An Improved Feature Selection Approach for Prediction of Students' Academic Performance in a Virtual Learning Environment

Felicia Ojiyovwi ADELODUN
LCU/PG/002519

Being a PhD Thesis Submitted to the Department of Computer Science, Faculty of Natural & Applied Sciences, Lead City University, Ibadan, Oyo State, Nigeria.

In Partial Fulfillment of the Requirements for the Award of Doctor of Philosophy Degree (PhD) in Computer Science.

2024

Certification

This is to certify that Felicia Ojiyovwi ADELODUN with matriculation number LCU/PG/002519 carried out this research work titled "An Improved Feature Selection Approach for Predictiohhn of Students' Academic Performance in a Virtual Learning Environment" in the Department of Computer Science, Faculty of Natural and Applied Sciences, Lead City University. Ibadan, Oyo State, for the award of Doctor of Philosophy Degree (PhD) in Computer Science and that this has not been previously submitted.

Dr. Wilson Sakpere
(Supervisor) **Date**

Dr. Wilson Sakpere
(Head of Department) **Date**

Dedication

This research work is dedicated to Almighty God who gave me the health, strength, and knowledge to complete this work.

Acknowledgement

I use this opportunity to thank Lead City University, Ibadan, Nigeria for allowing me to carry out this research work, I thank Google.com for the academic materials.

I thank my Supervisor and Head of Department, Dr. Wilson E. Sakpere for his dedication, words of encouragement and advice during this research work. I appreciate my lecturers, Professor S. O. Akinola, Dr. (Mrs) Afe, Dr. (Mrs.) Okesola, Dr. Ayoade, Dr. Sofowora, Dr. Waheed and Dr. Otobrise of the Department of Physics. They contributed in no small measure to ensure my progress in this work.

I thank TETFUND for giving me a scholarship to pursue this PhD.

My gratitude goes to my friend and colleague, Folakemi Famurewa, who has always encouraged me during this academic pursuit. I appreciate Mrs. Bukola Fawole's love and for checking on me always during my study.

I sincerely thank Mrs. Biodun Olubamiwa, Mrs.Dupe Fawale and Dr.Taiwo Abodunwa for always being there for me and for their financial support.

I sincerely thank Dr.(Mrs) Grace Tam-Nurse Man for her support and directions. I appreciate Dr. Awodele, Dr.(Mrs) Ayorinde, and Dr.Olusegun Omotosho for always encouraging me on my thesis.

I appreciate the love of my pastor, Pastor Atoyebi Oyelere and Minister Niyi Binuyo for their spiritual support.

My sincere appreciation goes to my sister, Sister Bernadette for her financial support, and for believing in me. May God always be with her.

I thank my children, Damisi and Anjola for their love and contributions. Their great understanding made it possible for me to complete my work on time. To the loving memory of my late husband, Kayode Adelodun, keep resting in peace.

Though the above–mentioned institutions and persons have assisted in the process of this research work, I alone stand responsible for the errors, if any found in the work.

Abstract

In the past, only machine learning algorithms were used for predicting Students' Academic Performance. In recent times both feature selection methods and machine learning algorithms are important in the prediction process. In previous researches, the focus have been on demographic information. Research specifically analyzing video interaction of learners is limited. This study provides an opportunity to investigate the interactions of learners in a Virtual Learning Environment (VLE), The study further investigated for clarification whether or not if Feature Selection should be skipped during the prediction process as some previous studies suggested. The study proposed a novel model named PF-PSO, as an improved Feature Selection (FS) method comprising a combination of three existing feature selection methods to improve machine learning models' accuracies in predicting students' academic performance in a VLE. The closen feature selection methods are Principal Component Analysis (PCA), Forward Selection Method (FOR) and Particle Swarm Optimization (PSO). Students' educational datasets were retrieved from secondary sources such as Kaggle.com. This unbiased study used two approaches- with FS and without FS to train machine learning models. The evaluation metrics include MSE, R^2 and MAE for the regression tasks. Accuracy, Precision, and F_1 measure for the classification tasks. The results from the study showed that while PSO proved promising, the proposed system achieved great success with Random Forest and Gradient Boosting performing very well in both regression and classification tasks and could explain 65% to 89% variance in the target variable. Logistic Regression was best for classification tasks with accuracy in the range of 61% and 75%. The proposed system can contribute to enhancing students' academic prediction. The findings of the study show the importance of incorporating a hybrid feature selection for predicting students' academic performance.

Keywords: Feature Selection, Pearson Correlation Coefficient, Principal Component Analysis, Forward Selection method and Particle Swarm Optimization.

Word Count: 301

Table of Contents

List of Tables

List of Figures

Chapter One

Introduction

1.1 Background to the Study

Computers are a major part of daily life in the modern world because of their many uses. Without computers, life might have been more difficult. Numerous professions, including engineering, architecture, the medical, economic, and entertainment sectors, employ computers. Teachers use computers in the classroom to help students do better academically and for research purposes. Massive volumes of student data are generated every day by educational institutions, computers, and the Internet. These data must be analyzed and preserved. Educational institutions make educated judgments about student retention and how best to understand students and their learning environment based on the findings of their analysis of student data. Predictive analytics, improved administrative support, and reduction of student dropout rates are examples of the benefits of education data mining (EDM). There are numerous data mining techniques available for various application types.

Two of the most important data mining techniques for EDM are regression and classification, which can be used to identify students who are failing, quitting school, or in need of extra help. They can also be used to offer information for curriculum changes and policy decisions. Various methods, like relationship mining, clustering, and visualization, can be applied to student data to identify trends and derive valuable insights that benefit both educators and learners. The technique of applying machine learning algorithms to analyze educational data to produce informative results is known as Educational Data Mining.

Educators are implementing digital systems like the Student Information System (SIS), Learning Management System (LMS), Massive Open Online Course (MOOC), and Virtual Learning System to support learning activities and improve students' academic performance in response to the recent growth and expansion of big data, artificial intelligence, and the Internet of Things (IoT). The level of focus, curiosity, enthusiasm, optimism, and passion students exhibit during a learning process is student engagement. This also includes how motivated they are to learn and further their education. Student engagement has three parts: behavioural, emotional, and cognitive. The amount of time a student dedicates to their academics is known as cognitive engagement. Students' actions in class, such as their attendance, participation, efforts, turn-in assignments, and questioning and answering, are all considered forms of behavioural engagement. Students' attitudes, reactions to teachers and other students, the course material, and their sense of belonging inside the school all constitute emotional engagement.

An interactive online space, like a virtual classroom, where students and teachers can communicate, is called a Virtual Learning Environment, (VLE). VLEs come in three primary varieties. They consist of hybrid, asynchronous, and synchronous systems. There are differences in the ways that students and teachers interact. Teachers are online at the same time as students when communicating are termed synchronous. They can speak with each other in real-time by using facilities like video conferencing, chat rooms, and virtual whiteboards. This type of learning is commonly used in courses with completion deadlines. Teachers and students do not log on at the same time in an asynchronous virtual learning environment. When it is most convenient for them, students can access the assignments and resources that their instructors provide online. For students who live in different time zones, this type of learning is beneficial. Synchronous and asynchronous learning are

combined in a hybrid virtual lea rning environment. Certain courses must be finished in isolation, while others must be finished instantly online.

E-learning is often referred to as virtual learning, remote learning, and distance online learning, It is the process of describing how students can use electronic devices to access educational materials outside of the traditional classroom.

Performance is the accomplishment of a task evaluated against predetermined benchmarks for speed, accuracy, and completeness.

Academic performance prediction is the measure of a student's future academic outcomes, such as grades, and overall achievement in a course or program, using data, statistical models, and algorithms. Predicting students' academic success helps experts create association rules so that they can make the right choices[1,2].

Feature selection is an important first step in creating a prediction model for students' academic achievement since it can help to discover the features that truly influence students' academic performance and boost prediction accuracy.

Machine learning uses a feature as input to generate predictions[3]. The features that affect student performance can be divided into four categories: demographic features, academic features, behavioural features, and other additional features[4]. Demographic features include student number, name, gender, and age. Academic features include school grades, class level, and semester. Behavioural features include discussion, access to resources, comments and other learning behaviour in an online learning environment.

A feature is a measurable attribute or variable that is utilized in a predictive model to predict a student's grades, likelihood of graduating, or other academic accomplishments.

The predictive algorithm uses features as inputs to find patterns and generate accurate predictions.

‘

There are three main types of feature selection methods: filter method, wrapper method and embedded method. Filler methods include methods such as ANOVA, Pearson correlation, and variance threshold. The wrapper method includes forward, backward, and stepwise selection, and embedded methods such as Lasso Regression, and Ridge Regression. Feature selection is the process of finding the best features out of all the available features using predetermined evaluation criteria. Feature selection eliminates redundant, unnecessary, or noisy data, lowers the dimensionality of the feature space, and improves the machine learning algorithm's speed and accuracy. Some machine learning algorithms have built-in feature selection methods like Classification and Regression Trees (CART)[5] and C4.5[6].

This study focuses on feature selection techniques like Principal Component Analysis (PCA), Forward Selection Method (FOR), and Particle Swarm Optimization (PSO). PCA is a statistical technique that reduces the complexity of high-dimensional data while maintaining trends and patterns. The Forward Selection Method is a stepwise regression statistical modelling strategy that is used to select a subset of important features from larger data. Starting with a blank model (no features), the approach adds features one at a time following a predetermined criterion, usually the one that yields the most significant improvement in model performance. PSO is a computer technique that attempts to enhance a potential solution iteratively to a specified quality metric. These three feature selection methods are existing techniques to improve students’ academic performance prediction. A hybrid strategy combines PCA with Random Forest (RF), Decision Tree (DT), Naïve Bayes (NB), and Support Vector Machine (SVM) to solve the

misclassification problem for improving the student's academic performance prediction[7]. PCA was used to reduce high-dimensional data complexity, allowing for more clearer presentation and analysis of complicated feature interactions[8]. To find relevant features, the forward feature selection method was utilized and then applied to Naive Bayes. The classification findings demonstrated a drop in model performance before feature selection and an increase in model performance following feature selection on student academic data[9]. PSO can be used to improve the prediction accuracy of students' academic performance[10].In a previous study, the performance of the machine learning algorithm is only focused on model interpretability; the feature selection phase is dropped which contradicts the classification model created to predict students' performance during the COVID-19 pandemic, where the prediction accuracy increased only when the dataset's ten most important features were incorporated.[11]. The BookRoll system used a variety of predictive models, including AdaBoost, K-Nearest Neighbour (KNN), Neural Network (NN), NB, and RF in its elementary information courses. The results indicated that when classified data was used, NB performed better than other algorithms, whereas RF had the best performance with raw data[12]. Research has indicated that the feature selection process may have an impact on classification models' accuracy[13]. In prediction research, feature selection plays an important role[14]. To solve an issue involving blast-induced ground vibration, feature selection approaches were used to choose features for the Random Forest algorithm[15]. Feature selection technique, in addition to the machine learning model, affects the accuracy or performance of a machine learning algorithm[16]. Feature selection can be used to improve the prediction of students' academic performance[17].

This study has followed the data mining method for building its predictive models. Data mining techniques have been used in many research to predict student performance. . Most educational institutions have switched from on-campus to online learning, the

COVID-19 pandemic brought attention to the necessity and significance of virtual education[18,19].

1.2 Statement of the Problem

Virtual learning environments (VLEs), have revolutionized education by providing accessible and adaptable teaching and learning platforms. However, because of the large and varied datasets that these platforms provide, forecasting students' academic achievement in VLEs continues to be a major challenge. These datasets contain demographic information, assessment records, activity logs, and engagement measures. In a virtual learning environment, predicting a student's academic achievement is a complicated process that depends on a number of variables, such as the student's behaviour, level of participation, and outside conditions. For example, virtual learning systems produce enormous volumes of data, such as interaction logs, job completion times, and login frequency. In many cases, these data are inconsistent or lacking. A student might overlook logging in. It can be challenging to quantify external factors that have a big influence on performance, like internet connectivity.

Many students are at risk of performing poorly or dropping out of VLEs due to the lack of reliable automated systems that forecast students' academic performance. while machine learning models have been widely applied for students' academic performance prediction, their effectiveness significantly depends on the quality of the features employed. Feature selection methods which identify the most relevant features for prediction, play an important role in improving model accuracy, reducing computational complexity, and enhancing interpretability. However, existing studies[20,21,22] often overlook the importance of systematically selecting features, leading to the inclusion of irrelevant or redundant features that could impair predictive performance. Notwithstanding the potential of sophisticated feature selection methods, little research has been done on how to use them

specifically to forecast academic achievement in a virtual learning environment. the focus has mostly been on demographic information and within the traditional classroom. The creation of effective, scalable, and precise prediction models specifically suited for virtual learning platforms is hampered by this gap.

Hence, this study proposed a novel feature selection model named PF-PSO, which comprises the combination of three existing feature selection methods: Principal Component Analysis (PCA), Forward Selection Method (FOR), and Particle Swarm Optimization (PSO) to increase the effectiveness and accuracy of feature selection. A previous study predictied academic performance of students by combining ranking and the heuristic approach(Genetic Algorithm)[23]. Our study improved the study by replacing the Genetic Algorithm (GA) with PSO for the following reasons: PSO exploits search space more efficiently and PSO requires fewer parameters to tune compared to Genetic Algorithm (GA). The proposed framework does not need to go through the various steps of crossover and mutation. The novel feature selection technique, PF-PSO suggested was used to predict students' academic performance in a virtual learning environment. The study further investigated for clarification whether or not if should Feature Selection be skipped during the prediction process and that the accuracy of prediction of students' academic performance should be based on only model interpretability as some previous studies suggested[24].

1.3 Aim and Objectives of the Study

This study aims to develop an improved feature selection model for the prediction of student's academic performance in a virtual learning environment.

The specific objectives of the study were to:

i. Pre-process student data collected from the publicly accessible Kaggle website.

ii. Develop a feature selection model using PF-PSO.

iii. Implement PF- PSO feature selection model.

iv Evaluation of PF- PSO feature selection model..

1.4 Research Questions

1. How can complex data with so many features be pre-processed?

2. In what way should the feature selection system that can predict students' academic performance with great accuracy be designed?

3. What metrics can be used to evaluate the performance of the proposed system?

4. Who are the beneficiaries of this proposed system and in what areas will they benefit?

1.5 Justification of the Study

The study will help;

- To improve students' academic performance
- To Identify students at risk for early intervention.
- For retention of students
- Provide information that will be used for decision-making concerning the students and the institutions.

1.6 Scope of the Study

The study considered personal information, demographical, social, and academic information and the behavioural patterns of students in virtual learning environments. The datasets used for the study were high-dimensional educational data.

1.7 Limitations of the Study

The datasets were obtained from relevant online databases consisting of education datasets, and popular databases such as the Kaggle.com repository. Real and local could be used to test the study.

1.8 Operational Definition of Terms

Pruning: Pruning involves removing branches from the tree to simplify it and improve generalization.

Over-fitting: This is an undesirable machine learning behaviour that occurs when the machine learning model gives accurate predictions for training data but not for new data.
Prediction: This is the forecast of what will happen in the future.

Academic Performance: This refers to how well students do in achieving their short and long-term educational goals.

Metaheuristic: A general-purpose algorithmic framework that can be applied to different optimization problems with relatively few modifications.

Hyperplane: This is the decision boundary that differentiates the two classes in the Support Vector Machine. It separates different classes in the feature space.

Log Odd: log odd converts the logistics regression model from a probability-based to a likelihood-based model. Log odd makes translating output easier.

Curse of Dimensionality: A situation where there is an extensively larger number of features compared to the number of instances.

Dimensionality Reduction: The transformation of feature space into lower dimensional space while preserving most of the relevant information.

Features Engineering: Is the process of selecting, transforming, manipulating and constructing new variables from raw data using domain knowledge.

Confusion Matrix: A table that shows information about actual and predicted classifications in given classification problem.

Endnotes

[1] L.Ye, X. Meng, Y.Hang & S Tayeb, "*Reseach on Visual tracking method for students' browsing data in art literacy online education*," **Journal of Network Intelligence**, 5, 2020.

[2] H. Xiong, X. Huang, M. Yang, L. Wang, & S. Yu, " *Unbounded and efficient revocable attribute-based encryption with adaptive security for cloud-assisted internet of things*," **IEEE Internet of Things Journal**, 9, no. 4, 2021:3097-3111.

[3]https://www.tecton.ai>Blog.

[4]B.K.Francis & S.B.Sasidhar,"*Predicting academic performance of students using a hybrid data mining approach,*" **Journal of Medical Systems** 24, no.6 2019: 3577- 3589.

[5]G. S. Kori & M. S. Kakkasageri,"*Classification And Regression Trees (CART) based resource allocation scheme for Wireless Sensor Networks*," **Computer Communications** 197, 2023, pages 242- 254.

[6]F.U. Widowati," *Application of C4.5 algorithm with PSO Feature Selection and Bagging Technique on Breast Cancer Classification*," **International journal of Management Science and Information Technique** 4 (2), 2024, pages 312 – 320.;

[7]P. Sokkhey & T. Okazaki, "*Hybrid Machine Learning Algorithms for Predicting Academic Performance*," **International Journal of Advanced Computer Science and Applications** 11, no. 1,2020.

[8]A. Selim, I. Ali & B. Ristevski," *University Information System's Impact on Academic Performance: A Comprehensive Logistic Regression Analysis with Principal Component Analysis and Performance Metrics*," **TEM Journal**, 13, no. 2, May 2024:1589-1598.

[9]A. Saifudin, E. Yulianti & T. Desyani, " *Forward Selection Technique to choose the Best Features in Prediction of Student Academic Performance Based on Naïve Bayes*," ***I*CComSET 2019.Journal of Physics**: Conference Series. IOP Publishing. 1477 2020,

[10]L. Yue, P. Hu & J. Zhu, " *Binary Particle Swarm Optimization Predicting Students' Academic Performance in College English*," **Journal of Network Intelligence. Taiwan Ubiquitous Information** 9, no. 2, May 2024.

[11] L. Ihab, H. Alsammak, A. H, Mohammed, & I. S. Nasir," *E-learning and COVID-19: Predicting Student Academic Performance using Data Mining Algorithms*," **Webology** 19, no. 1 2022: 3419-3432,

[12]G.Akcapinar, M.N.Hasnine, R.Majumdar, B.Flanagan, & H.Ogata," *Developing an early-warning system for spotting at-risk students by using eBook interaction logs*," **Smart Learning Environments**, 6(1) no. 4. 2019.

[13]E. D. Evangelista,"*A hybrid Machine Learning Framework for predicting students' performance in virtual learning environment*," **International Journal of Emerging Technologies in Learning (IJET***)* 16, no. 24 2021: 252- 272.

[14]F. Qiu, G. Zhang, X. Sheng, L. Zhu, Q. Xiang, B. Jiang & P. Chen," *Predicting, students' performance in e-learning using learning process and behaviour data*," **scientific report***s* 2022 www.nature.com/scentitificreports .

[15]H. Zhang, J. Zhou, D. J. Armaghani, M.M, Tahir, B. T. Pham, & V. Huynh, "*A Combination of Feature and Random Forest Techniques to Solve a Problem Related to Blast-induced Ground Vibration*," **Applied Science** 10, no.3 27 January 2020:869.

[16]P. Manikandaprablu,"*Feature Selection Methods: A study*," **compliance Engineering Journa***l*.12 no.5 2021.

[17]W. Nuankaew & J. Thongkam, "*Improving student academic performance prediction models using feature selection*,"17th International Conference of Electrical Engineering and Electronics Computer, Telecommunications and Information Technology (ECTI-CON) 2020.

[18]C. T. Martin, C. Acal, M. El-Honrani & A.C.M. Estrada, "*Impact on the Virtual Learning Environment Due to COVID- 19*," **Sustainability**, vol. 13, no. 2, pp 582, 2021.

[19]S.Kurbakova, Z. Volkova, & A. Kurbakov, "*Virtual Learning and Educational Environment: New Opportunities and Challenges Under the COVID-19 Pandemic*," The 4th International Conference on Education and Multimedia Technology*, 2020*, pp. 167 – 171.

[20]S. Verma, R.K. Yadav,& K.Kholiya, "*Prediction of Academic Performance of Engineering Students by using Data Mining Techniques*,"**International Journal of Information and Education Technology**, vol. 12 No.11, November,2022.

[21]M.G. Rao & K.K Kumar, "*Students Performance Prediction on Online Courses Using Machine Learning Algorithms*," **United International Journal For Research & Technology** Volume 02, Issue 11, ISSN: 2582-6832, 2021

[22]I. Khan, A. Al- Sadiri, A.R. Ahmad & N.Jabeur, " *Tracking Student Performance in Introductory Programming by Means of Machine Learning*" MEC International Conference on Big Data and Smart City (ICBDSC), Muscat, Oman, pp. 1-6, 2019

[23]W. Xiao, P. Ji & J. Hu, "*RnkHEU: A Hybrid Feature Selection Method for Predicting Students' Performance*," **Hindawi Scientific Programming** 2021, no.1 2021: 1670593.

[24]A. Al-Zawqari, D. Peumans & G. Vandersteen, "*A flexible feature selection approach for predicting the students' academic performance in online courses*," **Computers and Education: Artificial Intelligence** 3, 2022, 100103.

Chapter two

Literature Review

This chapter reviews some theoretical frameworks that are related to this study. An overview of the basic concepts which are fundamental to this research beginning from machine learning (ML), through feature selection method to student academic performance. Related works are also considered.

2.1 Machine Learning

Machine learning is a sub-field of AI where computers learn from data, discover patterns, and predict outcomes[1]. Large and complicated data sets can be automatically and swiftly analyzed by machine learning techniques, yielding accurate findings.

Based on the collected data, machine learning algorithms are helpful tools for early prediction of low-performance students. This technique is more sophisticated than traditional approaches, which employ student records such as attendance, quiz scores, exam results, and marks to assess and predict the academic performance of the students.

Assessment systems that employ data mining and machine learning techniques to predict student progress based on educational data are now available because of recent developments in the field of education.

Within huge and complicated datasets, machine learning has demonstrated the ability to reliably predict the academic success of students. However, the curse of dimensionality, where there are far more features than instances makes it difficult to create an appropriate prediction model based on educational datasets.

Feature selection, which aims to extract just the most relevant features and eliminate noisy, unnecessary, and redundant features from the dataset, improves the ability to generalize machine learning models.

2.2 Types of Machine Learning Algorithms

Large volumes of historical data are processed by machine learning algorithms so that they can discern patterns in the data.

Machine learning (ML) algorithms can be classified into two main categories. Supervised learning algorithm and unsupervised learning algorithm.

2.2.1 Supervised Learning Algorithm

With the help of the labelled data, this algorithm builds a learning model whose output is either a continuous numerical value or a categorical label. If the output of the learning model is a continuous numerical value, it is called regression; if it is a categorical label, it is called a classifier.

The most commonly used supervised machine learning algorithms are decision tree, logistic regression, linear regression, and support vector machine.

2.2.2. Unsupervised Machine Learning Algorithm.

An unsupervised machine learning method builds a learning model called a clustering model by analysing unlabeled data. Clustering models are used to classify or categorise the provided data to predict or identify a group or cluster. The most commonly used unsupervised machine learning algorithms include the k-means clustering algorithm, hierarchical clustering algorithm and apriori algorithm.

2.3 Approaches to Supervised Machine Lea,/rning

2.3.1 Random Forest

Random forest is one of the most well-liked and effective ensemble techniques in machine learning for regression and classification. Employing training data, random forest fits a linear function between the observed response (y') and the prediction (y). When there is a non-linear relationship between y and y', the linear model does not perform well[2]. Predictions from several decision trees are averaged using a random forest. Intuitively random forests average multiple huge trees to achieve good accuracy. Large trees typically have great variation and low bias. One can lessen the variation by averaging so that minimal bias and variance can be simultaneously achieved using random forests. Random forest accuracy has been shown to be competitive with several other supervised machine learning methods[3].

The majority of random forests are parameter-free because they frequently employ the default settings, and this makes it difficult to further improve random forest by parameter tuning[4].

2.3.2 Gradient Boosting

Gradient boosting is an ensemble learning method used for regression and classification tasks. Iteratively, it builds a group of weak learners, such as decision trees, with each new model resolving faults caused by the preceding models[5]. Gradient boosting's primary objective is to minimize the residuals in order to optimize a loss function. Boosting algorithms iteratively merge weak learners into a strong learner. Mathematically gradient boosting can be expressed as the following; The goal of gradient boosting, given a training dataset, $D = (x_i, y_i)N$, is to minimize the expected value of a given loss function, $L(y, F(x))$, to obtain an approximation, $F(X)$, of the function $F^*(x)$, which maps instances X to their

output value y. Gradient boosting builds an additive approximation of F*(x) as a weighed sum of functions.

$F_m(x) = F_{m-1}(x) + pmhm(x),$ Equation (2.1)

Where pm is the weight of the m^{th} function, hm(x). These functions are the models of the ensemble (for example: decision trees). Some popular variants of gradient boosting include XCGBoost (Extreme Gradient Boosting), Light GBM (light gradient Boosting Machine), and Cat Boost.

2.3.3 Decision Trees

A decision tree is a flexible machine learning approach which builds hierarchical structures to help with data-driven decision-making and prediction. It is imperative to comprehend the fundamental ideas and structure of decision trees to utilize them efficiently in a variety of machine learning applications. Random Forest and Gradient Boosting are two examples of ensemble techniques that can be used to improve decision trees.

These techniques reduce over-fitting and increase accuracy by combining the predictions of several decision trees. ID3, C4.5, C5 and CART are examples of decision trees. The basic structure of the decision tree consists of the following elements;

- The root node refers to the whole dataset.
- Internal nodes serve as decisions or feature tests.
- Edge: An edge is a node that links nodes and shows the result of a test or decision made at the parent node.
- The leaf node holds the final decision or prediction, which might be a numerical value in the case of regression or a class label in the case of classification.

- Features and thresholds are employed to determine the branching of the tree and to divide the data into subgroups.
- Branching; branches link nodes and depict the route from a root node to a leaf node, illuminating the series of choices taken in order to arrive at a conclusion or prediction.
- Depth: This is the longest path from a tree's root node to its leaf node.

2.3. 4 Extra Trees

Several randomized decision trees (Extra Trees) are fitted on different subsamples of the dataset using the Extra Trees ensemble learning technique. The predicted accuracy is enhanced and over-fitting is managed through the use of averaging.

Regression and classification applications can both benefit from the use of additional trees.

It can be applied to enhance the adaptability of the model.

2.3.5 K-Nearest Neighbour

K-Nearest Neighbour (K-NN) are instance based supervised lazy learning algorithms used for classification and regression tasks. The basic principle of K-NN is that a data point with similar characteristics is close to each other in feature space. K-NN works by first choosing the number of the nearest neighbours. Second, distance metrics like Euclidean Manhattan and Minkowski distances are used to calculate the distance between each data point and every other data point in the dataset. Thirdly, K-NN will determine which K data points in the dataset are the closest to the other data points.

Lastly, in the case of classification, voting is the process of assigning the class label that is most prevalent among the K-Nearest Neighbors to the target output; in the case of regression, K-NN predicts the target value as the average of the K-Nearest Neighbors'

target values K-NN is simple to use and comprehend. There is no need for a training phase, this makes it computationally efficient.

2.3.6 Multilayer Perception

Multilayer Perception (MLP) is a type of neural network. It belongs to a family of deep learning. MLPs are used for tasks such as classification and regression. MLP is a class of feed-forward artificial neural network (ANN). An MLP consists of at least three layers: an input layer, one or more hidden layers and output layers. Each layer contains neutrons that are fully connected to the next layer. Each neuron applies a non-linear activation function such as ReLU, sigmoid or tanh, to the weighted sum of its inputs the activation function can learn complex patterns in data. Data is passed through the network from the input layer to the output layer. Through the use of backpropagation methods, MLP learns by computing the gradient of the loss function and modifying the weights to decrease error. MLPs work especially well in situations where the data can be divided linearly after being transformed by hidden layers. MLPs are relatively easy to implement and can approximate any continuous function, but, they may have problems processing complex, high dimensional data.

2.3.7 Support Vector Machine

Support Vector Machine (SVM) was developed by Vladimir Vapuik and Alexey Chervonenkis in the 1960s, one of the most effective and reliable supervised machine learning algorithms is support vector machines (SVM), which may be used for both regression and classification problems[6]. The main idea behind SVM is to locate the best hyperplane, or decision boundary, to divide the data points belonging to various classes in the feature space, The training examples that influence the optimal hyperplane's position and orientation are known as support vectors.

To increase the generalisability of the SVM model, SVM maximizes the distance between the hyperplane and the closest data points.

Application areas of SVM include classification tasks like text categorization, image recognition and bioinformatics and regression tasks like Support Vector Regression (SVR) and time series prediction.

2.3.8 Support Vector Regression

Support Vector Regression (SVR) is the modification of the support Vector Machine (SVM) for regression problems. Finding a function that roughly represents the relationship between an input variable and a continuous target variable while retaining good generalizability to new data points is the main goal of support vector regression (SVR). SVR is better at handling outliers than traditional regression techniques.

There are several variants of SVR and they include Linear SVR, Polynomial SVR, RBF (Radical Basis Function) SVR, Nu-SVR, Epsilon-SVR, Laplacian SVR and Quadratic SVR.

2.3.9 Logistic Regression

Logistic Regression is a statistical technique used for binary classification. It makes predictions about the likelihood of a binary result (0 or 1) using one or more predictor factors. The sigmoid function, a logistic function, is used in logistic regression to model the likelihood that an input point falls into a specific class. To represent the relationship between predictor variables and a binary result, logistics regression utilizes log odds.

2.3.10 Adaboost

Boosting is a method for improving the predicting ability of the learning system and coordinating learning. Adaboost, which stands for Adaptive Boosting is a supervised

machine learning algorithm. It is a method for improving the predicting ability of the learning system. Adaboost creates a strong classifier by combining several weak classifiers.

Every new model in the iterative process is built upon errors committed by the one before it.

The process of Adaboost includes the following;

- Obtain a dataset
- Start weighing every sample.
- Develop a poor classifier (such as decision Stump).
- Calculate the classifier's error rate.
- Make weight updates
- Give the sample that was incorrectly classified more weights.
- Reduce the weights of the samples that were successfully classified.
- Determine the classifier's weight.
- Make classifier updates.
- Repeat steps until the error rate falls within a specified criterium level.
- Generate the final model.

2.3.11 Naïve Bayes

Naïve Bayes is a probabilistic algorithm based on Bayes' theorem used for classification tasks. In the Naïve Bayes algorithm, it is assumed that the features in each class label are independent of each other, this is advantageous because only a small amount of training data is required to estimate the parameters required for the classification. Naïve Bayes is robust to noise. Naïve Bayes is easy to construct, interpret and exhibits high accuracy and speed when applied to a large database.

Bayes' theorem:

$$P(A/B) = \frac{P(B/A)P(A)}{P(B)}$$ Equation (2.2)

Where,

P(A/B): the posterior probability

P(B/A): the likelihoods

P(A): prior probability

P(B): predictor prior probability

2.4 Feature Selection

Data from a variety of sources, including the Internet, banking, healthcare, and education, could contain thousands or even millions of attributes for each instance. These attributes are known as features. Data used as input for machine learning models to generate predictions is called a feature. Some of these features are redundant and irrelevant to the machine learning model. Irrelevant features are features that are unrelated to the intended outcome of the machine learning model. They lower the machine learning model's performance and accuracy. Redundant features are features that are duplicated or whose values are perfectly connected. It has been ,proven that having redundant features slows down machine learning and lengthens computation times[7].

The curse of dimensionality indicates the presence of redundant and irrelevant features in the dataset. Redundant features supply redundant information, while irrelevant features do not provide any valuable information for the machine learning process[8]. A high-dimensional dataset is one that has many more features than the dataset as a whole, leading to a large number of dimensions[9].

The accuracy or performance of a machine learning algorithm does not only depend on the machine learning model alone but also on the feature selection method[10].

The relevant features must be chosen in order to increase the accuracy of the machine learning algorithms that are used to predict students' academic performance.

High-dimensional data and noisy elements that can affect the predicted outcomes are common issues with educational datasets. Many characteristic include redundant and unnecessary features, which slows down the performance of the machine learning algorithm, complicates data interpretation, and produces inaccurate results. Feature selection is frequently used to solve problems with noisy features and large dimensional datasets. The primary goal of the feature selection technique is to maximize the subset of relevant features and reduce redundancy while retaining high accuracy and preserving crucial information. Feature selection is the technique of limiting the input variable to a predictive model by eliminating noise and using just relevant data:

- Feature selection can optimize predictive model several ways: prevent learning from noise (over-fitting), improved accuracy, and reduce training times. A model's ability to generalize successfully to unobserved cohorts depends on feature selection. The goal of feature selection is to produce a high-quality dataset with which to train the predictive model.
- The feature selection technique is applied to obtain a feature subset to lower the likelihood of prediction errors Furthermore, in order to achieve high predictive value accuracy, features that are very relevant are selected. Most feature selection methods, however, yield feature selection solutions that, lie in the zone between sub-optimal and nearly optimal. Throughout the search process, these searches are conducted locally as opposed to globally.
- Feature selection is one method that is crucial to improving the efficiency of machine learning algorithms. Feature selection is a method that helps reduce dataset size by removing redundant and unnecessary features from the dataset, which improves the performance of machine learning algorithms, accelerates the learning process, and builds simpler models.

2.5 Feature Extraction

Feature extraction is a process that entails converting high-dimensional data into a lower-dimensional space, finding a new set of K dimensions that are combinations of the original dimensions. The most popular and extensively utilized technique for extracting features is Principle Component Analysis.

2.6 Types of Feature Selection Method

One of the most important processes for assessing high dimensional data in the fields of data mining and machine learning is feature selection. The goal of feature selection is to minimize information loss while choosing the best features or a minimum subset of characteristics. Three primary categories of feature selection techniques exist. These consist of the filter, wrapper and the embedded methods. The three primary feature categories are the basis for further techniques like the hybrid and ensemble methods. The .ways in which these feature selection techniques vary include their incorporation into machine learning algorithms, assessment metrics, computational complexity, and capacity to identify feature interactions and redundancies.

2.6.1 Filter Method

These methods select features based on statistical properties like correlation, mutual information, or significance tests. Feature ranking is the evaluation metric for feature selection in the filter method. The features are ranked based on their scores in various statistical tests for their correlation with their class. The features that score above a certain threshold are removed. The filter methods are independent or separated from the machine learning algorithm. This makes filter method free from machine learning algorithm bias which reduces over-fitting and can be generalized and not fine-tuned to any specific machine learning algorithm and less computationally demanding and very suitable to high

dimensional data. Filter methods can be grouped into univariate and multivariate. Univariate methods include X^2 test, fisher's exact test, information gain, Euclidean distance, Pearson correlation, Mann-Whitney U test, and t-test) have attracted the most attention in domains that work with high dimensional dataset because of their speed and simplicity. Multivariate methods are more computationally heavy than univariate methods so cannot be effectively scaled or suitable for very high dimensional data. Filter method is fast and usually the good approach when the number of feature is huge and avoids over fitting but sometimes it may fail to select best features.

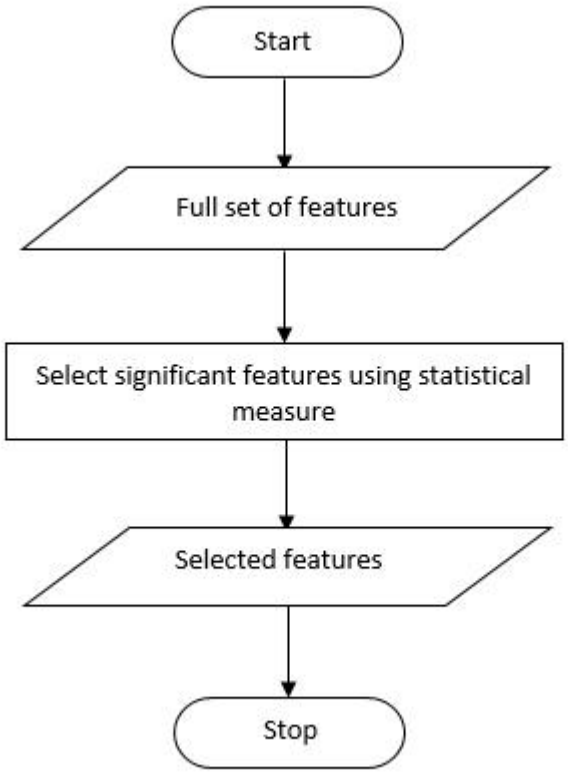


Figure 2.1: Filter feature selection method

(Source: Researcher, Adelodun F.O. 2024)

2.6.2 Wrapper Methods

Wrapper method uses the performance of a chosen machine learning algorithm as the evaluation metric to select the best feature subset. The wrapper method are dependent on the machine learning algorithms for selection of the best performing set of features for the chosen machine learning algorithm[11]. Wrapper method haven been proven to have a higher predictive performance than can be obtained in filter method[12]. Wrapper methods take into consideration of feature dependencies, interactions and redundancies during the selection of the best performing subsets while filter methods only take into considerations only of feature relevance. However, because of the high number of computations required to generate and evaluate feature subsets, wrapper methods are computationally heavy compared to filter and embedded methods.

Wrapper methods are dependent on the machine learning used, so there is no guarantee that the selected feature will remain optimum if another algorithm is used. Therefore, it cannot be generalized to external dataset. It will be prone to over-fitting Filter methods produces a ranked list of features and wrapper methods produces a best performing feature subset as the output. Examples of wrapper methods includes forward selection, backward elimination, and Recursive Feature Elimination (RFE).

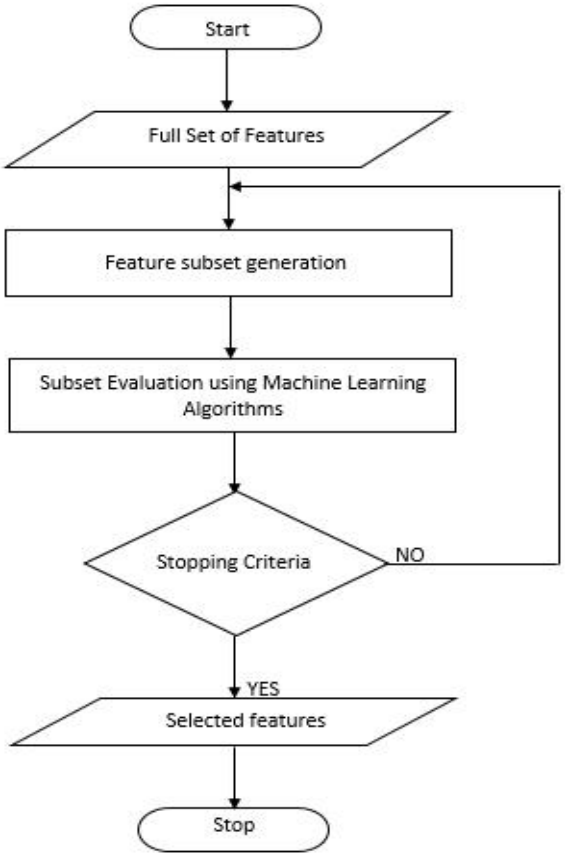


Figure 2.2: Wrapper selection methods

2.6.3 Embedded Method

Embedded method combines the qualities of both the filter and wrapper methods[13]. These methods perform feature selection as part of the model training process. The embedded method is built into machine learning algorithm. During the training stage, the machine learning algorithm adjusts its internal parameter and determines the appropriate weights given for each feature to produce the best accuracy. Examples of embedded methods include decision trees, LASSO (Least Absolute Shrinkage and Selection), Elastic Net, and Ridge Regression. Feature selection is optimized resulting in implied model performance, but they are limited to specific machine learning algorithms hence hindering their application across different models. They are computationally expensive. Advantages of Embedded methods include taking into consideration the interaction of features, they are

faster like filter methods, they are more accurate than filter method, they find the feature subset for the algorithm being trained and they are less prone to over-fitting.

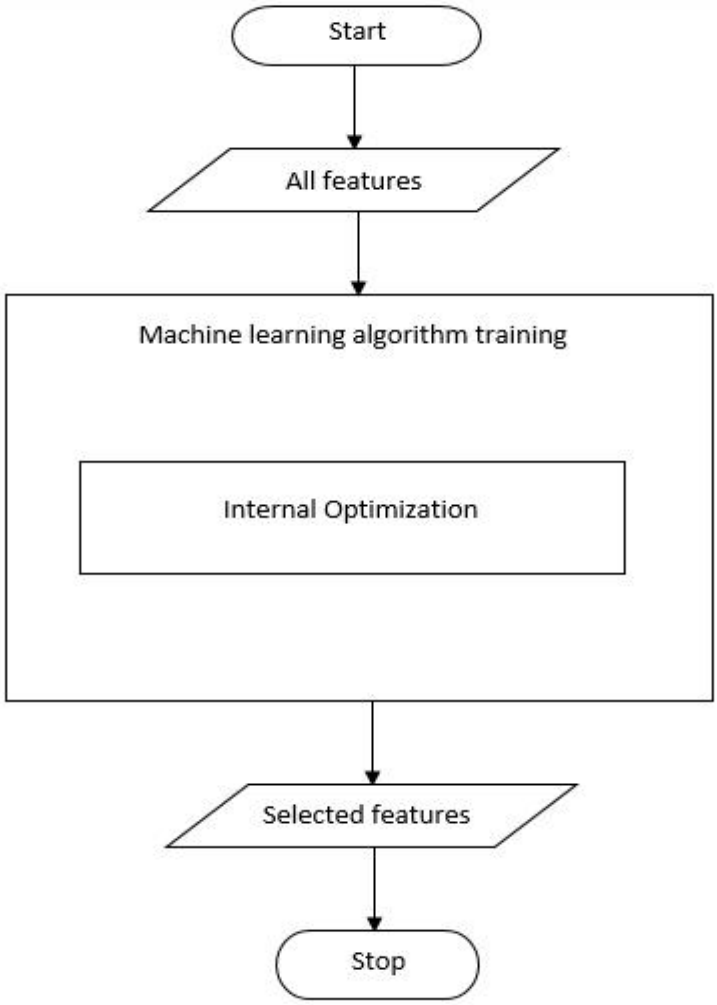


Figure 2.3: Embedded feature selection method

2.6.4 Ensemble Method

This method involves combining different feature selection methods and applying them to a dataset assuming that they will produce an output that is better than the output of a single algorithm. An ensemble of multiple feature selections combines the strengths of different methods while overcoming their weaknesses[14].

Ensemble methods try to increase the prediction accuracy by combining the results from multiple base classifiers. Random Forest is a class of ensemble methods that uses decision trees as weak learners Gradient Boosting (GB) is an ensemble classifier[15]. They are robust and versatile. They combine different feature selection methods leveraging the strength of each method to improve overperformance. They are applicable in different machine learning algorithms and problem domains. Implementing the ensemble method can increase complexity and computational cost. The results of the ensemble method may be difficult to interpret compared to those of individual methods and they are sensitive to parameter tuning.

2.6.5 Hybrid Method

The hybrid method is a feature selection that involves the combination of different feature selection approaches in order to take advantage of the strengths of the component methods for example, univariate filter-wrapper hybrid methods incorporate a univariate filter method as the first step to reduce the initial feature set size thus limiting the search space and computational load for the subsequent wrapper step. In this instance, the filter method is used because of its simplicity and speed. By contrast, the wrapper method is used because it can model feature dependencies and allow interactions with the learning algorithm, thus producing better performance.

There is always a tradeoff between computational complexity and performance in feature selection. In this context, hybrid methods can be considered as a 'middle ground' solution between the simple filter method and the more computationally complex wrapper and embedded methods, indeed many examples in the literature have shown that a hybrid method tends to produce better performance than filter method while also being less computationally expensive than a pure wrapper method. The hybrid of filter and wrapper

feature selection techniques performs better than those based solely on filter or wrapper techniques. The hybrid method is less computationally complex than those based on the wrapper technique while preserving its relatively higher accuracy than the filter technique. At present, no particular feature selection technique is superior to the others. Each technique has advantages and disadvantages.

2.7 Feature Selection Statistics

A feature selection model can be selected based on the input and output data. Use the Spearman's rank coefficient or the Pearson correlation coefficient if the input and output data are numerical. Use Kendall's rank coefficient and the ANOVA correlation coefficient if the input and output data are numerical and categorical, respectively, and apply the Chi-squared test (contingency tables) and mutual information if the input and output data are categorical in characteristics.

2.8 Students' Academic Performance Prediction.

A prediction is an anticipation of future events. Identifying successful students is less difficult than identifying students who could struggle. Academic performance refers to how well student do in achieving their short and long terms educational goals which can be measured through grades, test scores and their overall comprehension of academic material. The prediction of students' academic performance attempts to investigate data that is helpful to the student's learning process. Education institutions can profit from accurate student performance prediction by enhancing the student learning process and elevating the overall quality of their institutions. Machine learning models are often employed to predict to inform educators and institutions about early interventions, and personalized learning plans for improving students' future academic achievement.

2.9 E-Learning

The postal learning approach, which Sir Isaac Pitman pioneered in 1840 and is regarded as the first distance learning course, gave birth to e-learning[16].

Online classrooms are a sophisticated educational format known as e-learning.

The e-learning platform makes use of intelligent methods to gather important user data, like how often students access the system, how accurately they answer questions, and how many hours they spend reading and viewing tutorial videos. Massive Open Online Courses (MOOCs) and Learning Management Systems (LMS) are examples of e-learning platforms. The educational system is not new to e-learning, during the COVID-19 epidemic, it evolved into a useful supplement to traditional classroom-based instruction[17].

2.10 Data Mining

Data mining is the process of automatically extracting from raw data any relevant information. Educational data mining is the term for data mining techniques used in the field of education. Within the field of computer science, data mining is a recognized field. Late in the 1980s, data mining had its beginnings. It is a smaller procedure inside the larger Knowledge Discovery in Database (KDD) process.

Data cleansing, data integration, data selection, data transformation, pattern evaluation, and knowledge representation are additional sub-processes that are a component of the KDD processes. Finding Knowledge in Databases

- Knowledge Discovery in Database (KDD) are presented in the following order;
- Data cleaning: removing the unwanted data or noise in the datasets.
- Data integration: combining multiple data sources.
- Data selection: select related data to the task from the database
- Data transformation: converts the data into an appropriate form that will be easy to be processed.
- Data mining: a process such as regression, classification and association to extract data patterns.

- Pattern evaluation: evaluate the output of the data mining process and identify the interesting measures.

Knowledge representation: various techniques are used to present the mined or processed data to the users who will in turn make use of the acquired knowledge to make better decisions. The two major tasks of data mining are the predictive and descriptive tasks. The predictive tasks include classification, prediction and time-series analysis, and the descriptive tasks include association, clustering and summarization[18].

2.11 Predictive Modeling Approaches

Regression and classification are two predictive approaches in statistics and machine learning below is a short explanation of these approaches

2.11.1 Regression

Regression is a statistical method used for estimating the relationships among variables. It focuses on predicting a continuous outcome variable (the dependent variable) based on one or more predictor variables (the independent variable). The key point to the note is that the output variable in the regression is continuous, which means that it can take any value within a range. Common types of regression include linear regression, where the relationship can be more complex. Regression is often used for forecasting, determining the strength of predictors and trend analysis.

The evaluation Metrics of regression include Mean Square Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) etc. Examples of regression algorithms include Linear Regression, Polynomial Regression, Ridge Regression, and Lasso Regression.

2.11.2 Classification

Classification is categorizing or classifying an item into a predefined set of categories or classes. It involves building a model that assigns new observations to one of the several classes based on the features of the data.

The output variable in classification is categorical, not numeric. It can be binary (for example, yes/no, spam/not spam) or multi-class, for example, categories of disease. Common machine learning algorithms used for classification include logistic regression, decision trees, support vector machines and neural networks.

Classification is widely used in student academic prediction, email filtering (spam or not spam) image recognition, medical diagnosis and more. Evaluation metrics used in classification include Accuracy, precision, recall, F_i score, ROC curve, and confusion matrix.

The main difference between regression and classification is that regression predicts a continuous value while classification predicts a categorical value. Regression is used to predict a value, whereas, classification is used to separate data into classes and different metrics are used to evaluate the performance of regression and classification models. Classification problems can sometimes be more complex due to separating data into distinct categories.

2.12 Linear Regression

Linear regression is a fundamental statistical technique used to model the relationship between a dependent variable often denoted as *y* and one or more independent variables often denoted as x_1, x_2, …, x_n. It assumes that there is a linear relationship between the independent variables and the dependent variable.

The basic form of linear regression can be represented as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon \qquad \text{Equation (2.3)}$$

Where:

- *y* is the dependent variable (the variable we are trying to predict).

- $x_1, x_2, \ldots, x_n$ are the independent variables (also called predictors or features)
- β_0 is the intercept (the value of y when all independent variables are zero.
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients (also called slopes) that represent the change in for a one-unit change in the corresponding independent variable, holding all other variables constant.
- ε is the error term, representing the difference between the observed value of y and the value predicted by the model.

Linear regression is a modelling technique proposed by Sir Francis Galton in 1894 to predict the value of a dependent variable. It quantifies the relationship between the dependent variable and independent variables (predictors) Both correlation and linear regression provide the opportunity to understand the dependent variable and independent variable relationship, while, correlation provides a quantitative way of measuring the degree or strength of a relation between two variables, regression mathematically describes this relationship. In correlation analysis, the correlation coefficient "r" is a dimensionless.s number whose value ranges from -1 to +1. A value towards -1 indicates a negative relationship whereas a value towards +1 indicates a positive relationship. When there is a normal distribution, the Pearson correlation is used but with non-normal distributed data, the Spearson rank correlation is used. The coefficient of determination is the portion of the total variation in the dependent variable that can be explained by the variation in the independent variable(s). When R^2 is +1, there exists a perfect linear relationship between the dependent variable and independent variable, that is, 100 percent of the variation in the dependent variable is explained by the variation in the independent variable. When it is $0< R^2 < 1$, there is a weaker linear relationship between the dependent variable and independent variable, that is, some, but not all of the variations in the dependent variable can be explained by the variation In the independent variable.

The goal of linear regression is to estimate the coefficients (slopes and intercept) that minimize the difference between the observed values of the dependent variable and the values predicted by the model. This is typically done by minimizing the sum of the squared differences between the observed and predicted values, a method known as ordinary least squares (OLS) regression.

Once the coefficients are estimated, the linear regression model can be used to make predictions for new data by plugging in the values of the independent variables into the equation.

Linear regression is widely used in various fields such as science, economics, finance, biology, engineering, and social sciences for prediction, forecasting, and understanding the relationships between variables. However, it has assumptions, such as linearity, independence of errors, constant variance of errors, and normality of errors, which should be checked before interpreting the results. Additionally, it may not perform well with nonlinear relationships or when there are outliers in the data.

2.13 Ant Colony Optimization

Ant Colony Optimization (ACO) is an approach to problem-solving that is inspired by the foraying behaviour of some ant species. These ants deposit pheromones on the ground to mark some favourable path that should be followed by the other members of the colony. ACO exploits a similar mechanism for solving optimization problems. Other ants perceive the presence of pheromone and then follow paths where pheromone concentration is higher. In ACO, several artificial ants build solutions to the considered optimization problem at hand and exchange information on their quality via a communication scheme that is reminiscent of the one adopted by real ants. ACO is a type of metaheuristic algorithm. A metaheuristic is a set of algorithmic concepts that can be used to define

heuristic methods applicable to a wide set of different problems. In other words, a metaheuristic is a general purpose algorithmic framework that can be applied to different optimization problems with relatively few modifications.

There are three main ACO algorithms, proposed Ant System (AS), MAX-MIN Ant system and Ant Colony System. Traveling salesman problem is used to illustrate the ACO algorithm; at each iteration, the pheromone value are updated by all the ants that have built a solution in the iteration itself. The pheromone associated with the edge joining cities *i* and *j* is updated as follows

$$\tau_{ij} = (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad \text{Equation (2.4)}$$

Where ρ is the pheromone evaporation rate, m is the number of ants and ρ_{ij} is the quality of pheromone laid on edge (*i,j*) by ant K such that

$$\Delta\tau_{ij}^{k} = \begin{cases} \left(Q/L_k\right)_{ij} & if \quad ant \quad K \quad travels \quad on \quad edge \quad (i,j) \\ 0 \quad otherwise \end{cases} \qquad \text{Equation (2.5)}$$

Where Q is a constant and L_K is the length of the tour constructions of a solution, ants select the following city to be visited through a stochastic mechanism. When ant K is in city *i* and as so far constructed the partial solution S^P, the probability of going to city j is given by;

$$P_{ij}^{k} = \begin{cases} \dfrac{\gamma_{ij}^{\alpha}\ \eta_{ij}^{\beta}}{\sum_{\pi ijl \in N(S^p)} \tau_{ij}^{\alpha}\ \eta_{ij}^{\beta}} & if\ \tau_{ij} \in N(S^p), \\ O \qquad otherwise \end{cases} \qquad \text{Equation (2.6)}$$

$N(S^P)$ is the set of feasible components, that is, edges (I, L), where L is a city not yet visited by ant K and I represents the current city where ank K is located. The heuristic function, η_{ij} represents the heuristic value of moving from city i to city j which is given by

$$\eta_{ij} = \frac{1}{d_{ij}} \qquad \text{Equation (2.7)}$$

Where d_{ij} is the distance or cost between cities i and j.

2.14 Genetic Algorithm

The Genetic Algorithm (GA) is an adaptive technique used to learn and solve complex problems. GA is derived from natural selection and genetic concepts. It utilizes selection, crossover and mutation operators to effectively manage the search system strategy. It is a valuable tool to find solutions for problem optimization. GA is used to solve problems that do not have a well-defined efficient solution[19]. This approach is used to solve optimization problems such as scheduling, shortest path, modelling and simulation where randomness function is used[20]. Evolution generally starts with a community of randomized individuals and it is an interactive process with the population being viewed as a method of generation for each reproduction. For every generation, the fitness of everyone in the population is measured. When sufficiently fit individuals are probabilistically chosen from the existing population, and the gene is modified to create a new generation circle for all recombined and potentially mutated at random. A newer generation of candidate strategies would be utilized over the next generation of the process. The algorithm usually ends when either a maximum number of generations or satisfaction has been generated.

The following steps are used to obtain fitness:

- Consider population, P randomly

- Obtain the fitness of the population
- Repeated from the following steps until convergence.
- Choose any parent from the population individually.
- Generate a new population through the crossover process.
- Insert random genes in a new population to perform mutation.
- Obtain fitness for newly generated populations.

2.15 Related works

To increase the effectiveness and classification accuracy of SVM, two new methods were proposed: feature selection-score (FS–Score) and DWPSO-SVM, an enhanced particle swarm optimization algorithm[21].To select the features, the FS-Score approach was employed, and the parameters were optimized using the improved particle swarm optimization model with dynamic adjustment of inertia weight (DWPSO-SVM).

S/ix machine learning classifier techniques were used to extract a pattern from two datasets obtained from moodle learning platform to predict the performance of students in programming class course[22]. A binary categorization (pass/fail) was used to grade the first dataset and the second was graded in a three- level categorization (fail, good and excellent). The six machine learning classifiers were Logistic Regression, Naive Bayes, Support Vector Machine, Random Forest, Neural Network and Decision Tree. To choose the best forecasting features, the study uses every conceivable combination of the eleven features. On the two datasets, 24432 prediction models in total were analyzed. According to the results, Random Forest produced the greatest results on three-level grade categorization in terms of accuracy, precision, and recall, whereas Logistic Regression produced the best results on binary datasets.

The findings demonstrated that there is no one optimal prediction model and that the same classifiers on multiple features might produce very similar results. Limitation: all possible

feature combinations were used to select features instead of using features which can be computationally expensive and time-consuming.

A flexible predictive model was examined, in which a framework for predicting students' academic achievement was built directly from raw data[23]. Instead of using a feature selection process. The framework bases feature selection on model interpretability. The framework was tested with ANN and RF. Static features, dynamic features, category features, and continuous features were the four categories into which the features were divided. Course details and student demographics are among the static elements. The student click streams on various VLE pages are dynamic elements. The student's greatest level of education is the categorical feature. Students' clickstreams on the course page are continuous features. The framework included all four elements. The forecast was based on four forecast moments: the course's first quarter, midterm, third quarter, and two weeks prior to the last day of the course.

The experiment's findings demonstrated the predictive models' excellent 81 percent accuracy, 69 percent precision, and 57 percent recall.

To examine the connection between e-learning behaviour and learning performance, the Behaviour Classification based on E-learning Performance (BCEP) prediction framework was developed[24]. The Behaviour Classification based on the E-learning Performance (BCEP) prediction framework and the predictors in the traditional framework were compared for prediction performance. The number of login platforms, the number of participants in forum discussions, the homepage, page, subpage, glossary, resource, and other behavioural data are among the input variables. The Process-Behavior Classification (PBC) model serves as the foundation for the suggested BCEP prediction framework. The four steps of the BCEP are feature fusion, data cleaning, behaviour

catemgorization, and model training. Six machine learning techniques are included in the proposed model (BCEP): SVC(R), SVC(L), Naïve Bayes, K-NN(U), KNN(D), and softmax. The accuracy rate, F1-score, Kappa Coefficient, and prediction time were the evaluation criteria.

The results of the experiments demonstrate that the BCEP prediction framework-based learning performance predictor has a strong predictive impact and that the PBC model performs better in learning performance prediction than more conventional classifcation techniques.

A predictive model was developed for predicting student performance by utilizing log data from the Learning Management System (LMS)[25]. The model was built using four machine learning classifications: logistic regression, DT, RF, and SVM. When creating the model, they adhered to the data mining procedure, which included gathering datasets, preprocessing data (cleaning and transforming data), selecting features, classifying data, obtaining training data, and evaluating the model. A 10-fold cross-validation procedure and the evaluation metrics of recall, f-measure, accuracy, and precision were used to test the models. The study uses a dataset that was taken from the publicly accessible Kaggle website. Academic performance data and student engagement data were the two files that made up the data. The desired numerical data was converted into the nominal form using the discretization approach.

The study analyzed student data using statistical methods and machine learning algorithms. According to the results, RF is the best method for dividing student performance into three categories: high, medium, and low.

A thorough review of student academic performance prediction was carried out to determine the different Deep learning techniques for predicting student academic

performance using precise modelling and parameter measurements developed on reputable and publically accessible data sets[26]. Artificial Neural Network (ANN), Recurrent Neural Network (RNN), Convolutional. Neural Network (CNN), Multi-Layer Feed Forward Neural Network (MFFNN), and Long Short-Term Memory (LSTM) are among the deep neural networks that were taken into consideration in the study. Additionally, they provided a novel framework for the prediction task based on an enhanced LSTM algorithm. Adam and Nadam's algorithms are proposed as improvements that involve optimization to enhance learning and improve performance. A multi-layered neural network is trained by backpropagation to acquire the correct internal representation, which enables it to learn any random input-to-output mapping. The investigation and examination of research articles published in a number of reputable publications has led to the conclusion that either ANN or DNN is the most effective neural network design when it comes to predicting student performance. RNN is the deep neural network technique that is used the least.

In an attempt to predict students' performance in one of the undergraduate data structure courses at a large public research institute early in the course, a predictive model for student performance in the classrooms using student interaction with e-textbooks was conducted[27]. Two data mining techniques were used to achieve this goal. Regression analysis was utilized to predict the exam's final grade, and a classification method was employed to predict each student's performance, either good or bad. Support vector machines, Random Forest Classifier, Decision trees, K-Nearest Neighbor, and Logistic Regression were among the algorithms used for classification. Based on the confusion matrix, four distinct measures were used to evaluate the quality of the classification algorithm: accuracy, precision, recall, and F-measure. Random Forest Regression and Multiple Linear Regression were used for the Regression analysis. The quality of the

regression methods was evaluated using three distinct metrics: coefficient of determination (R^2), mean absolute percentage error, and root mean squared error (RMSE). The study employed the coefficient feature selection approach to identify the most crucial features for model construction.PE-total-time, PE-total attempts, PE-reset, PE-model, PE-exercise, and SS-total-time are some of the input variables.

The study's independent variable is the attribute "etest," and its predictors are the other features. For this set of data, the Random Forest classifier yielded the best overall classification result with an accuracy of 91.7%, and the Random Forest regression produced the best regression analysis with an R^2 of 0.977.

In a four-year computer technology program, a prediction model (CPM) was presented [28]. This model would provide students a predicted final CGPA based on their high school score average and second and third year grades. The Randon Forest technique was used on the dataset. Five hundred and twenty-five datasets in total were employed for the investigation. The dataset was split into training and testing subsets of seventy and thirty, respectively. To assess the accuracy of the model, the values were compared between the predictions and the real ones. Using eighty-two records, CPM was able to predict the final CGPA after the second and third years with an accuracy of 88.68 percent and 92.36 percent, respectively. One hundred and five recordings were used for the experiment's repetition. The second and third years experienced an increase in accuracy to 91.32 percent and 92.87 percent, respectively. The RF classification would improve with a larger dataset.

A model based on machine learning was presented to predict the final exam grades of undergraduate students[29]. Midterm test grades were used as primary data. The input variable is data-driven. Three parameters are used in the data-driven input variables:

departmental data, faculty data, and midterm test grades. The machine learning algorithms comprised the following: K-NN, LR, NN, SVM, RF, and NB. The prediction's outcome revealed a minimum accuracy of 69.9 percent for K-NN and a maximum accuracy between 70.75 percent and 74.6 percent for RF and NN respectively.

Dwarf Mongoose Optimization (DMO) and Quatum-Based Optimization (QBO) were combined to create a feature selection framework named DMOAQ, which benefits from both DMO and QBO's advantages[30]. To get beyond the search restrictions, Quatum-Based Optimization was used as a local search of the DMO, modifying the performance of the conventional Dwarf Mongoose Optimization. The well-known benchmarks and high-dimensional datasets, including traditional DMO, were used to test the study. The evaluation's findings demonstrated that DMOAQ far outperformed the original DMO's search capacity.

For feature selection applications, EGOA, a modified version of the grasshopper optimization algorithm,was designed[31]. To balance the exploitation and exploration mechanisms, the Grasshopper Optimization Algorithm (GOA) was utilized with a Gaussian strategy and elite opposition-based learning to boast its local and global search mechanisms. When the enhanced technique, EGOA, was tested using various numerical functions, it outperformed the original GOA.

Particle positions and velocities are updated for each iteration of an efficient feature selection approach based on a multi-objective PSO algorithm using feature ranks and particle rankings[32]. The performance of the enhanced PSO approach was evaluated using sixteen datasets; it outperformed eleven existing methods by a significant margin.

The Salp Swarm Algorithm (SSA) was used as a feature selection technique to improve the diagnosis of glaucoma[33]. When SSA was used with the Kernel-Extreme Learning Machine (KELM) classifier, the accuracy of the classification was much increased.

During the COVID-19 epidemic, an e-learning classification model was developed to determine the most relevant student attributes for prediction and to predict students' performance based on their satisfaction with e-learning[34].The model was constructed using machine learning classification models such as Decision Tree, Random Tree, Naïve Bayes, Random Forest, RepTree, Bagging, and K-Nearest Neighbor. Thirty-five features were present in the total dataset of one thousand. Employing all thirty-five features in the dataset, and then applying ten most significant features, the study's findings indicated that the prediction accuracy increased only when the ten most significant features were employed.

To build an e-learning environment model for a course, learners' e-learning experiences were reviewed, and log data from their interactions with learning activities was collected[35]. Interactions between students and other students, with materials, and with assessment activities were all included. The student information system was used to extract the data from the course registration forms and import it into the e-learning platform. For six weeks, or a semester, the students utilized the online learning platform, and the interaction logs of these activities were kept track of. These log data serve as both the study's data source and a representation of the student's online learning experiences.

Confirmatory Factor Analysis (CFA) was employed to determine whether e-learners' experiences and system components are related. Assessment, hypertext, video, content package, and instructional discussion (forum) were the components of the system. A total of sixty-two students made up the study's dataset. The CFA analysis's findings indicated

that there was a significant, moderately positive relationship between e-learning experiences and system components, with instructional discussions ranking highest and the content package ranking lowest. Discussion forums and the e-assessment component were also significant elements of e-learning.

A variety of publications on feature selection techniques were investigated for machine learning-based disease risk prediction employing high-dimensional patient genetic data[36]. The study reviewed feature selection techniques such as the filter, wrapper, embedded, and hybrid approaches.

The result of the review showed that there is no optimal feature selection. They conclude that while the filter method's speed and ease of use make it appropriate for high dimensions, it is only effective at identifying relevant features and is unable to remove redundant ones. They disregard feature interactions, which may result in the loss of important information. The wrapper approach handles redundant features and interactions to make up for the filter's shortcomings, but it comes at a significant computational cost because of the large dimensionality of the dataset. Given that the wrapper technique is not preferred for forecasting the illness risk in the patient's genetic high-dimensional data, they concluded that hybrid feature selection is a "middle ground." Several feature selection techniques are combined in a hybrid feature selection method to select features. Consider a hybrid filter-wrapper approach. The filter technique handles the reduction of features, and the wrapper approach processes the reduced data to provide a minimal feature subset that maximizes machine learning algorithm performance and has a possibility of accuracy.

Several researchers have used both supervised (classification) and unsupervised (clustering and association rule mining) educational data mining techniques to predict students' academic performance [37].

To forecast students' academic achievement, a deep neural network model was introduced [38]. This is the first time a deep neural network has been used to mine educational data and forecast student achievement.

The accuracy of the suggested model was 84.3%. Early intervention is made possible by the model's ability to forecast academic success and identify pupils who are more likely to fail.

A hybrid feature selection method named RnkHEU is designed to improve the accuracy of predicting students' academic performance[39]. The hybrid approach creates candidate feature sets for heuristic approaches using a combination of rank search and Naïve Bayes classifier. Heuristic search begins with the randomly generated feature subsets, which may produce inconsistent feature selection outcomes.

The ultimate result of feature selection was obtained by performing a heuristic search on the candidate set after employing rank search to produce subsets of ranking features and choosing the subset with the highest classifier accuracy as the candidate set.

The dataset size that was employed, which came from eight separate student data sources, ranges from one hundred and thirty-one to five thousand, eight hundred and twenty. Thirteen thousand, eight hundred and sixty-three is the entire dataset that was used.

Large and challenging datasets were handled by combining three well-known feature selection techniques[40]. By removing uncorrelated and inconsistent input values, the classifier's performance was improved by the use of three feature selection techniques: dominance-based rough set approach (DRSA), best first search (BFS), and correlation-based feature selection (CFS). The computational time complexity was another goal of the investigation. Pre-processing data, data reduction, feature selection, and data analysis

comprise the four stages of the methodology. The data cleaning step of pre-processing data involves structuring the data using randomization and normalization to create a uniform and objective dataset. A correlated features dataset is the result of applying CFS BFS to the standardized dataset during the data reduction step. The data reduction stage speeds up processing and conserves memory. Before the start of the data reduction process. A lot of subgroups would be formed if there were more than ten thousand occurrences. The CFS-BFS approach will then be applied to these subgroups in the same manner for reduction. The dataset's most optimal features are chosen during the data selection phase. Using SVM, decision analysis is conducted during the optimal data analysis phase. Next, the model's computational time and accuracy are assessed. NN, SVM, and Deep Learning (DL) are the classifiers that were evaluated. According to the results, NN had an accuracy rate of 82.1 percent, SVM 66.5 percent, and DL 49.9 percent. The model's building time using the chosen approach and the model's testing time on the testing dataset were used to calculate computational time. The outcome demonstrates that NN using the multilayer perception method was better suited for handling large datasets since it completed the analysis with less computational time and a high accuracy rate.

On two real-life education heterogeneous datasets, the effectiveness of the most popular machine learning classification models created by applying DT, NB, SVM, RF, and ANN were compared[41]. Predicting students' performance based on social, demographic, psychological, and family aspects served as the foundation for the comparison. One thousand and forty-four Portuguese datasets and four hundred and eighty xAPL datasets made up the two datasets. Due to the variability of the study's features, the two datasets' vastly different sizes, and how balanced or unbalanced datasets affected the models' performance, a comparison examination of the models proved challenging. Portuguese took into account factors like age and extracurricular activities, while xAPL took into

account factors like class percentage, announcements watched, discussion involvement, and satisfaction.

A hybrid machine learning framework consisting of three ensemble approaches and eight single machine learning classification algorithms was suggested to find the best predictive model for predicting students at risk[42]. To choose the best features from the dataset relevant to students' performance, filter-based and wrapper-based feature selection strategies were employed. For the filter-based approach, the framework employed the CFS subset Eval algorithm, and for the wrapper-based method, it used the classifier subset Eval algorithm. Comparing the experiment's outcome to the single classification algorithms, it was evident that all of the ensemble approaches employed were more accurate. The majority of the algorithms performed better when the combined approach of the filter-based and wrapper-based methods was used to select the best features. In total, 486 datasets were utilized in the investigation. The models' performance will be optimized using a larger dataset. F-measure and accuracy were the evaluation metrics.

A three-phase hybrid feature selection technique was created[43] to lower the computational cost by employing an enhanced integer PSO and correlation-guided clustering.

For high-dimensional datasets, a variable-size cooperative convolutional PSO was introduced[44]. It broke down a high-dimensional feature selection problem into several low-dimensional, computationally efficient sub-problems.

To increase the accuracy in predicting student academic performance, a genetic algorithm-based feature selection (GAFS) and four single classifiers (DT, NB, K-NN, and RF) classification framework were proposed[45]. The two stages of the system were feature selection and modelling. The experiment was run in two phases: single classifiers using feature selection techniques (GAFS) to decrease the dataset's dimensions and single

classifiers without GAFS using all features for classification. The study made use of 480 datasets from the Kaggle database. The classifiers' accuracy was assessed using four assessment metrics: F-measure, Precision, Recall, and Accuracy. The dataset was subjected to ten-fold cross-validation to speed up computation while maintaining accuracy. The outcome of the experiment showed that the use of GAFS with classifiers outperforms the use of classifiers with all features. RF is the most effective classifier. RF achieved 82.29 percent accuracy with GAFS. RF's accuracy was 79.79 percent without GAFS.

Six ensemble classifiers, including DT, Artificial Neural Network (ANN), RF, Voting, Bagging, and Boosting, were used in two experiments to analyze four hundred and eighty student performance data sets that included sixteen features that were obtained from the Kaggle repository[46]. The three primary categories of the dataset were demographic, academic, and behavioral. Training and testing data were validated using ten-fold cross-validation. The six metrics that were utilized to assess the performance of the six classifiers were Geometric Mean (G-mean), True Negative Rate (TNR), True Positive Rate (TPR), F1-Score, Precision, and Area under the curve (AUC). The confusion matrix served as the basis for calculating the evaluation measures. The dataset containing all sixteen features was subjected to the six ensemble classifiers in the initial experiments, and the six classifiers' performances were compared. The outcome demonstrated that RF outperformed the other classifiers. In the subsequent experiment, GA was used for feature selection, and the algorithms were re-executed using tenfold cross-validation and evaluation with evaluation metrics. RF outperformed all classifiers and improved top performances were seen in all classifiers.

Using machine learning algorithms[47], two predictive models were created to measure how learners' behaviours could affect their learning attainment in Massive Open Online

Courses (MOOCs). These models were students' evaluation grades and final students' performance models. Student performance on previous assessments of their interpersonal behaviour is taken into account for the first experiment. Feature selection is taken into account solely in the initial trial. The session homepage, sum-click forming, session resource, sum-click subpage, session quiz, sum-click quiz, and score evaluations were among the behavioural characteristics employed in the first experiment. In the second set of studies, only temporal features, demographic features, and behavioural aspects are utilized as input variables to calculate final exam marks. The previous assessment's grade was not considered. Regression analysis and machine learning models, including Random Forest (RF) and Multi-layer perception (MLP) with two hidden layers, were used for both experiments. The Generalized Linear Model, Gradient Boosting Machine (GBM), and Single Hidden Layer Neural Networks (Nnet) were used. R-squared (R^2) and Root Mean Square Error (RMSE) were the metrics used in regression analysis. The discrepancy between the expected and actual observations is measured by RMSE.

The predictive models perform better, as evidenced by the lowest RMSE value. The outcome demonstrates that RF, with a score of 8.131 for the student evaluation grade model, obtained the lowest RSME. With an average value of 0.086, GBM produced the best accuracy in terms of final students' performance.

A set of guidelines was developed using the Apriori Association Rule Algorithm to investigate the relationship between academic performance and student involvement[48]. In a mixed learning environment, the relationship between student's academic achievement and their degree of involvement was demonstrated using confidence and lift indicators. Nine engagement metrics were used, including time-related and student grades that were commonly associated. Other engagement indicators, including emotions, and variables

like age and gender that may have an impact on student involvement were not taken into account.

By employing data mining techniques on datasets gathered from the Learning Management System (LMS) and other systems inside the educational institution, a framework for predicting student performance was proposed[49]. Three predicted output factors were the student's academic profile, the analysis and prediction of academic performance, and the strategy planning based on reports and results.

Expectation Maximization (EM) clustering technique, non-hierarchical clustering method, and hierarchical clustering approach were recommended to be applied to the dataset to identify the student profile. To predict students' academic success. Data mining techniques like K-means clustering and hierarchical clustering algorithms were recommended to be applied to the datasets gathered from LMS (Moodle), student record systems, and student information systems (SIS) to predict students' academic success. It was recommended to apply data mining techniques including association, classification, and clustering for strategy development and decision making.

To select features for machine learning algorithms like RF, CART, CHAID, ANN, and SVM to predict Peak Particle Velocity (PPV), feature selection methods were used[50].

Two cancer datasets were classified using several algorithms including Bayes Network, NB, SVM, MLP, KNN, DT, and RF[51]. The features in the two breast cancer datasets, BCDW11 and WBCD32, varied in terms of quantity and type. Whereas WBCD32 had thirty-two features and five hundred and sixty-nine instances, BCDW11 had eleven features and six hundred and ninety-nine instances. Upon applying classification methods to these datasets, the Bayesian Network outperformed other classifiers, yielding the best

accuracy of 97.13 percent for BCDW11 with fewer featured data. For WBCD32 datasets with more featured data, SVM provided the best accuracy of 97.89 percent.

A predictive model was created in a study[52] to predict, early in the semester, the final grades of students registered in introductory courses. The researchers used WEKA software to test eleven machine learning algorithms in order to compare their effectiveness.

The correlation between first-year students' Grade Pant Average (GPA) at Qassim University in Saudi Arabia and a range of personal and academic characteristics, including parents' educational attainment, the amount of time they spend on cell phones, and their level of academic achievement[53]. Six machine learning techniques were used to measure the admission test. The six algorithms include Neural Network, K-Nearest Neighbour, Decision Trees, Random Forest, AdaBoost and Multi-Layer Perception. They also applied association rule to check if some rules demonstrated the existence of direct relationship between personal and educational factors and students' performance. Accuracy was used to measure each of the six algorithms' performance.

To enhance the multiple regression model through feature selection, the performance of the model constructed using an academic performance dataset from the Kaggle repository was examined and assessed[54]. GRE, SCORE, Tofel Score, University rating, SOP, and CGPA are among the features taken into consideration. Backward elimination and other feature selection techniques were used to build the model with multiple features by determining the statistical relationship between the features. The optimized model is then tested with a variety of machine learning classifiers, including Logistic Regression, K-NN, Kernel SVM, Naïve Bayes, Decision tree, and Random Forest. Classifier efficiency is evaluated using metrics including precision, recall, F-score, and accuracy. Without using backward elimination, RF produced the best accuracy of 0.91 and followed by accuracy of

0.74 for Kernel SVM and followed by NB with the accuracy of 0.70. The Kernel SVM achieved the best accuracy of 0.89 with backward elimination, followed by RF (0.88) and NB (0.80).

Using cut-point and feature discretization, PSO was able to search a wider range of gene expression datasets and selected fewer featumres while maintaining similar classification accuracy[55].

The Open University (OU) in the United Kingdom[56] used student data from a social science course to test six different types of machine learning classifiers, including DT, J48, JRIP, Gradient-Boosted Tree (GBT), Classification and Regression Trees (CART), and Naïve Bayes Classifier (NBC), to predict low student engagement in an e-learning system. Three categories of data make up the student data: learning behaviour (the total number of clicks on VLE activities throughout the first course assignment), performance (final results and assessment scores), and demographic (highest education level). Data plus, forming, glossary, oucolloborate, oucontent, resources, subpage, homepage, and URL were among the VLE activities. The study's entire dataset, which came from the July 2013 session, numbered three hundred and eighty-four. Based on a significant level of 0.05, statistical analyses such as Spearman's correlation using SPSS are used to determine the effect of these data on student engagement. The analysis's findings demonstrated a substantial correlation between the dependent variable (the total number of clicks on the VLE activity) and the assessment's final outcomes and ratings. As a result, the highest education level was left out, and the final outcomes, assessment score, and total clicks on VLE activities were used to determine how engaged the students were.

The result of the application of the machine learning models on the remaining data showed that J48, GBT, DT and JRIP performed better than the other models in terms of Accuracy, Kappa value and Recall.

Decision Tree was also built using a decision tree classifier on VLE activities. The result showed that students' clicks on the homepage, foruming, oucontent and subpage were the most important predictors of student engagement in the VLE course because these features appeared most frequently in the classification model. Lastly, using the study's findings as a basis, they created an engagement prediction system. The VLE portal, source data, the best Machine Learning classifiers (J48, DT, JRIP, and GBT) for model selection, and the Instructor Dashboard made up the engagement prediction system. An instructor can provide low-engagement students with intervention recommendations by using the Instructor Dashboard, a computer program that analyzes and provides useful data regarding student participation in VLE activities. The source data is the student's interaction with the VLE system to finish a course assessment, which is documented in the log file. The student database contains the student performance data.

To optimize logistic regression (classifier) accuracy, a comparison of filter and wrapper feature selection techniques was carried out[57]. Based on the classification accuracy, Bayesian Information Criteria (BIC), Akaike Information Criteria (AIC), Area Under Receiver Operator Curve (AUC), sensitivity, and specificity of logistic regression (classifier), various feature selection techniques were evaluated. Data generation for continuous features and a single binary dependent variable for various sample sizes are part of the simulation study. Information gain and correlation-based feature selection techniques were employed as filter techniques, and sequential forward and sequential backward elimination techniques were employed as wrapper techniques. Three genuine

datasets—Pima Indian Diabetes, Breast Cancer Wisconsin, and Spam—were subjected to these feature selection techniques. The simulation study's findings demonstrated how the kind of features affect the feature selection process. There were only continuous features in the three genuine datasets. When evaluating the effectiveness of various approaches, it was found that wrapper methods perform better for choosing accurate continuous features than filter methods.

Big data techniques are used to predict every student's performance across all classes to identify the high-risk individuals who could drop out, perform below expectations, or fail a course[58]. To test simple classifiers like Decision Trees and Linear SVM and sophisticated classifiers like RF and Gradient Boosting (GB), features were taken from past grade data. Since RF required less training time to reach comparable performance to GB, they were chosen. The classifiers were assessed using five-fold cross-validation. Four partitions were utilized to train the classifiers, and one partition was used for testing using evaluation measures such as precision, recall, F1 score, ROC, and AUC for each fold. Based on the F1 score and AUC, the experiment's results indicate that GB is the top classifier, closely followed by the RF classifier. DT performs the worst. The students at risk were substantially imbalanced, with a much lower F1 score, whereas the A-students were mostly accurately predicted. The original dataset, which spans more than thirteen years, was acquired from the University of Minnesota; however, the total number of datasets was not disclosed. The learning management system's features have been used to estimate the results as fail, pass, good, and exceptional.

To provide a qualitative assessment of the student's performance, binary classification was used. Feature selection technique based on a basic Particle Swarm Optimization was applied to decrease the search space and increase search effectiveness[59].

2.16 The Summary of Gaps

The e-learning classification model developed to determine the most relevant student attributes for prediction and to predict students' performance based on their satisfaction with e-learning, the reviewed literature of the study revealed that there was a gap in using all features totally thirty-five (35) in the dataset. Better prediction accuracy was achieved only when the ten most significant features were applied in the dataset compared to application of all the thirty- five. features of the dataset.

With a total dataset of 62, confirmatory Factor Analysis (CFA) was to find the relationship between e-learners' experiences of learners and system components. The limitation of the study was the size of the dataset, so cannot be generalized.

The use of two heterogeneous datasets that were unbalanced affected the performance of the models built by the application of DT, NB, SVM, RF and ANN. The use of the unbalanced datasets was the gap.

A proposed framework for predicting student performance by applying data mining techniques on a dataset. Data mining techniques like K-means clustering and hierarchical clustering algorithms were recommended to be applied to the datasets gathered from LMS (Moodle), student record systems, and student information systems (SIS) to predict students' academic success. But the gap here is that the study is yet to be implemented or carried out.

The model based on machine learning to predict the final exam grades of undergraduate students was data driven utilising only three parameters- faculty data, departmental data and midterm exam grades. This study can be improved by including other parameters such as demographic variables and other machine learning algorithms to the modelling process.

The study of a predictive model for student performance in classrooms can be broadened to incorporate additional distinguishing features to acquire more accurate data that can be utilized to improve student learning outcomes. Various data mining techniques such as clustering and association can be used to compare and analyze them.

The correlation between first-year students' Grade Pant Average (GPA) at Qassim University in Saudi Arabia and a range of personal and academic characteristics was evaluated by accuracy only to measure model performance, but accuracy alone can give misleading results. Accuracy and F-measures to evaluate performance will give better results. The primary study focus was parental education and personal traits, which are insufficient to influence academic performance.

Endnotes

[1]N. K. Yadav & S. S. Deshmukh, "*Prediction of Student Performance using Machine Learning Techniques: A review*," Proceedings of the International Conference on Applications of Machine Intelligence and Data Analytics (ICAMIDA 2022), ACSR 105, May, 2023: 735-741

[2]L. Chen, P. W. Gamage & J. Ryan," *Debias Random Forest Regression Predictors*," **Journal of Statistical Research** 56, no. 2, 2022: 115 – 131,

[3]S. Udding, A. Khan, M. E. Hossain & M. Ali, "*Comparing different supervised machine learning algorithms for disease prediction*," **BMC Medical Informatics and Decision Making** 19, no. 281, 2019.

[4]C. Bentejac, A. Csorgo & G. Martinez – Munoz," *A comparative Analysis of XGBoost*," **Artificial Intelligence Review** 54, 2020: 1937 – 1967

[5]S. P. Dwaraka, L. A Vijaya, D. T. David, S. Aditya, & S. G. Thippanna, "*A Forest of Possibilities: Decision Trees and Beyond*," **Journal of Advancement in Parallel Computing** 6, no. 3 2023: 29 – 37.

[6]J. Cervantes, F. Garcia- Lamont, L. Rodriguez & A. Lopez-Chau," *A comprehensive survey on support vector machine classification: Applications, challenges and trends*," **Neurocomputing** 408, 30 september, 2020: 189 – 215.

[7]A.A.G.S,Danasingh, A.a.B Subramanian, & J.L. Epiphany, "*Identifying Redundant Features Using Unsupervised Learning for High Dimensional Data*,". **SN. Appl. Sci**. 2, 1367 2020. .

[8]B. Mustafa, & C. O. Mehmet, "*A Comprehensive Review of Feature Selection and Feature Selection Stability in Machine Learning*," **Gazi University,Journal of Science. GUJ Sci**. 36, no.4 ,2023:1506 – 1520.

[9]N. Narisetty, S. R. Ami, S, Rao, & C.R Rao, "*Principle and Methods for Data Science*," First Edition, 43,(2020) Handbook ISBN: 1780444642110 eBook ISBN: 9780444642127.

[10]P.Manikandaprablu,"*Feature Selection Methods: A study*," **compliance Engineering Journal**.12 no.5 2021.

[11]B Remeseiro, & V Bolon-Canedo,"*A Review of Feature Selection Methods in Applications*," **Comput. Biol. Med**. 2019.

[12]M. Ghosh, R. Guha, R. Sarkar & A. Abraham, "*A wrapper-filter feature selection technique based on Ant Colony Organization*," **Neuro comput. Applic** 32. 2020: 7839-7857.

[13]Y Guo, F Chung, G Li, & L Zhang, "*Multi label Bioinformatics Data classification with Ensemble Embedded Feature selection*," **IEEE Access** 7 2019.

[14]B. Pes," *Ensemble Feature Selection for High-Dimensional Data: a Stability Analysis across Multiple Domains*," 32 2020:5951- 5973 **Neural Computing and Applications** 32 2020:5951- 5973.

[15]C,,F, Tsai, & Y., T. Sung,"*Ensemble Feature Selection in High Dimension, Low Size Sample Datasets:Paralleland Serial Combination Approaches*," **Knowledge- Based System** 203, 106097.

[16]D. Archibald & S. Worsley,"*The Father of Distance Learning*"**Association for Educational Communications and Technology(AECT)** 63 pp.100-101, 2019.

[17]A. Pregowska, K. Maszlalerz, M. Garlinska & M. Osial, "*A Worldwide Journey through Distance Education – From the Post Office to Virtual, Augmented and Mixed Realities, and Education during the COVID- 19 Pandemic*" **Education Science** 11, 118, 2021.

[18]S. P. Barus, "*Implementation of Naïve Bayes Classifier-based Machine Learning to Predict and Classify New Students at Matana University*," **Journal of Physics Conference Series** 2021: 1842(1)012008.

[19]A. Tanweer, Q. Shamimul, D. Amit & B. Mohamed, "*Genetic Algorithm: Reviews, Implementations and Applications*," **International Journal of Engineering Pedagogy (IJE.P***)* 2020.

[20]S. MirJalili, J. S Dong, A. S. Sadiq & H. Faris, "*Genetic Algorithm: Theory, Literature Review, and Application in Image Reconstruction: Methods and Application studies in Computational Intelligence*," **In Nature- inspired Optimizers** Jan. 2020: 69 – 85.

[21]J. Wang. X. Wang. X. Li & J. Yi, "*A Hybrid Particle Swarm Optimization Algorithm with Dynamic Adjustments of Inertia Weight Based on a new Feature Selection Method to Optimize SVM Parameters*," **Entropy** 25, no. 531 2023.

[22]Ivan Peraic & Ani Grubisic, " *Predicting Academic of Students in a Computer Programming Course Using Data Mining*,"**International Journal of Engineering Education**, July, 2023.

[23]A. Al-Zawqari; D. Peumans & G. Vandersteen, "*A flexible feature selection approach for predicting the students' academic performance in online courses*," **Computers and Education: Artificial Intelligence** 3 2022 100103.

[24]F. Qiu, G. Zhang, X. Sheng, Leiiang, L. Zhu, Q.Ziang, B. Jiang & P. Chen," *Predicting, students' performance in e-learning using learning process and behaviour data*," **scientific reports** 2022.

[25]J. Usama, M. Shahid, R. Saba, W. Muhammad, A. Sabar, Z. Muhammad & J. I. Muhammad, "*Student Performance Prediction in E-learning Environment Using Machine Learning*," Jilin, Daxue Xuebao (GongXueban)/**Journal of Jilin University (Engineering and Technology Edition**. 41, no.12 2022: ISSN: 1671

[26]E. Ismanto, H. AbGhani, N. Izrin M. Saleh, J. Al Amien, & R. Gunawan. "*Recent Systematic review on student performance prediction using back propagation algorithms*,"

Telecommunication computing Electronics and control. ISSN 1693-6930 e-ISSN2302-9293 20, no. 3 June 2022:597-606,

[27]A. Abd Elrahman, T. H. A. Soliman, A. I. Taloba, & M. F. Farghally, "*A predictive Model for Student Performance in classrooms using student interactions with an e Textbook*," *2022* 10th International Japan- Africa Conference Research Square: Inf. Sci.Let*t*. 12, no. 1 2023:9-22.

[28]N. Mirna, & A. N. Mahmoud, "*Predicting Student Performance to Improve Academic Advising Using the Random Forest Algorithm*," **International Journal of Distance Education Technologies** 20, no.1 Jan 2022.

[29]M. Yagci, "*Educational Data Mining: Prediction of students' academic performance using machine learning algorithms*," **smart learning environments** 9, no.11 2022s.

[30]M. Abd Elaziz. A. A. Ewees, M. A. A. Al-qaness, S. Alshathri & R. A. Ibrahim," *Feature Selection for High Dimensional Datasets Based on Quantum-Based Dwarf Mongoose Optimization*," **Mathematics** 10, no. 23 2022,4565.

[31]Z Xu, A A Heidari, F Kuang, A Khalil, M Mafarja, S Zhang, H Chen, & Z Pan, "*Enhanced Gaussian Bare-Bones Grasshopper Optimization: Mitigating the Performance Concerns for Feature Selection*,". **Expert Systems with Applications** 212, February **2022.**, 118642.

[32]A. Rashno, M. Shafipour, & S. Fadaei, " *Particle ranking: An Efficient Method for Multi Objective Particle Swarm Optimization Feature Selection*," **Knowl.-Based Syst**. 245, 7 June **2022**, 108640.

[33]K Balasubramanian, & N Ananthamoorthy, "*Correlation-based feature selection using bio-inspired algorithms and optimized KELM classifier for glaucoma diagnosis,*" **Appl. Soft Comput**. 128,2022,109432.

[34]I. L, H. Alsammak, A. H. Mohammed, & I. S. Nasir," *E-learning and COVID-19: Predicting Student Academic Performance using Data Mining Algorithms*," **Webology** 19, no. 1 January2022: 3419-3432.

[35]S. Keskin, & H. Yurdugul, "*E-learning Experience: Modeling students' e-learning interactions using log data*," **Journal of Educational Technology& Online Learning** 5, no.1 2022:1- 13

[36]N. Pudjihartono, T. Fadason, A. W. Kempa –Liehr & J. M. O'Sullivan, "*A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction,"* **Frontiers in Bioinformatics** 27 June, 2022.

[37]S.Verma, R.R. Yadav, & K. Khaliya. "*Prediction of Academic Performance of Engineering Students by using Data Mining Techniques*," **International Journal of Information and Education Technology**, vol. 12, No. 11, November,2022.

[38]M.G. Rao & K.K. Kumar, " *Students Performance Prediction on Online Courses using Machine Learnlng Algorithms*," **United International Journal For Research and Technology**, volume 02, issue 11, ISSN :2582-6832, 2021.

[39]W. Xiao, P. Ji & J. Hu, "*RnkHEU: A Hybrid Feature Selection Method for Predicting Students' Performance*," **Hindawi Scientific Programming** 2021, no.1 2021: 1670593.

[40]M. Mohamad, A. Selamat, O. Krejcar, R. G. Crespo, E. Herrera – Viedma & H. Fujita, "*Enhancing Big Data Feature selection using a hybrid correlation based feature selection,*" **ELECTRONICS** 10, no. 23 2021:2984**.**

[41]I. Leila, M. Huned, & H. Alain, "*Comparative Analysis of Machine Learning Models for students' performance prediction*," International Conference on Advances in Digital Science 1352 2021: 149-160.

[42]E. D. Evangelista, "*A hybrid Machine Learning Framework for predicting students' performance in virtual learning environment,*" **International Journal of Emerging Technologies in Learning** *(IJET)* 16, no. 24 2021: 252- 272,

[43]X. F. Song, Y. Zhang, D. W. Gong, & X. Z. Gao, "*A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high dimensional Data*," **IEEE Transactions on Cybernetics** 99, 2021: 1–14.

[44]X.F. Song, Y. Zhang, Y.N. Guo, X.Y. Sun, & Y.L. Wang, "*Variable-size cooperative convolutionary particle swarm optimization for feature selection on high-dimensional data*," **IEEE. Transactions on Evolutionary Computation** 24, no. 5 2020: 882–895.

[45]Al Farissi, H. M. Dahlan, & Samsuryadi, "*Genetic Algorithm Based Feature Selection for Predicting Student's Academic Performance,*" s Conference paper 2019: 110-117.

[46]Al Farissi, & H. M. Dahlan, "*Genetic Algorithm Based Feature selection with Ensemble Methods for Student Academic Performance Prediction*". **Journal of Physics: Conference Series**, 2020: 1500 012110.

[47]R. Al-shabandor, A. Hussain, R. Keight & W. Khan," *Students performance prediction in Online Course Using Machine Learning Algorithms*" 2020 International Joint Conference on Neural Networks (IJCNN), 19 July 2020- 24 July 2020, Glasgow, UK

[48]A. Moubayed, M. N. Injadat, A. Shami, & H. Lutfiyya," *Relationship between Student Engagement and Performance in e-learning Environment using Association Rules*," **IEEE World Engineering** 1 March 2018. Corpus ID:52149154

[49]M. A. Job & J. Pandey,"*Academic Performance Analysis Framework for Higher Education by Applying Data Mining Techniques*," 1 June 2020. 2020 8th International Conference on Reliability, infocom Technologies and optimization (Trends and Future Directions). Corpus 1D: 204782835.

[50]H. Zhang, J. Zhou, D. J. Armaghani, M.M, Tahir, B. T. Pham, & V. Van Huynh, "*A Combination of Feature and Random Forest Techniques to Solve a Problem Related to Blast-induced Ground Vibration*," **Applied Science** 10, no.3 27 January 2020.

[51]A Kumar, R Sushi, & A K Tiwari, "*Comparative study of Classification Techniques for Breast Cancer Diagnosis*," **JCSE International Journal of Computer Science and Engineering** 7, no.1 2019.

[52] I. Khan, A. Al Sadiri, A.R. Ahmad & N.Jabeur, "*Tracking Student Performance in IntroductorybProgramming by Means of Machine Learning*," MEC International Conference on Big Data and Smart City (ICBDSC), Muscat, Oman, pp.1-6, 2019

[53]Ayshal, Mohammed, Hessah, & Meznah, "*Performance evaluation and Comparison of Classification Algorithms for Students at Qassim University, Kingdom of Saudi Arabia (KSA)*," **International Journal of Science and research** 8, no 11 2019: ISSN: 2319-7064.

[54]R. Suguna, S. Devi, R. A. Bagate, & A. S. Joshi, "*Assessment of feature selection for student academic performance through machine learning classification*," **Journal of statistics and management systems** 22 no. 4 25 Jun 2019: 729-739, ISSN:0972-0510.

[55]X. Huang, Y. Chi, & Y. Zhou, "*Feature Selection of High Dimensional Data by Adaptive Potential Particle Swarm Optimization*," in Proceedings of the IEEE Congress on Evolutionary Computation *(CEC)* June 2019: 1052–1059 Wellington, New Zealand.

[56]H. Mushtaq, Z. Wenhao, Z. Wu, & M. R. A. Syed, "*Student Engagement Predictions in an e-Learning System and Their Impact on Student Course Assessment Scores*," **Hindawi Computational Intelligence and Neuroscience** 2018, no. 6347186 2018: 21 pages.

[57]B. W. Yap, N. Ibraham, S. Abdul-Rahman, S. Fong & H. A. Hamid, "*Feature selection methods: case of filter and wrapper approaches for maximising classification accuracy*," **Pertanika Journal of science and Technology** 26, no. 1 January, 2018: 329 – 340.

[58]A. Polyzou & G Karypis, "*Feature extraction for classifying students based on their academic performance*," proceedtngs of the 11th International Conference on Educational Data Mining .International Educational Data Mining Society, 1 July 2018, Corpus ID: 52172242

[59]B. Tran, B. Xue, & M. Zhang, "*A new representation in Particle Swarm Optimization for discretization-based feature selection*," I**EEE Transactions on Cybernetics** 48, no. 6 2018: 1733–1746.

Chapter Three

Methodology

3.1 Research Approach

The research approach for this study is introduced in this chapter. It explains the research methodology and design that were employed in this investigation. The study describes the locations and different techniques for gathering data, in addition to the system architecture and experimental configuration.

The purpose of this study is to investigate the impact of feature selection techniques on the accuracy and effectiveness of student performance prediction on machine learning models. It is expected that feature selection will play a crucial role in improving model performance by identifying and utilizing the most relevant and correlated features from educational datasets. To achieve this aim, different feature selection methods were evaluated to know how they can influence the prediction accuracy of machine learning models and to understand how useful they can be for educational purposes. Student performance prediction's complex nature often originates from datasets with high dimensionality demands a sophisticated feature selection strategy. This study offers a unique model and optimization technique, named PF-PSO comprised of a combination of three existing feature selection techniques such as Principal Component Analysis (PCA), Forward Feature Selection (FOR) and Particle Swarm Optimization (PSO) that is used to enhance the feature selection process and for lowering model complexity and strengthening the models' capacity to predict student success. The proposed PF-PSO model addresses this challenge by combining dimensionality reduction, iterative feature evaluation, and metaheuristic optimization. This hybrid approach aims to pinpoint the most predictive features, potentially enhancing model performance and offering a deeper understanding of student outcomes. The PSO algorithm optimises the feature selection by

updating particle positions and velocities, evaluating fitness, and updating personal best and global best positions. The final selected features are based on the global best position achieved by the PSO algorithm, by concentrating on the most pertinent features. Forward Selection is a useful and simple feature selection technique that can enhance model performance and lessen overfitting. To prevent overfitting, it might not take feature interactions into account and necessitate carefully adjusting the stopping criterion. We combined all of these treatments' advantages with PF-PSO's limitations and carried out the cornerstone of our work by combining the power of PCA, FOR and PSO.

In approaching this study, we thought about all possible combinations and the order of such combinations. We considered several ways to combine Principal Component Analysis (PCA), Forward Selection Method (FOR), and Particle Swarm Optimization (PSO).

Firstly, we could do PCA -> FOR -> PSO. This approach will reduce the dimensionality of the data by first using PCA, making it easier for FOR to select a smaller set of relevant features. Then, PSO is used to further refine the feature selection. This method can be computationally efficient, as PCA reduces the search space for FOR and PSO. It also helps remove noisy or irrelevant features early in the process.

Secondly, we could discard FOR and do PCA -> PSO. Similar to the first option, we could start with PCA for dimensionality reduction and then apply PSO directly to the reduced feature set, potentially leading to a different selection of features compared to FOR. While this approach avoids the computational overhead of FOR, it might be less effective at selecting features if the initial PCA does not capture all the relevant information.

Thirdly, we considered FOR -> PCA -> PSO. In this approach, we use FOR on the original features to get a subset. Then, PCA is applied to this subset, and PSO is used to further refine the selection from the principal components. While this method prioritizes the iterative selection of features with high predictive power by FOR before applying dimensionality reduction, it would be more computationally intensive.

Lastly, we considered PSO -> FOR. In this case, we started with PSO to find an initial set of features, and then we use FOR to further refine the selection. This approach prioritizes exploring a wide range of feature combinations using PSO before focusing on the most relevant features with FOR. However, it would be computationally expensive if the original dataset is large. We decided to choose the first option (PCA -> FOR -> PSO) because of its efficiency. PCA significantly reduces the dimensionality of the data, potentially making FOR and PSO more efficient. This is crucial since our dataset has a large number of features.

We can expect significant noise reduction since PCA helps to remove noise and irrelevant features, ensuring that subsequent feature selection techniques (FOR and PSO) focus on the most relevant features. We believe this approach offers a balanced strategy between exploration and exploitation. PCA does a broad exploration, FOR does a systematic selection and it is a more directed method for exploring feature space and locating features that enhance model performance and PSO does a fine-grained optimization by exploiting several possible bests.

However, we recognise that we need to carefully consider the number of components to retain in PCA to ensure we are not losing too much information. Also, we need to choose an appropriate number of features to select in FOR.

By combining these techniques, we achieved a more comprehensive feature selection process and improved our model's performance. However, we anticipated that the benefits and synergies would greatly exceed the drawbacks, providing a net positive outcome generally superior to other approaches. For example, PSO is known to be a computationally expensive method and often requires a limited search space to save computing time and expenses. Nonetheless, we are positive that the search area accessible to the final PSO phase is not only appropriately reduced but also a very good and nearly optimal space, given that it was preceded by the potent approaches of PCA and FOR. Additionally, it facilitates the optimization of hyperparameters like cognitive, and social factors, inertia weight, and swarm size. However, by identifying a subset of fundamental contributing features, the ensuing Forward Selection processes could help recover clarity, though PCA alone might

requires a limited search space to save computing time and expenses. Nonetheless, we are positive that the search area accessible to the final PSO phase is not only appropriately reduced but also a very good and nearly optimal space, given that it was preceded by the potent approaches of PCA and FOR. Additionally, it facilitates the optimization of hyperparameters like cognitive, and social factors, inertia weight, and swarm size. However, by identifying a subset of fundamental contributing features, the ensuing Forward Selection processes could help recover clarity, though PCA alone might at times diminish direct interpretability. On the other hand, we acknowledge that overfitting, a situation in which the model's ability to generalize to new data is compromised by the features chosen might be a significant drawback to watch out for. An excessive number of features may also cause overfitting.

Figure 3.1 below is the Architectural Workflow of the model. It comprises the following steps: 1. Pre-process the dataset 2. Analyze and visualize the data. 3. Split data into a training set and testing set. 4. Run machine learning models without feature selection. 5. Evaluate the models. 6. Run the machine learning models with feature selection using feature selection methods like the Pearson Correlation Coefficient (PCC), the Principal Component Analysis (PCA), the Forward Feature Selection (FOR), the Particle Swarm Optimization (PSO) separately 7. Select features with proposed, PF- PSO model. 8. Train the machine learning models with the features selected by PF-PSO. 9. Evaluate the models. 10. Compare results with feature selection methods and without feature selection methods.

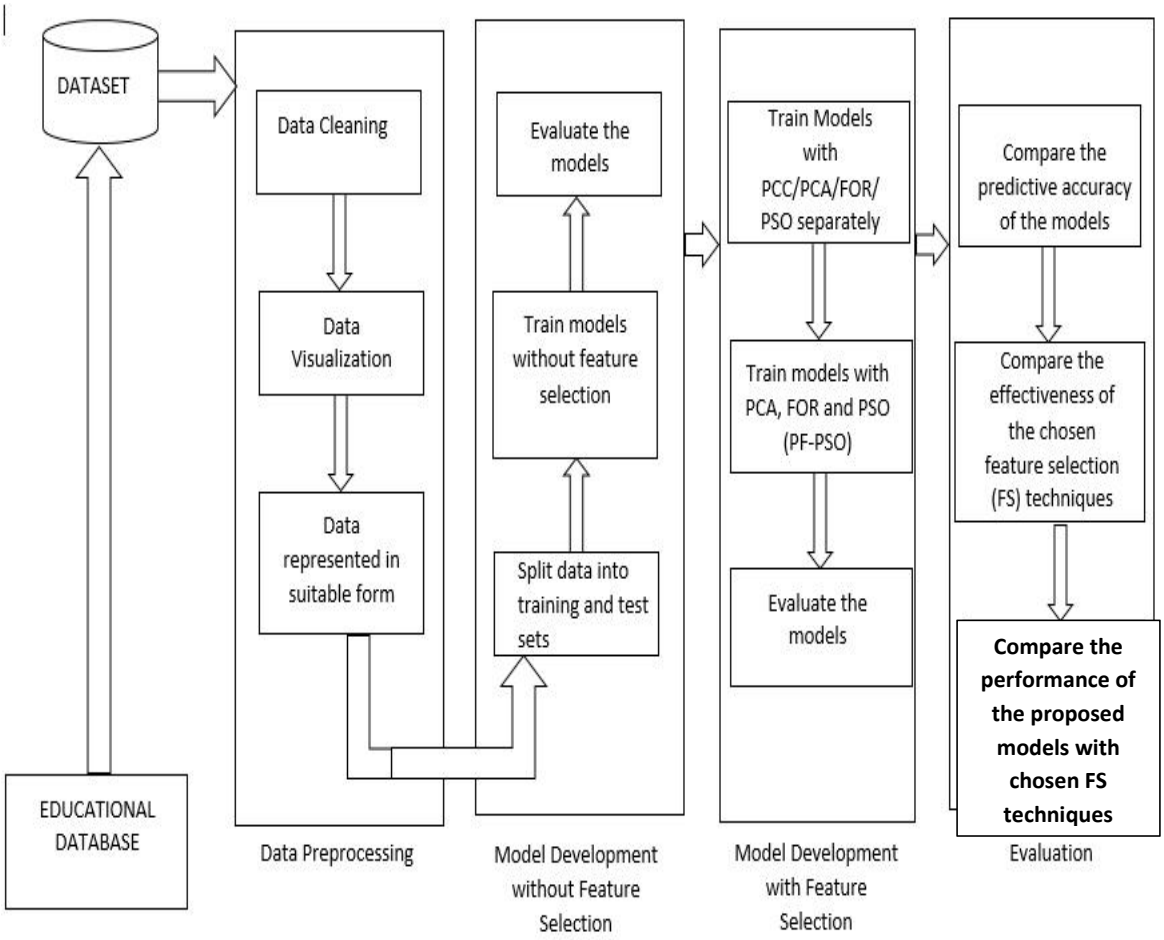


Figure 3.1: Architectural Workflow (Source: Researcher, Adelodun F.O. 2024)

3.2 Research Design

This research is a quantitative study. It employs an experimental and comparative design. The experimental aspect lies in evaluating the impact of different feature selection techniques on machine learning model performance. To properly understand the impacts of our tests, we conducted our experiments in a comparative and iterative manner allowing us to step through and repeat processes to understand the test results. We repeated these processes for each of the selected datasets and compare the results across the datasets. The comparative aspect involves comparing the prediction accuracy of models built with and without feature selection and the effectiveness of five chosen feature selection techniques. The five feature selection techniques that were utilized include Pearson Correlation Coefficient (PCC), Principal Component Analysis (PCA), Forward Selection method

(FOR), Particle Swarm Optimization (PSO) and a novel approach of combining PCA, FOR and PSO referred to as PF-PSOl. As we approach our target PF-PSO, we believe it is important to perform separate and combined approaches of the elements of our proposed model to serve as comparators. Considering that we have three methods in our proposed model (Principal Component Analysis, Forward Selection, and Particle Swarm Optimization), we have six (6) possible sets of singular methods namely No Feature Selection, PCC, PCA, FOR, PSO and lastly, we have our proposed PF-PSO method. Consequently, we have a total of six (6) feature selection approaches in our experimentation. We repeated these six approaches for all three datasets resulting in a total of eighteen (18) experiment sets. The table 3.1 below shows how the eighteen experiments were carried out. Feature selection technique is the focus of this research and forms the pivot point of our methodology. The first method was to run the dataset on our prediction models without feature selection.

This would form the control set against which we can compare results and rationalize. We trained and evaluated the models using the dataset's entire original feature set, potentially including irrelevant, redundant, or noisy features. We carefully tracked the model's performance using the evaluation metrics discussed in 3.14. Using this method as our starting point acts as a control set, allowing us to quantify any improvement or potential degradation in model performance. The second step was to apply feature selection techniques to select features to train the models. The first feature selection technique to be applied is a simple Pearson Correlation Coefficient (PCC) analysis to select top correlated features. Then, we run the model with the selected features. This serves as one of the comparator against our proposed feature selection model. Furthermore, examining this baseline model may offer vital insights into which original features are the most significant predictors of student success, even in the event that feature selection does not

result in appreciable performance gains. For example, we expect in cases where model performance metrics remain the same with or without feature selection, selecting features may result in more robust and generalizable models that can more easily generalize to new data. This approach can help assess the overall computational expense of the chosen model when working with all original features, probably highlighting the potential need for dimensionality reduction through feature selection. It is important to note that datasets with a large number of features might increase the risk of overfitting, with this baseline method. Some machine learning algorithms also have embedded feature selection mechanisms, which is crucial to consider when interpreting results in no-feature-selection approaches. We believe some researchers have overlooked this fact in their works.[1] The next step after PCC feature selection is to use the Principal Component Analysis (PCA) technique to the dataset to reduce the dimensionality. PCA facilitates dimensionality reduction by projecting the data onto a lower-dimensional space composed of orthogonal principal components. These components are ranked by the amount of variance they explain in the original data by judiciously selecting the principal components responsible for the majority of variance. Each feature selection technique is applied to each dataset separately, the feature techniques such as PCC, PCA, FOR, and PSO after which the proposed system, PF- PSO comprising of PCA, FOR and PSO is applied to the datasets.

Table 3.1 Experiment Sets of the Study

	Dataset 1	Dataset 2	Dataset 3
No FS	1	7	13
PCC	2	8	14
PCA	3	9	15
FOR	4	10	16
PSO	5	11	17
PF- PSO	6	12	18

Source: Researcher, Adelodun F. O 2024

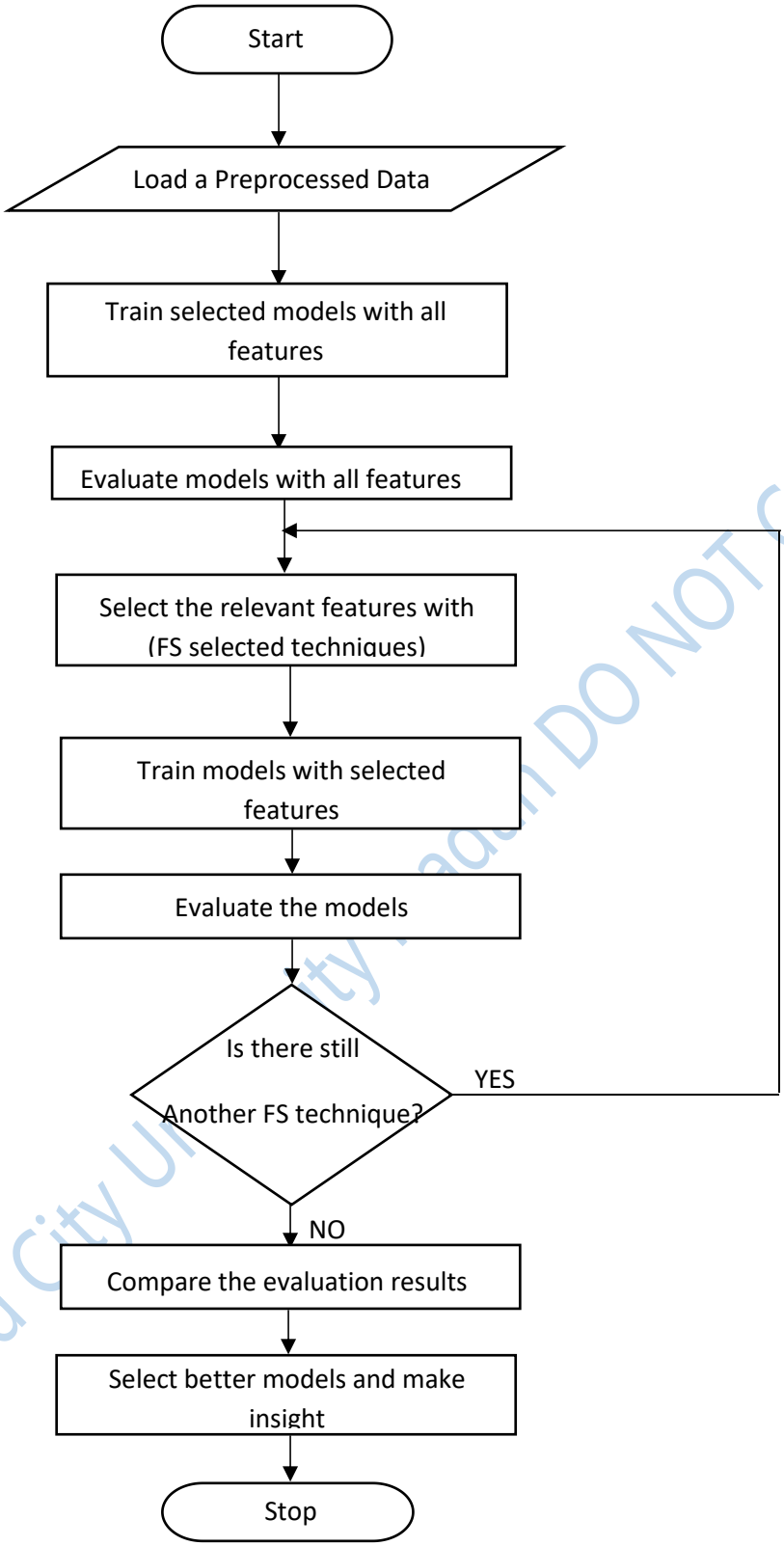


Figure 3.2: The Flowchat of the Proposed Model (source: Researcher, Adelodun, F. O. 2024)

3.3 The Experimental Procedures

This is a brief description of the total of the eighteen (18) experiments performed on the three educational datasets refered to as Dataset 1,Dataset 2, and Dataset 3. Regression tasks were carried out on Dataset 1 and Dataset 3. Nineteen (19), regression models were run into the datasets. The regression models included Linear regression, DT, RF, SVR (RBF), K-NN, MLP, GradientBoosting, XGBoost, LightGBM, Extra Trees, AdaBoost, SVR(linear), SVR (Poly), Gaussian Process, K-NN(tuned), Bayesian Ridge, Ridge Regression, Lasso Regression and ElasticNet Regression. For Dataset 2 experimentation we deviated from regression models to classification models. we performed multi-classification using nine (9) classification models which comprises of the following: logistic regression (Multinominal), K-Nearest Neighbours (K-NN), Support Vector Machine, Random Forest, Gradient Boosting, XGBoosting, LightGBM, MLP classifier (Neural Network), and Naïve Bayes. The experiments were arranged according to Table 3.1 For example, experiments 1, 7 and 13 were carried out without feature selection method. Experiments 1,and 13 of Dataset 1 and Dataset 3 respectively were ran into the 19 regression models respectively without feature selection, likewise experiment 7 of Dataset 2 were ran into the 9 classification models without feature selection. The proposed system (PF-PSO) were experimented as experiments 6, 12, and 18.

PCC selector were ran on expreiments 2, 8, and 14. For experimentt 2, PCC selector selected 8 features as top correlated features with a threshold greater than 0.6. For experimentt 8 at the selection threshold values of 0.1 and 0.5. A threshold of 0.1 selected more features and had a better model performance metric than PCC threshold of 0.5. Feature selection reduced as we increased the threshold. PCC threshold at 0.1; selected 14 features. PCC threshold at 0.5, the selected features were 4 features.For experiment 14,

when we ran the PCC selector with a selection threshold greater than 0.6, we selected 5 features.

Applying PCA in experiments 3, 9, and 15: we selected features with a 0.95 value for n-components, unfortunately, we cannot print selected features for PCA like PCC do. Experiment 3 selected 10 features. Experiment 9 selected 4 features and experiment 15 selected 29 features.Fitting the models on the PCA transformed dataset, we got results shown in chapter four

Experimentation for Forward Selection for experiments 4, 10 and 16 shows the following: Experiment 4: we chose "Random Forest Regressor" and "Decision Tree Regressor" as our FOR estimators. "Gird Search" was used to perform an hyperparameter search on "n feature to select" from 3 to 23 to find the optimal value.

Experiment 10: An ensemble of logistic Regression and Randomforest classifier were used as estimators for our sequential feature selectiom and we passed the selected feature into all the models nine (9) features were selected

Experiment 16: Random Forest as our estimator, we used GridSearch to perform an hyperparameter search on n features to select" we select 30 features and used to train the models

PSO: (Experimentts 5, 11 and 17): For the PSO setup, we ensured that the dimentions of the search space matched the total number of features in our dataset. we initialized the total particles to search this space we also set the cognitive, social and inertia parameters (C1, C2, W1) to 0.5, 0.3, and 0.9 respectively, as a balanced starting point. The PSO searches, and extracts the features that are consistently selected in the best solutions. The features are inputed in the regression models and classification models as applicable.

For PF-PSO: Experiments 6, 12, 18: We ran the PCA on the datasets, afterwards ran FOR on datasets and finally, we fed the features selected by PCA and FOR into PSO which selected features at best cost. We then trained our models on this final selection.

3.4 Data Acquisition

A total of three datasets were used in this research and they were sourced from public repositories. The first step in data collection was to scrape the internet using Google's Dataset Search engine for datasets with the following keywords and phrases: student, virtual, academic, performance, scores, student academic performance dataset, student performance dataset, student academic performance prediction, student performance prediction, academic performance prediction, student virtual academic performance prediction, and educational datasets.

On Google's Dataset Search Engine, the search with the keyword combination of "student, virtual academic, performance, prediction, datasets" returned 18 datasets. A second search with "student, academic, performance, prediction, datasets" yielded 72 dataset results. It is expected that fewer keywords would make the search more generic and yield increasingly larger but probably less relevant results. For example, the second search only contained 10 of the earlier 18 results repeated. Nonetheless, we accessed and analysed all 72 results for our selection criteria. The results clustered around three sources – Kaggle, the Public Library of Science (PLOS) on Figshare, and Mendeley.

We assessed and analysed every result in these searches and picked three (3) sources: Kaggle.com, Academia.edu and Figshare which we had identified as top sources from our Google Search and Google Scholar.

On Academia.edu we found an educational dataset named "Dataset of Students' Performance Using Student Information System (SIS), Modular Object-Oriented Dynamic

Learning Environment (Moodle) and the Mobile Application "eDify"[2]. Edify simplifies information associated with higher education.

Dataset 1 was collected from the student information system (SIS), the learning management system (LMS) called Moodle, and video interactions from the mobile application called "eDify." The dataset, from the higher educational institution (HEI) in the Sultanate of Oman, comprises five modules of data from Spring 2017 to Spring 2021. The original dataset as provided by the study is accessible[3].

Further descriptions of this dataset are provided later in chapter four. Dataset 2 is educational data consisting of student records from two semesters collected from the libaba Cloud Tranchi platform. Xapl-Edu data gathered using the Kalboard 360[4].

On Kaggle, a search for "student performance" and "learning analytics dataset" yielded 307 and 701 results respectively. We searched and filtered all the results for relevance based on our selection criteria. Finally, dataset 3 was selected by settling for "Student Performance Data"[5]. We traced the dataset to a source which is the UC Irvine Machine Learning Repository[6].

Sources of the selected datasets are presented in Table 3.2 below. Further details are provided in later sections.

Table 3.2: Sources of Selected Datasets

Attribute	Dataset 1	Dataset 2	Dataset 3
Source	Higher Educational Institution (HEI) in the Sultanate of Oman.	Xapl-Edu Data was gathered using the Kalboard 360 Learning Management System (LMS) which employs the experience of APL	Published in Proceedings of 5th Annual Future Business Technology Conference by P. Cortez, A. M. G. Silva. 2008.
	Retrieved via Academia.edu	Retrieved from Alibaba Cloud Tranchi platform.	Retrieved from the UC Irvine (UCI) archives

Critically, we ensured that these datasets included variables that have been identified by domain experts as relevant and correlated to students' performances. The performance is based on scores in tests and exams. The variables cover such related factors as:

1. **Academic Records:** Including grades, attendance, exam scores, and coursework completion status.
2. **Demographic Information:** Such as age, gender, ethnicity, and socioeconomic background.
3. **Parental Education and Occupation:** Providing insights into the educational and occupational backgrounds of students' parents.
4. **Behavioural Patterns:** Including disciplinary records, extracurricular activities participation, participation in virtual classrooms, and engagement in educational programs.

There is no consensus on the dataset used for predicting student academic performance. Many of these datasets include not only Learning Management Systems (LMS) interaction information but also other data such as gender, average self-study, student family size, and father and mother qualification[7,8]. However, if the dataset use LMS, there is a lot of variation from first login[9], total number of visits and clicks[10]

3.5 Data Preprocessing

The following pre-processing steps were followed to pre-process the datasets.

1. Exploratory Data Analysis: We calculated descriptive statistics and created various visualizations to examine data distributions, identify outliers, and explore feature relationships.

2. Data Cleaning and Missing Data: We employed several methods as required and as we consider appropriate in every case. We manually found the missing values by finding corresponding labels from similar instances. In other cases, affected instances were removed or addressed using mean imputation for numerical features. We carefully detected and corrected outliers using box plots and Z-scores, retaining those that appeared to contain meaningful information.

3. Encoding Categorical Variables: We applied one-hot encoding to transform categorical features into numerical representations.

4. Feature Scaling: We considered running our experiments on scaled and unscaled data but the number of iterations will double and would increase the complexity of this work. Consequently, we considered our target feature selection methods and machine learning models in selecting a method. We have chosen the

Standardization method of feature scaling as this is what fits Principal Component Analysis, where we usually prefer standardization over Min-Max scaling (normalization) since we are interested in the components that maximize the variance.

3.6 Dataset 1

The dataset contains Higher Educational Institutions (HEI) in the Sultanate of Oman, comprising five data modules from Spring 2017- Spring 2021. It integrates data from the student information system (SIS), Moodle (the LMS), and the "eDify" mobile video application. The dataset includes 326 student records with 40 features, covering acasdemic information (24 features), Moodle activity (10 features), and eDify video interactions (6 features).

This dataset is particularly relevant to this study which has virtual learning within its scope. Numerous studies have employed data mining approaches to predict student performance. However, the primary emphasis has been on demographic data. Research specifically analyzing video interactions of learners in video-assisted courses is limited. The dataset, integrating data from an SIS, Moodle, and a video-assisted course (eDify), offers a unique opportunity to explore video learning analytics.

3.6.1 Exploratory Data Analysis (EDA) of Dataset 1

The dataset was structured into three segments:

The presented dataset is organized into three distinct categories:

1. Student Academic Information: This data was collected directly from the institution's Student Information System (SIS).

2. Student Activity: This encompasses student interactions and activities performed within the Moodle learning management system.

3. Student Video Interactions: This data captures student engagement with video content through the "eDify" mobile application.

Figure 3.3 adapted from[7], illustrates the structure of the dataset and maps how these categories were integrated.

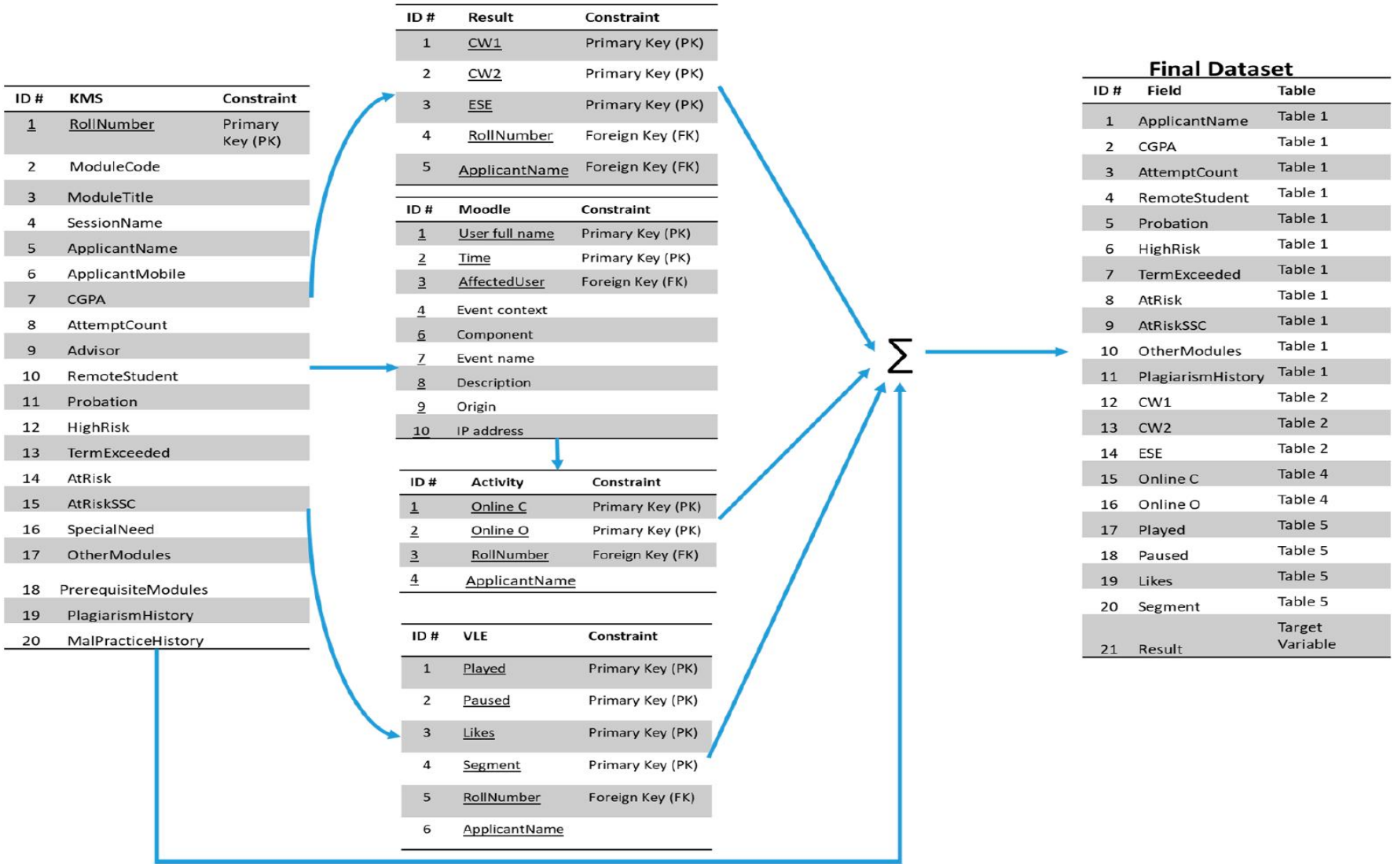


Figure 3.3: Dataset of Students' Performance using Student Information System[11]

3.6.2 Data Preprocessing of Dataset 1

The original work that collated the dataset had pre-processed in such a way that they coded results and several other metrics into ordinal text values.

We will take th e data to python and begin further Exploratory Data Analysis (EDA) and pre-processing.

3.7 Dataset 2

The xAPI-Edu-Data dataset is an educational classification dataset designed to analyze and predict student academic performance. Collected from the Alibaba Cloud Tianchi platform, the dataset includes 480 student records from two semesters, encompassing various countries and gender. The primary objective of this dataset is to support research in e-learning, predictive models, and educational data mining. The data was gathered using the Kalboard 360 Learning Management System (LMS), which employs the Experience API (xAPI) to track learner activities and progress.

Kalboard 360 is a multi-agent LMS aimed at enhancing learning through advanced technology, providing synchronous access to educational resources from any device with an internet connection. The Experience API, a component of the Training and Learning Architecture (TLA), enables detailed monitoring of learning progress and actions, such as reading articles or watching training videos. This API helps learning activity and identifies the learner, activity, and objects involved in a learning experience.

The dataset consists of 480 instances with 17 attributes each, categorized into demographic, academic background, and behavioural features. Demographic features include gender and nationality, academic background features cover educational stages and grade levels, and behavioural features track activities like raising hands in class,

visiting resources, and parental involvement. Notably, the dataset does not contain any missing values, making it robust for analysis. Student demographics show a gender distribution of 305 males and 175 females. Nationality-wise, the students originate from diverse countries, with significant representations from Kuwait (179 students) and Jordan (172 students). Other nationalities include Palestine, Iraq, Lebanon, Tunis, Saudi Arabia, Egypt, Syria, USA, Iran, Libya, Morocco, and Venezuela. This diverse composition allows for a comprehensive analysis of academic performance across different backgrounds.The dataset spans two educational semesters, with 245 records from the first semester and 235 from the second. Attendance records indicate that 191 students had more than seven absence days, while 289 students had fewer than seven absence days. Parental involvement was also tracked, with 270 parents responding to school surveys and 210 not responding. Additionally, 292 parents expressed satisfaction with the school, whereas 188 did not.

Academic performance is categorized into three levels based on total grades: Low (0-69), Middle (70-89), and High (90-100). The majority of students fall within the middle category, with high-scoring students (90-100) comprising 29.58% of the total. This distribution highlights the dataset's utility in identifying factors that contribute to different levels of academic success.

Overall, the xAPI-Edu-Data dataset is a valuable resource for this study which has virtual learning within its scope. The comprehensive set of LMS-tracked attributes, combined with detailed demographic and behavioural data, provides a robust foundation for developing predictive models and enhancing educational strategies.

Table 3.3: Dataset Overview of Dataset 2

Aspect	Description
Number of Instances	480
Number of Attributes	17
Area	E-learning, Education, Predictive Models, Educational Data Mining
Attribute Characteristics	Integer/Categorical
Date	2016-11-8
Associated Tasks	Classification
Missing Values	No
File Formats	Comma-separated Values (csv)
Source	Elaf Abu Amrieh, Thair Hamtini, and Ibrahim Aljarah, The University of Jordan, Amman, Jordan

Table 3.4: Data Collection of Dataset 2

LMS	Experience API (xAPI)
Tool	Kalboard 360
Function	Tracks learner activities and progress, monitors actions like reading articles or watching videos
System	Multi-agent LMS designed for synchronous access to educational resources

Table 3.5: Demographic Distribution of Dataset 2

Attribute	Description
Gender Distribution	Males: 305, Females: 175
Nationality Distribution	Kuwait: 179, Jordan: 172, Palestine: 28, Iraq: 22, Lebanon: 17, Tunis: 12, Saudi Arabia: 11, Egypt: 9, Syria: 7, USA: 6, Iran: 6, Libya: 6, Morocco: 4, Venezuela: 1

Table 3.6: Semester and Attendance of Dataset 2

Aspect	Description
Semesters	First: 245 records, Second: 235 records
Attendance	Absence days > 7: 191, Absence days ≤ 7: 289

Table 3.7: Parental Involvement of Dataset 2

Aspect	Description
Parent Answering Survey	Yes: 270, No: 210
Parent School Satisfaction	Yes: 292, No: 188

Table 3.8: Academic Performance of Dataset 2

Performance Level	Grade Range	Percentage of Students
Low	0-69	26.46%
Middle	70-89	43.96%
High	90-100	29.58%

3.7.1 Exploratory Data Analysis (EDA) of Dataset 2

As stated earlier, the dataset consists of 480 instances with 16 attributes each, categorized into demographic, academic background, and behavioural features. Demographic features include gender and nationality, academic background features cover educational stages and grade levels, and behavioural features track activities like raising hands in class, visiting resources, and parental involvement

Below is a table detailing the 16 attributes (features) and their data types. The "Class" attribute is our data type and we would have to adopt classification rather than regression models as we are unable to obtain the continuous scores of the students and rather have only the classifications of Low, Mid, and High corresponding to scores ranges 0-69, 70-89, and 90-100 respectively. We are unable to modify this classification in any way as we do not have access to the original scores.

1. Class Distribution

The class distribution plot indicates a significant imbalance in the target variable (Class):

- **High (H)**: Relatively fewer instances.
- **Medium (M)**: Largest group.
- **Low (L)**: Intermediate size.

This imbalance can affect model performance, particularly for the minority class (H). Techniques like SMOTE (Synthetic Minority Over-sampling Technique) can address this. However, we are relying on class weighting abilities of our models to address this.

2. Correlation Analysis

The correlation matrix for numerical features (raisedhands, VisITedResources, AnnouncementsView, Discussion) shows:

- **raisedhands** and **VisITedResources** have a moderate positive correlation.
- Other correlations are weaker, indicating that these features might provide complementary information.

3. Distribution of Numerical Features

The histograms for numerical features show:

- **raisedhands**: Skewed distribution with most values on the lower side but a long tail.
- **VisITedResources**: Similarly skewed distribution as “raisedhand”.
- **AnnouncementsView**: Follows a similar pattern but with fewer extreme values.
- **Discussion**: Most values are concentrated on the lower side.

4. Distribution of Categorical Features

The count plots for categorical features show:

- **Gender**: Balanced distribution between male and female students.
- **NationalITy** and **Place of Birth**: Dominated by a few categories (e.g., 'KW' for Kuwait).
- **StageID**, **GradeID**, and **SectionID**: Fairly balanced with slight variations.
- **Topic**: Heavily skewed towards certain subjects (e.g., IT).
- **Semester**: Equal distribution between Fall and Spring.
- **Relation**: Predominantly 'Father'.

- **Parent Answering Survey and Parent school Satisfaction**: Skewed distributions with most responses being 'Yes' and 'Good'.
- **Student Absence Days**: Mostly 'Under-7' days of absence.

5. Feature Distributions Across Classes

The boxplots showing numerical features across different classes highlight:

- **raisedhands**: Higher values tend to correlate with better performance (more hands raised in the high-performing class).
- **VisITedResources**: Similar trend, with more resource visits in higher-performing classes.
- **Announcements View**: Higher counts for better-performing studentsy, but with less clear separation.
- **Discussion**: More discussions participated in by higher-performing students, with overlap.

6. Suitability for Modeling

- **Numerical Features**: Show distinct distributions across performance classes, indicating suitability for prediction.
- **Categorical Features**: Offer potential insights into demographic and behavioural factors influencing performance.

We can conclude that the dataset is suitable for modelling predictors of academic performance, given the evident relationships between features and the target variable. We therefore proceeded to preprocessing.

3.7.2 Data Preprocessing of Dataset 2

The original work that collated the dataset had pre-processed in such a way that they coded results and several other metrics into ordinal text values. We are unable to get the raw dataset so we will work with it as such and switch over to classification.

A quick Excel view of the raw pre-processed file showed no columns are required to be dropped. The next step is to check for missing values. We found no missing values across all the attributes. Values of 0 are taken as 0 and not missing.

Afterwards, we perform encoding for the categorical values and standardization scaling of the numerical values.

To ensure the best encoding for each attribute, we carefully considered the characteristics and cardinality of each feature. So, we will use one-hot encoding for low-cardinality nominal features, ordinal encoding for ordinal features, and frequency encoding for high-cardinality nominal features.

With the above, below is our encoding plan:

1. **Gender**: Nominal, low cardinality -> One-hot encoding
2. **Nationality**: Nominal, high cardinality -> Frequency encoding
3. **Place of birth**: Nominal, high cardinality -> Frequency encoding
4. **Educational Stages**: Ordinal -> Ordinal encoding (Lower Level < Middle School < High School)
5. **Grade Levels**: Ordinal -> Ordinal encoding (G-01 < G-02 < ... < G-10)
6. **Section ID**: Nominal, low cardinality -> One-hot encoding
7. **Topic**: Nominal, high cardinality -> Frequency encoding
8. **Semester**: Nominal, low cardinality -> One-hot encoding

9. **Parent responsible for student**: Nominal, low cardinality -> One-hot encoding
10. **Parent Answering Survey**: Nominal, low cardinality -> One-hot encoding
11. **Parent School Satisfaction**: Nominal, low cardinality -> One-hot encoding
12. **Student Absence Days**: Ordinal -> Ordinal encoding (Under-7 < Above-7)
13. **Grade Classification**: Ordinal -> Ordinal encoding (Low < Middle < High)

With this, we conclude preprocessing and have our data ready for the experiment sets. In dataset 2 experimentation, we have to deviate from regression models to classification models, and we will be performing multi-class classification using Logistic Regression (Multinomial), k-Nearest Neighbors (k-NN), Support Vector Machine (SVM), Random Forest, Gradient Boosting, XGBoost, LightGBM, MLP classifier (Neural Network) and Naive Bayes.

Table 3.9: Attributes and Descriptions of xAPI-Edu-Data

Attribute	Description	Data Type
Gender	Student's gender (e.g., "Male", "Female")	Nominal
Nationality	Student's nationality (e.g., "Kuwait", "Lebanon", "Egypt")	Nominal
Place of birth	Student's place of birth (e.g., "Kuwait", "Lebanon", "Egypt")	Nominal
Educational Stages	Educational level student belongs to (e.g., "Lower Level", "Middle School", "High School")	Ordinal
Grade Levels	Grade level student belongs to (e.g., "G-01", "G-02", "G-03")	Ordinal
Section ID	Classroom student belongs to (e.g., "A", "B", "C")	Nominal
Topic	Course topic (e.g., "English", "Spanish", "French")	Nominal
Semester	School year semester (e.g., "First", "Second")	Ordinal
Parent responsible for student	Parent responsible for the student (e.g., "mom", "father")	Nominal
Raised hand	Number of times the student raises his/her hand in class (0-100)	Numeric
Visited resources	Number of times the student visits course content (0-100)	Numeric
Viewing announcements	Number of times the student checks new announcements (0-100)	Numeric
Discussion groups	Number of times the student participates in discussion groups (0-100)	Numeric
Parent Answering Survey	Whether the parent answered surveys provided by the school (e.g., "Yes", "No")	Nominal
Parent School Satisfaction	Degree of parent satisfaction with the school (e.g., "Yes", "No")	Nominal
Student Absence Days	Number of absence days for each student (e.g., "above-7", "under-7")	Ordinal
Grade Classification	Student's grade classification based on total grade (e.g., "Low", "Middle", "High")	Ordinal

3.8 Dataset 3

The dataset was collected as part of an educational research initiative aimed at understanding the factors influencing academic success in secondary education. This dataset was originally compiled by (Paulo Cortez and his colleagues) and has been widely used in the academic and machine learning communities for various analyses and predictive modelling tasks.

3.8.1 Data Collection

The dataset was collected from two Portuguese secondary schools, Gabriel Pereira (GP) and Mousinho da Silveira (MS). The data encompasses a range of variables that capture not only academic performance but also socio-economic, demographic, and behavioural aspects of the students. The objective was to gather comprehensive information that could shed light on the multifaceted nature of student performance.

3.8.2 Data Description

1. **Demographic Information**:

    - **School**: The dataset records the school attended by each student, either Gabriel Pereira or Mousinho da Silveira. This variable provides context for the academic environment and potentially different teaching methods or resources available at each school.

    - **Gender and Age**: Gender and age are crucial demographic factors. The age range of the students is from 15 to 22 years, which is typical for secondary education in Portugal.

2. **Family Background**:

    - **Address and Family Size**: The type of home address (urban or rural) and the family size provide insight into the socio-economic background of the students. Larger families and those living in rural areas may face different challenges compared to smaller, urban families.

    - **Parental Status and Education**: Information on parents' cohabitation status, education levels, and occupations offers a view into the socio-economic and educational support system available to the students.

3. **Educational and Extra-Curricular Activities**:

    - **Study and Travel Time**: The dataset includes how much time students spend travelling to school and studying each week. These factors can significantly impact the amount of time available for academic and extracurricular activities.

    - **Extra Support and Activities**: Variables like additional educational support, family support, paid classes, and participation in extracurricular activities highlight the various forms of academic assistance and enrichment activities available to the students.

4. **Personal and Social Attributes**:

    - **Family Relationships and Free Time**: The quality of family relationships, amount of free time, and social activities like going out with friends are included to understand their impact on student well-being and performance.

    - **Alcohol Consumption and Health**: Workday and weekend alcohol consumption are recorded along with the student's health status, acknowledging the potential influence of lifestyle choices on academic outcomes.

5. **Academic Performance**:

    - **Grades and Absences**: The key performance indicators in the dataset are the grades from three periods (G1, G2, and G3) and the number of school absences. These metrics provide a direct measure of academic success and consistency.

3.8.3 General Description

The dataset is structured with 395 instances and 33 attributes, providing a rich and detailed snapshot of student life and performance. Each record in the dataset corresponds to a unique student and includes both categorical and numerical variables. The final grade (G3) serves as the target variable for regression analysis, making this dataset suitable for predicting academic performance based on the myriad factors recorded.

This dataset is particularly valuable for its comprehensive nature, encompassing a broad spectrum of influences on student performance. It allows for the exploration of how personal, familial, and educational factors intertwine to affect academic outcomes. Researchers can leverage this data to identify key predictors of success, understand the impact of different socio-economic backgrounds, and develop strategies to support students in achieving their full potential.

3.8.4 Dataset Structure

The "Student Performance Data" dataset comprises records of 395 students, focusing on their performance in secondary education, specifically in Mathematics. The dataset includes 33 attributes that capture a wide array of factors such as demographic information, social and family dynamics, educational support, and academic performance. The target variable for this regression problem is the final grade (G3).

1. **Demographic Attributes**:

    - **school**: The school attended by the student (binary: 'GP' for Gabriel Pereira, 'MS' for Mousinho da Silveira).
    - **Gender**: Gender of the student (binary: 'F' for female, 'M' for male).
    - **age**: Age onf the student (numeric, ranging from 15 to 22).

2. **Family Attributes**:

    - **address**: Type of home address (binary: 'U' for urban, 'R' for rural).
    - **famsize**: Family size (binary: 'LE3' for less than or equal to 3, 'GT3' for greater than 3).
    - **Pstatus**: Parent's cohabitation status (binary: 'T' for living together, 'A' for apart).
    - **Medu**: Mother's education level (numeric: 0 to 4, where 0 means none and 4 means higher education).
    - **Fedu**: Father's education level (numeric: 0 to 4).
    - **Mjob**: Mother's job (nominal: 'teacher', 'health', 'services', 'at home', 'other').
    - **Fjob**: Father's job (nominal: same categories as mother's job).
    - **reason**: Reason for choosing the school (nominal: 'home', 'reputation', 'course', 'other').
    - **guardian**: Guardian of the student (nominal: 'mother', 'father', 'other').

3. **Study and Travel Attributes**:

- **traveltime**: Time taken to travel from home to school (numeric: 1 to 4, where 1 means <15 min and 4 means >1 hour).
- **studytime**: Weekly study time (numeric: 1 to 4, where 1 means <2 hours and 4 means >10 hours).
- **failures**: Number of past class failures (numeric: 0 to 4).

4. **Support and Activities**:

- **schoolsup**: Extra educational support (binary: yes or no).
- **famsup**: Family educational support (binary: yes or no).
- **paid**: Extra paid classes within the course subject (binary: yes or no).
- **activities**: Participation in extra-curricular activities (binary: yes or no).
- **nursery**: Attended nursery school (binary: yes or no).
- **higher**: Aspiration to pursue higher education (binary: yes or no).
- **internet**: Internet access at home (binary: yes or no).
- **romantic**: Engagement in a romantic relationship (binary: yes or no).

5. **Personal and Social Attributes**:

- **famrel**: Quality of family relationships (numeric: 1 to 5).
- **freetime**: Amount of free time after school (numeric: 1 to 5).
- **goout**: Frequency of going out with friends (numeric: 1 to 5).
- **Dalc**: Workday alcohol consumption (numeric: 1 to 5).
- **Walc**: Weekend alcohol consumption (numeric: 1 to 5).
- **health**: Current health status (numeric: 1 to 5).

- **absences**: Number of school absences (numeric: 0 to 93).

6. **Academic Performance**:

- **G1**: First-period grade (numeric: 0 to 20).
- **G2**: Second-period grade (numeric: 0 to 20).
- **G3**: Final grade (numeric: 0 to 20, this is the target variable for the regression problem).

3.8.5 Exploratory Data Analysis (EDA) of Dataset 3

The dataset consists of 395 student records and 33 attributes, capturing a wide range of demographic, social, and academic factors. Here are some key points from the summary statistics:

- **School**: The dataset includes students from two schools, with the majority (349) attending Gabriel Pereira (GP).
- **Sex**: There are 208 female and 187 male students.
- **Age**: The age of students ranges from 15 to 22 years, with an average age of about 16.7 years.

Family-related attributes:

- **Address**: Most students (307) live in urban areas.
- **Family Size**: A larger portion of students (281) come from families with more than three members.
- **Parental Status**: Most students (354) have parents who live together.

- **Parents' Education**: The education levels of mothers and fathers vary, with averages around secondary education level (Medu mean: 2.75, Fedu mean: 2.52).
- **Parents' Occupations**: The most common job for mothers is 'other' (141), and for fathers, it is also 'other' (217).

Study and Travel Time:

- **Travel Time**: Most students have a travel time of less than 30 minutes to school.
- **Study Time**: Weekly study time is varied, with a notable portion of students studying less than 5 hours a week.
- **Past Failures**: The number of past class failures is relatively low, with the majority having no failures.

Support and Activities:

- **Extra Educational Support**: A mixed distribution of students receiving extra educational support.
- **Family Support**: Similarly mixed distribution for family support.
- **Paid Classes**: Around half of the students take extra paid classes.
- **Extracurricular Activities**: Participation in extracurricular activities is also balanced.
- **Nursery School Attendance**: Most students attended nursery school.
- **Higher Education Aspiration**: A large majority of students aspire to pursue higher education.
- **Internet Access**: Most students have internet access at home.
- **Romantic Relationships**: About half of the students are in romantic relationships.

Personal and Social Attributes:

- **Family Relationships**: Generally good, with an average rating of 3.94 out of 5.
- **Free Time**: Moderate, with an average rating of 3.24 out of 5.
- **Going Out with Friends**: Moderate, with an average rating of 3.11 out of 5.
- **Alcohol Consumption**: Workday alcohol consumption is low (average 1.48 out of 5), but weekend consumption is higher (average 2.29 out of 5).
- **Health**: Generally good, with an average rating of 3.55 out of 5.
- **Absences**: Absences vary widely, with some students having as many as 75 absences, but the majority have fewer than 10.

Academic Performance:

- **G1 (First Period Grade)**: Average grade is about 10.91 out of 20.
- **G2 (Second Period Grade)**: Average grade is about 10.71 out of 20.
- **G3 (Final Grade)**: Average grade is about 10.42 out of 20, which is our target variable for regression analysis.

Data Distribution and Visualizations

We visualized the distributions of some key numerical variables (age, absences, grades) and categorical variables (school, gender, address). This will be presented in chapter four.

3.8.6 Data Preprocessing of Dataset 3

The dataset presents us with a continuous value for “G3”, our target variable. So, we have a regression problem.

A quick Excel view of the dataset showed no columns are required to be dropped. The next step is to check for missing values. We found no missing values across all the attributes. Values of 0 are 0 and not missing.

Next is for us to encode the variables and get the dataset ready for feature selection modelling.

The dataset has 395 entries and 33 columns, with a mix of numerical and categorical data. Here's a brief overview of the dataset structure:

- The Numerical features (int64) are: age, Medu, Fedu, traveltime, studytime, failures, famrel, freetime, goout, Dalc, Walc, health, absences, G1, G2, G3.
- The Categorical features (object): school, gender, address, famsize, Pstatus, Mjob, Fjob, reason, guardian, schoolsup, famsup, paid, activities, nursery, higher, internet, romantic.

We can further consider binary features while also considering the ordinality of some features.

Based on the dataset structure and the types of variables provided, we can identify the following:

Categorical Variables Encoding:

1. **Binary Variables**:

- school, genders, address, famsize, Pstatus, schoolsup, famsup, paid, activities, nursery, higher, internet, romantic.

For binary categorical variables like these (which have only two unique values), you can use label encoding:

- Assign 0 or 1 for each category. For example, for school, 'GP' could be 0 and 'MS' could be 1. This ensures each category is represented by a numerical value without introducing unnecessary dimensions.

2. **Multi-category Variables**:

   - Mjob, Fjob, reason, guardian

   For variables with more than two categories, we can use one-hot encoding to create new features. For example,

   - Create binary columns for each category. For instance, Mjob_teacher, Mjob_healths for Mjob.
   - This approach ensures that each category is represented independently without assuming any ordinal relationship among them.

3. **Numeric Variables:** Variables like age, Medu, Fedu, traveltime, studytime, failures, famrel, freetime, goout, Dalc, Walc, health, absences, G1, G2, G3 are already in numeric format and do not require encoding. We would only apply scaling.

Meanwhile, it is important to note that some of the categorical features are ordinal. For example, famsize (family size), has ordinal characteristics. In such cases, it's important to encode them in a way that preserves their ordinal nature, especially when using models that can benefit from understanding ordinal relationships. Models like Linear Regression, Decision Trees, Random Forests, and Gradient Boosting can potentially benefit from understanding ordinal relationships in features like famsize, Medu, and Fedu. These models can learn and utilize the order of values to make more informed predictions.

Based on the foregoing consideration, we have the following as our encoding plan.

Numerical Features: we will apply Standarization scaling to numerical features.

1. **Demographic Attributes**:
    - age: Numeric attribute representing age of the student. This is numeric and naturally ordinal.
2. **Family Attributes**:
    - Medu: Numeric attribute representing Mother's education level (ordinal).
    - Fedu: Numeric attribute representing Father's education level (ordinal).
    - traveltime: Numeric attribute representing time taken to travel from home to school (ordinal).
    - studytime: Numeric attribute representing weekly study time (ordinal).
    - failures: Numeric attribute representing number of past class failures (ordinal).
    - famrel: Numeric attribute representing quality of family relationships (ordinal).
    - freetime: Numeric attribute representing amount of free time after school (ordinal).
    - goout: Numeric attribute representing frequency of going out with friends (ordinal).
    - Dalc: Numeric attribute representing workday alcohol consumption (ordinal).
    - Walc: Numeric attribute representing weekend alcohol consumption (ordinal).
    - health: Numeric attribute representing current health status (ordinal).

- absences: Numeric attribute representing number of school absences (ordinal).

3. **Academic Performance**:

   - G1: Numeric attribute representing first-period grade (ordinal).
   - G2: Numeric attribute representing second-period grade (ordinal).
   - G3: Numeric attribute representing final grade (ordinal, target variable).

Categorical Features:

1. **Demographic Attributes**:

   - school: Binary attribute ('GP' for Gabriel Pereira, 'MS' for Mousinho da Silveira). This is nominal as no school is inherently better than the other.
   - gender: Binary attribute ('F' for female, 'M' for male). This is also nomnal as no gender is higher than another

2. **Family Attributes**:

   - address: Binary and nominal attribute ('U' for urban, 'R' for rural).
   - famsize: Binary attribute ('LE3' for less than or equal to 3, 'GT3' for greater than 3) (ordinal).
   - Pstatus: Binary attribute ('T' for living together, 'A' for apart). Categorizing this feature as ordinal because we believe a student who has the support of both parents might be more advantaged than a student with one.
   - Mjob: Nominal attribute representing Mother's job ('teacher', 'health', 'services', 'at_home', 'other').
   - Fjob: Nominal attribute representing Father's job ('teacher', 'health', 'services', 'at_home', 'other').

- reason: Nominal attribute representing reason for choosing the school ('home', 'reputation', 'course', 'other').
- guardian: Nominal attribute representing guardian of the student ('mother', 'father', 'other').

3. **Study and Support Attributes**: These attributes are very subjective and we chose to just code them as binary with 1 for yes and 0 for no. We believe any ordinal bias that might introduce is subective when you consider the probable impact of these variables.
    - schoolsup: Binary attribute indicating extra educational support (yes or no).
    - famsup: Binary attribute indicating family educational support (yes or no).
    - paid: Binary attribute indicating extra paid classes within the course subject (yes or no).
    - activities: Binary attribute indicating participation in extra-curricular activities (yes or no).
    - nursery: Binary attribute indicating attendance at nursery school (yes or no).
    - higher: Binary attribute indicating aspiration to pursue higher education (yes or no).
    - internet: Binary attribute indicating internet access at home (yes or no).
    - romantic: Binary attribute indicating engagement in a romantic relationship (yes or no).

Encoding Plan:

1. **Binary Variables**:
    - Use label encoding (0 or 1) for binary variables where categories are non-ordinal:

- school, gender, address, Mjob, Fjob, reason, guardian, schoolsup, famsup, paid, activities, nursery, higher, internet, romantic.

2. **Ordinal Variables**:
    - Use ordinal encoding for variables that have a natural order:
        - Medu, Fedu, traveltime, studytime, failures, famrel, freetime, goout, Dalc, Walc, health. These are numeric and would already retain their ordinality when scaled using standardization.
        - Ensure that ordinal encoding preserves the order of categories (e.g., 'LE3' as 0 and 'GT3' as 1 for famsize, 'T' as 0 and 'A' as 1 for Pstatus).
3. **Nominal Variables**:
    - Use one-hot encoding for nominal variables to create binary variables for each category. This removes any ordinal bias we might introduce by any other form of encoding.
        - Mjob, Fjob, reason, guardian

3.9 Pearson Correlation Coefficient

The Pearson Correlation Coefficient (PCC), often referred to as 'r', is a tool in statistics used to measure the direction and strength of a linear relationship between two continuous variables. The value of 'r' falls between -1 and +1. A value of -1 means a perfect negative linear correlation, while +1 indicates a perfect positive linear correlation. A value of 0 means there is no linear correlation between the two variables. The sign of the PCC value (+ or -) shows the direction of the relationship. A positive sign means that as the value of one variable increases, the value of the other variable also tends to increase. Conversely, a negative sign indicates that as one variable increases, the other tends to decrease. It's important to note that the PCC specifically measures linear

relationships, and may not be a good indicator of the strength of a relationship if the association between the variables is non-linear. Also, two strongly correlated variables are not automatically causative. It is generally understood that correlation doesn't imply causation. Other factors may be influencing the relationship, or the strong correlation might just be a coincidence.

3.10 Principal Component Analysis (PCA)

The Principal Component Analysis (PCA) is a foundational dimensionality reduction technique widely used in machine learning and data analysis. At its core, PCA aims to transform a dataset consisting of multiple correlated variables into a set of linearly uncorrelated variables called principal components (PCs). Its primary objective is to find a new coordinate system (formed by principal components) that captures the maximum variance within a dataset, providing a more compact and informative representation. Imagine a dataset where features are highly correlated this implies that some features carry redundant information. PCA aims to:

1. Identify Directions of Maximum Variance: Find the directions, or axes, along which the data exhibits the greatest spread.
2. Create Orthogonal Components: Generate new features (principal components) that are linear combinations of the original features and are uncorrelated (orthogonal) to each other.
3. Ordered Importance: Rank the principal components (PCs) by the amount of variance they explain. The first principal component accounts for the most variance, then the second, and so on.

These PCs are ordered in terms of the variance they capture in the original data, with the first PC capturing the maximum variance and subsequent PCs capturing decreasing amounts of variance. This transformation is achieved through a series of mathematical computations that involve eigenvalue decomposition and linear algebra operations.

Firstly, the data is centred by subtracting the mean of each feature across samples, ensuring that the data is centred around zero. This step is crucial as PCA is sensitive to the mean of the data.

Step 1: Define the data.

Step 1: Calculate the mean vector $\bar{x}$ across each feature:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \text{Equation (3.1)}$$

Step 2: Subtract the mean from each feature to centre the data:

$$X_{centered} = X - \bar{x} \qquad \text{Equation (3.2)}$$

Step 3: Compute the covariance matrix S of the centred data:). Obtain covariance of the matrix. S

$$S = \begin{bmatrix} S_{it} & S_{12}, & ..., & S_{1p} \\ S_{21} & S_{22}, ..., S_{2p} \\ S_{p1} & S_{2p} & ....S_{pp} \end{bmatrix} \qquad \text{Equation (3.3)}$$

such that=

$$S_i = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \qquad \text{Equation 3.4}$$

and

$S_{11} = S_{22}, \quad S_{12} = S_{21}$ Equation (3.5)

Where

$$S_{ij} = \frac{\sum_{i=1}^{p} \left(x_{ii} - \overline{x_i}\right)^2}{n-1}, \quad i = j$$ Equation (3.6)

$$\frac{\Sigma \left(X_{ij} - \bar{x}_i\right)\left(X_{ij} - \bar{x}_j\right)}{n-1} \; for = i \pm j$$ Equation (3.7)

$S = n-11 \quad (X_{centered} \quad)T \cdot X_{centered}$ Equation (3.8)

Step 4: Perform eigenvalue decomposition on S to obtain eigenvalues $\lambda_1 \ ,\lambda_2 \ ,...,\lambda p$ and corresponding eigenvectors $v_1 \ ,v_2 \ ,...,v_p$. These eigenvectors represent the principal components. The covariance matrix E of the centred data x is Y:

$E = 1/n\text{-}1 \ x^T x$ Equation (3.9)

Where x^T is the transpose of the centered data matric and

$x^T x$ c; computes the dot product of the transpose and the centred data.

1/n-1 is used to get an unbiased estimator of the covariance.

Step 5: Select the PCs by sorting the eigenvalues in descending order and selecting the top k eigenvectors corresponding to the highest eigenvalues. These eigenvectors form the new basis for the transformed data.

Step 6: Transform the data by projecting the original data onto the new basis to obtain the transformed dataset X transformed :

$$X_{transformed} = X_{centered} \cdot V_k \quad \text{Equation (3.10)}$$

Where V_k is the matrix containing the top k eigenvectors as columns.

3.11 Forward Selection Method (FOR)

Subsequently, Forward Selection (FOR) augments the feature selection process. This greedy algorithm works iteratively, starting from the reduced representation obtained from PCA. At each step, FOR examines the impact of adding individual features on the performance of a chosen machine learning model. The feature that yields the greatest performance improvement is then incorporated into the permanent feature set. This stepwise approach ensures we include features that demonstrably enhance model capability

Forward Selection as a feature selection process works iteratively, starting with a null model, adding only one feature at a time to the model and if a feature is selected, it never drops from the model. The central idea behind Forward Selection is to build a feature set incrementally. Here's how it operates:

1. **Start Small:** Initialize an empty set of features.
2. **Iterate and Evaluate:** For each feature not already in the selected set,
    - Temporarily add the feature to the current set of features,
    - Train a machine learning model with this newly expanded feature set, and
    - Evaluate the model's performance using a chosen metric (e.g., accuracy, F1-score).
3. **Select the Best:** Identify the feature that, when added to the existing set, resulted in the greatest improvement in the performance metric.

4. **Grow the Feature Set:** Permanently add the selected feature to the feature set.
5. **Repeat:** Continue the iterative process until a stopping criterion is met (e.g., no significant performance improvement, reaching a maximum number of features).

Mathematically, let's represent the dataset as X with p features and y as the target variable. The Forward Selection process can then be expressed as follows:

1. Start with an empty set of selected features: $S=\emptyset$
2. For each feature X_j not in S, evaluate the performance metric (e.g., R^2 for regression, accuracy for classification) when combining x_j with the features in S.
3. Select the feature X_k that maximises the performance metric:

   xk =argmaxXj ∉S Performance Metric(S ∪ {xj })

Add the selected feature to the set of selected features: $S=S \cup \{x_k \}$

4. Repeat the process until a stopping criterion is met (e.g., maximum number of features, no improvement in performance).

Notably, Forward Selection is a practical and intuitive feature selection method that can improve model performance and reduce overfitting by focusing on the most relevant features. However, it may not consider feature interactions and may require careful tuning of stopping criteria to avoid overfitting. We believe an earlier PCA solves this and FOR selects the best of the PCs.

3.12 Particle Swarm Optimization (PSO)

Lastly, the integration of Particle Swarm Optimization (PSO) introduces a global, metaheuristic component that complements the localized search of Forward Selection. Particle Swarm Optimization (PSO) was inspired by the social behaviour of bird flocks or fish schools. It explores all possible combinations of features. Hence, it is commonly used

in optimization problems, including feature selection in machine learning. PSO is widely used in both science and industry optimization algorithms. The PSO algorithm was put forward by Eberhart and Kennedy[12]. It is a population intelligent algorithm designed by imitating the study on the predation behaviour of birds while observing the predation situation[13].

PSO is based on the concept of particles moving through a search space to find the optimal solution by iteratively updating their positions and velocities. In this framework, each particle within the swarm represents a candidate feature subset. The "fitness" of a particle is determined by evaluating the model's performance using its corresponding feature subset. Through a collaborative process of updating particle positions and velocities based on individual and swarm-wide best experiences, PSO converges towards an optimal or near-optimal feature combination.

This is the major idea of PSO is the search space for the best position of each particle and the best positions of all particles:

- **Particles as Potential Solutions:** Each particle represents a candidate solution (in our case, a potential feature subset).
- **Positions and Velocities:** Particles have positions (representing their current solution) and velocities (indicating their movement direction and speed).
- **Collaborative Search:** Particles are influenced by:
    - **Personal Best:** Their own best-known position found so far.
    - **Global Best:** The best position discovered by any particle in the entire swarm.
- **Movement Updates:** At each iteration, particles adjust their velocities and positions based on their personal best, the global best, and some degree of

randomness. This enables them to explore the search space while being drawn towards promising areas.

Mathematically, PSO can be understood as:

Step 1. Initialization: Initialize a population of particles randomly within the search space. Each particle represents a potential solution to the optimization problem.

Step 2. Velocity Update: Update the velocity of each particle based on its previous velocity, personal best position, and global best position. The velocity update formula for particle i in dimension d is given by:

$$v_i(t+1) = wxv_i(t) + C_1 r_1\left(pbest_i - x_i(t)\right) \quad \text{Equation (3.11)}$$

$(inertia \quad past/cognitive\ component)$

$$+ C_2 r_2 \left(gbest - xi(t)\right)$$

$social\ component$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad \text{Equation (3.12)}$$

Where:

1. $v_{i\ d}(t+1)$ is the updated velocity of particle i in dimension d at time t+1.
2. w is the inertia weight controlling the particle's momentum.-
3. c_1 and c_2 are acceleration coefficients.
4. r_1 and r_2 are functions generating random values between 0 and 1.
5. p_{id} is the personal best position of particle i in dimension d.
6. p_{gd} is the global best position among all particles.
7. x_{id} (t) is the current position of particle i in dimension d at time t.

Step 3. Position Update: Update the position of each particle based on its velocity. The position update formula for particle i in dimension d is given by:

$x_{id}\ (t+1)=x_{id}\ (t)+v_{id}\ (t+1)$ Equation (3.13)

Step 4. Fitness Evaluation: Evaluate the fitness (objective function value) of each particle based on its current position.

Step 5. Update Personal Best and Global Best: Update the personal best position p_{id} for each particle and the global best position p_{gd} among all particles based on their fitness values.

Step 6. Termination Criteria: Repeat steps 2-5 until a termination criterion is met (e.g., maximum number of iterations, convergence criteria).

As such, PSO leverages the collective intelligence of particles to explore and exploit the search space effectively, making it suitable for optimization tasks such as feature selection.

Key things to consider in these implementations include a few parameters and arguments to be passed such as:

1. Fitness Function: The heart of PSO for feature selection lies in the fitness function design. It should accurately evaluate how well a feature subset contributes to the model's performance.

2. Hyperparameters to consider:

    a. swarmsize: Size of the particle swarm (e.g. 30)

    b. maxiter: Maximum number of iterations (e.g. 100)

c. w (inertia weight) and c1, c2 (acceleration coefficients): These influence the balance between particle personal exploration and convergence towards the global best.

3. Discrete PSO: This PSO has been adapted to discrete search spaces (what we have in feature selection problem) where positions require binary values (feature included or excluded) and termed lb and ub (lower and upper bounds respectively).

3.13 Model and Model Evaluation

Academic performance prediction can either be a regression or classification problem depending on the target prediction. If we want to predict continuous score values, we have a regression problem. Typical models that fit this problem class are linear regression models such as Simple Linear Regression and Multiple Linear Regression, and non-linear regression models such as Polynomial Regression, Support Vector Regression, Decision Tree Regression, and Random Forest Regression. However, if we want to predict whether the student will pass or fail, then we have a binary classification problem. Typical models for these problems are Logistic Regression, K-Nearest Neighbours, Support Vector Machines, Kernel SVM, Naïve Bayes and Decision Tree Classification, and Random Forest Classification models. Nonetheless, we recognize that these typical binary classification models can be tweaked with binary transformations to perform multi-class classification which can be useful if we classify students into fail, pass and distinction. This introduces a certain complexity to our work. We have to properly consider our datasets to determine if they have binary targets if we would transform them to have a target binary feature with variables of pass and fail or if they have nominal targets.

Now, Dataset 1 had three grades – CW1, CW2 AND ESE. CW1 represents marks obtained in the first course week and contains discrete data like 86.5. The same applies

for CW2 and ESE which represent the scores for the second course week and the end of the semester respectively. In this case, we have a regression problem and we need not apply classification except we define a cut-off mark and transform the scores into pass and fail. Dataset 2 is a classification problem, it was evaluated by Accuracy, Precision, F-mean and Recall. Dataset 3 has three target variables G1, G2, and G3 representing grades. G1 is the first period grade (numeric: from 0 to 20), G2 is the second period grade (numeric: from 0 to 20), and G3 is the final grade (numeric: from 0 to 20, output target). This is a regression problem.

Model results of datasets1 and datasets3 were evaluated using standard Linear Regression performance metrics – Mean Absolute Error (MAE), Mean Squared Error (MSE) and R-squared. The performance of the models was assessed with and without feature selection to compare the impact of different feature selection techniques on model accuracy and predictive power. This comprehensive evaluation ensures a thorough analysis of the effectiveness of the different feature selection methods in enhancing the predictive capabilities of machine learning models for students' performance prediction.

3.14 Evaluation Metrics

Regression model performance is often evaluated utilizing these three metrics: R-squared, Mean Squared Error (MSE), and Mean Absolute Error (MAE).

R-squared

R-squared (R^2) is a statistical measure also called the coefficient of determination, which shows how much of the variance of the target variable can be anticipated or explained by the features used as input in a regression model

A higher score denotes a more favourable model-data fit. R-squared has a scale from 0 to 1 or 0 percent to 100 percent. 100 percent, or 1, means that all of the variability in the r.esponse data around the mean is explained by the model. Put in another way, the dependent variable is correctly predicted by the independent variables. R-squared indicates that independent variables have no predictive power over the dependent variable if it is zero or 0 percent.

$$R^2 = 1 \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$ Equation (3.14)

Where:

y_i = actual value

$\hat{y}_i$ = predicted value

$\bar{y}$ = mean of the actual values

n = number of data point

Mean Squared Error

Mean Squared Error (MSE) is a statistic that shows the average squared difference between actual and predicted outcomes It measures the correctness of the model; lower values suggest better performance[14,16]. By virtue of the squaring of differences, MSE is more sensitive to large errors.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2$$ Equation (3.15)

Mean Absolute Error

The Mean Absolute Error (MAE) is determined by calculating the average absolute deviation between the actual and predicted values, In comparison with MSE, the Mean Absolute Error is less prone to outliers[14,15,16].

$$MAE = \frac{1}{n} \sum_{i=1}^{b} | y_i - \hat{y}_i | \qquad \text{Equation (3.16)}$$

Several measures such as accuracy, recall, precision and F-measure has been proposed to evaluate model performance in classification problems.

Accuracy:

Accuracy measures the percentage of true positives and true negatives, or accurate predictions, among all predictions made.

Precision:

This measures the percentage of actual positive predictions among all the classifier's positive predictions.

Recall (Sensitivity):

Recall measures the percentage of real positives that the classifier accurately identified.

F- Measure (F1-Score):

This is the harmonic mean of recall and precision, to provide a balance between the two, particularly in cases where the distribution of classes is not uniform.

The performance of the models was examined using performance measures based on the confusion matrix. A two-class confusion matrix is shown in Table 3.10.

Where,

TP =the proportion of positive cases that were correctly identified. For instance, predict as positive when actual positive.

TN = the proportion of negative cases that were classified correctly. Predict negative when actual negative.

FN = the proportion of positive cases that were incorrectly classified as negative. Predict negative when actual positive.

FP = the proportion of negative cases that were incorrectly classified as positive. Predict positive when actual negative.

TPR = True Positive Rate

PPV = Positive Predictive Value

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP} \qquad \text{Equation (3.17)}$$

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Equation (3.18)}$$

$$\text{Recall} = \frac{TP}{TP + FN} \qquad \text{Equation (3.19)}$$

$$\text{F-measure} = \frac{2(\text{Pr}\,ecision * \text{Re}\,call)}{\text{Pr}\,ecision + \text{Re}\,call} \qquad \text{Equation (3.20)}$$

Table 3.10 Confusion Matrix

		Predict Class	
		Class Success	**Class Failure**
Actual	Class = Success	TP	FN
Class	Class = Failure	FP	TN

3.15 Tools, Language and Materials

The experiments in this research were conducted with Python as the programming language. Our Python version is 3.7.0 with details shown in our output below:

3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)]

We used Python because of its readability, vast community, and rich ecosystem of libraries that make it serve as the foundation for data analysis, feature engineering, modelling, and visualization required in this project. The use of Python 3.7 ensures compatibility with modern libraries and features.

The key libraries we used are:

1. **Core Data Science Libraries:**

a. **Pandas (1.5.3):** Provided Data Frames and a wide range of tools for data loading, cleaning, manipulation, transformation, and analysis.

b. **NumPy (1.24.1):** Forms the basis for all numerical computing, multi-dimensional arrays, and mathematical functions used in our experiments.

c. **SciPy (1.9.3):** Builds upon NumPy, offering advanced tools for scientific computing, including optimization, linear algebra, and statistical analysis.

2. **Visualization Libraries:**

a. **Matplotlib (3.7.0):** The cornerstone of static plotting in Python. We used it for creating various plots (line charts, scatter plots, histograms, etc.).

b. **Seaborn (0.12.2):** Provides statistically informative and aesthetically pleasing plots based on Matplotlib.

3. **Machine Learning Libraries:**

a. **Scikit-learn (1.2.0):** A comprehensive library which we used for:

i. **Preprocessing tools:** For data scaling, normalization, and encoding.

ii. **Model evaluation metrics:** To assess model performance.

4. **Feature Selection Libraries:**

a. **Scikit-learn's feature selection module:** Provided the PCC method.

b. **MLxtend (0.21.0):** Offers Forward Feature Selection (FOR) implementation.

c. **pyswarms (1.4.0):** Implemented the Particle Swarm Optimization (PSO) for feature selection..

For code editing and version control, we used VisualStudio Code. We employed Visual Studio Code because of its versatility and robust integration with the Python RE. It also afforded us a streamlined workflow and robust version control.

We used Microsoft Excel for Microsoft Excel initial data exploration and cleaning. We also used it to create some simple visualizations and tables.

Our primary computing environment was on a local Windows-based system. However, we ran codes simultaneously on Google Cloud to ensure there were no errors from computing limitations.

Endnotes

[1]A. Al-Zawqari, D. Peumans & G. Vandersteen, "*A flexible feature selection approach for predicting the students' academic performance in online courses*," **Computers and Education: Artificial Intelligence** 3 2022 100103.

[2]https://www.academia.edu/72282730/Dataset_of_Students_Performance_Using_Student_Information_System_Moodle_and_the_Mobile_Application_eDify_

[3]https://zenodo.org/record/5591907

[4] https://www.kaggle.com/datasets/aljarah/xAPI-Edu-Data/data

[5]https://www.kaggle.com/datasets/devansodariya/student-performance-data

[6]https://archive.ics.uci.edu/dataset/320/student+performance

[7]S.Verma, R. K. Yadav & K. Kholiya," *Prediction of Academic Performance of Engineering Students using Data Mining Techniques*," **International Journal of Information and Education Technology**, vol 12, no.11, November 2022.

[8]I. Peraic & A. Grubisic, "*Predicting Academic Performance of Students in a Computer Programming Course using Data Mining,*" **International Journal of Engineering Education**, July 2023

.[9]S.Palmar, "*Modelling Engineering Student Academic Performance using Academic Analytics,* " **International Journal of Engineering Educatio***n*. Vol. 29 No. 1, pp 132- 138, 2013.

[10]M. Pokharel, " *Educational data mining in Moodle data,"* **International Journal of Informatics and Communication Technology (IJICT***)* Vol. 10, No.1, pp.9 – 18, 2021

[11]R. Hasan, S. Palaniappan, S. Mahmood, A. Abbas & K. U. Sarker, " *Dataset of Students' Performance Using Student Information System, Moodle and the Mobile Application "eDify"* " **Data** 6, no.110 2021,

[12]M. Jain , V. Saihjpal, N. Singh & S. B. Singh,"*An Overview of Variants and Advancements of PSO Algorithm*" **Applied Science**, 2022, 12 (17) 8392.

[13]A. A. Firdaus, O. Penangsang, A. Soeprijanto,& D. Uman, "*Distribution Network Reconfiguration Using Binary Particle Swarm Optimization to Minimize Losses and Decre ase Voltage Stability Index*" **Bulletin of Electrical Engineering and Informatics** 7, No. 4, December 2018: 514 – 521.

[14]T. Hodson, "*Root-mean-square error(RMSE) or mean absolute error(MAE). When to use them or not*" **Geoscientific Model Development**: 2020

[15]C. Anisha & N. Arulanand, "*Tuned Homogenous Ensemble Regressor Model for Early Diagnosis of Parkinson Disorder Based on Voice Features Modality*" **Journal of Artificial Intelligence and Capsule Networks** September 2022.No. 3 pp. 188 – 199.

[16]D. M. A. Ali, Y. M. Mohialde & N. M. Hussien, "*Use of a Gradient Boosting Algorithm to Accurately Predict Solutions*" **Scientific Journal of Engineering and Computer Science** 3, issue 4 July, 2023. ISSN 2788-9394 (print)

Chapter Four

Results and Discussion of Findings

4.1 Introduction

This chapter discusses the results and evaluation outcomes of the experiments carried out iteratively and comparatively to assess and compare feature selection methods for predicinting students' academic performance in a Virtual Learning Environment.

4.2 Results and Discussion

4.2.1 Descriptive Visualisation Results

Below are some descriptive visualization results after the Data Pre-processing of Dataset 1

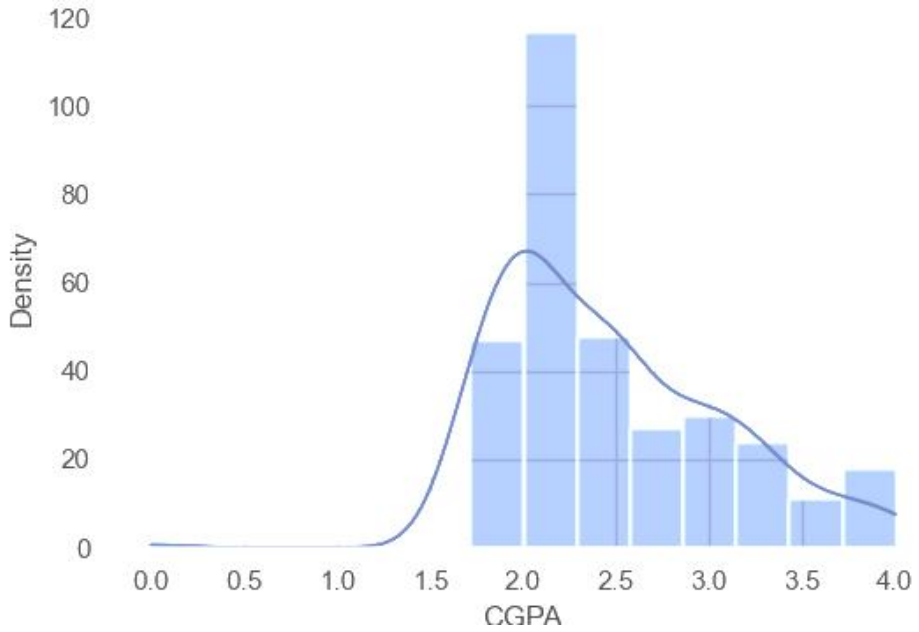


Figure 4.1: Distribution of CGPA (Source: Researcher, Adelodun, F. O. 2024)

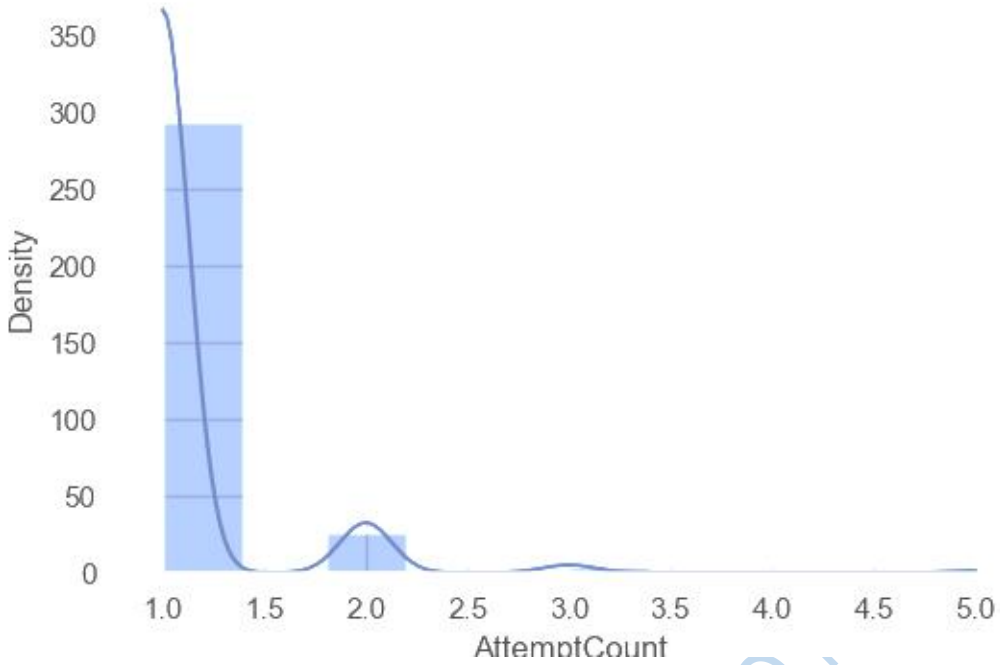


Figure 4.2: Distribution of AttemptCount (Source: Researcher, Adelodun, F.O. 2024)

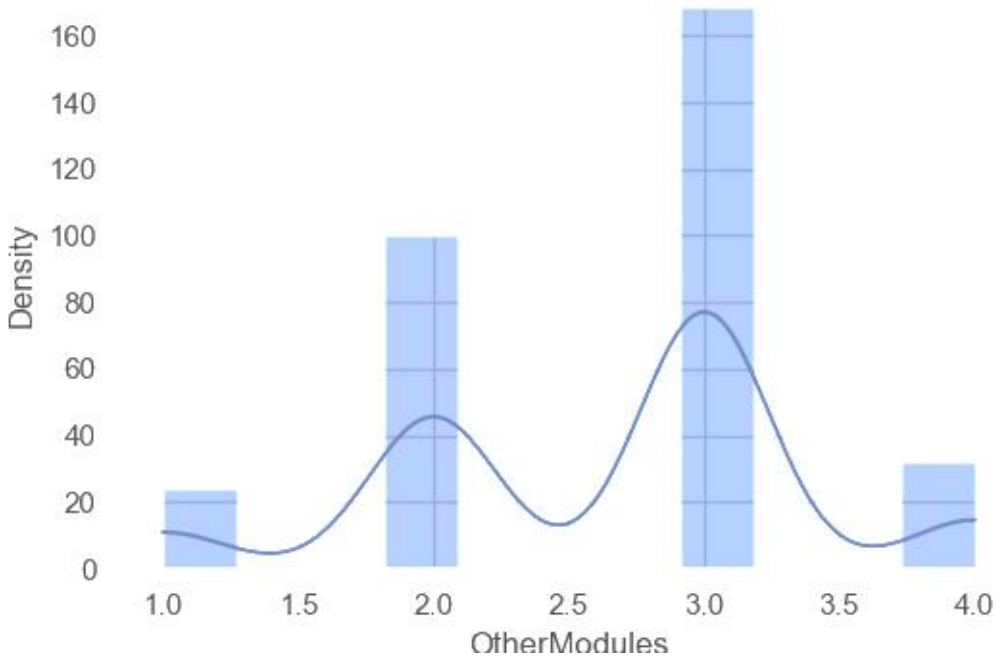


Figure 4.3: Distribution of OtherModules

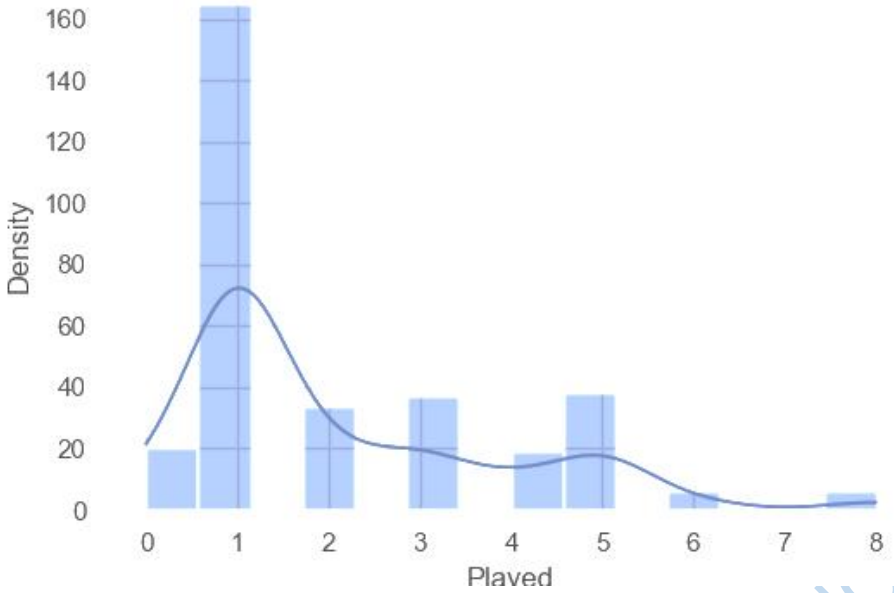


Figure 4.4: Distribution of Played

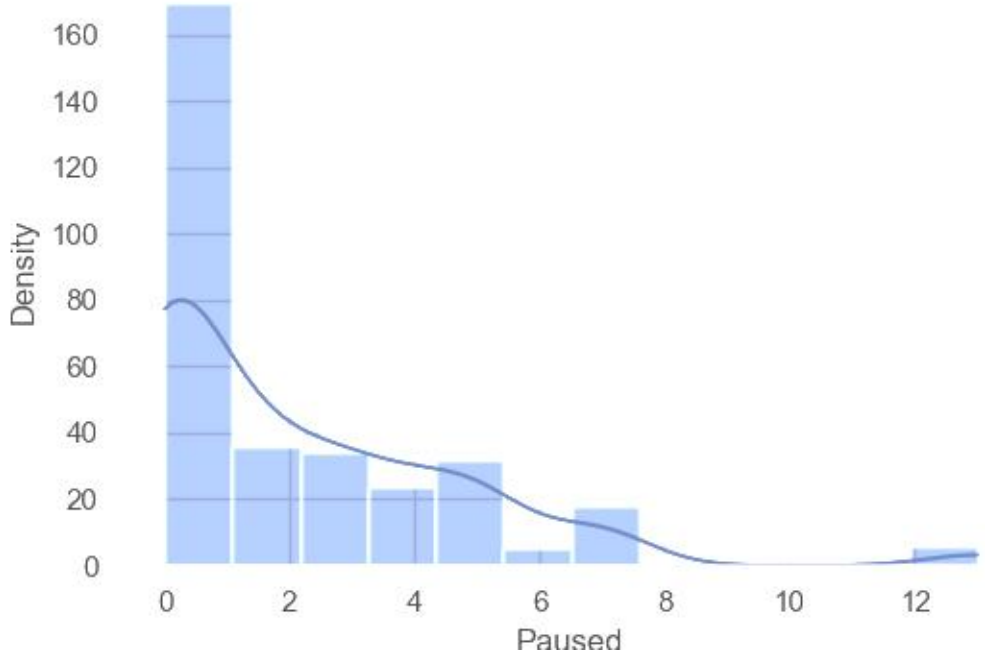


Figure 4.5: Distribution of Paused

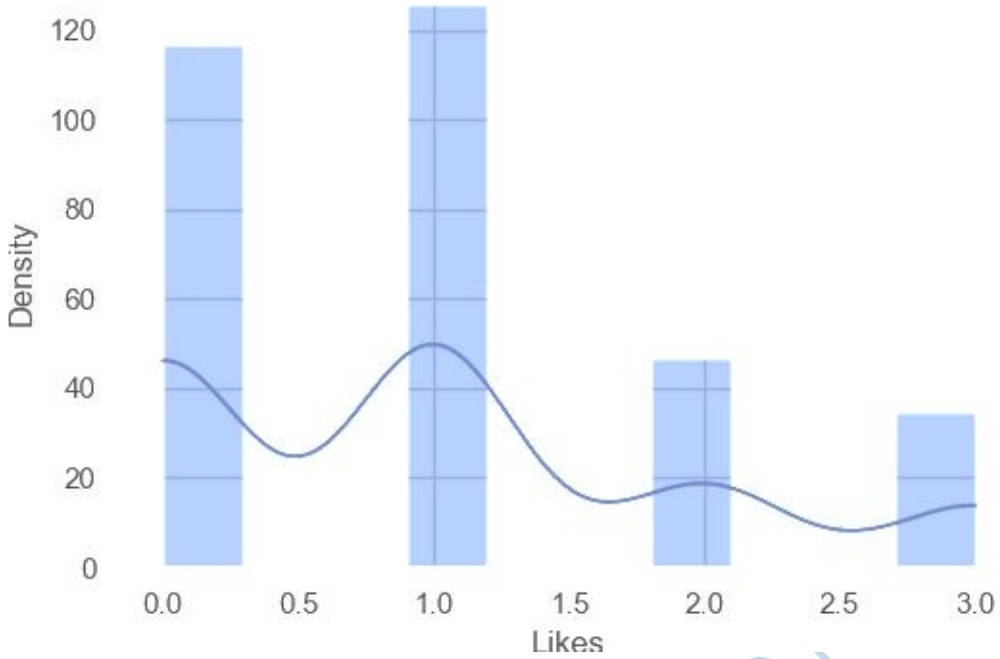


Figure 4.6: Distribution of Likes

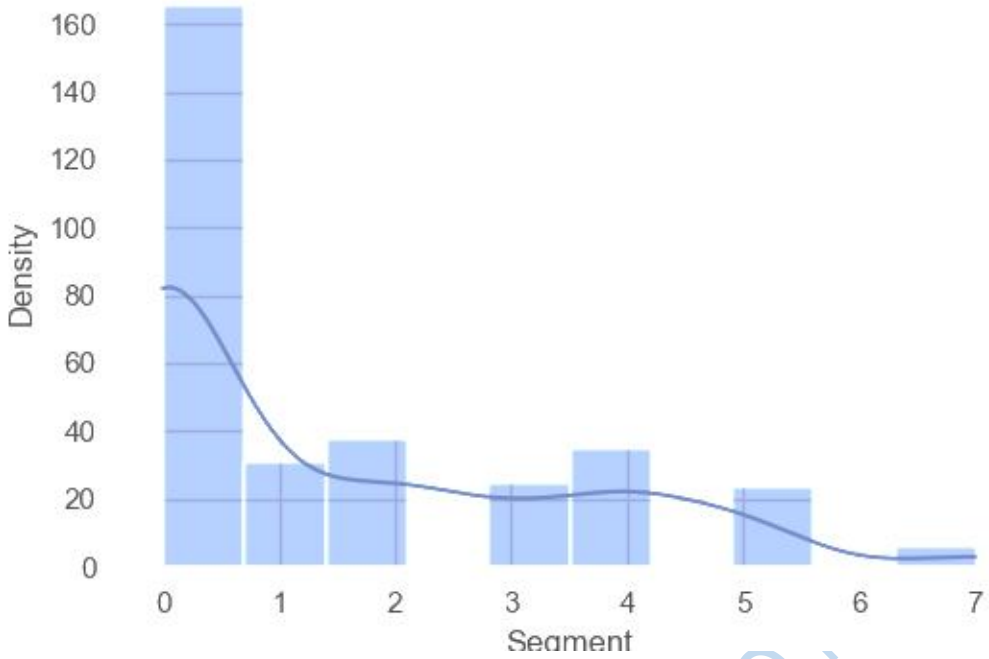


Figure 4.7: Distribution of Segment

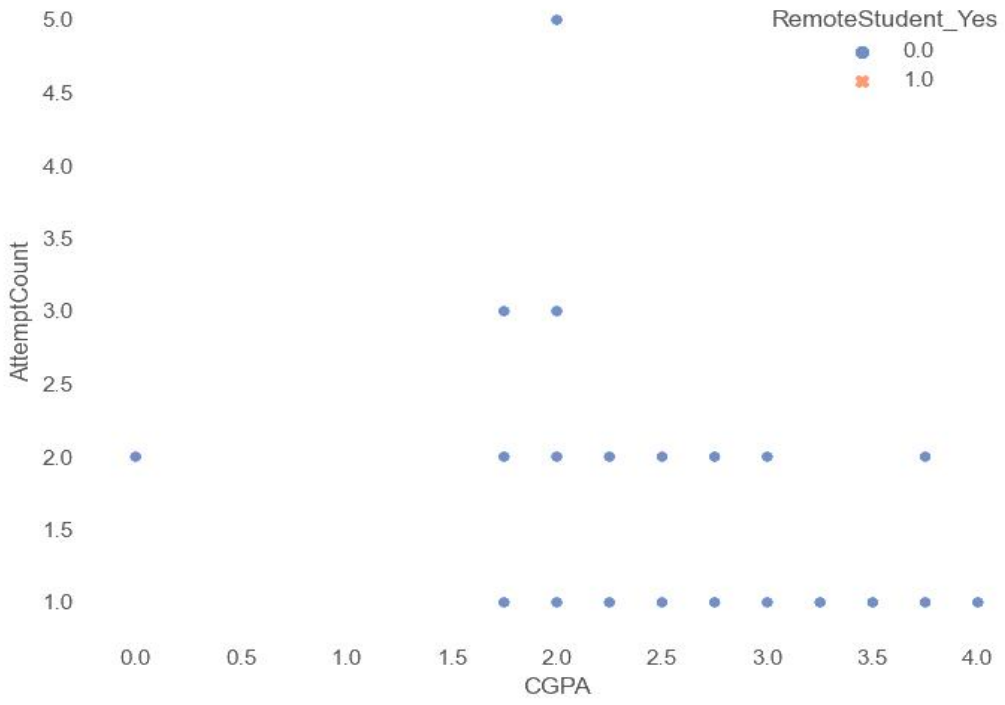


Figure 4.8: Relationship between CGPA and AttemptCount (Coloured by remote)

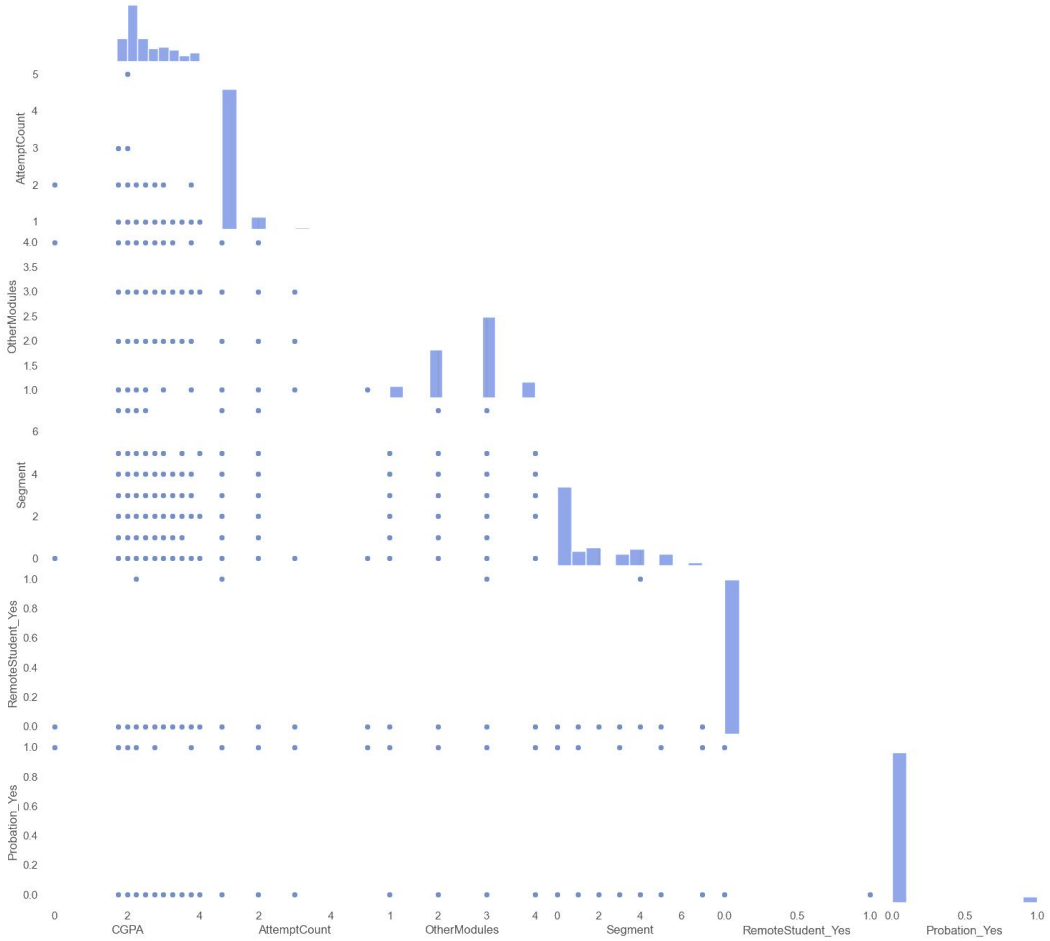


Figure 4.9: Relationship between features

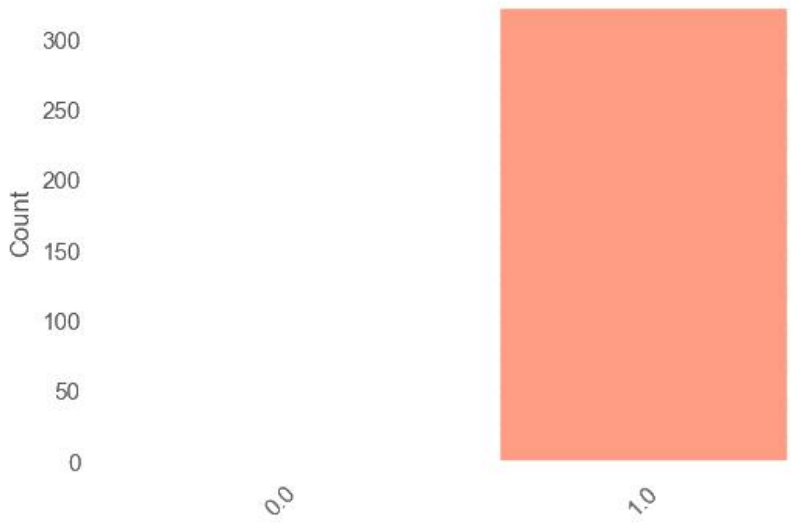


Figure 4.10: Distribution of RemoteStudent_No

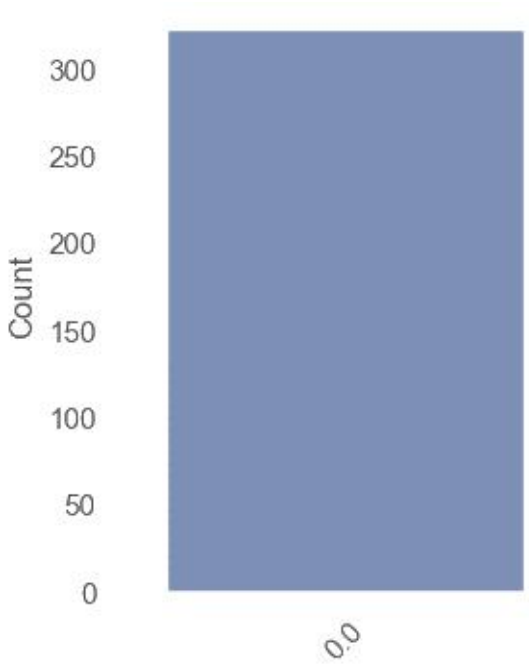


Figure 4.11: Distribution of RemoteStudent_Yes

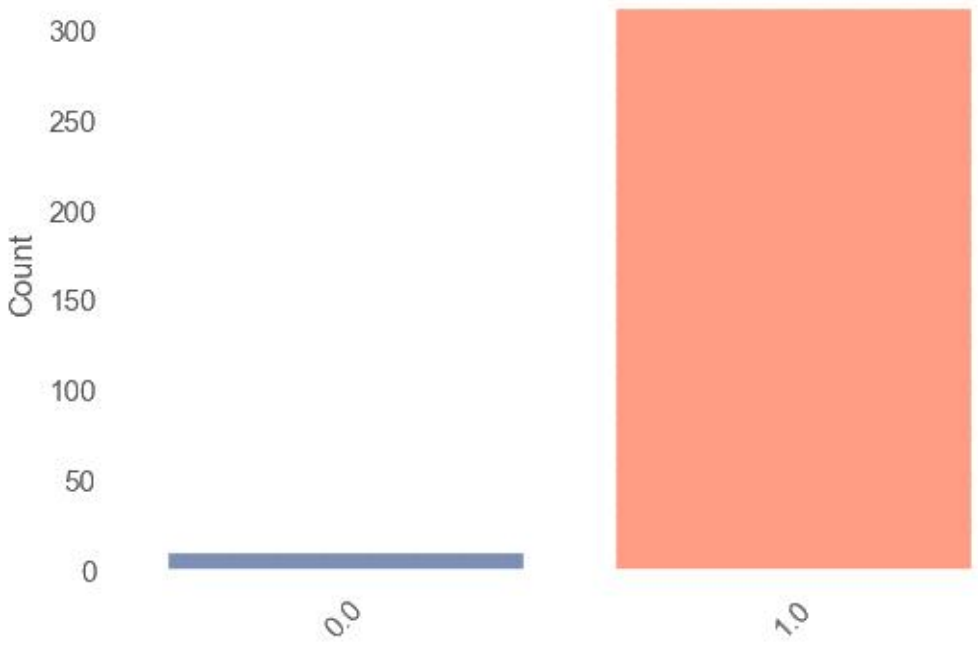


Figure 4.12: Distribution of Probation_No

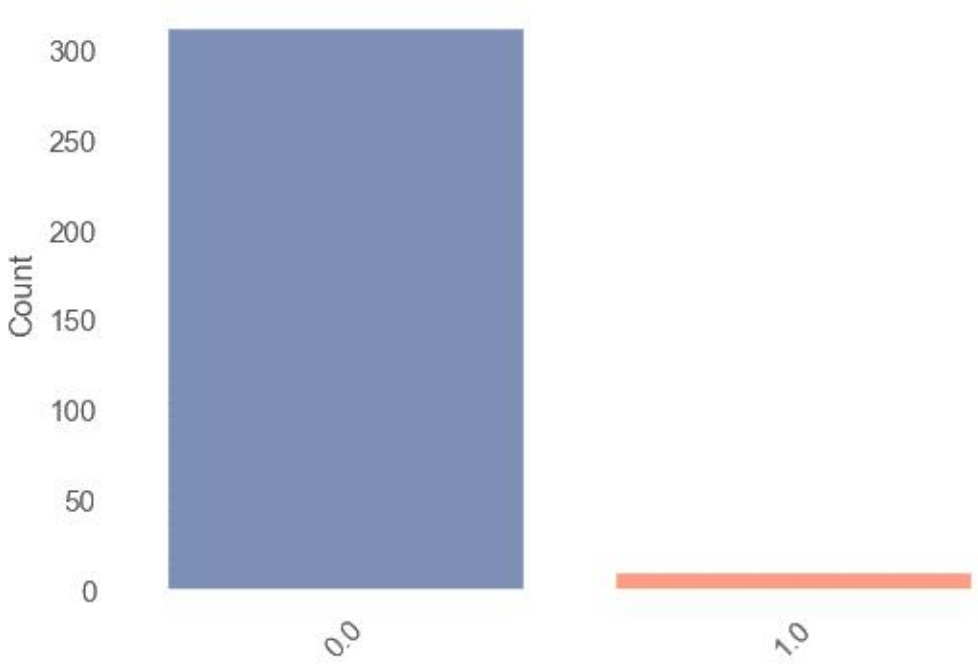


Figure 4.13: Distribution ofProbation_Yes

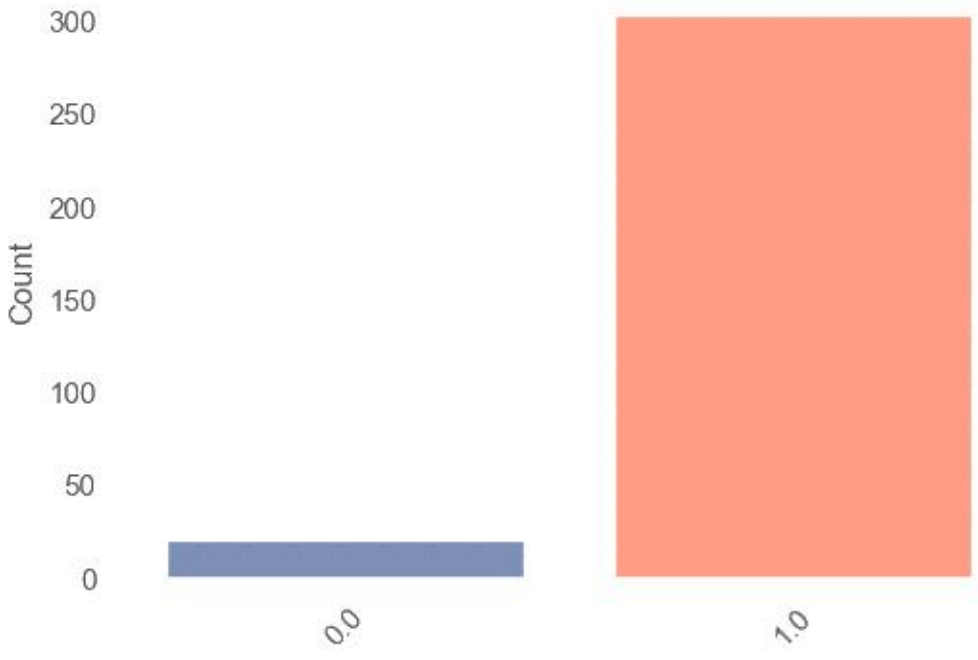


Figure 4.14: Distribution of HighRisk_No

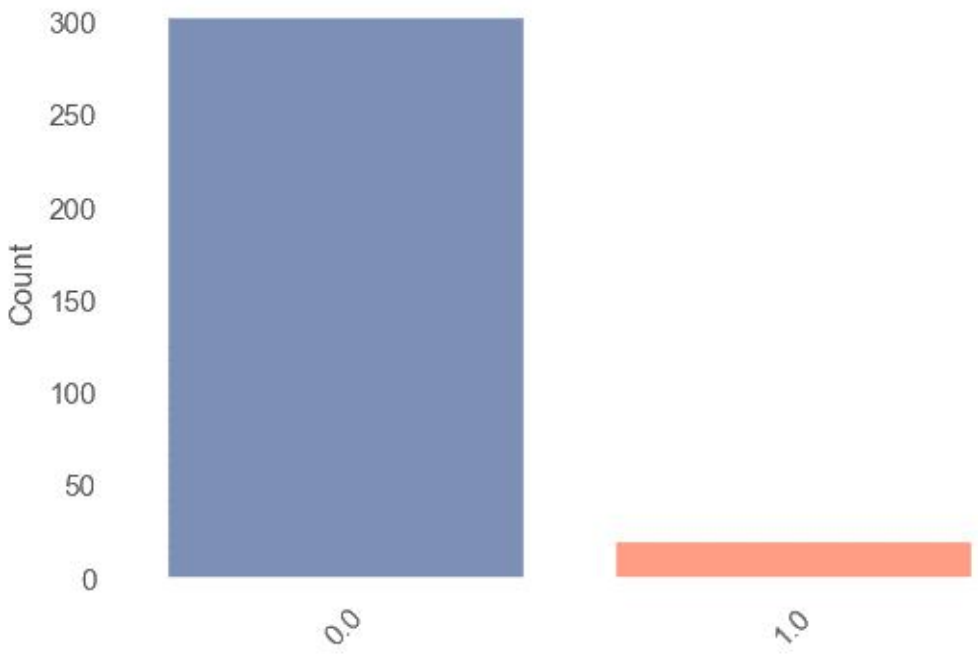


Figure 4.15: Distribution of HighRisk_Yes

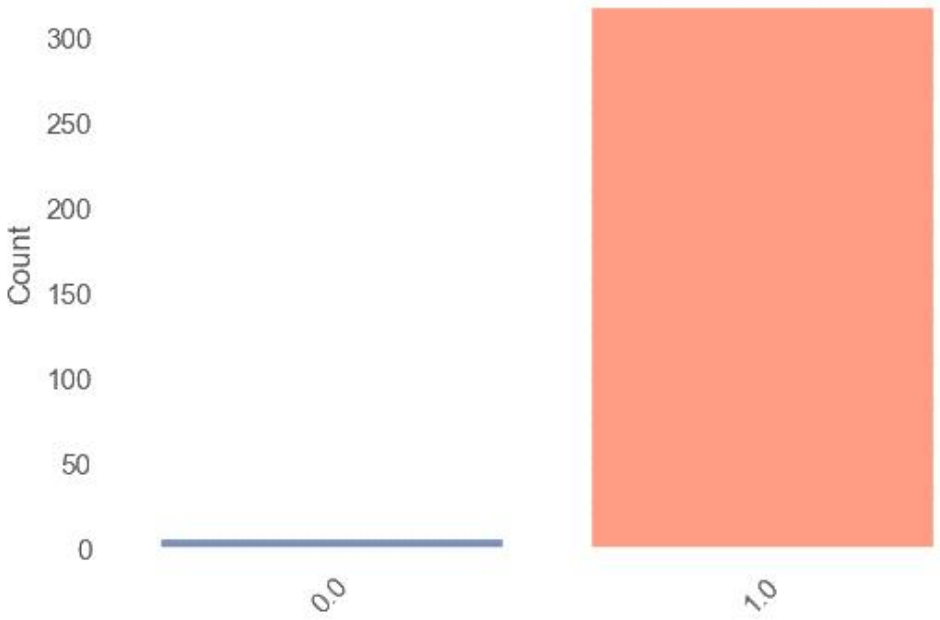


Figure 4.16: Distribution of TermExceeded_No

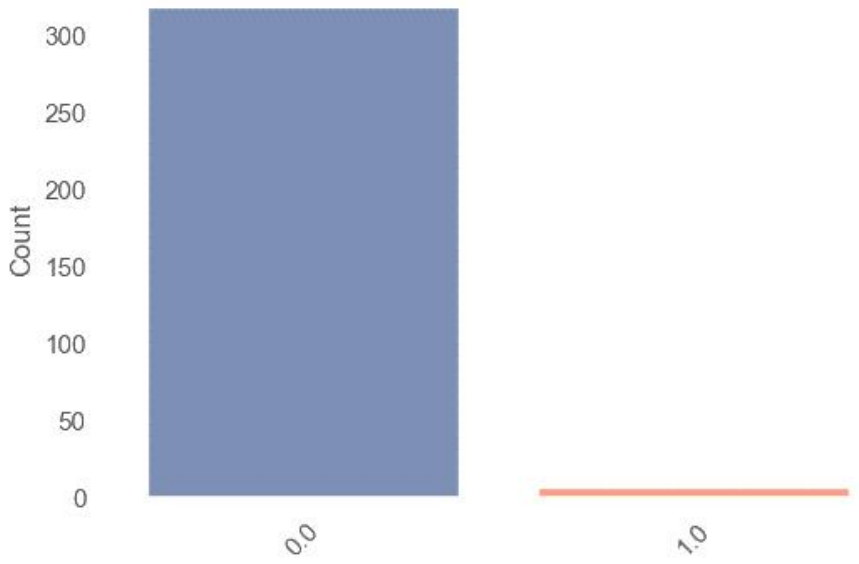


Figure 4.17: Distribution of TermExceeded_Yes

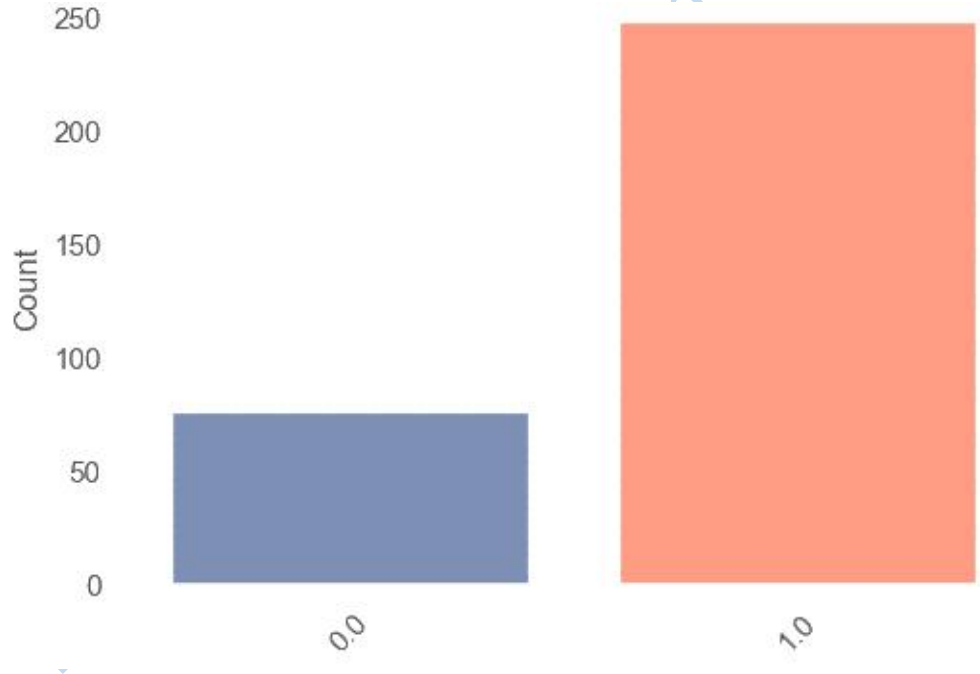


Figure 4.18: Distribution of ArtRisk_No

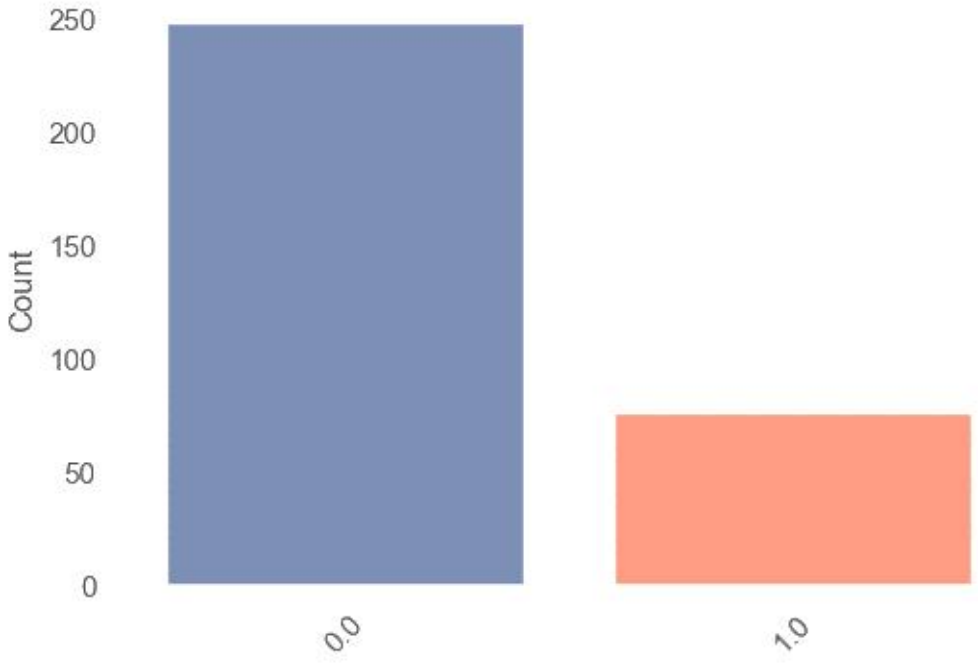


Figure 4.19: Distribution of ArtRisk_Yes

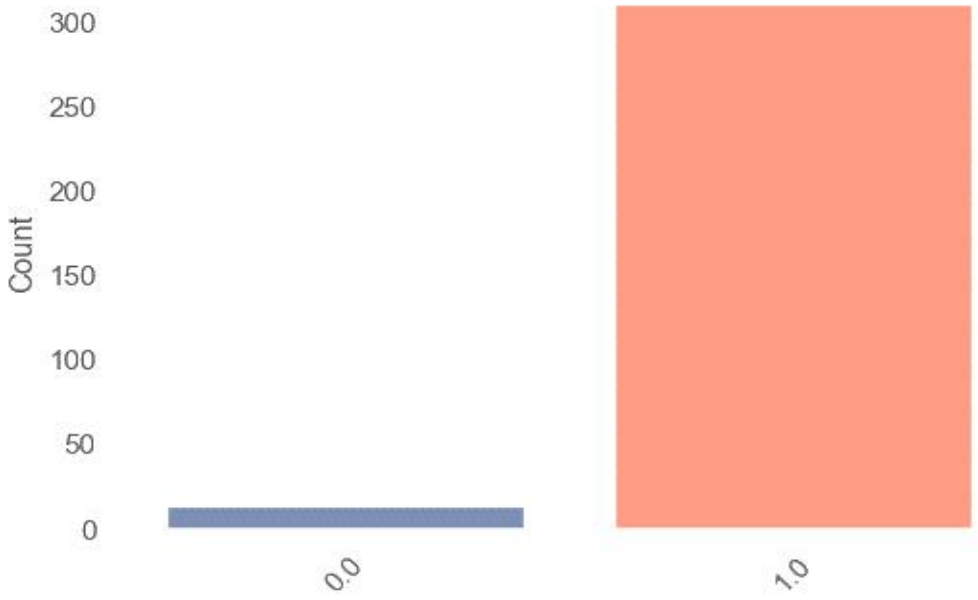


Figure 4.20: Distribution of ArtRiskSSC_No

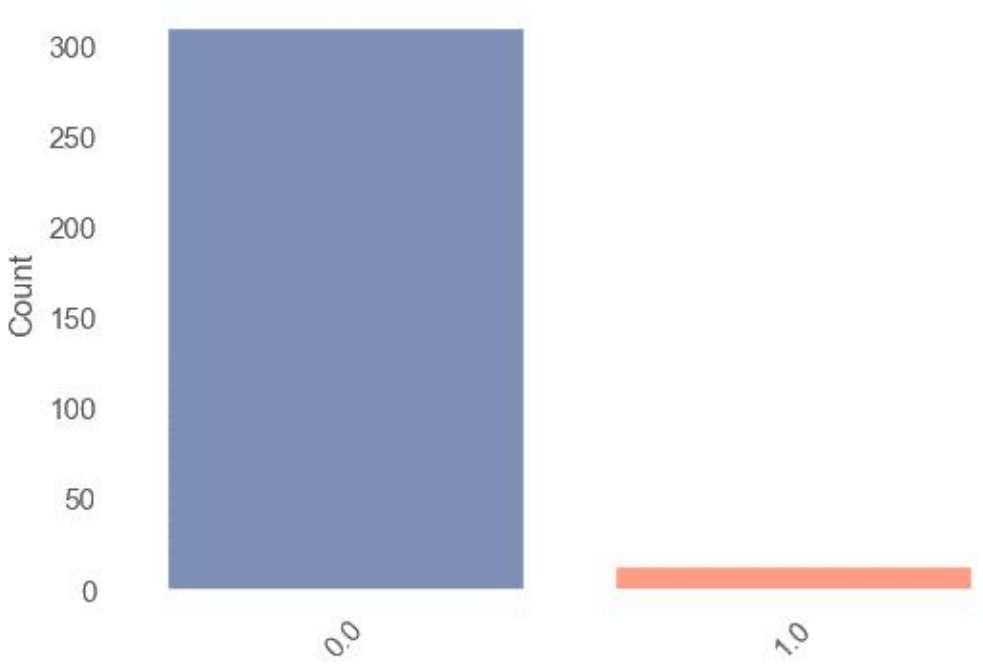


Figure 4.21: Distribution of ArtRiskSSC_Yes

Below are some descriptive visualization results of Dataset 2.

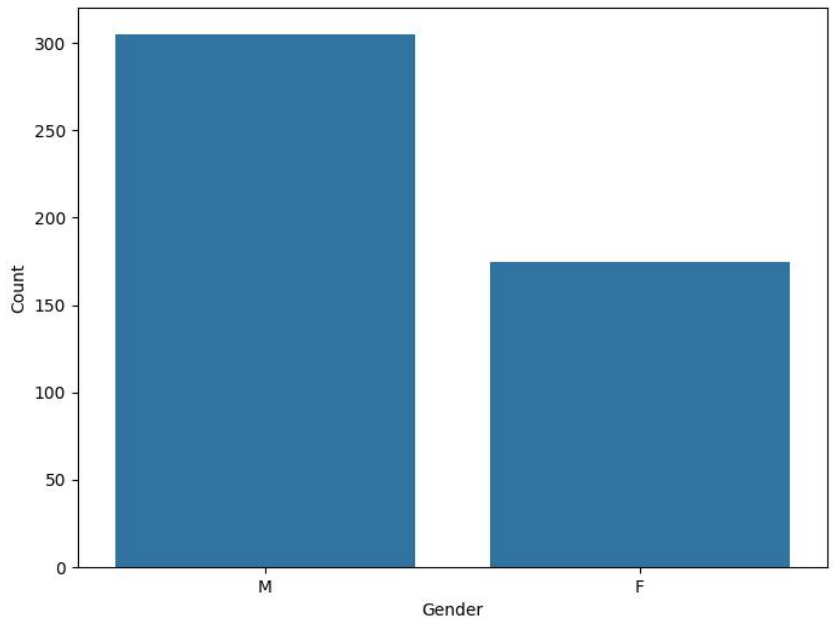


Figure 4.22: Gender Distribution of Dataset 2

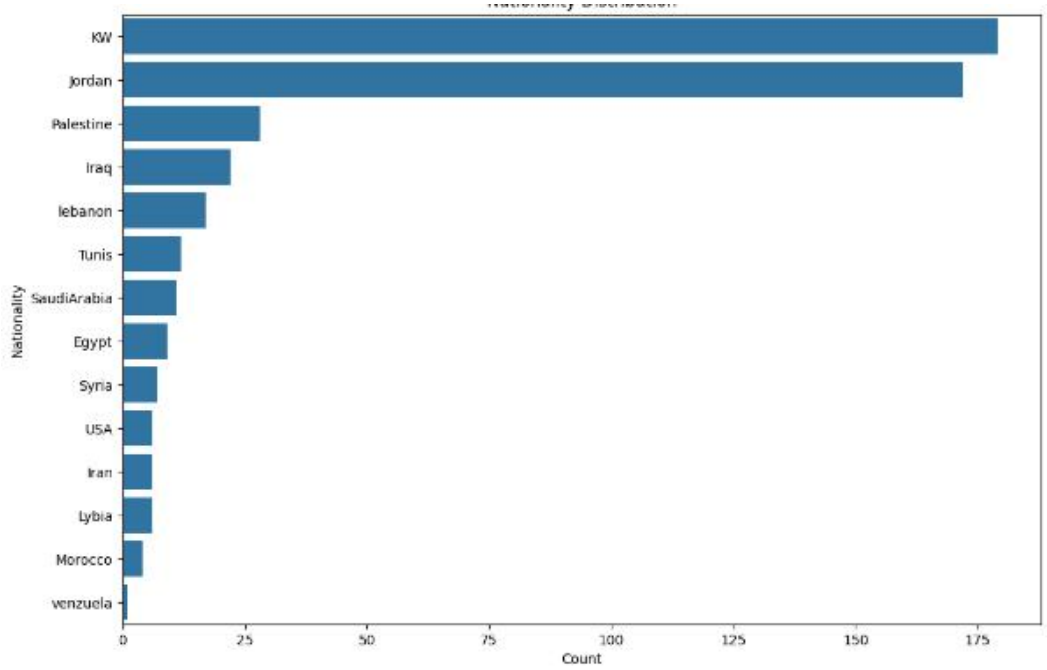


Figure 4.23: Nationality Distribution of Dataset 2

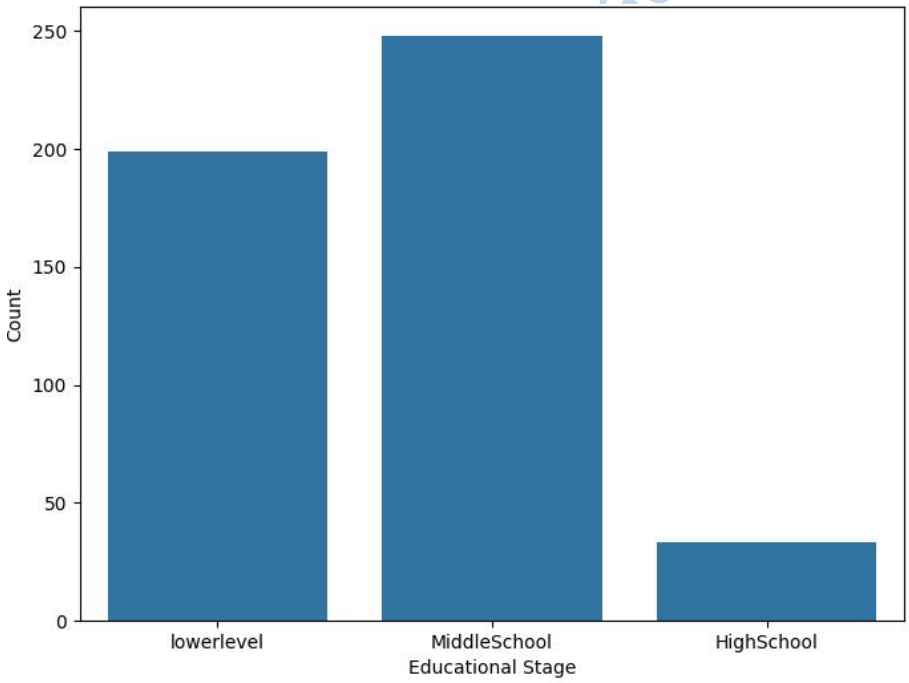


Figure 4.24: Educational Stages Distribution of Dataset 2

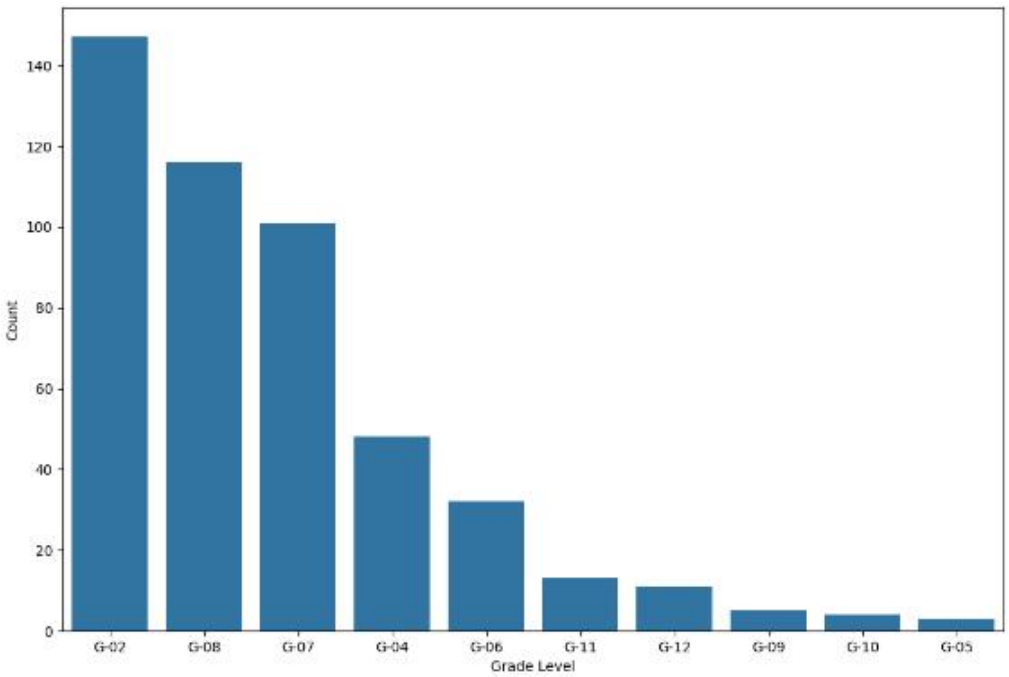


Figure 4.25: Grade Levels Distribution of Dataset 2

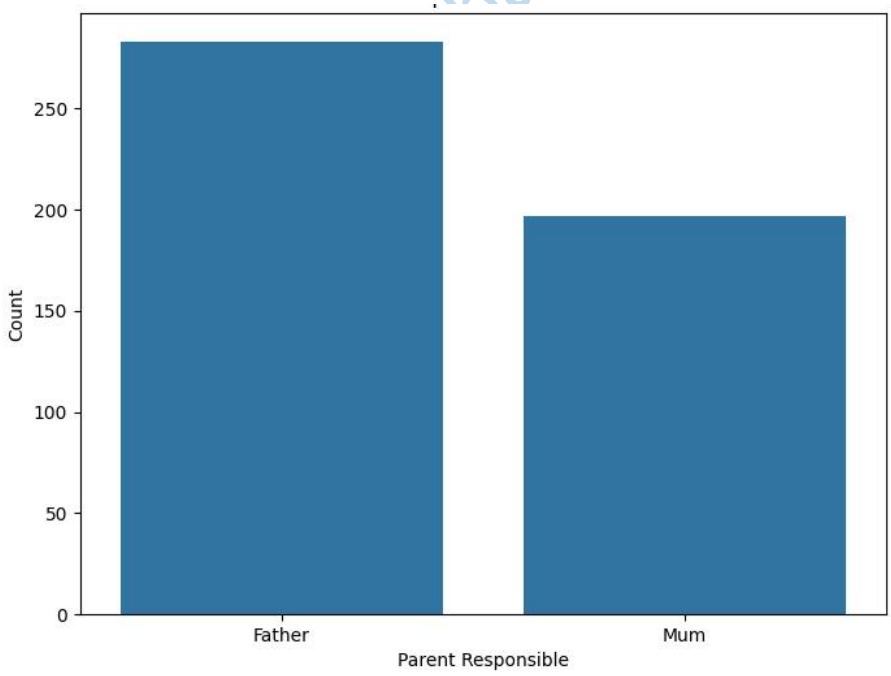


Figure 4.26: Parent Responsible for Student of Dataset 2

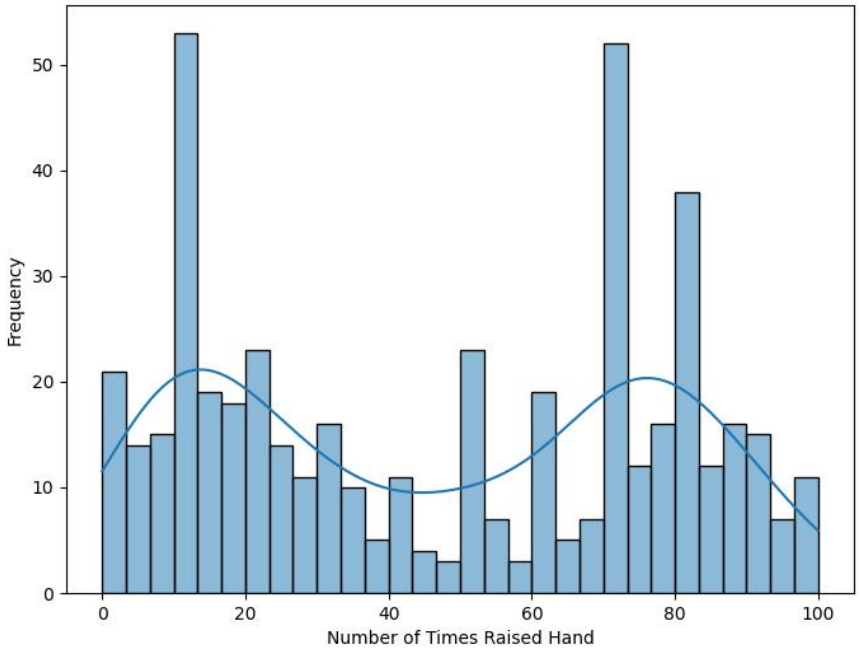


Figure 4.27: Distribution of Raised Hands of Dataset 2

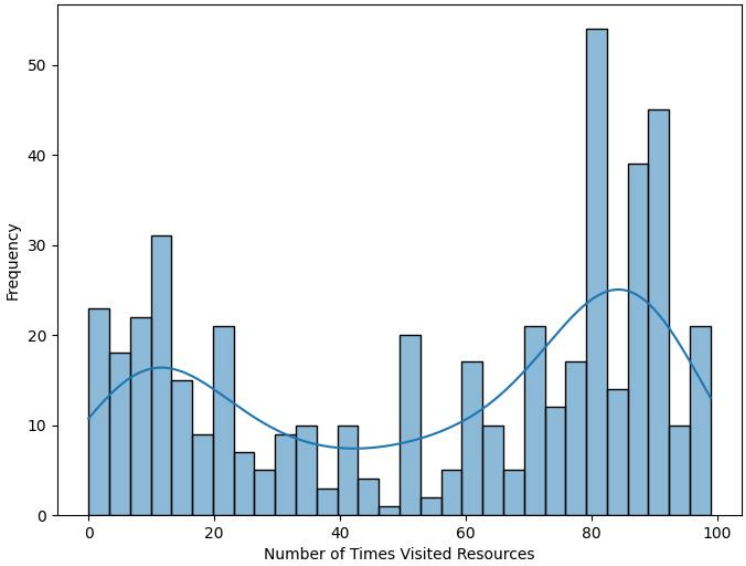


Figure 4.28: Distribution of Visited Resources of Dataset 2

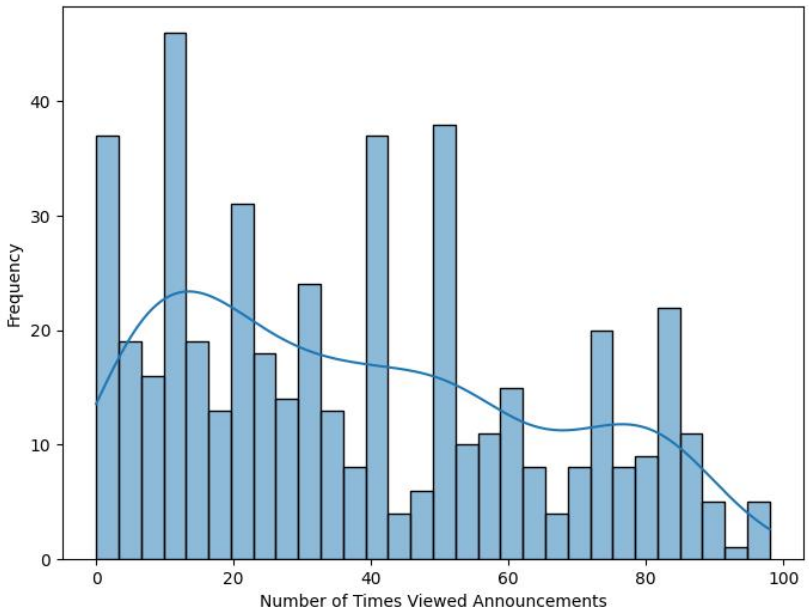


Figure 4.29: Distribution of Viewing Announcements of Dataset 2

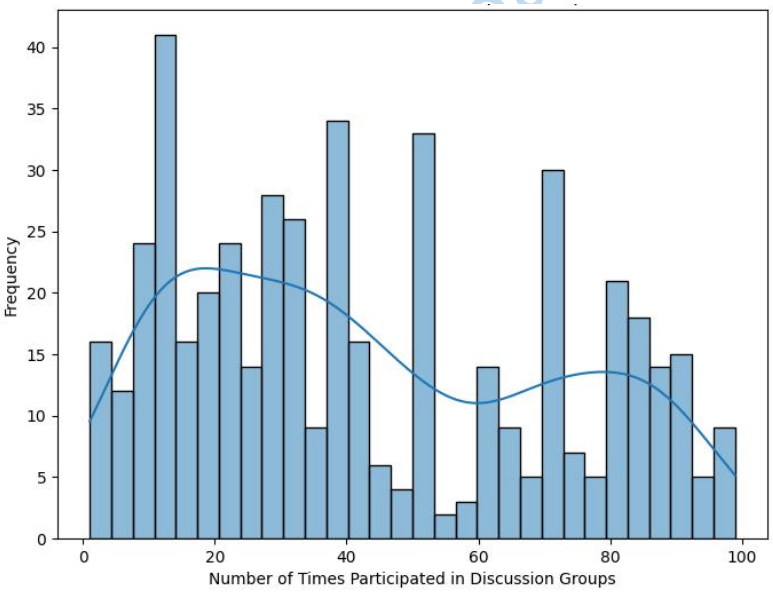


Figure 4.30: Distribution of Discussion Groups Participation of Dataset 2

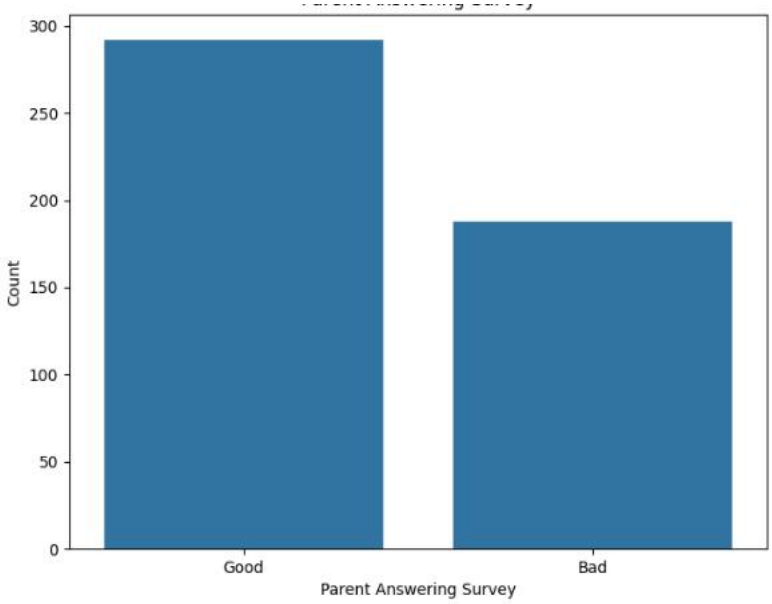


Figure 4.31: Distribution of Answering Survey of Dataset 2

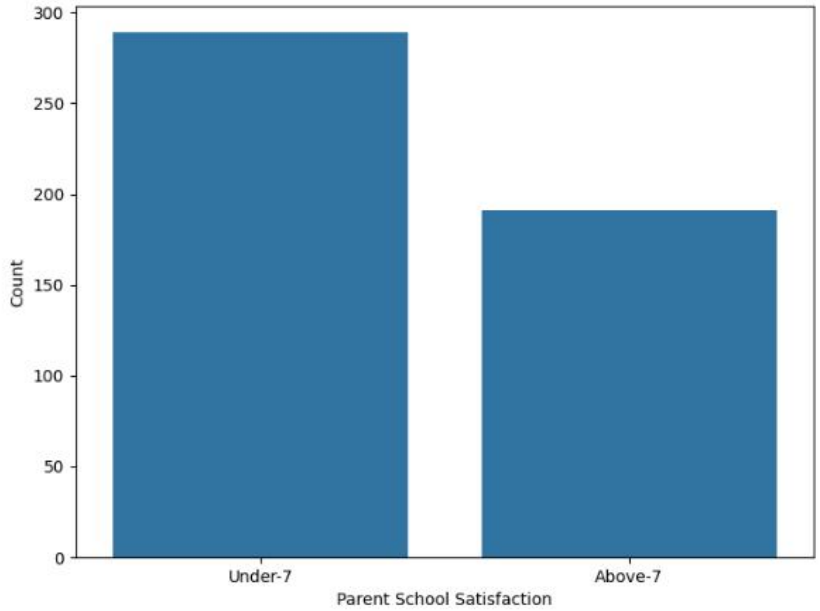


Figure 4.32: Parent School Satisfaction of Dataset 2

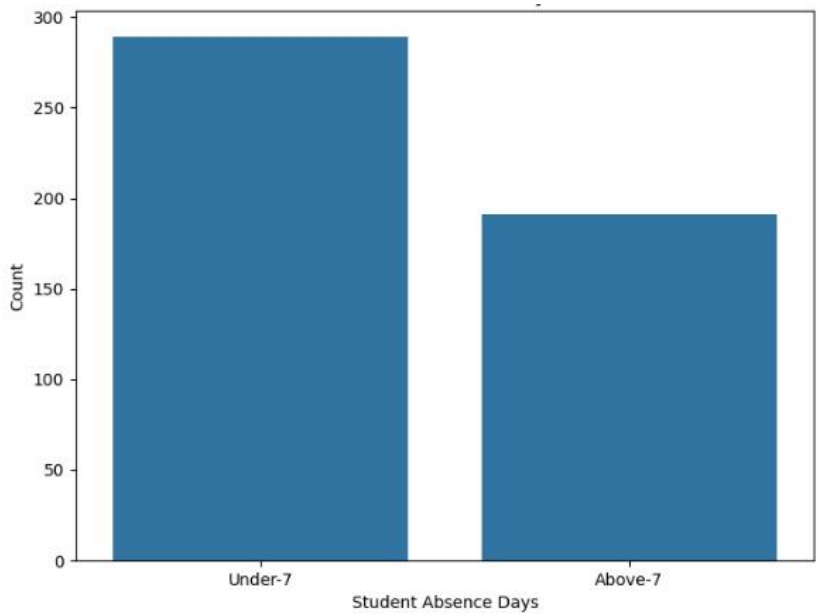


Figure 4.33: Student Absence Days of Dataset 2

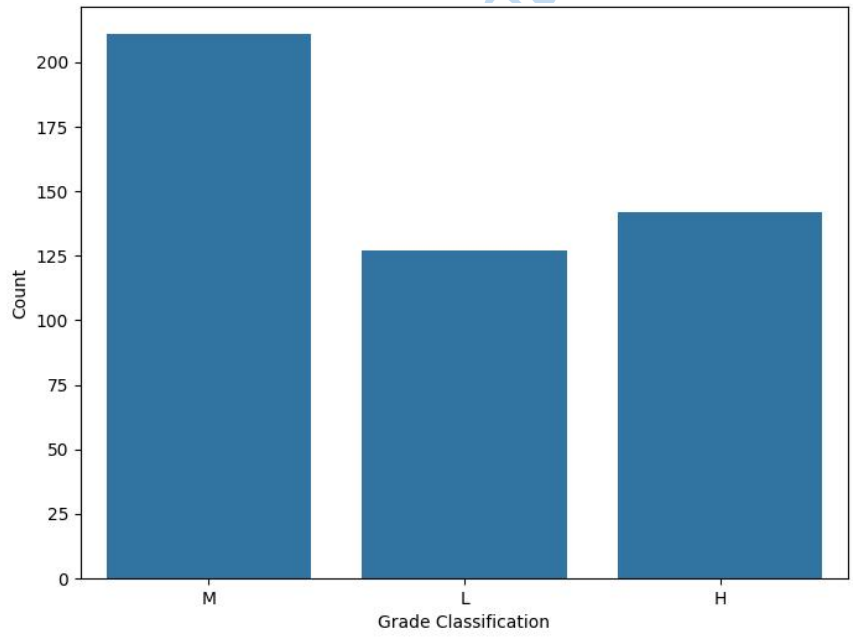


Figure 4.34: Grade Classification of Dataset 2

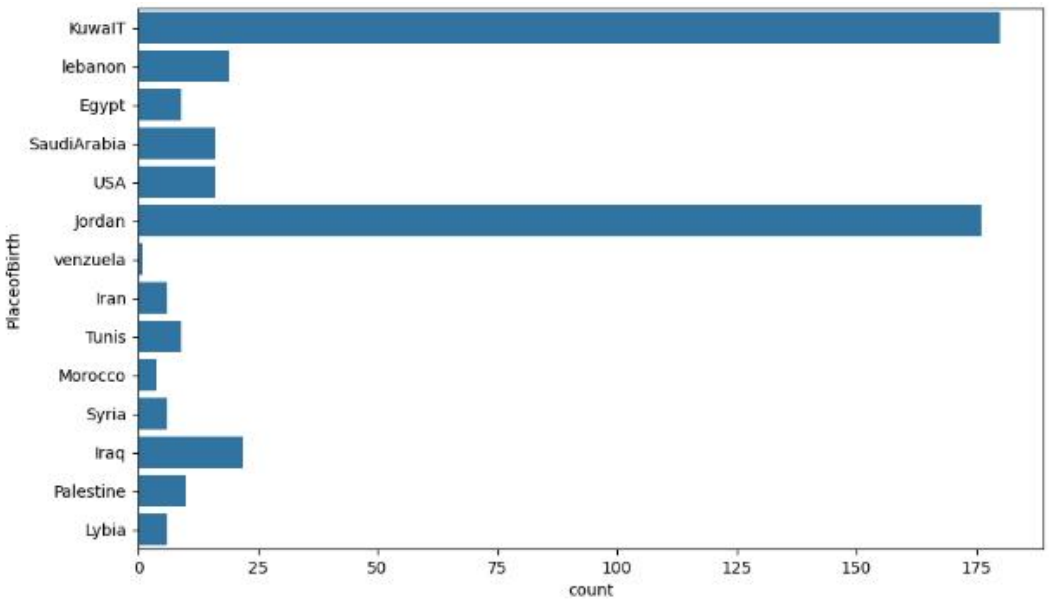


Figure 4.35: Distribution of Place of Birth of Dataset 2

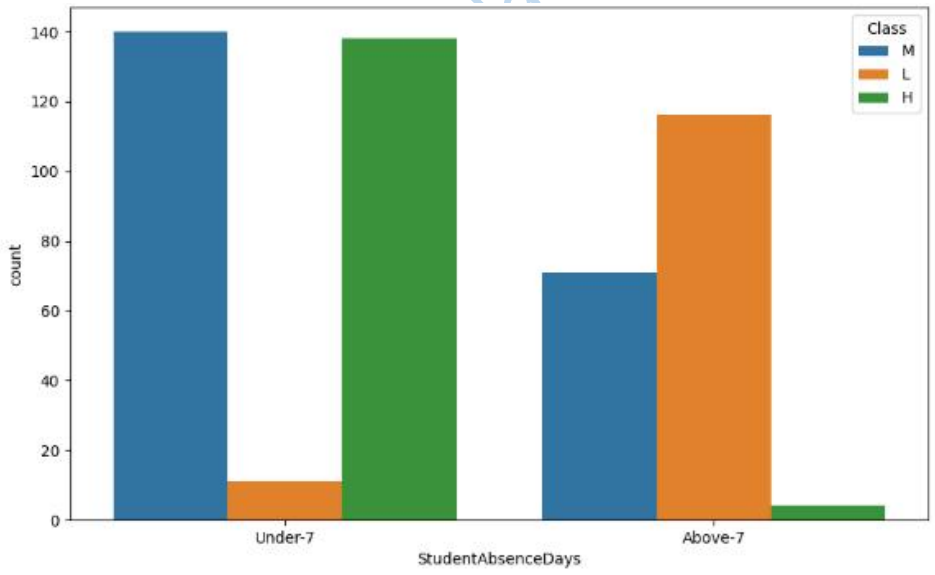


Figure 4.36: Count plot of StudentAbsenceDays grouped by Class of Dataset 2

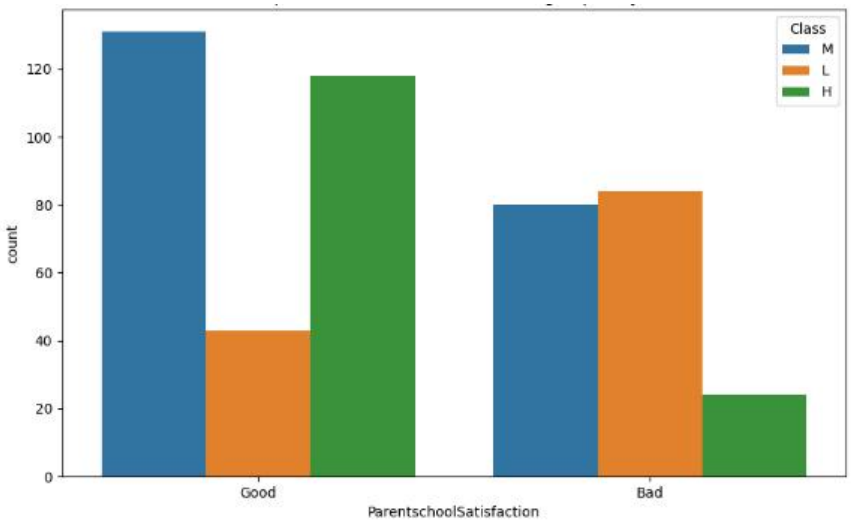


Figure 4.37: Count plot of ParentschoolSatisfacton grouped by Class of Dataset 2

Below are some descriptive visualization results after the Data Pre-processing of Dataset 3.

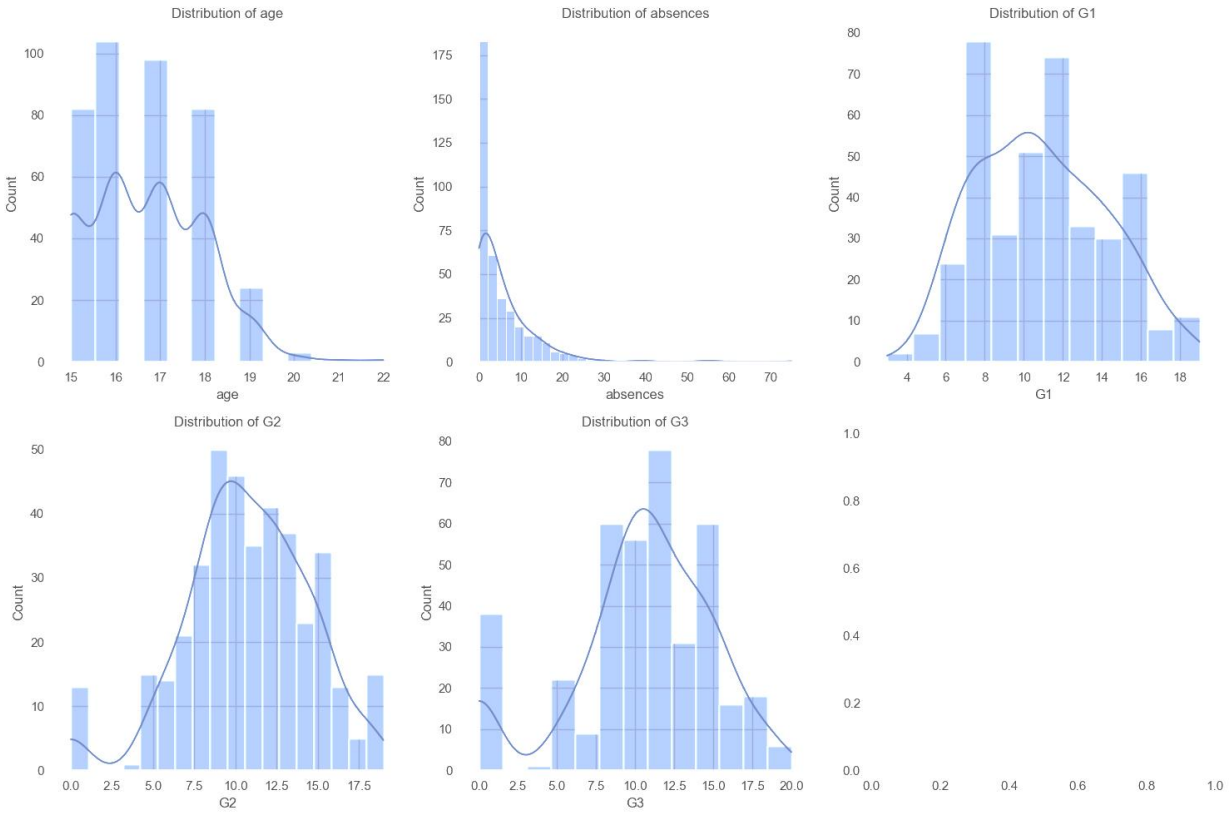


Figure 4.38: The Distributions of Numerical Features of Dataset 3

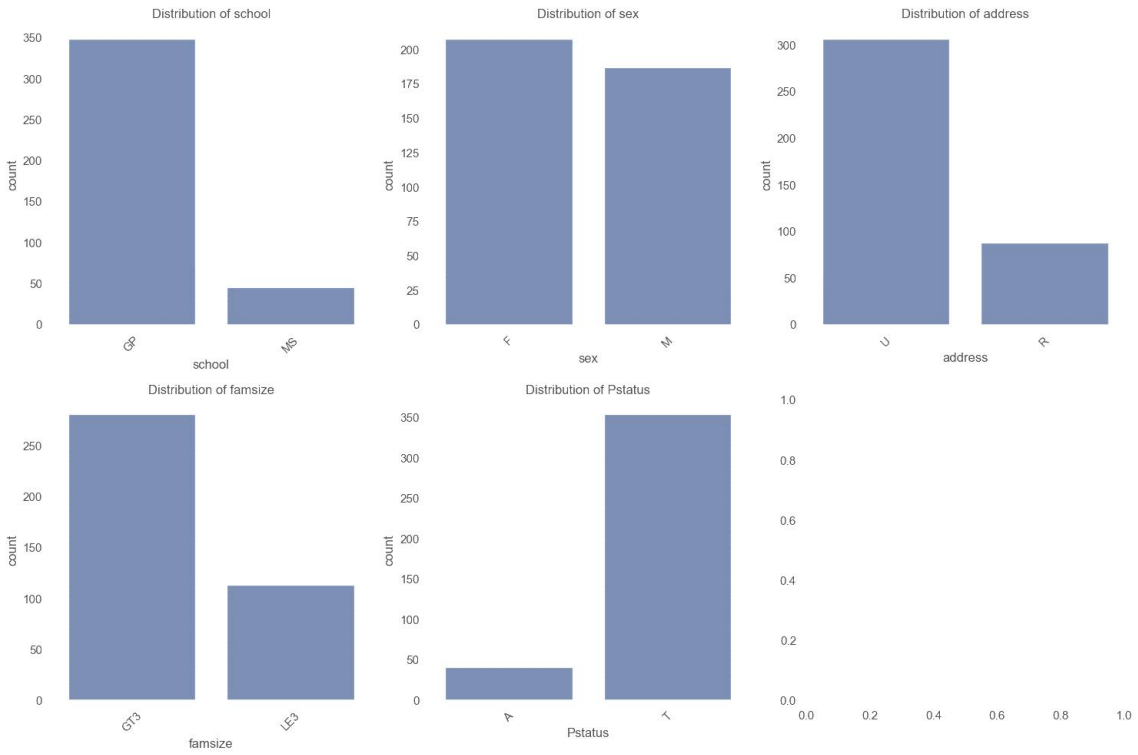


Figure 4.39: Categorical Features and their Distribution of Dataset 3

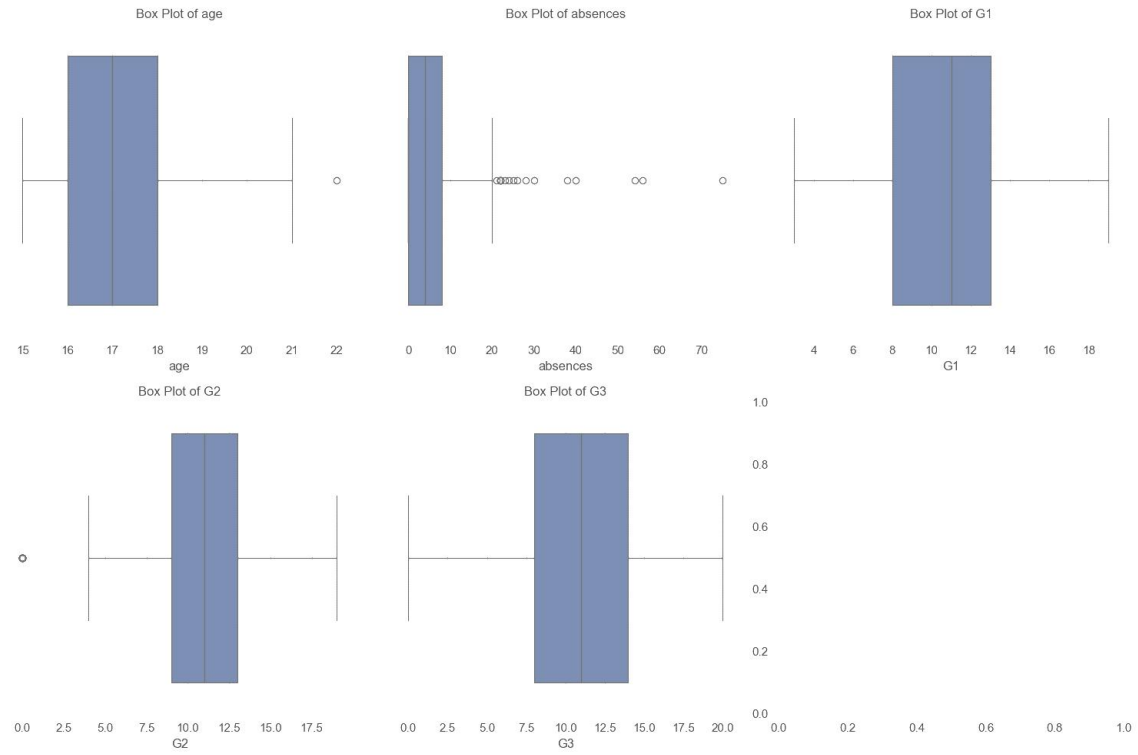


Figure 4.40: Numerical features highlight the presence of outliers of Dataset 3

4.2.2 Model Training and Evaluation Results

4.3 Dataset 1 Experiment Sets

Table 4.1 The Results of Experiment Set 1

		Average MSE	Average R-squared	Average MAE
1	Linear Regression	314.81	-0.09247	13.69697
2	Decision Tree	481.6768	-0.66606	14.91809
3	Random Forest	223.4295	0.229647	10.6209
4	SVR (RBF)	292.9208	-0.01037	13.01048
5	KNN	325.8456	-0.11548	13.62819
6	MLP	318.5454	-0.10272	13.78158
7	Gradient Boosting	251.7244	0.127525	11.25462
8	XGBoost	249.7095	0.133933	11.42567
R10	Extra Trees	242.7997	0.167313	11.11915
11	AdaBoost	279.7785	0.028615	13.25315
12	SVR (linear)	308.1571	-0.06394	13.47966
13	SVR (poly)	286.8212	0.010756	12.8311
14	Gaussian Process	1751.601	-5.25361	36.47341
15	KNN (tuned)	308.6294	-0.05458	13.37506
16	Bayesian Ridge	295.2953	-0.01847	13.18661
17	Ridge Regression	312.524	-0.08352	13.64762
18	Lasso Regression	293.3104	-0.00991	13.16934

19	ElasticNet Regression	294.4782	-0.01581	13.18315

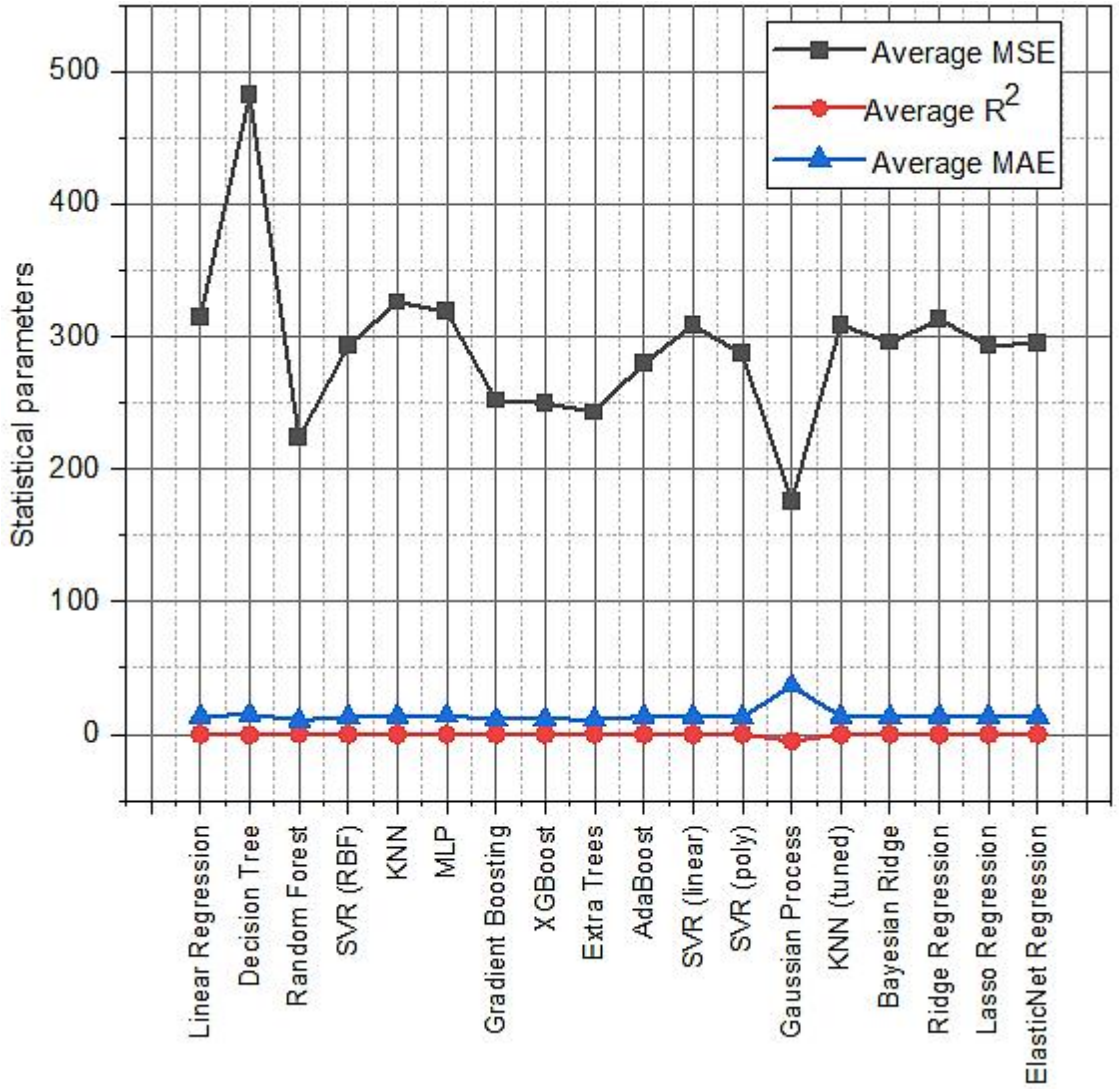


Figure 4.41: **Visual representation of Experiment Set 1 Result**

Experiment Set 2

The following is the result of running the PCC selector at a threshold greater than 0.6. It selected eight (8)which was used to train the machine learning models.

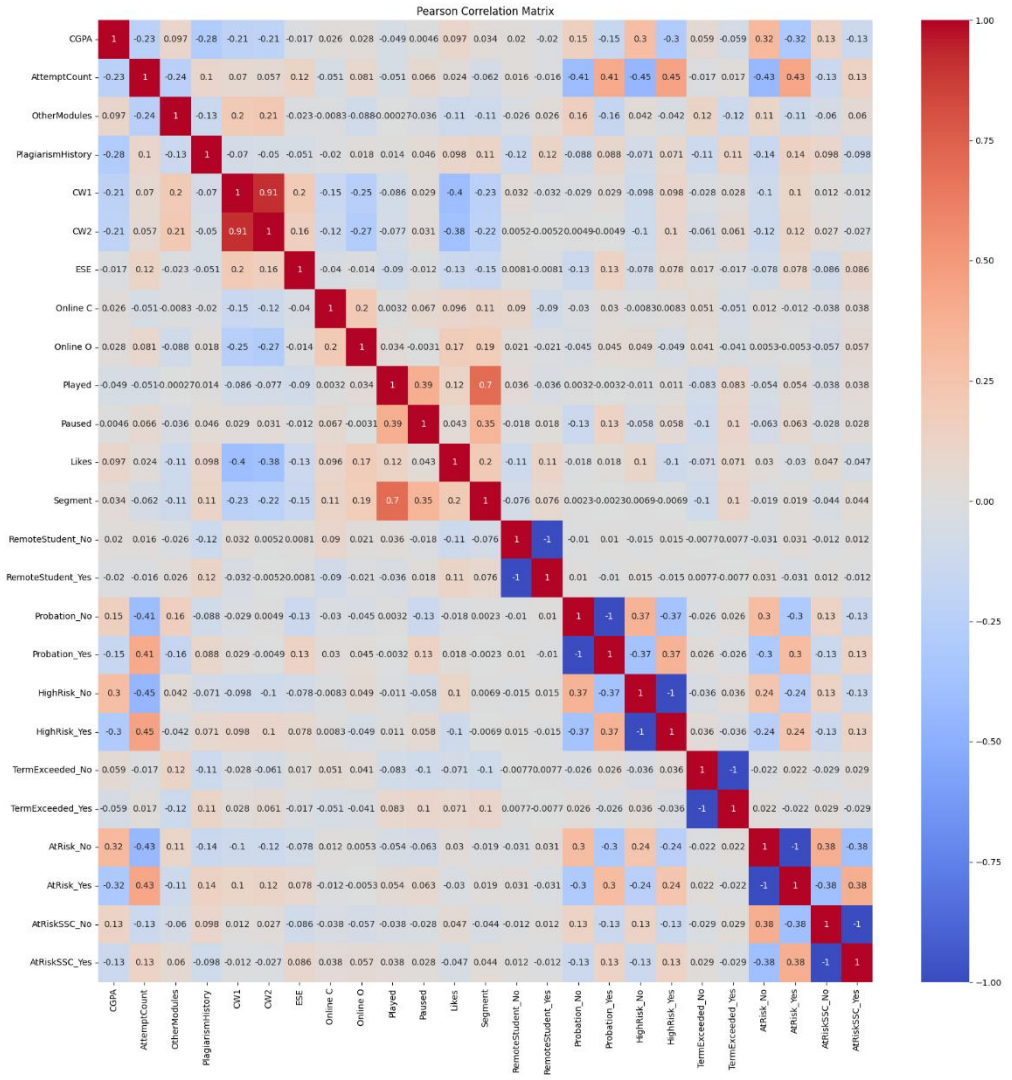


Figure 4.42: PCC Matrix of Experiment 2 (Dataset 2)

Table 4.2: The Result of Experiment Set 2

	Average MSE	Average R-squared	Average MAE
Linear Regression	302.2859	-0.04226	13.35342
Decision Tree	384.8053	-0.34693	14.05801
Random Forest	263.4042	0.087904	12.13148
SVR (RBF)	296.0745	-0.01857	12.99936
KNN	296.3355	-0.01491	13.2783
MLP	313.2623	-0.07168	13.59177
Gradient Boosting	264.8725	0.077715	12.23619
XGBoost	304.1845	-0.05046	12.63343
LightGBM	271.6999	0.066384	12.52942
Extra Trees	332.7279	-0.16231	13.633
AdaBoost	267.8909	0.072366	12.79653
SVR (linear)	294.2838	-0.00801	13.00514
SVR (poly)	288.4291	0.006913	12.95524
Gaussian Process	3877291	-14718.1	420.3828
KNN (tuned)	293.5627	0.001015	13.17142
Bayesian Ridge	294.5166	-0.01177	13.14262
Ridge Regression	299.7191	-0.03225	13.28294
Lasso Regression	296.7647	-0.02215	13.20578
ElasticNet Regression	296.1098	-0.02033	13.20679

Experiment Set 3

Unfortunately, we cannot print selected features for PCA because PCA doesn't directly provide "feature names" the way feature selection methods like SelectKBest in PCC do. This is because PCA transforms the data into a new set of components that are linear combinations of the original features. It doesn't strictly select specific features.

Also, the resulting principal components (the new 'features') are represented by directions in the transformed space. We did a PCA Explained Variance analysis (elbow method) to determine optimal n components value which we found to be between 0.9 and 0.96 at 10 components.

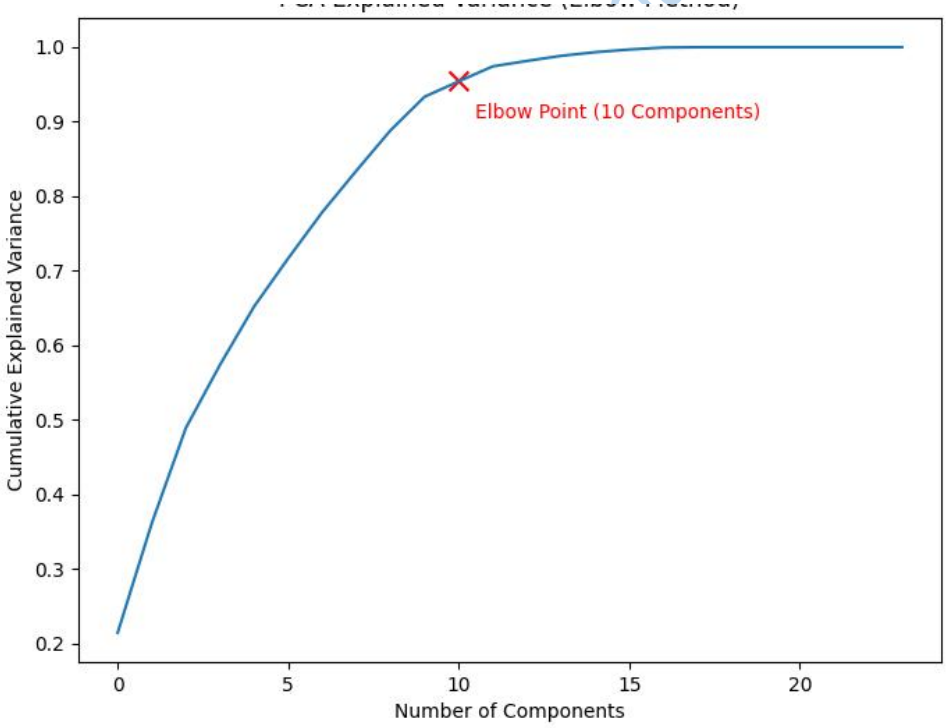


Figure 4.43: PCA Explained Variance of Experiment 3

So, fitting the models on the PCA-transformed dataset, we get the following results:

Table 4.3: The Result of Experiment 3

	Average MSE	Average R-squared	Average MAE
Linear Regression	302.8952	-0.04552	13.38349
Decision Tree	578.8873	-1.07215	17.92467
Random Forest	300.9406	-0.03019	13.1167
SVR (RBF)	292.7527	-0.00994	12.98768
KNN	323.8624	-0.11549	13.57187
MLP	328.7724	-0.11996	13.97091
Gradient Boosting	328.5207	-0.13361	13.98714
XGBoost	341.362	-0.17653	13.87984
LightGBM	349.0052	-0.21111	13.95963
Extra Trees	306.1674	-0.04383	13.27959
AdaBoost	319.1056	-0.11116	13.87409
SVR (linear)	307.5677	-0.06011	13.41835
SVR (poly)	293.6241	-0.0143	13.05535
Gaussian Process	1525.781	-4.44261	33.2103
KNN (tuned)	313.051	-0.07384	13.50823
Bayesian Ridge	295.5758	-0.01959	13.17439
Ridge Regression	302.7898	-0.04515	13.38076
Lasso Regression	293.5146	-0.01128	13.07444
ElasticNet Regression	292.3099	-0.00802	13.07271

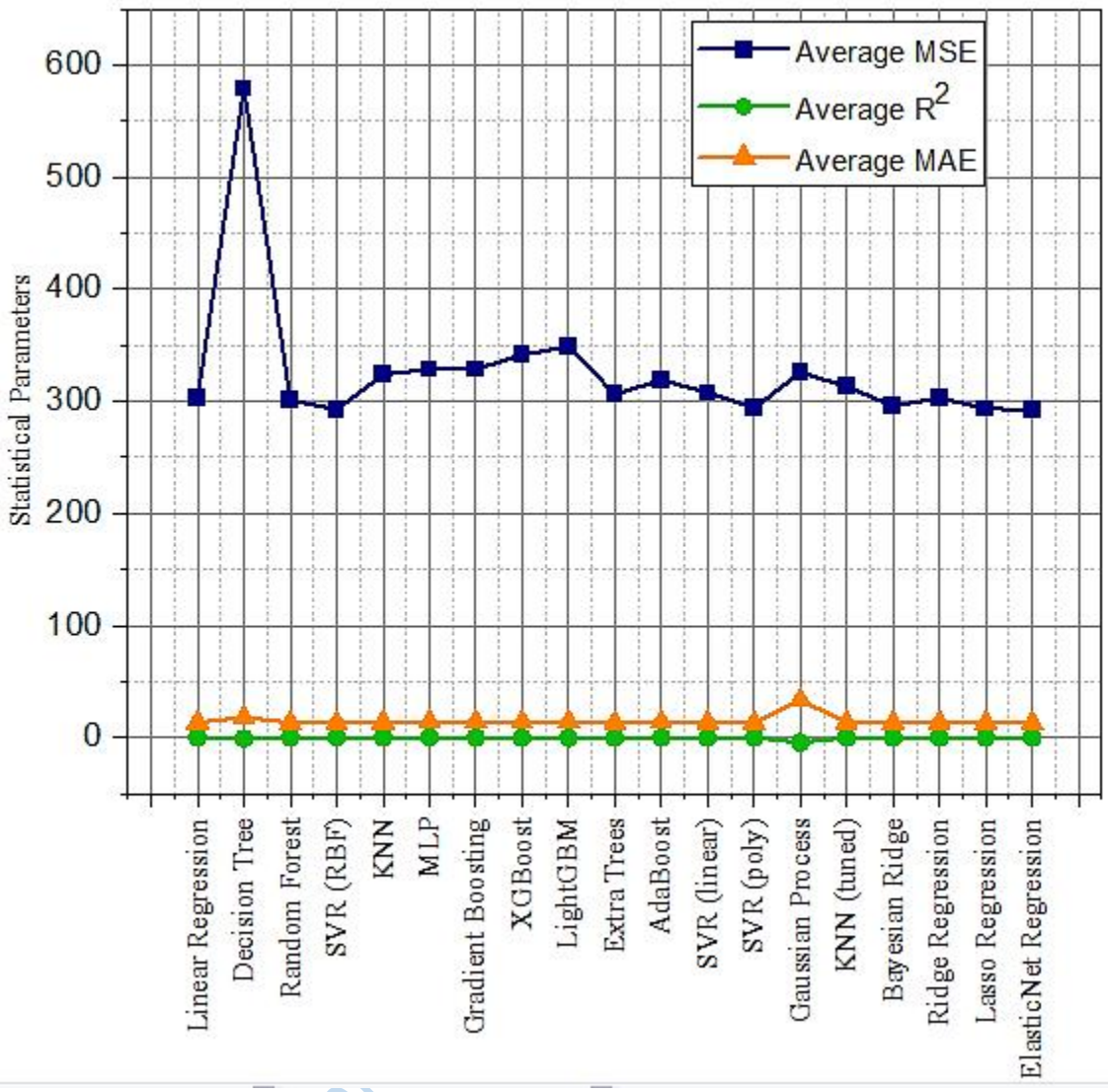


Figure 4.45: Visual representation of Experiment Set 3 Result

Experiment Set 4

We run the scaled dataset with FOR only as feature selection. Knowing the non-linearity of our dataset, we chose "RandomForestRegressor" and "DecisionTreeRegressor" as our FOR estimators. We used GirdSearch to perform a hyperparameter search on "n_features_to_select" from 3 to 23 to find the optimal value.

Table 4.4: The result of Experiment 4

	Average MSE	Average R-squared	Average MAE
Linear Regression	315.2957	-0.09501	13.68136
Decision Tree	426.0438	-0.51315	15.03338
Random Forest	237.9175	0.181225	11.44543
SVR (RBF)	292.9108	-0.0109	12.98584
KNN	332.6935	-0.14387	13.79145
MLP	322.5508	-0.11947	13.71777
Gradient Boosting	267.4766	0.0691	12.20902
XGBoost	274.8883	0.048817	12.53197
LightGBM	270.2428	0.07956	12.18283
Extra Trees	267.5636	0.077452	12.04694
AdaBoost	292.0608	-0.01456	13.11031
SVR (linear)	311.9695	-0.07795	13.50007
SVR (poly)	287.8989	0.006328	12.80858
Gaussian Process	1623.462	-4.79849	34.39143
KNN (tuned)	315.9393	-0.08975	13.58896
Bayesian Ridge	297.6101	-0.02864	13.23092
Ridge Regression	313.3432	-0.08741	13.63518
Lasso Regression	297.7939	-0.02821	13.22409
ElasticNet Regression	296.7159	-0.02559	13.19829

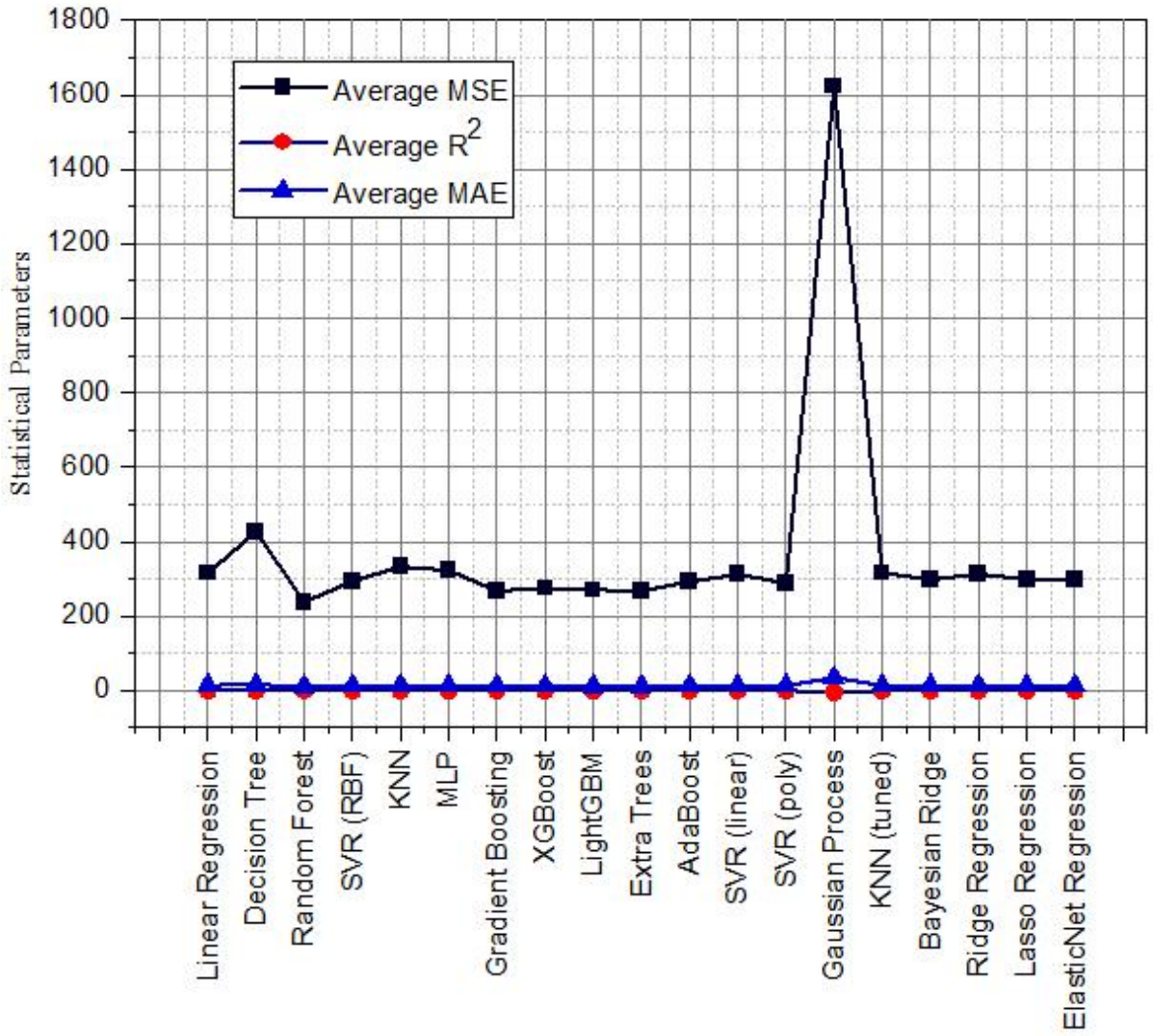


Figure 4.46: Visual representation of Experiment Set 4 Result

Experiment Set 5

For the PSO setup itself, we ensured that the dimensions of the search space matched the total number of features in our original dataset (24). So, we initialized 24 particles to search this space. We also set the cognitive, social, and inertia parameters (`$C_1$`, `$C_2$`, and `W`) to 0.5, 0.3, and 0.9, respectively, as a balanced starting point. Once the PSO finished searching, it extracted the features that were consistently selected in the best solutions

(those with positions greater than 0.5) and we input that in the regression models we have evaluated. We found the following result from our models.

Table 4.5: The result of Experiment 5

	Average MSE	**Average R-squared**	**Average MAE**
Linear Regression	314.81	-0.09247	13.69697
Decision Tree	481.6768	-0.66606	14.91809
Random Forest	223.4295	0.229647	10.6209
SVR (RBF)	292.9208	-0.01037	13.01048
KNN	325.8456	-0.11548	13.62819
MLP	318.5454	-0.10272	13.78158
Gradient Boosting	251.7244	0.127525	11.25462
XGBoost	249.7095	0.133933	11.42567
LightGBM	250.9639	0.143603	11.41556
Extra Trees	245.7222	0.156451	11.22059
AdaBoost	268.522	0.069014	12.95594
SVR (linear)	308.1571	-0.06394	13.47966
SVR (poly)	286.8212	0.010756	12.8311
Gaussian Process	1751.601	-5.25361	36.47341
KNN (tuned)	308.6294	-0.05458	13.37506
Bayesian Ridge	295.2953	-0.01847	13.18661
Ridge Regression	312.524	-0.08352	13.64762
Lasso Regression	293.3104	-0.00991	13.16934
ElasticNet Regression	294.4782	-0.01581	13.18315

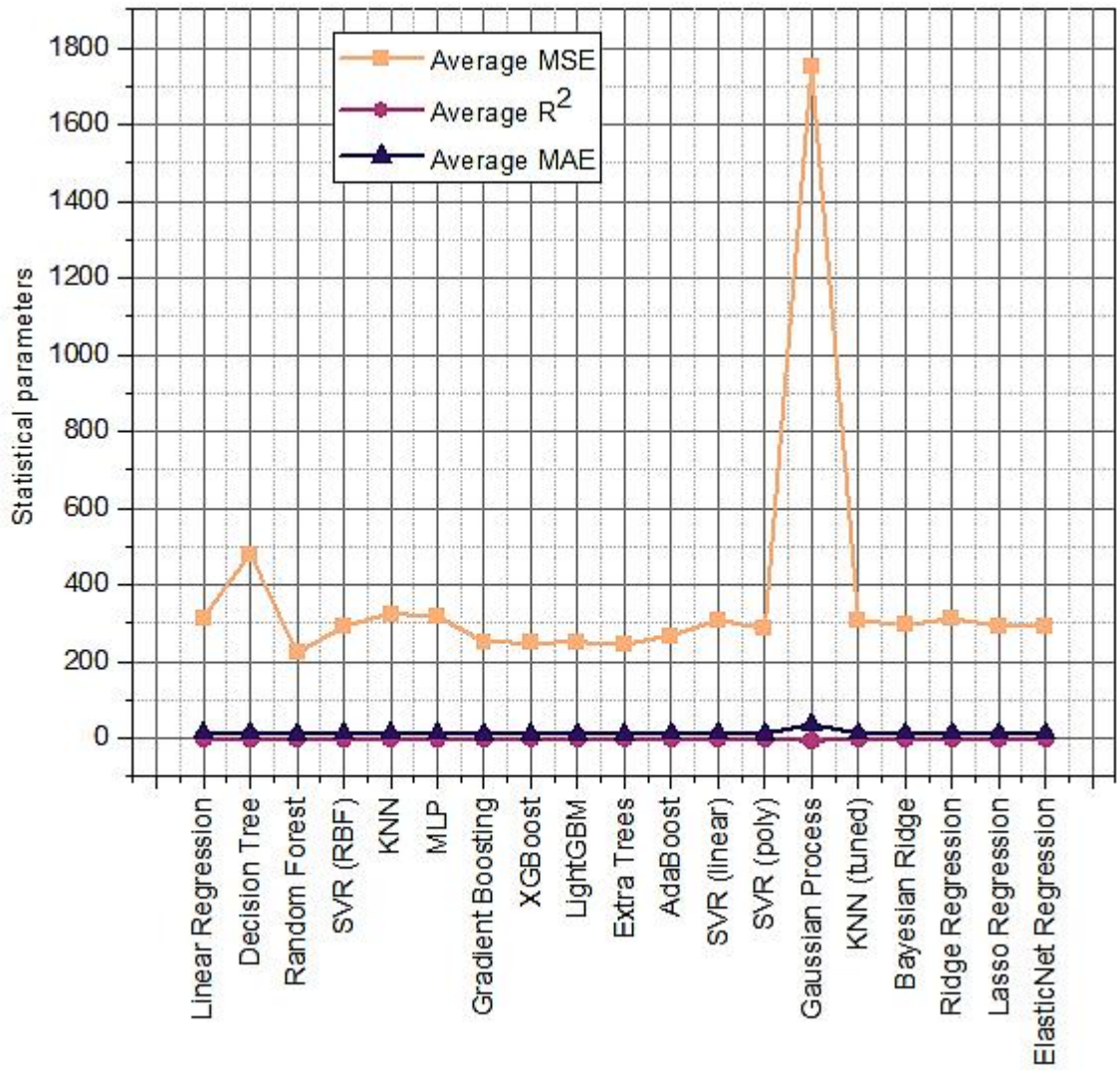


Figure 4.47: Visual representation of Experiment Set 5 Result

Experiment Set 6

This is the proposed model, PF- PSO, a combination of PCA, FOR and PSO. Combining these techniques, we hope to achieve a more comprehensive feature selection process and improve our model's performance.

Table 4.6: The result of Experiment 6

	Average MSE	Average R-squared	Average MAE
Linear Regression	310.6982	-0.07888	13.77405
Decision Tree	663.8221	-1.35404	19.43788
Random Forest	324.0596	-0.14001	13.61323
SVR (RBF)	300.5625	-0.04531	13.28329
KNN	335.8528	-0.17283	13.91777
MLP	326.6266	-0.13815	14.04312
Gradient Boosting	351.3551	-0.24592	14.46499
XGBoost	385.4649	-0.37225	15.184
LightGBM	355.879	-0.26008	14.38811
Extra Trees	334.276	-0.17135	13.68018
AdaBoost	355.4155	-0.2629	14.64434
SVR (linear)	317.4653	-0.10213	13.84965
SVR (poly)	318.1219	-0.10192	13.87862
Gaussian Process	46854.78	-169.347	138.2104
KNN (tuned)	337.3727	-0.17243	14.05427
Bayesian Ridge	304.2629	-0.05563	13.57231
Ridge Regression	310.5051	-0.0782	13.76995
Lasso Regression	304.5223	-0.05669	13.59095
ElasticNet Regression	304.4046	-0.05646	13.59885

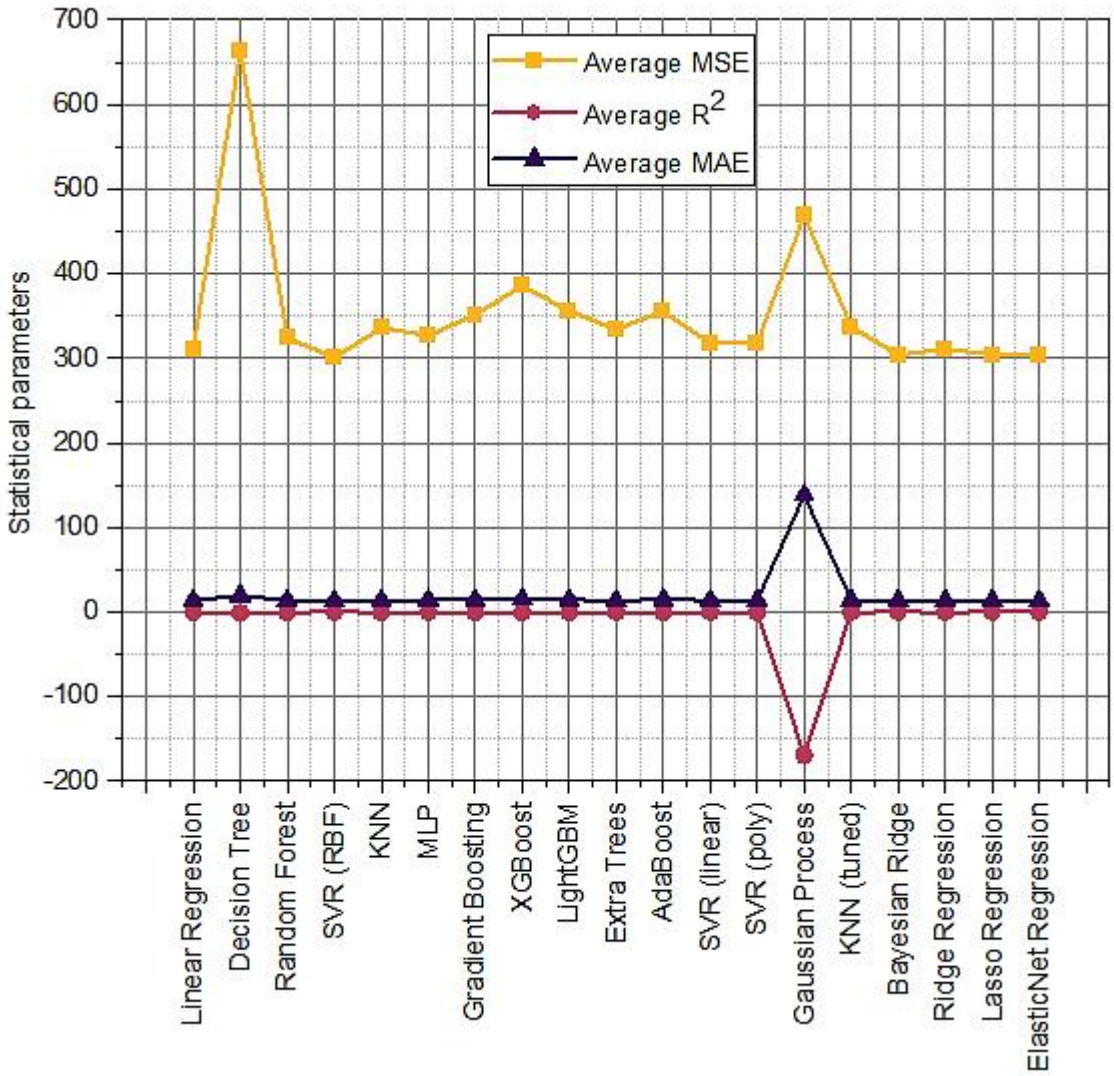


Figure 4.48: Visual representation of Experiment Set 6 Result

4.4 Dataset 2 Experiment Sets

Now, we perform a standardization scaling on our dataset and begin the experiment sets.

Experiment Set 7

We ran the scaled dataset without feature selection into the 9 models and tested with a 10-fold cross-validation. Now, we have two ways to approach the classification.

In one way, we encode “Class” values of “L”,”M” and “H” into 0, 1, and 2 respectively which also conveys the ordinality of the classes. In this case, the models perform multi-class classifications to predict class values of 0, 1 and 2.

In the second way, we encode the “Class” variable into three new variables – “Class_L”, “Class_M” and “Class_H” and assign binary values appropriately to represent the class value. So, a “Class” value of “L” will work out as 1, 0, 0 across “Class_L”, “Class_M” and “Class_H”; “Class” value of “M” works out to 0, 1, 0; and a “class” value of “H” works out to 0, 0, 1. In this case, we let the models perform three multi-class predictions across “Class_L”, “Class_M” and “Class_H” and test the precision for prediction each class correctly.

For the first case, we got the following results:

Table 4.7: Result of Experiment 7 (Single Class Prediction)

Model	Accuracy Mean	Accuracy Std	F1 Mean	F1 Mean Std	Recall Mean	Recall Std	Precision Mean	Precision Std
Logistic Regression	0.7188	0.0598	0.7258	0.0621	0.7291	0.0597	0.7391	0.0674
k-NN	0.6000	0.0708	0.6020	0.0764	0.6082	0.0782	0.6266	0.0737
SVM	0.5917	0.0752	0.5855	0.0809	0.6032	0.0824	0.6397	0.0993
Random Forest	0.6813	0.0575	0.6798	0.0679	0.6831	0.0664	0.7142	0.0527
Gradient Boosting	0.6542	0.0711	0.6591	0.0700	0.6626	0.0663	0.6835	0.0803
XGBoost	0.6646	0.0628	0.6690	0.0617	0.6716	0.0617	0.6973	0.0679
LightGBM	0.6583	0.0660	0.6653	0.0667	0.6667	0.0703	0.6897	0.0679
Naive Bayes	0.6896	0.0451	0.6996	0.0458	0.7175	0.0540	0.7202	0.0573
Neural Network	0.6688	0.0694	0.6714	0.0687	0.6763	0.0666	0.7080	0.0760

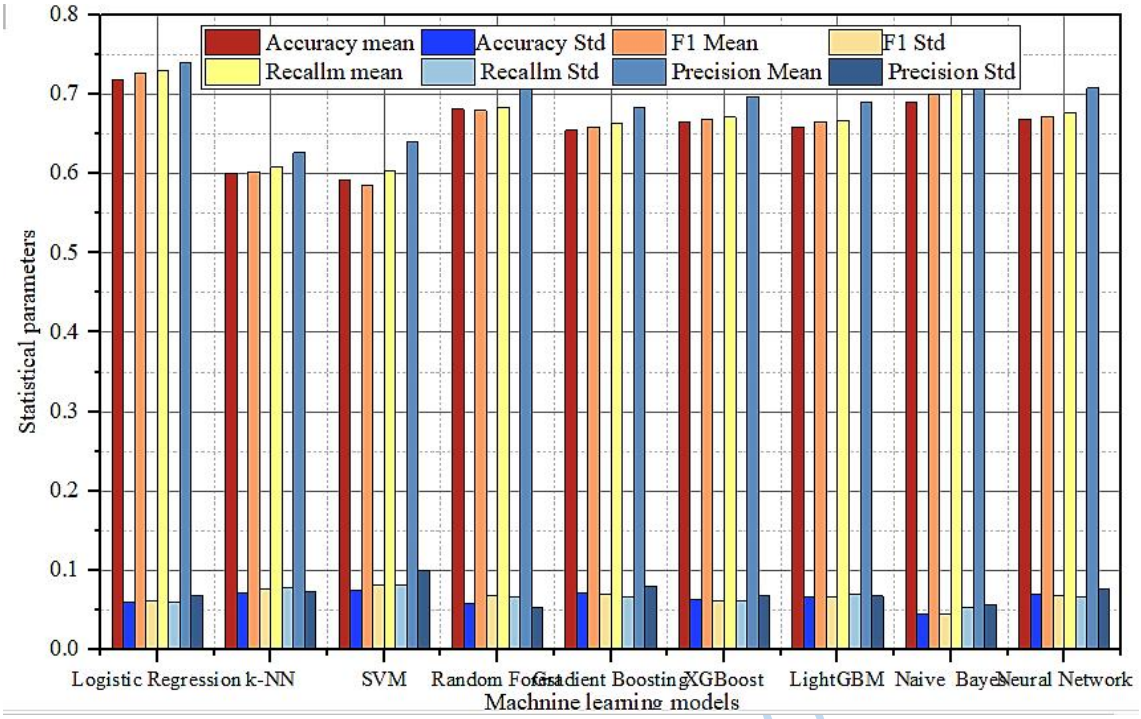


Figure 4.49: Visual representation of Experiment Set 7 (Single-Class Prediction) Result

In the second case, we got the following results:

Table 4.8: Result of Experiment 7 (Multi-Class Prediction)

Model	Class	Accuracy Mean	Accuracy Std	F1 Mean	F1Mean Std	Recall Mean	Recall Std	Precision Mean	Precision Std
Logistic Regression	L	0.9000	0.0580	0.8190	0.0961	0.8301	0.1273	0.8379	0.1574
Logistic Regression	M	0.5604	0.0782	0.4099	0.1469	0.3758	0.1690	0.5133	0.1479
Logistic Regression	H	0.8104	0.0681	0.6745	0.1050	0.6743	0.1713	0.7091	0.1239
k-NN	L	0.8292	0.1061	0.6768	0.1851	0.6731	0.2174	0.7456	0.2321
k-NN	M	0.6021	0.0655	0.5504	0.0487	0.5552	0.0909	0.5642	0.0842

k-NN	H	0.7354	0.0559	0.5183	0.1090	0.5043	0.1822	0.5890	0.1363
SVM	L	0.8563	0.0804	0.7313	0.1530	0.7455	0.2060	0.7666	0.2012
SVM	M	0.6354	0.0376	0.5590	0.0869	0.5506	0.1559	0.6026	0.0543
SVM	H	0.7271	0.0957	0.4771	0.1855	0.4662	0.2590	0.6088	0.2198
Random Forest	L	0.9063	0.0376	0.8203	0.0642	0.8051	0.1278	0.8724	0.1414
Random Forest	M	0.6729	0.0834	0.5997	0.1088	0.5690	0.1469	0.6511	0.1143
Random Forest	H	0.7563	0.0672	0.5547	0.1382	0.5533	0.2312	0.6448	0.1786
Gradient Boosting	L	0.8938	0.0555	0.8089	0.0817	0.8205	0.1124	0.8315	0.1661
Gradient Boosting	M	0.6792	0.0667	0.6244	0.0648	0.6067	0.0924	0.6586	0.0930
Gradient Boosting	H	0.7479	0.0635	0.5631	0.1306	0.5752	0.1972	0.5941	0.1461
XGBoost	L	0.8979	0.0531	0.8107	0.0906	0.8212	0.1387	0.8302	0.1436
XGBoost	M	0.6771	0.0627	0.6292	0.0675	0.6262	0.0953	0.6434	0.0837
XGBoost	H	0.7625	0.0723	0.5937	0.1228	0.5971	0.1773	0.6249	0.1670
LightGBM	L	0.8938	0.0547	0.8018	0.0959	0.8051	0.1410	0.8304	0.1494
LightGBM	M	0.6813	0.0774	0.6175	0.1090	0.6026	0.1541	0.6511	0.0981
LightGBM	H	0.7500	0.0652	0.5542	0.1186	0.5414	0.1757	0.6242	0.1732
Naive	L	0.8688	0.1008	0.7903	0.1352	0.8532	0.1349	0.7743	0.2025

Bayes									
Naive Bayes	M	0.4958	0.0967	0.3472	0.2006	0.3610	0.2455	0.3845	0.2416
Naive Bayes	H	0.7958	0.0921	0.7063	0.1140	0.8086	0.1393	0.6416	0.1405
Neural Network	L	0.8833	0.0723	0.7797	0.1353	0.7756	0.1690	0.8154	0.1788
Neural Network	M	0.6646	0.0700	0.6175	0.0619	0.6119	0.0690	0.6327	0.0928
Neural Network	H	0.7729	0.0938	0.6197	0.1299	0.6248	0.1747	0.6564	0.1687

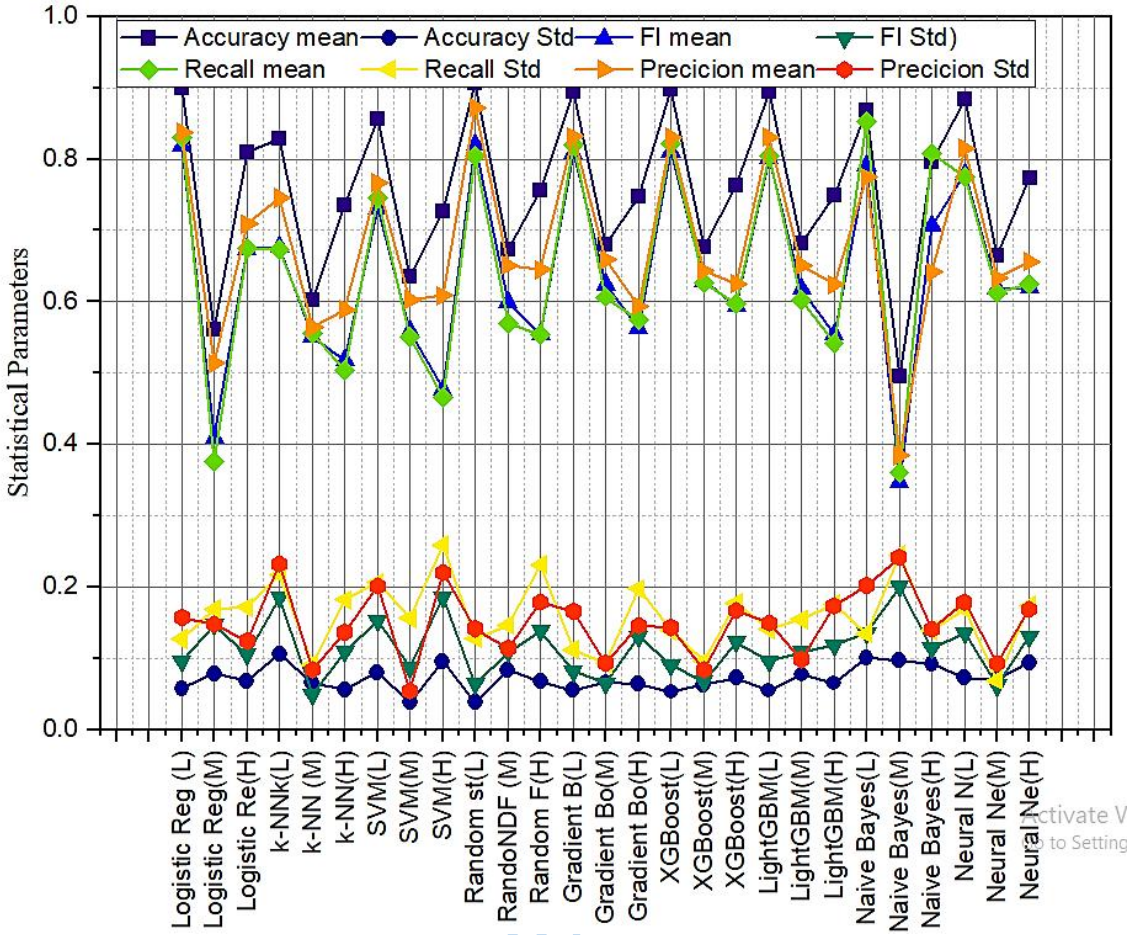


Figure 4.50 Visual representation of Experiment Set 7 (Multiple-Class Prediction) Result

Experiment Set 8

Here, dataset 2 was ran with Pearson Correlation Coefficient feature selection. We did this with selection threshold values of 0.1 and 0.5.

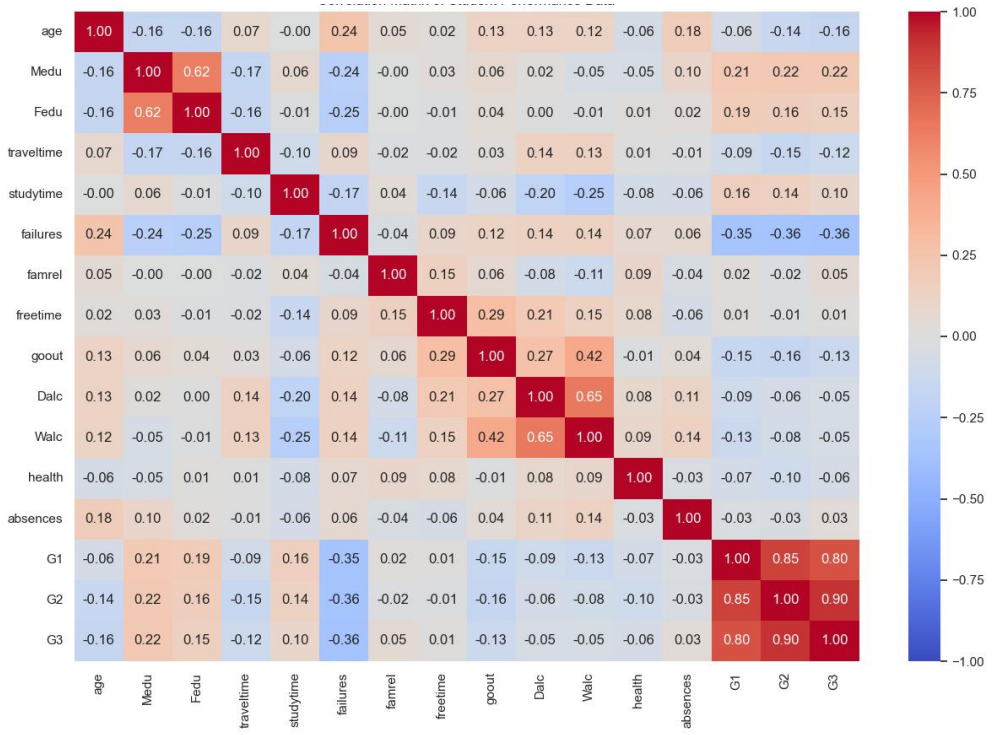


Figure 4.51: the correlation matrix of relationships between different features and the target variable (for Dataset 2)

Table 4.9: Result of Experiment 8 (at Threshold 0.1)

Model	Accuracy Mean	Accuracy Std	F1 Mean	F1 Std	Recall Mean	Recall Std	Precision Mean	Precision Std
Logistic Regression	0.7333	0.0565	0.7411	0.0591	0.7427	0.0523	0.7529	0.0598
k-NN	0.6042	0.0801	0.6069	0.0879	0.6128	0.0895	0.6282	0.0852
SVM	0.5958	0.0769	0.5894	0.0837	0.6071	0.0845	0.6436	0.0996
Random Forest	0.7333	0.0657	0.7377	0.0694	0.7380	0.0713	0.7602	0.0638
Gradient Boosting	0.6792	0.0529	0.6866	0.0489	0.6876	0.0555	0.7013	0.0531
XGBoost	0.6688	0.0820	0.6740	0.0812	0.6766	0.0800	0.6947	0.0849
LightGBM	0.6667	0.0828	0.6733	0.0825	0.6735	0.0852	0.6942	0.0867
Naive Bayes	0.7188	0.0627	0.7303	0.0624	0.7446	0.0657	0.7453	0.0647
Neural Network	0.6896	0.0614	0.6913	0.0681	0.6963	0.0722	0.7149	0.0604

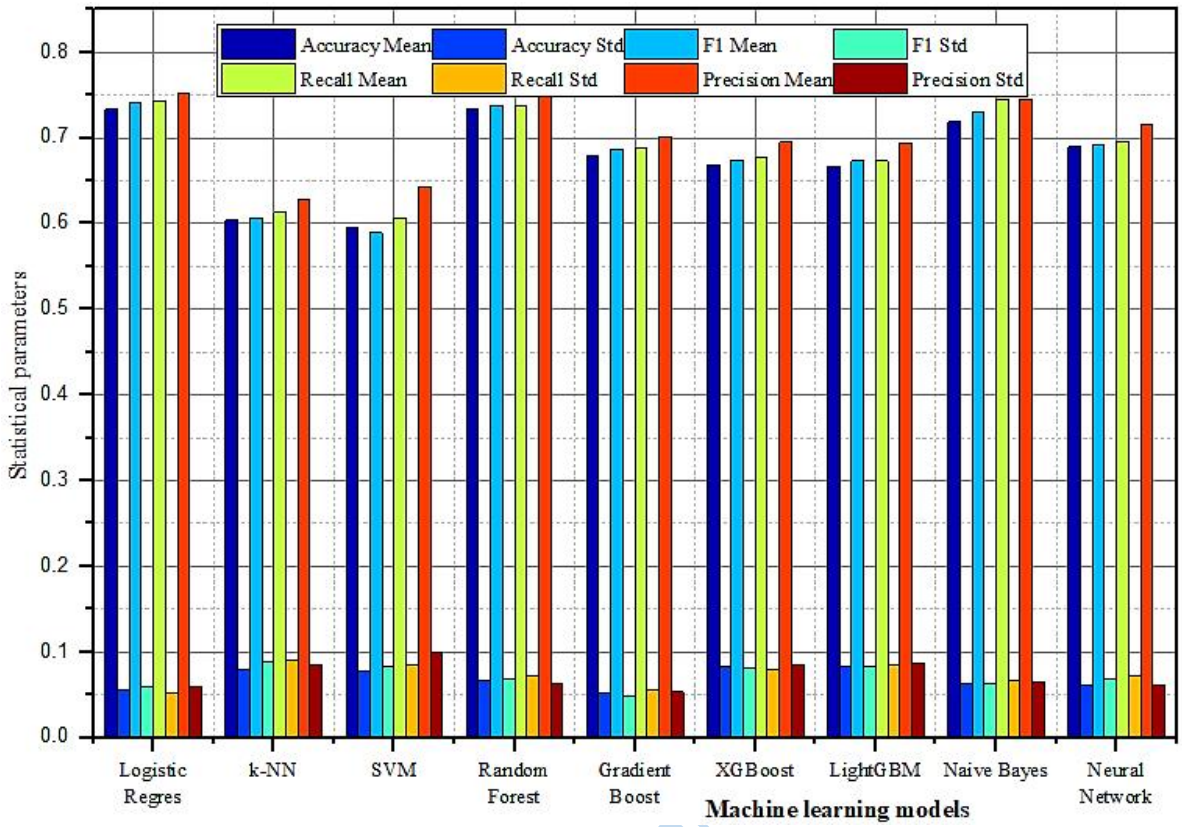


Figure:4.52 Visual representation of Experiment Set 8 (at Threshold 0.1) Result

Table 4.10: Result of Experiment 8 (at Threshold 0.5)

Model	Accuracy Mean	Accuracy Std	F1 Mean	F1 Std	Recall Mean	Recall Std	Precision Mean	Precision Std
Logistic Regression	0.7146	0.0884	0.7212	0.0951	0.7273	0.0998	0.7426	0.0910
k-NN	0.5521	0.0819	0.5502	0.0970	0.5596	0.0985	0.5743	0.0908
SVM	0.6063	0.0920	0.6085	0.1012	0.6229	0.1067	0.6476	0.0925
Random Forest	0.6792	0.0946	0.6864	0.0942	0.6870	0.0991	0.7105	0.0961
Gradient Boosting	0.6604	0.0950	0.6690	0.0942	0.6754	0.0945	0.6920	0.0976
XGBoost	0.6354	0.0919	0.6478	0.0901	0.6476	0.0926	0.6646	0.0948
LightGBM	0.6333	0.0890	0.6417	0.0895	0.6427	0.0925	0.6711	0.0912
Naive Bayes	0.6896	0.0788	0.7023	0.0786	0.7224	0.0838	0.7212	0.0845
Neural Network	0.6563	0.0537	0.6606	0.0699	0.6793	0.0698	0.6841	0.0734

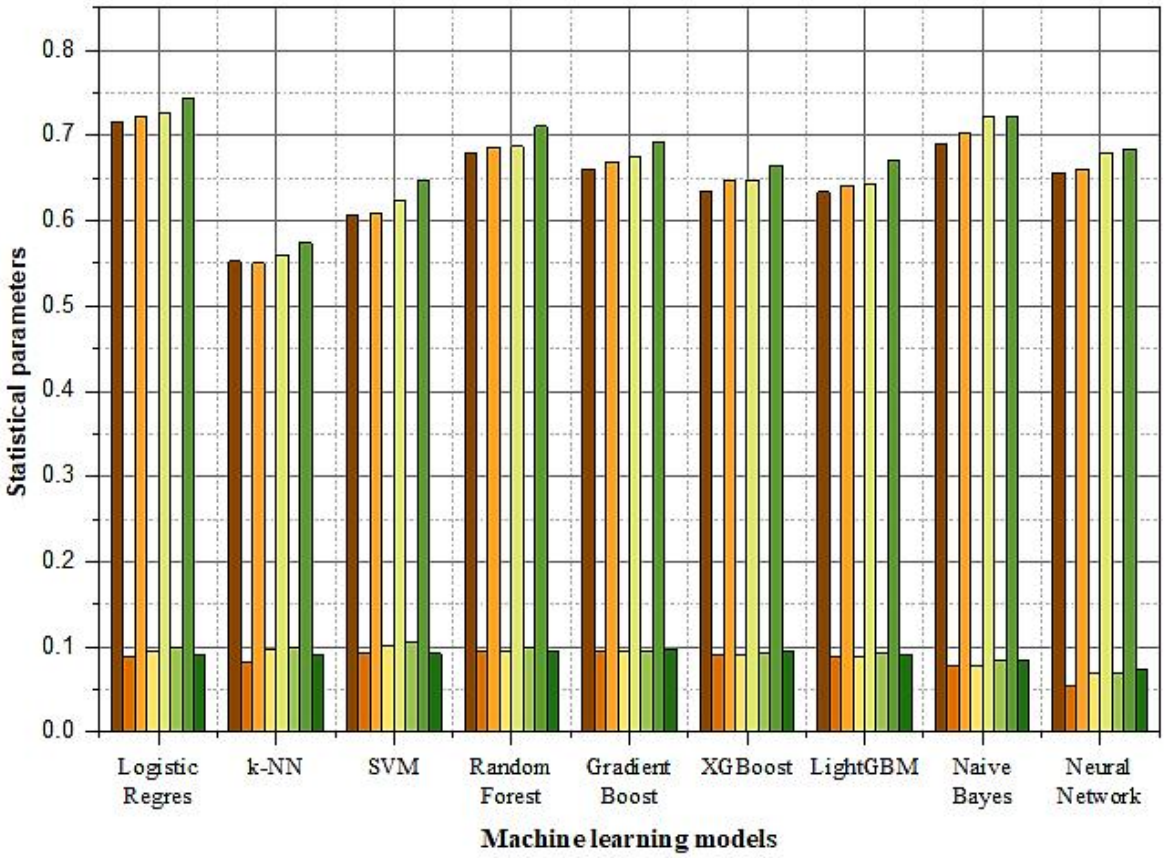


Figure 4.53 Visual representation of Experiment Set 8 (at Threshold 0.5) Result

Experiment Set 9

Here, we perform PCA only for feature selection with a 0.95 value for n component

Again, we performed an Explained Variance Analysis to identify a good value for n components

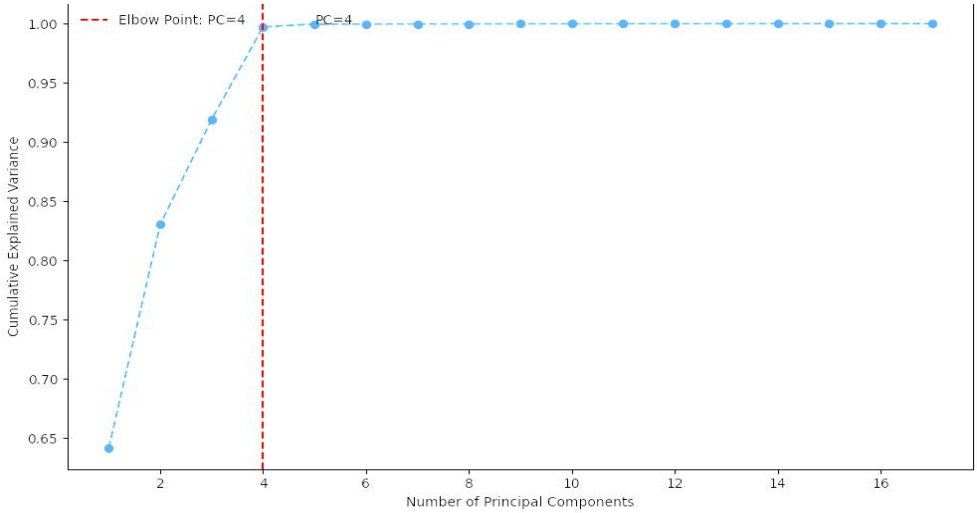


Figure 4.54: PCA Explained Variance Analysis for Experiment 9

So, fitting the models on the PCA-transformed dataset, we get the following results:

Table 4.11: Result of Experiment 9

Model	Accuracy Mean	Accuracy Std	F1 Mean	F1 Std	Recall Mean	Recall Std	Precision Mean	Precision Std
Logistic Regression	0.6063	0.0924	0.6066	0.1003	0.6199	0.1030	0.6423	0.0903
k-NN	0.6063	0.0771	0.6089	0.0853	0.6144	0.0872	0.6309	0.0813
SVM	0.6063	0.1039	0.6062	0.1111	0.6151	0.1115	0.6430	0.1108
Random Forest	0.5917	0.0933	0.5905	0.1043	0.5959	0.1093	0.6257	0.1045
Gradient Boosting	0.5313	0.0693	0.5298	0.0840	0.5309	0.0888	0.5690	0.0760
XGBoost	0.5563	0.1025	0.5591	0.1089	0.5606	0.1170	0.5953	0.0952
LightGBM	0.5688	0.0828	0.5670	0.0960	0.5720	0.1030	0.6074	0.0830
Naive Bayes	0.5792	0.0945	0.5857	0.1003	0.6095	0.1011	0.6109	0.1029
Neural	0.5792	0.0784	0.5857	0.0837	0.5957	0.0809	0.6100	0.0856

Network

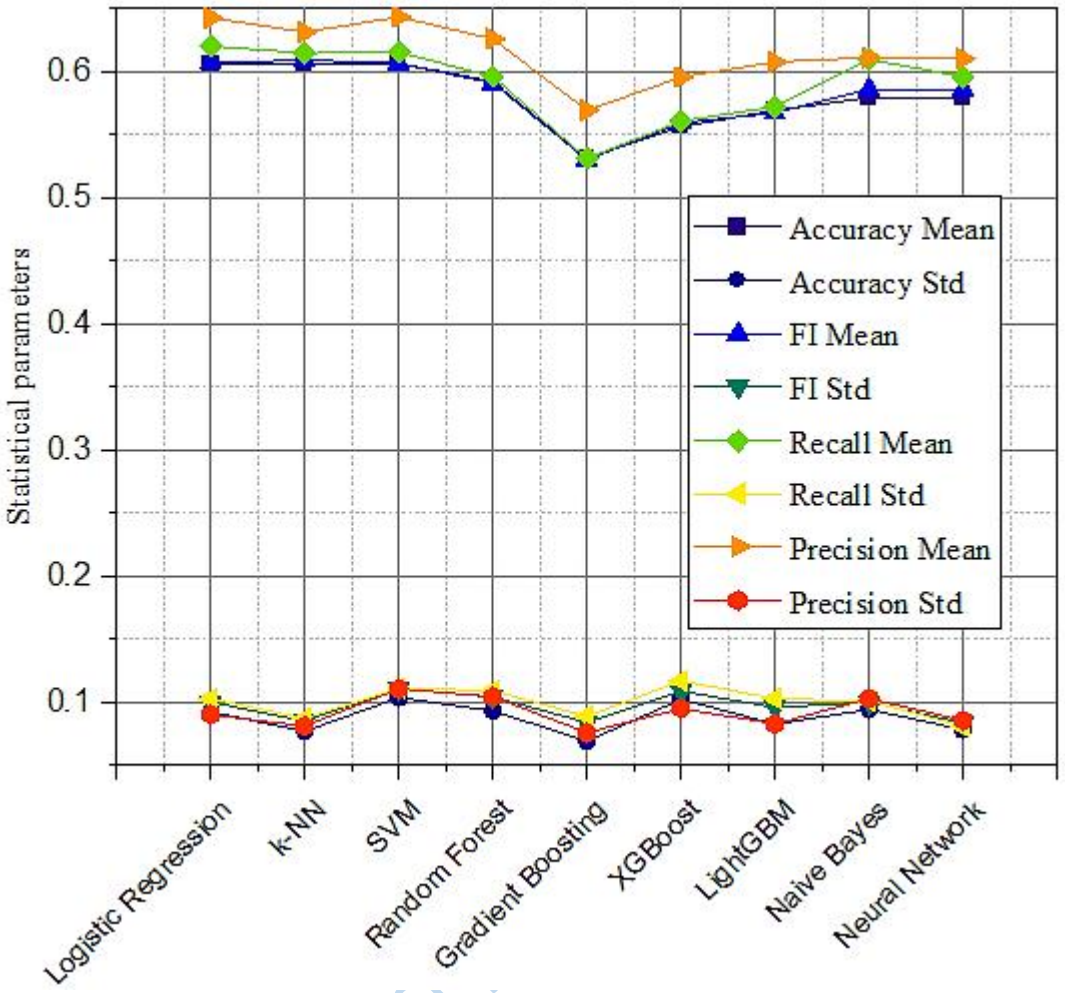


Figure: 4.55 Visual representation of Experiment Set 9 Result

Experiment Set 10

Here, we ran our dataset with FOR only as the feature selection we decided to create an ensemble of Logistic Regression and RandomForest Classifier as estimators for our sequential feature selection then we passed the selected features into all the models. We got the following results.

Table 4.12: Result of Experiment 10

Model	Accuracy Mean	Accuracy Std	Precision Mean	Precision Std	Recall Mean	Recall Std	F1 Mean	F1 Std
Logistic Regression	0.7375	0.0598	0.7602	0.0651	0.7478	0.0627	0.7440	0.0613
k-NN	0.5896	0.0982	0.6106	0.1106	0.6016	0.1104	0.5929	0.1068
SVM	0.6000	0.0868	0.6448	0.1136	0.6084	0.0927	0.5921	0.0940
Random Forest	0.7000	0.0375	0.7267	0.0353	0.7050	0.0518	0.7054	0.0446
Gradient Boosting	0.6833	0.0500	0.7127	0.0522	0.6848	0.0550	0.6868	0.0487
XGBoost	0.6542	0.0619	0.6839	0.0738	0.6625	0.0669	0.6620	0.0613
LightGBM	0.6813	0.0646	0.7089	0.0749	0.6886	0.0680	0.6870	0.0649
Naive Bayes	0.7125	0.0306	0.7418	0.0382	0.7402	0.0465	0.7222	0.0340
Neural Network	0.6729	0.0536	0.7178	0.0499	0.6859	0.0619	0.6708	0.0692

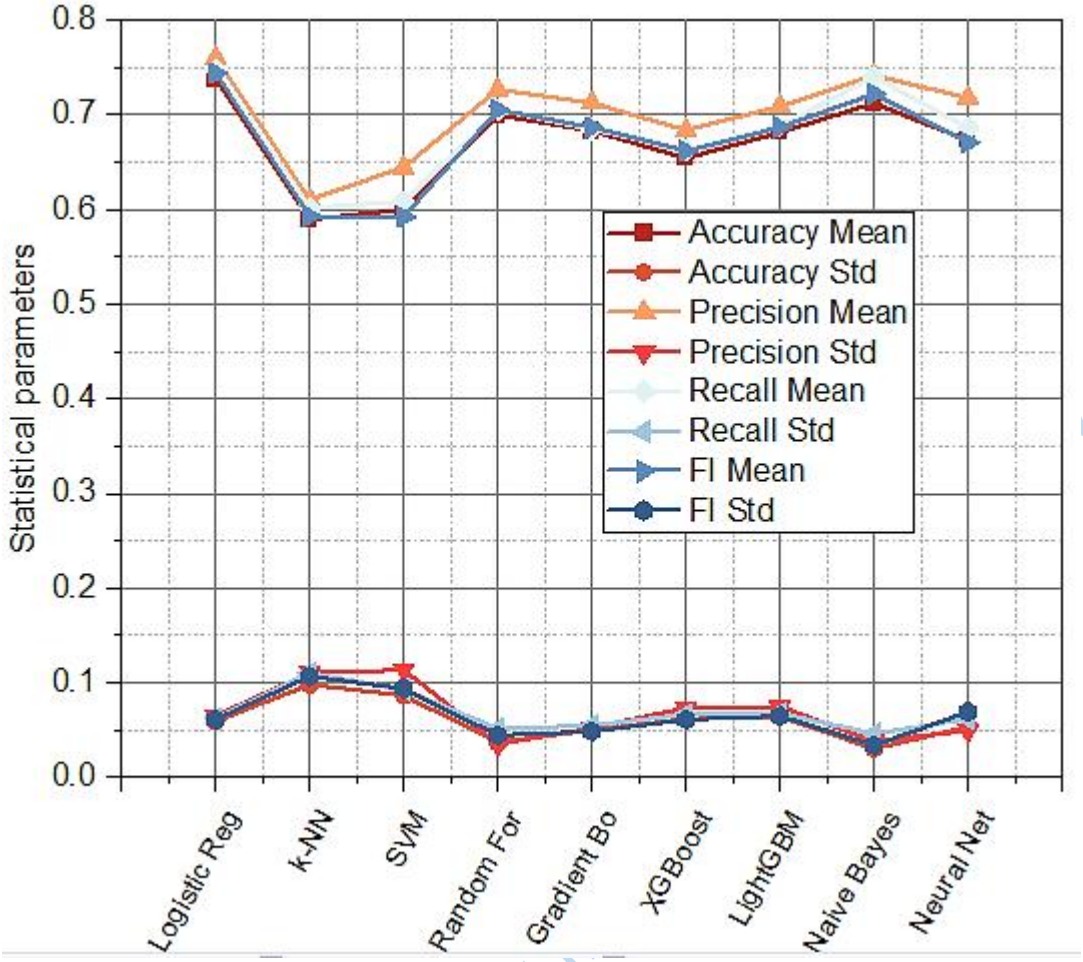


Figure 4.56 Visual representation of Experiment Set 10 Result

Experiment Set 11

Here, we ran our dataset with PSO only as the feature selection method. We found the following result from our models.

Table 4.13: Result of Experiment 11

	Avg. Accuracy	Avg. Precision	Avg. Recall	Average Mean	F1
Logistic Regression	0.752083	0.762559	0.756876	0.755334	
k-NN	0.610417	0.620669	0.617673	0.607957	
SVM	0.610417	0.616677	0.630555	0.619232	
Random Forest	0.79375	0.803606	0.79219	0.794638	
Gradient Boosting	0.7625	0.768078	0.760758	0.760412	
XGBoost	0.754167	0.762672	0.755929	0.754554	
LightGBM	0.76875	0.780891	0.761748	0.76667	
Naive Bayes	0.710417	0.718859	0.730219	0.714991	
Neural Network	0.714583	0.72595	0.721131	0.714711	

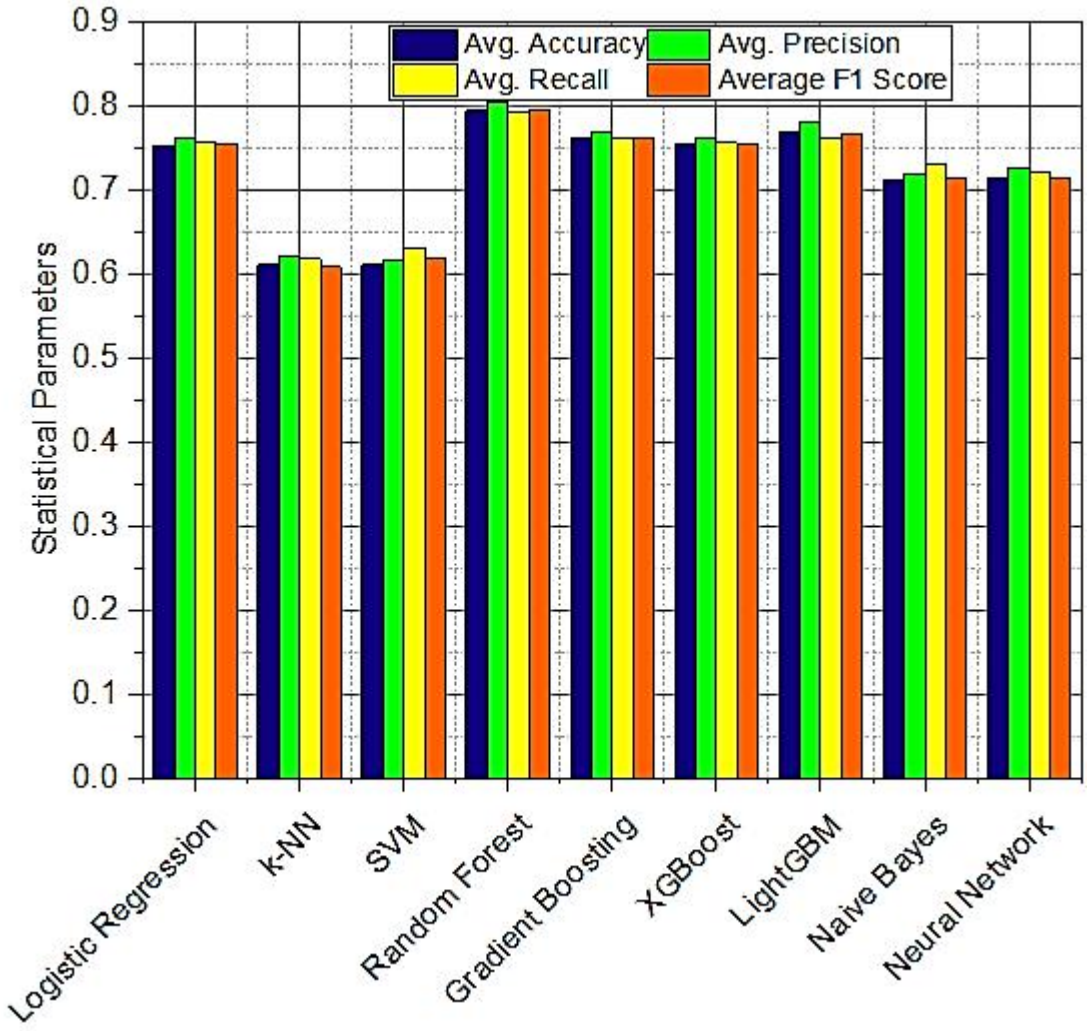


Figure 4.57: Visual representation of Experiment Set 11 Result

Experiment Set 12

Here, we use our PF-PSO approach for feature selection. We found the following results from our models.

Table 4.14: Result of Experiment 12

	Avg. Accuracy	Average Precision	Average Recall	Average F1 Mean
Logistic Regression	0.641667	0.644493	0.652968	0.643064
k-NN	0.61875	0.6294	0.6257	0.620159
SVM	0.639583	0.64472	0.658631	0.644098
Random Forest	0.654167	0.658717	0.666966	0.658124
Gradient Boosting	0.608333	0.618611	0.61687	0.609822
XGBoost	0.604167	0.611548	0.613044	0.607973
LightGBM	0.6	0.60988	0.611888	0.6041
Naive Bayes	0.645833	0.648899	0.678399	0.65182
Neural Network	0.61875	0.628545	0.640701	0.622643

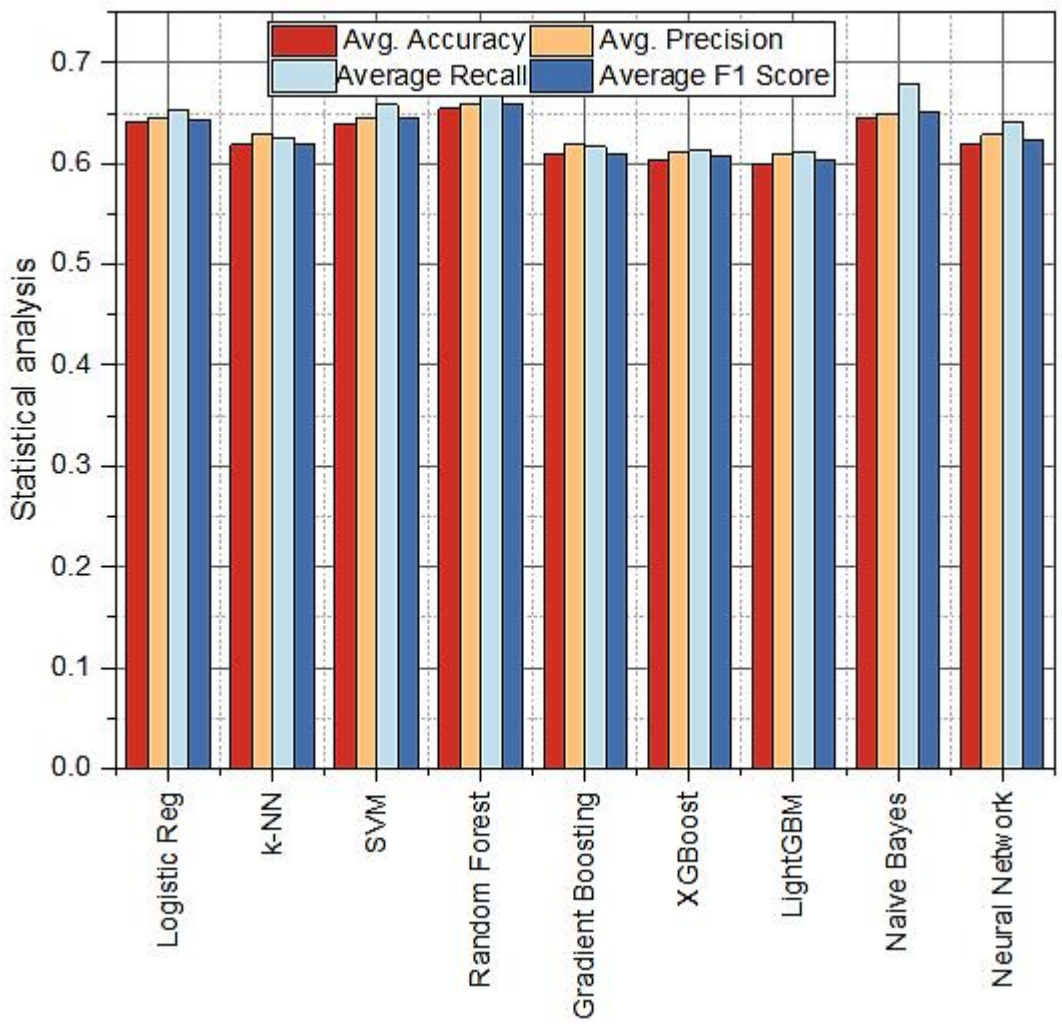


Figure 4.57 Visual representation of Experiment Set 12 Result

4.5 Dataset 3 Experiment Sets

Now, we perform a standardization scaling on our dataset and begin the experiment sets.

Experiment Set 13

Here, we ran the processed dataset without feature selection into the 19 models .We got the following results:

Table 4.15: Result of Experiment 13

	Average MSE	Average R-squared	Average MAE
Linear Regression	4.167037473	0.793348758	1.362017551
Decision Tree	5.943653846	0.701857363	1.276987179
Random Forest	2.534196673	0.878131433	0.977377564
SVR (RBF)	6.631440243	0.681717595	1.570204329
KNN	8.664735897	0.566625751	2.108423077
MLP	4.878417487	0.753700671	1.554396686
Gradient Boosting	2.769427716	0.865406866	1.044217105
XGBoost	3.018841715	0.850669077	1.019573491
LightGBM	2.711731538	0.869119483	1.061075629
Extra Trees	3.57945591	0.82609068	1.151033333
AdaBoost	3.095818299	0.845694729	1.175335621
SVR (linear)	4.03774909	0.808761183	1.021310769
SVR (poly)	5.663651744	0.723184995	1.466979659
Gaussian Process	125.8750997	-5.483777034	10.26894336
KNN (tuned)	7.770882602	0.620520526	1.94532634
Bayesian Ridge	4.029990154	0.800593917	1.319374309
Ridge Regression	4.14182235	0.794668374	1.354239671
Lasso Regression	4.824687657	0.770646329	1.347107983
ElasticNet Regression	5.836188808	0.720569846	1.54858554

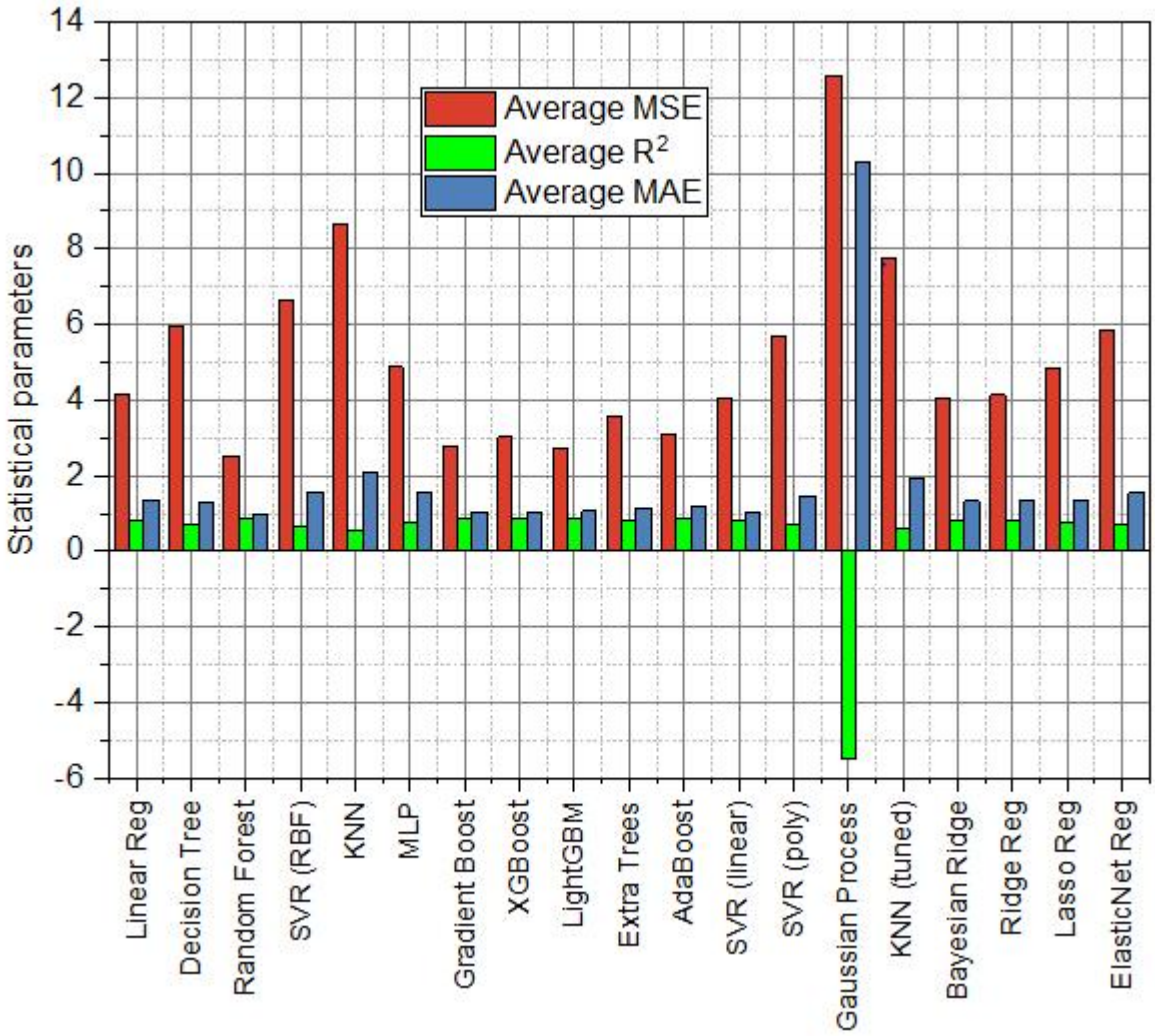


Figure 4.58 Visual representation of Experiment Set 13 Result

Experiment Set 14

Here, we ran dataset with Pearson Correlation Coefficient feature selection. We did this with a selection threshold at 0.6.

Below is the PCC matrix:

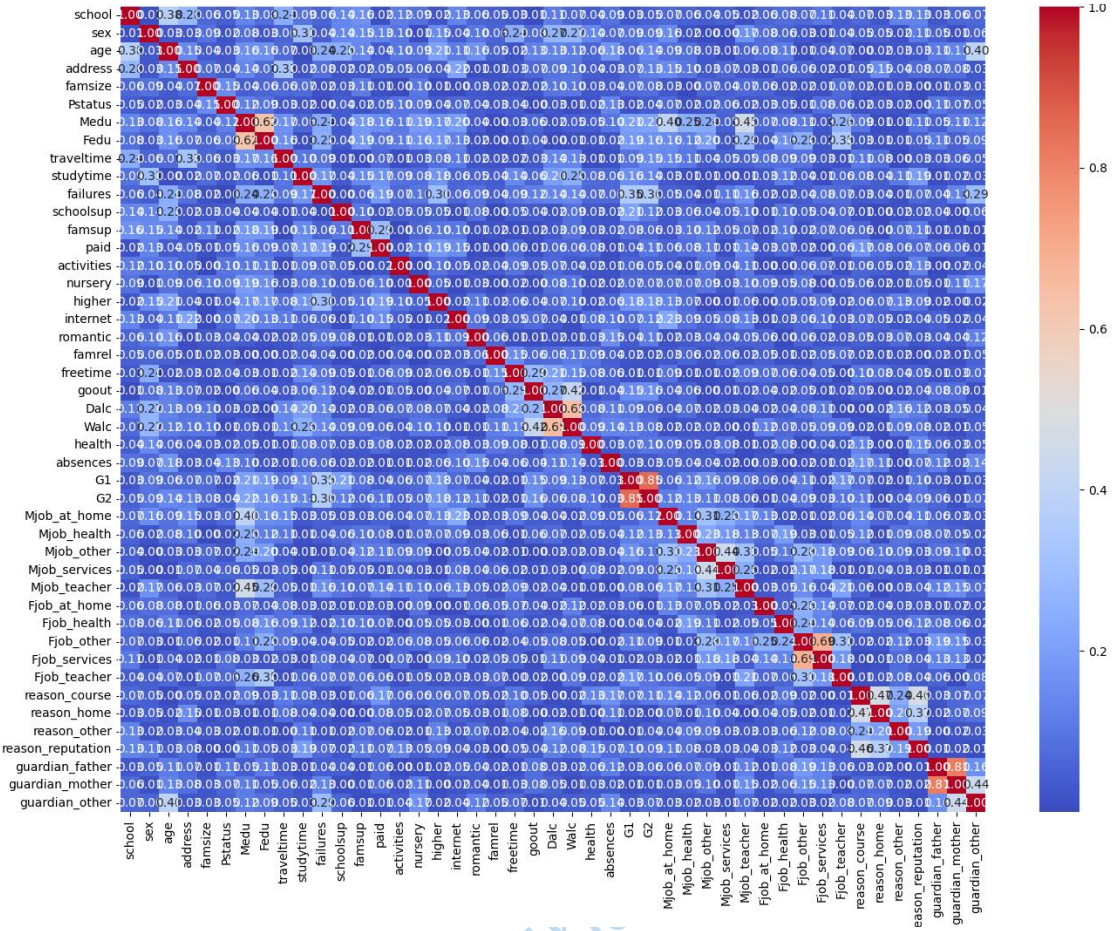


Figure 4.59: PCC Matrix of Experiment 14 (for Dataset 3)

Now, running the models on these features produced the following results:

Table 4.16: Result of Experiment 14

	Average MSE	Average R-squared	Average MAE
Linear Regression	3.801511	0.81745	1.179896
Decision Tree	7.566166	0.619835	1.490128
Random Forest	5.110064	0.74687	1.334034
SVR (RBF)	5.127085	0.754779	1.277935
KNN	5.608195	0.730465	1.564385
MLP	4.745322	0.772009	1.355507
Gradient Boosting	4.257462	0.792503	1.266013
XGBoost	6.956212	0.658305	1.527786
LightGBM	4.4855	0.782048	1.359939
Extra Trees	6.475354	0.679382	1.449573
AdaBoost	4.952309	0.752401	1.53436
SVR (linear)	4.064841	0.807427	1.006872
SVR (poly)	6.767886	0.674616	1.655411
Gaussian Process	883268.6	-40750.2	88.62502
KNN (tuned)	5.539747	0.733024	1.507393
Bayesian Ridge	3.801091	0.817508	1.176369
Ridge Regression	3.801076	0.817506	1.176555
Lasso Regression	4.824555	0.770655	1.346927
ElasticNet Regression	6.773973	0.676691	1.737694

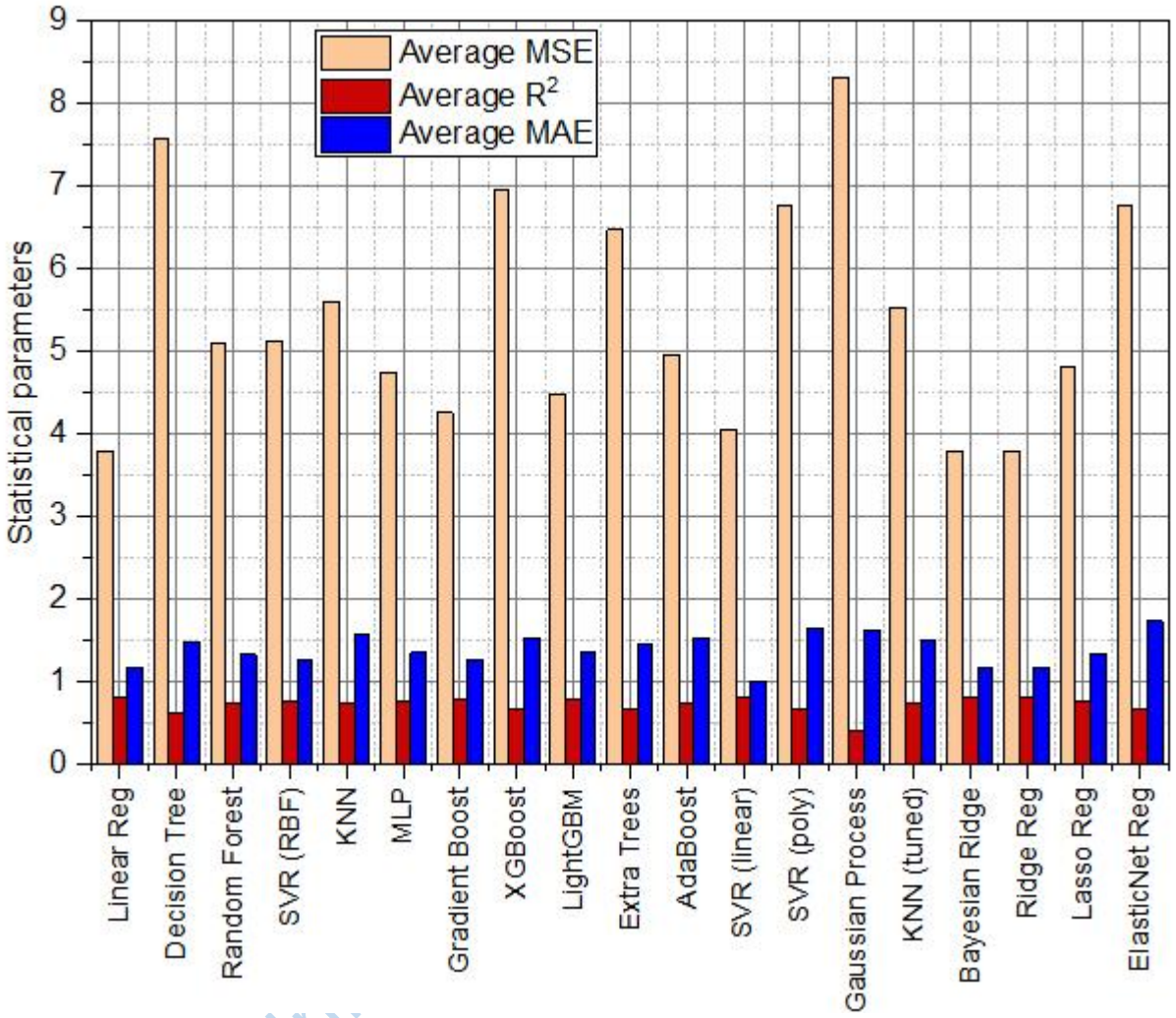


Figure 4.60: Visual representation of Experiment Set 14 Result

Experiment Set 15

We run the dataset with PCA only for feature selection with a 0.95 value for n_components. This selected 29 columns.

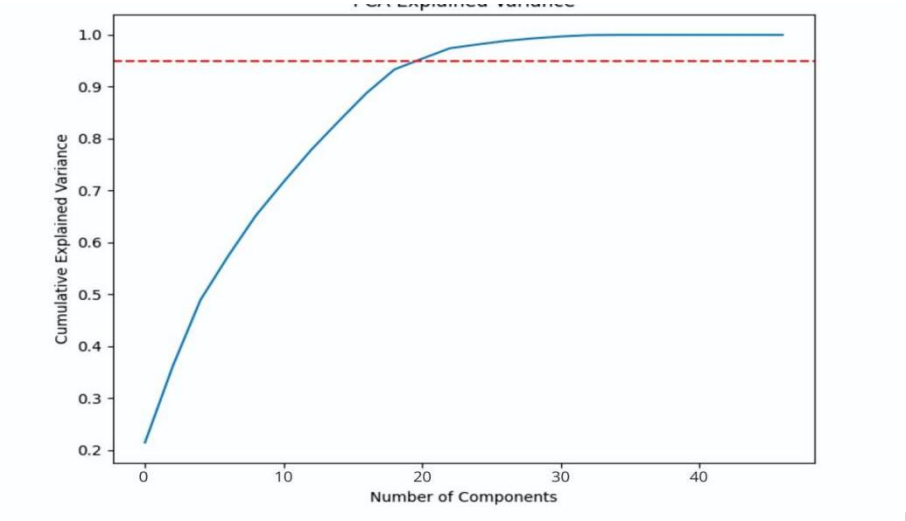


Figure 4.61: PCA Explained Variance of Experiment 15

Fitting the models on the PCA-transformed dataset, we get the following results:

Table 4.17: Result of Experiment 15

	Average MSE	Average R-squared	Average MAE
Linear Regression	3.943025	0.805574	1.319135
Decision Tree	12.81077	0.331617	2.412051
Random Forest	6.346573	0.697071	1.694247
SVR (RBF)	7.077807	0.660296	1.641759
KNN	8.291603	0.590026	2.052628
MLP	6.467473	0.673641	1.944505
Gradient Boosting	5.643729	0.721785	1.594324
XGBoost	6.807857	0.675741	1.770869
LightGBM	5.736735	0.722214	1.662418
Extra Trees	5.963539	0.712357	1.619804
AdaBoost	7.518536	0.633796	2.071861
SVR (linear)	4.014464	0.808969	1.136138
SVR (poly)	7.57469	0.630727	1.794704
Gaussian Process	123.2046	-5.34491	10.15998
KNN (tuned)	7.55379	0.631294	1.93834

Bayesian Ridge	3.921699	0.806536	1.305111
Ridge Regression	3.937712	0.80582	1.316371
Lasso Regression	6.990429	0.662641	1.760004
ElasticNet Regression	6.959303	0.66431	1.761868

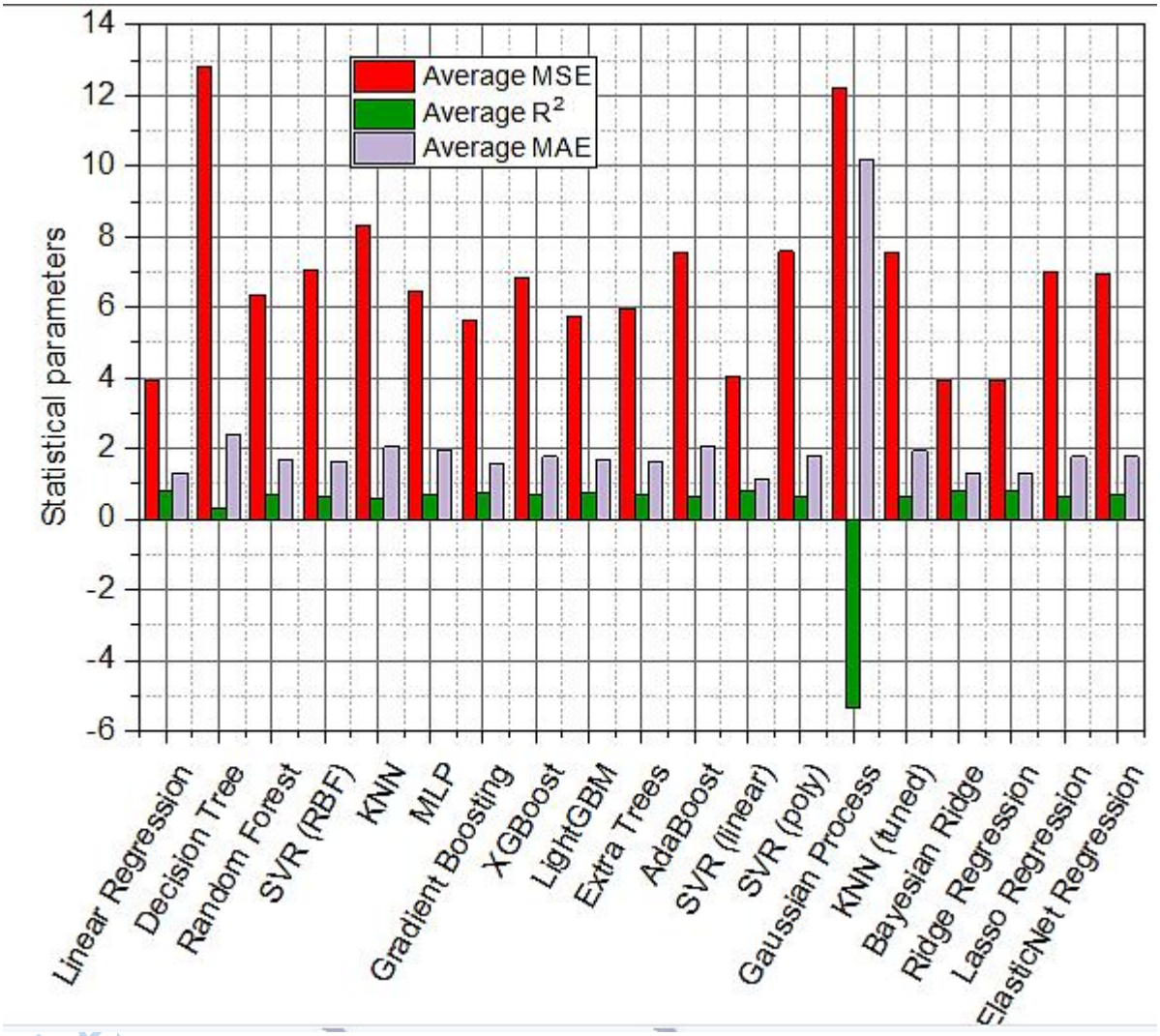


Figure 4.62 Visual representation of Experiment Set 15 Result

Experiment Set 16

Here, we do the same as above but with FOR-only feature selection. We used Random Forest as our estimator and GridSearchCV to perform an hyperparameter search on "n_features_to_select". We got the following results:

Table 4.18: Result of Experiment 16

	Average MSE	Average R-squared	Average MAE
Linear Regression	4.118546	0.797603	1.309694
Decision Tree	5.324487	0.740587	1.198077
Random Forest	2.688065	0.870081	0.977721
SVR (RBF)	5.714514	0.726377	1.437079
KNN	6.822226	0.66682	1.760256
MLP	6.468993	0.683327	1.7786
Gradient Boosting	2.898625	0.860079	1.027734
XGBoost	3.450973	0.824524	1.086595
LightGBM	3.103504	0.848246	1.124777
Extra Trees	3.514586	0.830325	1.121525
AdaBoost	3.475162	0.824737	1.247554
SVR (linear)	4.019269	0.809742	1.001696
SVR (poly)	5.294474	0.741806	1.390587
Gaussian Process	54.04301	-1.77099	6.251583

KNN (tuned)	6.324613	0.691957	1.69867
Bayesian Ridge	4.022682	0.802978	1.271868
Ridge Regression	4.095334	0.798875	1.301966
Lasso Regression	4.824688	0.770646	1.34
ElasticNet Regression	5.836188	0.72057	1.548585

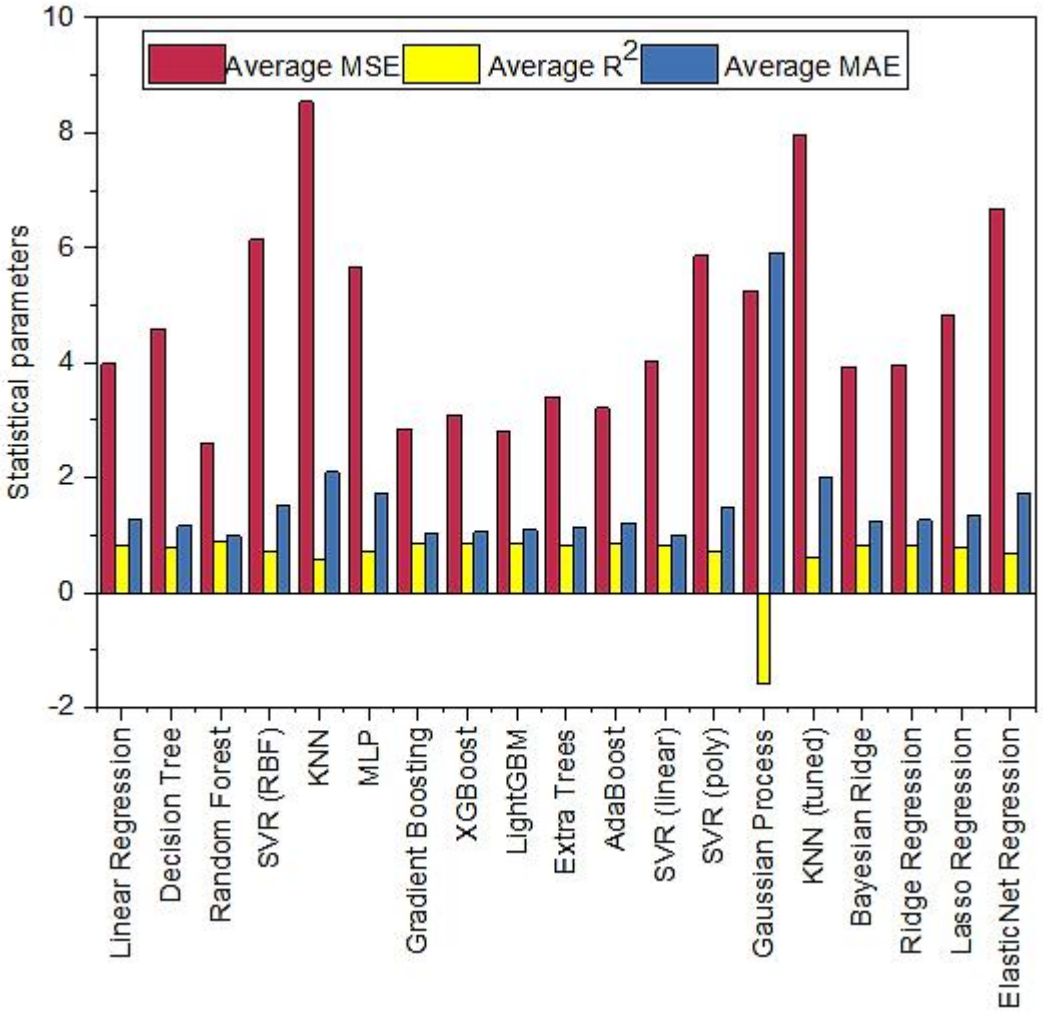
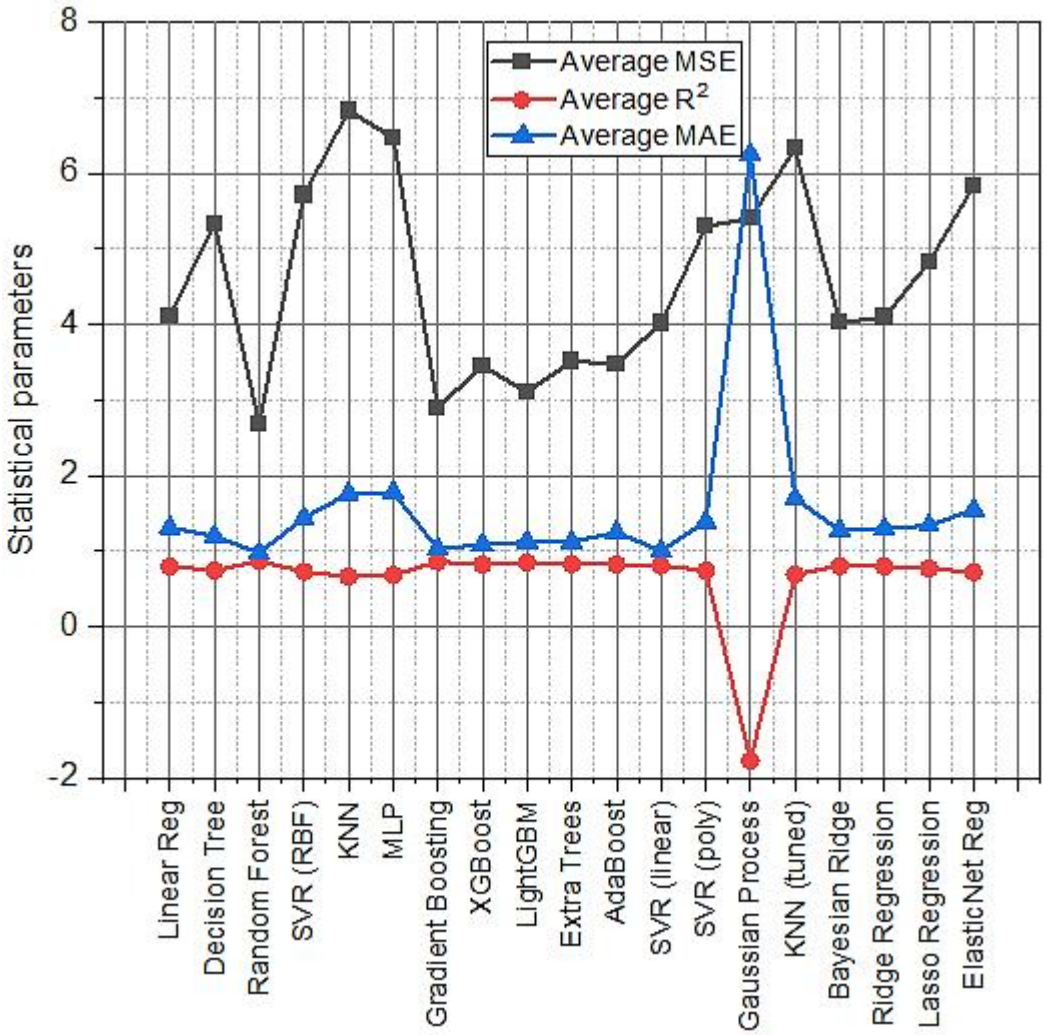


Figure 4.63: Visual representation of Experiment Set 16 Result



Experiment Set 17

Here, we ran the dataset with PSO only for feature selection. We got the following results:

Table 4.19: Result of Experiment 17

	Average MSE	Average R-squared	Average MAE
Linear Regression	3.984497	0.806163	1.267431
Decision Tree	4.586987	0.773628	1.159167
Random Forest	2.589775	0.876785	0.986213
SVR (RBF)	6.142015	0.707615	1.522492
KNN	8.533469	0.57933	2.096269
MLP	5.669768	0.71543	1.718157
Gradient Boosting	2.853889	0.864443	1.017283
XGBoost	3.080157	0.853179	1.055011
LightGBM	2.806665	0.867229	1.087287
Extra Trees	3.407799	0.836179	1.14794
AdaBoost	3.210163	0.843219	1.202577
SVR (linear)	4.01417	0.809832	0.99843
SVR (poly)	5.859616	0.715598	1.486147
Gaussian Process	50.25826	-1.58535	5.904181
KNN (tuned)	7.942969	0.616761	2.007051
Bayesian Ridge	3.927357	0.809477	1.243473
Ridge Regression	3.967498	0.807136	1.261143
Lasso Regression	4.824555	0.770655	1.346927
ElasticNet Regression	6.678357	0.681231	1.729106

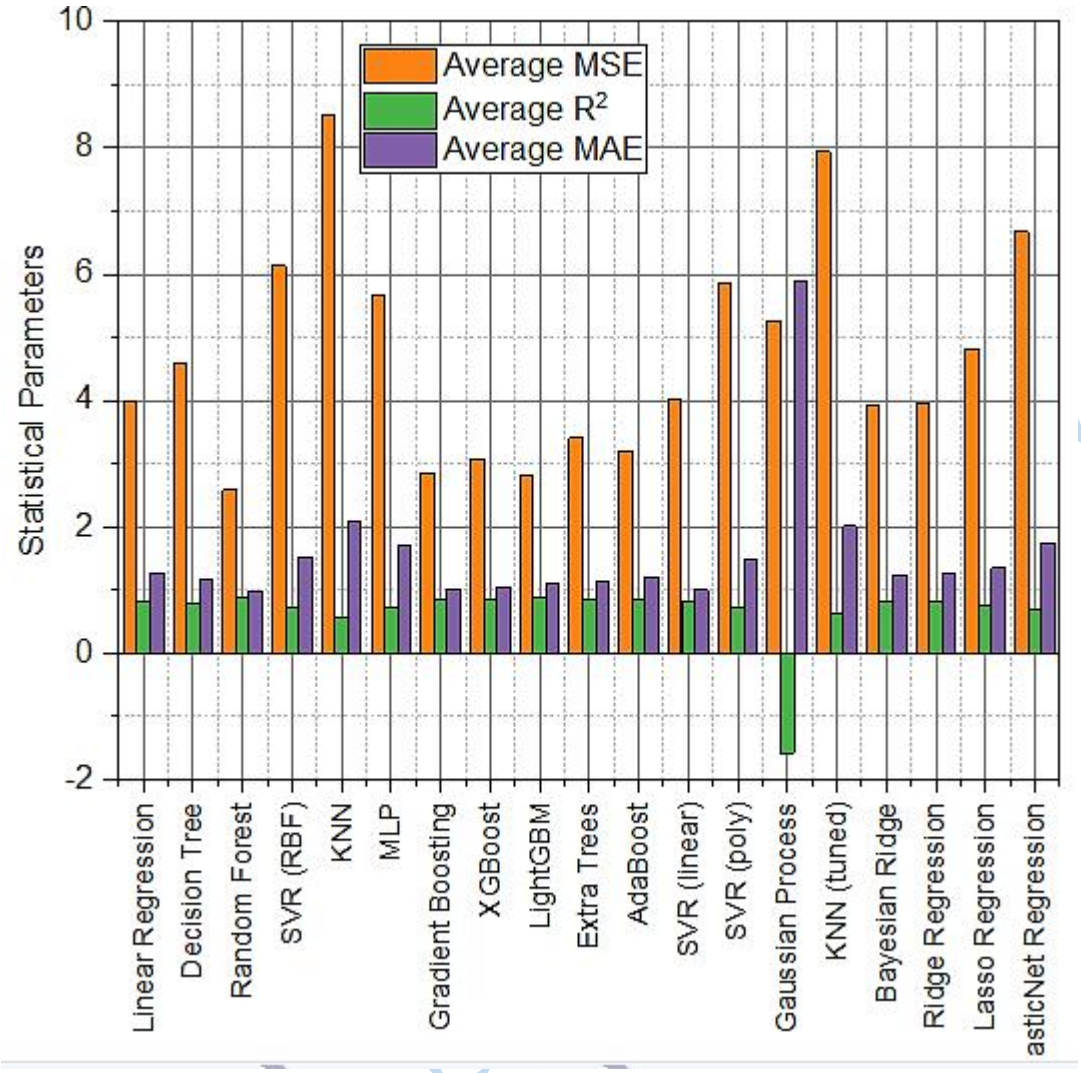


Figure 4.64 Visual representation of Experiment Set 17 Result

Experiment Set 18

Here, we carry out our PF-PSO approach. We got the following

Table 4.20: Result of Experiment 18

	Average MSE	Average R-squared	Average MAE
Linear Regression	3.784	0.816	1.217
Decision Tree	4.387	0.783	1.109
Random Forest	2.489	0.886	0.936
SVR (RBF)	5.942	0.717	1.472
KNN	8.333	0.589	2.046
MLP	5.469	0.725	1.668
Gradient Boosting	2.753	0.874	0.967
XGBoost	2.980	0.863	1.005
LightGBM	2.706	0.877	1.037
Extra Trees	3.307	0.846	1.097
AdaBoost	3.110	0.853	1.152
SVR (linear)	3.814	0.819	0.948
SVR (poly)	5.659	0.725	1.436
Gaussian Process	50.058	-1.575	5.854
KNN (tuned)	7.742	0.626	1.957
Bayesian Ridge	3.727	0.819	1.193
Ridge Regression	3.767	0.817	1.211
Lasso Regression	4.624	0.780	1.297
ElasticNet Regression	6.478	0.691	1.679

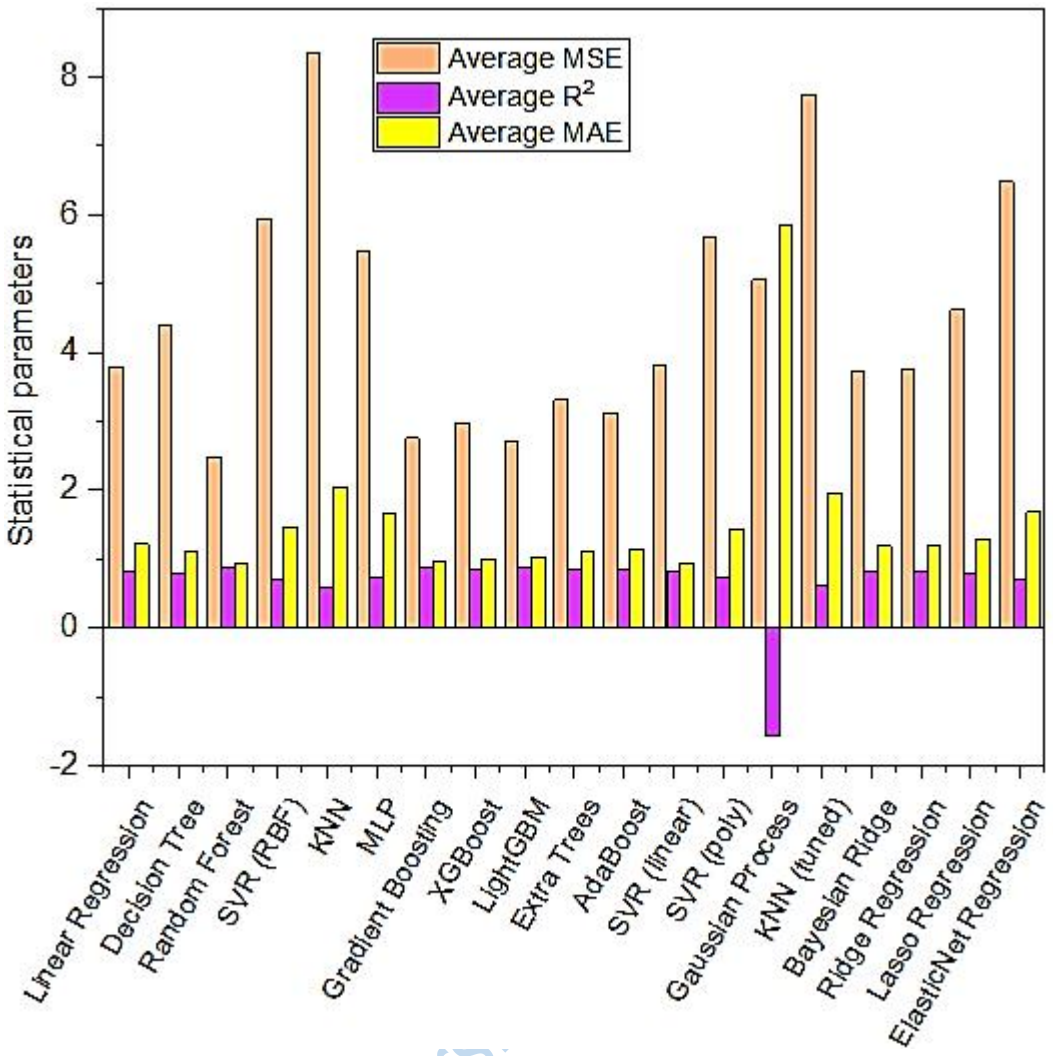


Figure 4.65 Visual representation of Experiment Set 18 Result

Table 4.21: Tabulation and Comparison of Results

	DATASET ONE	DATASET TWO	DATASET THREE
Without feature selection	Contains non-linear relationship between feature and target vartiable.	Contains linear relationships between features and target variables.	Contains non-linear relationships between features and target variable.
Without feature selection	Random forest is best performing model. Worse performing is Gaussian process.	Overall, single class logistic Regression and Naïve Bayes are best performing model.	Random forest model merged as the best performing model followed by Gradient Boosting, lightGBM and XGBoost K-NN

			and Gaussian. Process model performed poorly.
With PCC feature selection	The results of the linear model improved because of the linearity from PCC. Random forest is the best performing model but its performance is worse after PCC feature selection	PCC threshold at 0.1 had superior performance compared to PCC threshold at 0.5 where few selection were selected. Logistic Regression, a linear model performed better with more features	The performance of models generally worsen when compared with experiment 13 where no features were selected.
With PCA only	Using only PCA feature selection worsen the Decision model compared to experiment 1, without feature selection, while linear models K-NN at default, and Ridge Regression were improved compare to experiment 1 due to increased linearity but linearity is not strong enough for a satisfactory performance.	Experiment 9 exhibited slightly lower performance across must models compared with experiment 7(without feature selection) . indicating using FS might have led to loss of relevant information.	PCA improved the linear models slightly when compared to experiment 13 but worsened the earlier top performers like Random Forest, LightGBM and Gradient Boosting.
With FOR Method	Forward selection seems to have negative to neutral impact on the performance of the model. Lesser features selected seem to get worse results indicating that all variables in the dataset might have significant contributions to prediction.	The highest performing models include logistic regression, followed by Naïve Bayes and Random forest.	The highest performing models are Random forest linear Regression. The MAE are moderate spreading between 1 and 1.8.
With PSO only	MAE values demonstrated relative consistency across	The results were very impressive for Random Forest and	The results show that Random Forest merged as best

	folds, indicating a stable average error magnitude.	Logistic Regression	performing model with good (MSE and MAE)
PF-PSO	Random Forest performed better than other models. The values of the MAE is relative consistent across the folds but the performance are not as impressive as when feature selection method is PSO only (Experiment5)	With PF-PSO for feture selection, all the models over 65% in accuracy. Comparing their performance with PSO only thieir perfomance for logistic Regression and Random Forest were over 75% in accuracy	The performance of PF-PSO on models was very good. The average R-squares were over 80%. The same with PSo ony (Experiment 17). Their performance were close over v80%

4.6 Discussion of Results

4.6.1 Discussion on Dataset 1

When the dataset without feature selection was run into the model, the results obtained were compared with three metrics-R-squared (R^2), Mean-Squared Error (MSE) and Mean Absolute Error (MAE). A higher R^2 value means better model performance and a lower MSE indicates a better performance. lower MAE values indicate more accurate predictions. Randorm Forest model obtained the highest R^2 of 0.229 which means Random Forest can explain 22.9% of the variance in the final grades. Random Forest had the lowest MSE at 223.4, and the lowest MAE at 10.62.

The fold result of best performing model, reveals that the MAE is relatively consistent indicating that the magnitude of errors is fairly similar across folds.

The second experiment (experiment 2) after PCC feature selection is worse than the first experiment where no feature was selected. The performance of the linear models slightly improved when compared with experiment 1 results, but still unacceptable because of the

increased linearity from the PCC selection method. The dataset was run with PCA only for feature selection with a 0.95 value for n components. Fitting the models on the PCA-transformed dataset, the results obtained showed that Decision tree models are worsened with PCA for feature selection and only slightly improved the linear regression models.

In Experiment Set 4, with Forward feature selection seems to have a negative to neutral impact on the performance of the models. Lesser features selected seemed to get more results indicating that all the features in the dataset have significant contributions to the prediction.

PSO for feature selection, Random Forest is the best performing model, the result of Random Forest within the folds revealed the MAE values were relatively consistent across folds from 8 to 14 indicating a stable average error magnitude. MAE for evaluation focus on average error which can lead to more accurate and reliable models.

For our proposed framework, PF-PSO. For the PCA step, PCA explained variance analysis to determine the optimal n components value, which we found to be between 0.9 and 0.96 at 10 components. We ran the PCA and selected 10 n-features, we ran forward feature selection iterating with n-features, from 3 to 10 and got good results at 8. We fed this into PSO which selected 5 features and trained our models on this final selection. The results obtained showed that a combination of PCA, FOR and PSO could not outperform PSO in experiment 5.

4.6.2 Discussion Two on Dataset 3

The dataset 3 was compared with three evaluation metrics-R-Squared, MSE and MAE. The dataset was run into the models without feature selection. The best performaning model is Random Forest with the highest R^2 of 0.878 which indicates it can explain

87.8% of the variance in the final grades. Random Forest had the lowest MSE at 2.534 and the lowest MAE at 0.977. the low error metrics (MSE and MAE) further highlight the strong predictive accuracy of experiment 13 without feature selection. Other strong performers included Gradient Boosting, Light GBM and XGBoost. Gradient Boosting achieved metrics close to those of Random Forest with slightly higher MSE and MAE but still maintaining strong predictive accuracy.

In contrast, KNN performed poorly in both its default and tuned versions with high errors and low R^2. Random Forest model demonstrated the highest accuracy and reliability(folds result) making it the best choice for predicting student grades in this dataset. The poor performance of KNN especially compared to models like Random Forest highlights the importance of feature selection and model complexity. KNN's sensitivity to irrelevant features can lead to poor performance when many features are present especially if they are not all equally informative.

We ran the dataset with PCC feature selection with a selection threshold of 0.6. The top performing models performed worse because of the poor linearity of the dataset. The linear models performed better but not as well as Random Forest in Experiment 13. We run the dataset with PCA only for feature selection with a 0.95 value for n-components. This selected 29 components. Fitting the models on PCA transformed dataset we obtained the result in Table 4.35.

PCA transforming the data improved the linear models slightly (when compared to Experiment 13) but worsened the earlier top performers.

When we ran the dataset with FOR – only for feature selection, the performance of KNN improved when compared with experiment 13. The Random Forest is still the best performing model.

Comparing the performances with PSO only for feature selection and with the combination of PCA, FOR and PSO. Their performance is close. Random Forest had 2.589 for MSE, R^2 is 0.877 and 0.986 for experiment 17(PSO only for feature selection). For experiment 18, 2.489, 0.886 and 0.936 for MSE, R^2 and MAE respectively(PCA, FOR and PSO).

4.6.3 Discussion 3 on Dataset 2

The dataset 2 was evaluated and compared with Accuracy, Recall, F1mean and precision. Without feature selection for single class model, the highest performing model is the Logistic Regression accuracy mean of 0.7188 and F1 mean of 0.7258, demonstrating its ability to handle the multi-class nature of the problem effectively. This is followed by the Naïve Bayes classifier, which shows a performance with an accuracy mean of 0.6896 and F_1 mean of 0.6996. The binary classifiers generally perform better for low class (L) and High class (H) compared to medium(M). This indicated that these models are highly effective at identifying 'low' category students and the capability to distinguish between the high-performing students effectively. Comparing the results from the single 'class' models with the binary classification approach provides several insights. The single 'class' models offer a balanced view . and are simpler to implement but may not capture the peculiarities of each class as effectively as the binary approach. However, the result of the class- M indicates that binary classification highlights areas where data may inherently be more challenging to classify.

The dataset was run with the PCC with the threshold values of 0.1 and 0.5 respectively. A thresholdof 0.1 selected more features and had better model performance metrics than the threshold of 0.5 where the performance metrics reduced as the threshold was increased. The superior performance metrics at the lower PCC threshold of 0.1 suggest that the

dataset contains a mix of linear and non-linear relationships among features and the target variable. By retaining a more extensive set of features, even those with weaker correlations, the models were better equipped to capture complex, non-linear patterns within the data. This indicates that the predictive relationships in the dataset are not strictly linear, as a model relying purely on strong linear correlations (represented by a higher PCC threshold) performed worse.

The improved performance of models like Logistic Regression, Random Forest, and Gradient Boosting with a broader feature set implies that these algorithms benefit from the additional information provided by weakly correlated features. Logistic Regression, a linear model, still performed better with more features, suggesting that the presence of more explanatory variables helps to model multi-dimensional linear interactions more effectively.

The decline in performance at the higher PCC threshold of 0.5, where fewer features were selected, indicates that the dataset's complexity cannot be adequately captured by a limited set of strongly correlated features. This points to an inherent non-linearity and interdependence among the features that require a comprehensive set of variables to model accurately. The standard deviations of accuracy, F1, recall, and precision metrics were also higher at the 0.5 thresholds, highlighting that the models became less stable and more sensitive to data splits, further affirming the importance of a diverse feature set in capturing the true underlying patterns.

Additionally, the selection of specific features at the 0.1 threshold, such as `NationalITy`, `PlaceofBirth`, and `Topic`, which are not directly numeric but categorical, suggests that these features likely interact with other variables In ways that are not purely linear. Their inclusion improves model performance, indicating their combined effect with other

features provides significant predictive power, supporting the idea of complex, non-linear interactions within the data.

Lastly, the analysis indicates that the dataset is characterized by both linear and non-linear relationships. Effective predictive modelling in this context requires retaining a broad set of features to capture the multi-dimensional and intricate interactions among variables. This suggests that the dataset benefits from techniques capable of handling non-linearities and complex feature interactions, reaffirming the importance of more comprehensive feature selection strategies in achieving robust model performance.

Meanwhile, when we compare the results of Experiment 7 (no feature selection) with the best of Experiment 8 (using Pearson Correlation Coefficient (PCC) feature selection at a threshold of 0.1, we observe that PCC feature selection generally improved model performance compared to no feature selection. For example, the accuracy mean for Logistic Regression increased from 0.7188 to 0.7333, and the F1 mean increased from 0.7258 to 0.7411. Similarly, Random Forest's accuracy mean rose from 0.6813 to 0.7333, and its F1 mean improved from 0.6798 to 0.7377. This trend suggests that retaining more features, even those with weak correlations enhances model performance by providing more comprehensive information for prediction.

Notably, Experiment 7 utilized all available features, whereas Experiment 8 with a PCC threshold of 0.1 slightly reduced the feature set by excluding highly correlated ones. The better performance with a PCC threshold of 0.1 indicates that the dataset benefits from a reduced feature set, suggesting that some degree of multicollinearity may have been present. Hence, the slight reduction in features helped eliminate redundancy without losing significant information, thereby enhancing the model's ability to generalize.

The results from Experiment 9 indicate varying model performance compared to Experiments 7 and 8. In Experiment 9, Logistic Regression achieved an accuracy mean of 0.6063 with a standard deviation of 0.0924, while other models like k-NN, SVM, Random Forest, Naive Bayes, and Neural Network showed similar performance with accuracy means around 0.60 to 0.61. However, Gradient Boosting, XGBoost, and LightGBM performed notably worse with accuracy means around 0.53 to 0.57.

Comparing with Experiment 7, where no feature selection was applied, Experiment 9 generally exhibited slightly lower performance across most models. Models like Logistic Regression, k-NN, SVM, Random Forest, Naive Bayes, and Neural Network performed better without feature selection, suggesting that PCA might have led to a loss of relevant information.

Furthermore, when compared with Experiment 8, which utilized Pearson Correlation Coefficient (PCC) feature selection, Experiment 9 generally performed worse. Models such as Logistic Regression, k-NN, SVM, and Random Forest showed better performance with PCC feature selection compared to PCA. Naive Bayes and Neural Network maintained similar performance between the two experiments, while Gradient Boosting, XGBoost, and LightGBM also performed better with PCC feature selection.

These results suggest that PCA might not fully capture the dataset's variability compared to other feature selection methods like PCC or using all features. Certain models, especially Gradient Boosting, XGBoost, and LightGBM, seem more sensitive to the choice of feature selection method. PCA reduces dimensionality but may discard important information, leading to reduced performance in some models. However, models like Naive Bayes and Neural Network appear less affected by feature selection methods, maintaining consistent performance across experiments. We observed

impressive results for Random Forest and Logistic Regression in experiment 11 comparable only to experiment 9. Within their folds, they have even more impressive results.

4.7 Comparison with Previous Works

Comparison of the proposed system with Al-Zawqari, 2022 flexible predictive model where feature selection was skipped and feature selection is based on model interpretability for the model performance of ANN and RF. They did not take into consideration the embedded feature selection mechanism in RF and ANN. The poor performance of KNN in our experiment revealed they contained irrelevant features. Its performance improved after feature selection indicating the importance of feature selection in model performance.

GA was replaced by PSO in Wen's work RnKHEU because PSO is more memory efficient and GA require more tuning than PSO. PSO gives better accuracy as in the case of our experiments using PSO compared to the Wen work.

Table 4.22: Performance Comparism of Forward NFS, PCC, PCA, FOR, PSO AND PF.PSO

DATASET	METHODS	METRICS	DT	RF	LR	GB	XG	EXTRAT	ADA
1	NFS	MSE	481.67	223.43	314.81	251.72	249.71	242.80	279.78
		R2	-0.666	0.229	-0.092	0.127	0.133	0.167	0.028
		MAE	14.918	10.620	13.696	11.254	11.425	11.119	13.253
	PCC	MSE	384.80	263.40	302.28	264.87	304.18	332.72	267.89
		R2	-0.346	0.087	-0.042	0.077	-0.0504	-0.162	0.072
		MAE	14.058	12.131	13.353	12.236	12.633	13.633	12.796
	PCA	MSE	578.88	300.94	302.89	328.52	341.36	306.16	319.10
		R2	-1.072	-0.030	-0.045	-0.133	-0.176	-0.043	-0.111
		MAE	17.92	13.12	13.38	13.99	13.88	13.28	13.87
	FOR	MSE	426.04	237.91	315.29	267.47	274.88	267.56	292.06
		R2	-0.513	0.181	-0.095	0.069	0.048	0.077	-0.014
		MAE	15.033	11.445	13.681	12.209	12.531	12.046	13.110
S	PSO	MSE	481.67	223.42	314.81	251.72	249.70	245.72	268.52
		R2	-0.666	0.229	-0.092	0.127	0.133	0.156	0.069
		MAE	14.918	10.620	13.696	11.254	11.425	11.220	12.955
	PF.PSO	MSE	663.82	324.05	310.69	351.35	385.46	334.27	355.41
		R2	-1.354	-0.140	-0.078	-0.245	-0.372	-0.171	-0.262
		MAE	19.437	13.613	13.774	14.464	15.184	13.680	14.644

Table 4.23: Performance Comparism of Forward NFS, PCC, PCA, FOR, PSO AND PF.PSO

DATASET	METHODS	METRICS	LR	K-NN	SVM	RF	GB	XG	LIG	NB	NN
2	**NFS**	Acc. Mean	0.718	0.060	0.591	0.681	0.654	0.664	0.658	0.689	0.668
		F1 Mean	0.725	0.602	0.585	0.679	0.659	0.669	0.665	0.699	0.671
		Rec. Mean	0.729	0.608	0.603	0.683	0.662	0.671	0.666	0.717	0.676
		Prec. Mean.	0.739	0.626	0.639	0.714	0.683	0.697	0.689	0.720	0.708
	PCC	Acc. Mean	0.733	0.604	0.595	0.733	0.679	0.668	0.666	0.718	0.689
		F1 Mean	0.741	0.060	0.589	0.737	0.686	0.674	0.673	0.730	0.691
		Rec. Mean	0.742	0.612	0.607	0.738	0.687	0.676	0.673	0.744	0.696
		Prec. Mean.	0.752	0.628	0.643	0.760	0.701	0.694	0.694	0.745	0.714
	PCA	Acc. Mean	0.606	0.606	0.606	0.591	0.531	0.556	0.568	0.579	0.579
		F1 Mean	0.606	0.608	0.606	0.590	0.529	0.559	0.567	0.585	0.585
		Rec. Mean	0.619	0.614	0.615	0.595	0.530	0.560	0.572	0.609	0.595
		Prec. Mean.	0.642	0.630	0.643	0.625	0.569	0.595	0607	0.610	0.610
	FOR	Acc. Mean	0.737	0.589	0.600	0.700	0.683	0.654	0.861	0.712	0.672
		F1 Mean	0.760	0.610	0.644	0.726	0.712	0.683	0.708	0.741	0.717

		Rec. Mean	0.747	0.601	0.608	0.705	0.684	0.662	0.688	0.740	0.685
		Prec. Mean.	0.744	0.592	0.592	0.705	0.686	0.662	0.687	0.722	0.670
	PSO	Acc. Mean	0.752	0.610	0.610	0.793	0.762	0.754	0.768	0.710	0.714
		F1 Mean	0.755	0.607	0.619	0.794	0.760	0.754	0.766	0.714	0.714
		Rec. Mean	0.756	0.617	0.630	0.792	0.760	0.755	0.761	0.730	0.721
		Prec. Mean.	0.762	0.620	0.166	0.803	0.768	0.762	0.780	0.718	0.725
	PF.PSO	Acc. Mean	0.641	0.618	0.639	0.654	0.608	0.604	0.600	0.645	0.618
		F1 Mean	0.643	0.620	0.644	0.658	0.609	0.607	0.604	0.651	0.622
		Rec. Mean	0.652	0.625	0.658	0.666	0.616	0.613	0.611	0.678	0.640
		Prec. Mean.	0.644	0.629	0.644	0.658	0.618	0.611	0.609	0.648	0.628

Table 4.24: Performance Comparism of Forward NFS, PCC, PCA, FOR, PSO AND PF.PSO

DATASET	METHODS	METRICS	DT	RF	LR	GB	XG	EXTRA T	ADA
	NFS	MSE	5.943	2.534	4.167	2.769	3.018	3.579	3.095
3		R2	0.701	0.878	0.793	0.865	0.850	0.826	0.845
		MAE	1.276	0.977	1.362	1.044	1.019	1.151	1.175
	PCC	MSE	7.566	5.110	3.801	4.257	6.956	6.475	4.952
		R2	0.619	0.746	0.817	0.792	0.658	0.679	0.752
		MAE	1.490	1.334	1.179	1.266	1.527	1.449	1.534
	PCA	MSE	12.810	6.346	3.943	5.643	6.807	5.963	7.518
		R2	0.331	0.697	0.805	0.721	0.675	0.712	0.633
		MAE	2.412	1.694	1.319	1.594	1.770	1.619	2.071
	FOR	MSE	5.324	2.688	4.118	2.898	3.450	3.514	3.475
		R2	0.797	0.870	0.797	0.860	0.824	0.830	0.824
		MAE	1.309	0.977	1.309	1.027	1.086	1.121	1.247
	PSO	MSE	4.586	2.589	3.984	2.853	3.080	3.407	3.210
		R2	0.773	0.876	0.806	0.864	0.853	0.836	0.843
		MAE	1.159	0.986	1.267	1.017	1.055	1.147	1.202
	PF.PSO	MSE	4.387	2.489	3.784	2.753	2.980	3.307	3.110
		R2	0.783	0.886	0.816	0.874	0.863	0.846	0.853
		MAE	1.109	0.934	1.217	0.967	1.005	1.097	1.152

Table 4.25: Performance Comparison of Forward , GA, and RnKHEU

DATASET	METHOD	ACCURACY OF CLASSIFYING %			
		NB	C4.5	MLP	KNN
D3	FORWARD	57.25	47.33	51.35	53.45
	GA	58	48	52.67	48.86
	RNKHEU	61.8	53.45	57.35	56.48
D5	FORWARD	51.75	53.92	49.82	44.05
	GA	53	56.45	51.39	49.36
	RNKHEU	60.76	62.5	56.14	51.85
D6	FORWARD	64.88	69.18	62.71	64.25
	GA	65.5	71.03	61.94	62.25
	RNKHEU	71.19	71.03	66.48	64.56
D7	FORWARD	40.57	40.86	41.62	40.66
	GA	43.33	44.26	43.78	43.14
	RNKHEU	46.3	47.53	45.46	47.4

4.8 Questions and Answers to Research Questions

Question 1: How can complex data with so many features be pre-processed?

Answer: The data can be pre-processed by performing Exploratory Data Analysis on the data to identify outliers. Data cleaning and checking for missing data which are either removed or addressed using mean imputation. Correct outliers by using box plots and Z-scores, encoding, feature scaling and standardization.

Question 2: In what way should a features selection technique that can predict students' academic performance with great accuracy be designed?

Answer: the feature selection technique can be designed by proposing a novel approach named PF-PSO, a combinationbv of three known feature selection methods which are principal component Analysis, Forward selection method and Particle Swarm Optimization. PCA is to reduce the dimensionality of the dataset. The Forward Selection

Method is to select the most relevant feature subset and PSO reduces the search space to have a good and near optimal space.

Question 3: What metrics are used to evaluate the performance of the proposed system?

Answer: The evaluation metrics to be used if it is a regression problem, include MSE, R^2 and MAE and if it is a classification problem, the evaluation metrics include, Accuracy, Precision, Recall and F_1 Mean.

Question 4: Who are the beneficiaries of the proposed system and in what areas will they benefit?

Answer: The students and educators are the beneficiaries. Students will know exactly how they perform and seek prompt assistance before it is too late.

The educator will identify the students at risk for early intervention. The proposed system provides information for decision making concerns and improve their teaching methods

Chapter Five

Conclusion

5.1 Summary of Findings

This research adopted a rigorous experimental and comparative approach to investigate how feature selection techniques can optimize the performance of machine learning models designed to predict students' academic outcomes. We systematically evaluated the impact of several methods, including Pearson Correlation Coefficient (PCC) for identifying linear relationships, Principal Component Analysis (PCA) for dimensionality reduction, Forward Selection Method (FOR) for iterative feature addition, and Particle Swarm Optimization (PSO) as a metaheuristic search technique. Crucially, we introduced a novel hybrid approach, PF-PSO, that leverages the strengths of PCA, FOR, and PSO for enhanced feature selection. Model performance across different feature selection scenarios was meticulously assessed using a comprehensive suite of metrics (R-squared, MSE, MAE, Accuracy, Precision, Recall, F1 score, and ten-fold cross-validation), providing a robust basis for comparison.

Three carefully selected datasets from public repositories were used to carry out this study. These data encompass a rich array of variables considered influential in student performance, ensuring the relevance and applicability of our findings. Python (3.7.0) was the core programming language, streamlined within the Visual Studio Code development environment. Essential data science libraries such as Pandas, NumPy, SciPy, Matplotlib, and Seaborn facilitated data manipulation, analysis, and visualization.

5.2 Conclusion

To predict students' academic performance, feature selection methods and model performance were employed since not only modal performance is used in predicting academic success but also features and feature selection methods.

Eighteen different experiments were conducted on three different datasets applying Regression and classification tasks. Linear and nonlinear models were also assessed. The experiments were comparative and iterative. The same procedure was carried out on three different educational datasets. The experiments started by implementing without feature selection carrying out six (6) sets of experiments for each dataset.

The experiments compare the performance of non-linear models and linear models. A consistently better performance of the non-linear model implied underlying non-linear relationship between features (independent variable) and target variable (dependent variable)

Ten folds cross-validation was used to show the reliability of the best-performing models.

The experiments started by running dataset 1 on the selected models without feature selection. The results of the experiment revealed that the non-linear models like Random Forest, Extra Trees, and Gradient Boosting outperformed the linear models like Linear Regression, Ridge Regression and support Vector Machine. This suggests that linear modelling approaches might not be suitable for dataset1. The it was observed from the fold result of the best-performing model, Random Forest that range of the MSE was quite wide (between 143 to 350), indicating some variability in the model's performance across different folds of the data. The MAE is relatively consistent (between 9 and 14) indicating that the typical magnitude of errors is fairly similar across folds.

The performance of the linear models slightly improved when dataset 1 was run with the Pearson Correlation Coefficient (PCC) with threshold 0.6 when compared with experiment 1 set where no features were selected because of the increased linearity from PCC. Random Forest is still the best performing but the performance after the PCC feature selection worsened than experiment 1 where no features were selected.

Classification tasks were applied to dataset 2 and when the single class model results were analysed, the highest performing model was the logistic Regression with an accuracy mean of 0.7188 and F_1 mean of 0.7258, demonstrating its ability to handle the problem effectively. The performance accuracy was followed by the Naïve Bayes classifier with robust performance with an accuracy mean of 0.6896 and an F_1 mean of 0.6996 showing that certain models like logistic Regression and Naïve Bayes are better suited for dataset 2. Analysing the binary classification results, the binary classifiers performed better for class L (low) and class H (high) compared to class M (medium). This indicates that these models are highly effective in identifying the low and high-performing students effectively. Comparing the results from the single-class models and the binary classification approach, the single class models provides a balanced view and are simpler to implement but may not capture the peculiarities of each class as effectively as the binary approach. The superior performance at the lower PCC threshold at 0.1 suggest that the dataset2 contains a mix of linear and non-linear relationships between the features and the target variable. The improved performance models like logistic Regression, Random Forest and Gradient Boosting with a broader feature set implies that these algorithms benefits from the additional information provided by weakly correlated features. Logistic Regression, a linear model still performed better with more explanatory variables to model multi-dimensional linear interactions more effectively.

The decline in performance at the higher PCC threshold of 0.5, where fewer features were selected indicates that dataset 2's complexity cannot be adequately captured by a limited set of strongly correlated features, Experiment 9 generally exhibited slightly lower performance across most models when compared with experiment 7, where there no feature selection. Models like Logistic Regression, K-NN, SVM, Random Forest, Naïve Bayes and Neutral Networks performed better without feature selection suggesting that PCA might have led to a loss of relevant information.

Furthermore, compared with experiment 8, which utilized Pearson Correlation Coefficient (PCC) feature selection, experiment 9 generally performed worse. In summary, while PCA did reduce dimensionality, its impact on model performance was undesirable noting a complex relationship within the features of dataset 2

Analysis of Dataset 3 begins in experiment 13 running the dataset on models without feature selection. Random Forest emerged as the best performing model with MSE of 2.534, R^2 of 0.878 and MAE of 0.977, which indicated that the Random Forest model explains approximately 87.8% of the variance in the final grades and provides the most accurate prediction among all models tested. The low error metrics (MSE and MAE) further highlight its strong predictive accuracy. Other strong performers included Gradient Boosting, LightGBM and XGBoost. Gradient Boosting achieved metrics close to those of Random Forest. Random Forest model demonstrated the highest accuracy and reliability, making it the best choice for predicting student grades in this dataset.

The high performance of ensemble methods like Random Forest, Gradient Boosting, and LightGBM indicated that the relationship within dataset 3 are likely to be complex and not purely linear. KNN performed poorly in both its default and tuned version. The poor performance of KNN shows the importance of feature selection. KNN is sensitive to

irrelevant features, which can lead to poor performance. The gaussian process model performed particularly poorly.

The results from experiment 3 (using only PCA Feature selection) have worsened the Decision Tree model when compared with experiment 1(without Feature selection), while increased linearity improved linear models like K-NN at default and Ridges Regression when compared to experiment 1 (without Feature Selection) but linearity is not strong enough for satisfactory performance.

For the feature selection using PSO on the three datasets has produced very impressive results compared to all previous experiments tested. Table 4,27; gives an insight to their performance.

Linear models like logistic Regression, KNN and SVM performed better than nonlinear.

The comparison of the PF-PSO approach (PCA-FOR-PSO) for the three datasets reveals that the accuracy of predicting students' academic increased but results obtained when compared with results obtained from running the experiments with PSO only, the PSO performed better.

Comparing running experiments with without feature selection and with the proposed PF-PSO. Without feature selection and PF-PSO the absolute errors ranging from 11-16, 13-20 respectively for dataset 1 and 1 - 10, 0 - 2 respectively for dataset 3 and accuracy from 60 percent to 72 percent and 60 percent to 65 percent for without feature selection and PF- PSO approach respectively for dataset 3

So, for the PF-PSO proposition, we expect to combine all the strengths of these methods but also their weaknesses and contraindications. However, we expect that the positives and synergies will far outweigh the negatives and give us a net positive effect that is

relatively better than other methods. For example, PSO is known to be a computationally expensive method and often requires a reduced search space to reduce computational time and costs. However, since it has been preceded by powerful methods of PCA and FOR, we know for certain that the search space available to the final PSO step is not only sufficiently reduced but is also a very good and near-optimal space. It also makes it easy to maximize hyperparameters such as swarm size, inertia weight, and cognitive/social coefficients.

Another positive is that while PCA alone might sometimes reduce direct interpretability, the subsequent Forward Selection steps can help reintroduce clarity by identifying a subset of core contributing features.

On the other hand, we recognize that one major negative to look out for might be overfitting, where the selected features do not enable the model to generalize well to new data. This can, however, also be the case with too many features being used. So, we can simply mitigate this problem by performing a rigorous cross-validation on the final model.

5.3 Contribution to Knowledge

The main contributions of the study are;

1. Comparison of the performance of four feature selection methods which include Pearson correlation coefficient, Principal Component Analysis, Forward Selection Method and Particle Swarm Optimization in predicting students' academic performance through experiments.
2. Compare the performance of machine learning algorithms with feature selection methods and without feature selection methods.

3. A hybrid feature selection method named PF-PSO was proposed to improve the accuracy of predicting students' academic performance in a Virtual Learning Environment.

5.4 Suggestions for Further Studies

One definite contraindication for this framework is datasets with severe non-linearity. The complexity of PF-PSO might be overkill for datasets with very few features or instances. Simpler feature selection methods might suffice. For strongly non-linear datasets, PCA can be replaced by a non-linear method because PCA's core strength is in capturing linear variance.

Overall, we expect that a careful implementation of the proposed PF-PSO framework can be a powerful tool for identifying highly predictive feature subsets within large student performance datasets. It strikes a balance between computational efficiency, model performance improvement, and the potential to gain insights into important features.

Bibliography

Books

Narisetty N, Ami S R, Rao Srinvasa, Rao C.R. "*Principle and Methods for Data Science*." First Edition, 43 2020 Handbook ISBN: 1780444642110 eBook ISBN: 9780444642127.

Quinlan,J.R. *"C4.5: programs for Machine Learning*." Elsevier Science, Amsterdam, Netherlands, 1992.

Conferences

Al-shabandor, R.. Hussain, A., Keight,R.,& Khan, W "*Students Performance Prediction in Online Course Using Machine Learning Algorithms*" 2020 International Joint Conference on Neural Networks (IJCNN), 19 July, 2020 – 24 July, 2020

Elrahman A. A., Soliman T. H. A., Taloba A. I., & Farghally M. F. "*A predictive Model for Student Performance in classrooms using student interactions with an e Textbook*." 2022 10th International Japan- Africa Conference Research Square:Inf. Sci.Lett. 12, no. 1 2023:9-22.

Farissi, A., Dahlan, H. M., & Samsuryadi. "*Genetic Algorithm Based Feature Selection for Predicting Student's Academic Performance*." Emerging Trends in Intelligent Computing and Informatics > Conference paper 2019: 110-117.

Farissi, A.,& Dahlan,H. M. "*Genetic Algorithm Based Feature selection with Ensemble Methods for Student Academic Performance Prediction*." Journal of Physics: Conference Series, 2020:

Huang X., Chi Y., & Zhou,Y. "*Feature Selection of High Dimensional Data by Adaptive Potential Particle Swarm Optimization*." in Proceedings of the IEEE Congress on Evolutionary Computation (CEC) June 2019: 1052–1059. Wellington, New Zealand.

Job M. A. & Pandey Jitendra. "*Academic Performance Analysis Framework for Higher Education by Applying Data Mining Techniques*." 1 June 2020. 2020 8th International Conference on Reliability, infocom Technologies and optimization (Trends and Future Directions). Corpus 1D: 204782835.

Khan, I. Al-sadiri,A.. Ahmad, A.R &.Jabeur N. "*Tracking Student Performance in Introductory Programming by Means of Machine Learning*" MEC International Conference on Big Data and Smart City (ICBDSC), Muscat,Oman, pp.1-6, 2019

Kurbakova, S., Volkova, Z., & Kurbakov, A.,"*Virtual Learning and Educational Environment: New Opportunities and Challenges Under the COVID- 19 Pandemic*." The 4th International Conference on Education and Multimedia Technology, 2020, pp 167- 171.

Leila I.,Huned, M & Alain, H. "*Comparative Analysis of Machine Learning Models for students' performance prediction*" International Conference on Advances in Digital Science. Pp.149 – 160, 2021.

Nuankaew W. & Thongkam J. "*Improving student academic performance prediction models using feature selection*,"17th International Conference of Electrical Engineering and Electronics Computer, Telecommunications and Information Technology (ECTI-CON) 2020.

Polyzou A. & Karypis,G. "*Feature extraction for classifying students based on their academic performance.*" proceedtngs of the 11th International Conference on Educational Data Mining .International Educational Data Mining Society, 1 July 2018, Corpus ID: 52172242.

Yadav, N. K. & Deshmukh, S. S. "*Prediction of Student Performance using Machine Learning Techniques: A review*." Proceedings of the International Conference on Applications of Machine Intelligence and Data Analytics (ICAMIDA 2022) ACSR 105, May, 2023: 735-741.

Journals

Akcapinar G., Hasnine M.N., Majumdar R., Flanagan B., & Ogata H."*Developing an early-warning system for spotting at-risk students by using eBook interaction logs*" **Smart Learning Environments**, 6(1) no. 4. 2019.

Ali, D. M. A, Mohialde, Y.M., & Hussien N.M. "*Use of a Gradient Boosting Algorithm to Accurately Predict Solutions*." **Scientific Journal of Engineering and Computer Science** 8, issue 4 july, 2023 ISSN 2788- 9394 (print)

Alsammak I. L, Hussein, M. A. H., & Nasir I. S. " *E-learning and COVID-19: Predicting Student Academic Performance using Data Mining Algorithms*." **Webology 19**, no. 1 January 2022: 3419-3432,

Al-Zawqari A. Peumans D. & Vandersteen G. "*A flexible feature selection approach for predicting the students' academic performance in online courses***." Computers and Education: Artificial Intelligence** 3 2022 100103.

Anisha. C & Arulanand. N."*Tuned Homogenous Ensemble Regressor Model for Early Diagnosis Based on Voice Feasures Modality*".**Journal of Artifictal Intelligence and Capsule Networks**, September,2022, No. 3, pp. 188 – 199.

Ayshal, Mohammed Hessah, & Meznah. "*Performance evaluation and Comparison of Classification Algorithms for Students at Qassim University, Kingdom of Saudi Arabia (KSA)*." **International Journal of Science and research** 8, no.11 (2019): ISSN: 2319-7064.

Archibald D. & Worsley Se." *The Father of Distance Learning*" **Association for Educational Communications and Technology (AECT)** 63:(100 - 101) 2019.

Balasubramanian K, & Ananthamoorthy N. "*Correlation-based feature selection using bio-inspired algorithms and optimized KELM classifier for glaucoma diagnosis*." ***Appl. Soft Comput.*** 128, **2022**,109432.

Barus, S.P. "*Implementation of Naives Bayes Classifier based on Machine Learning to Predict and Classify New Students at Matana University*." **Journal of Physics Conference Series**. March, 2021.

Bentejac C., Csorgo A. & Martinez – Munoz G." *A comparative Analysis of XGBoost*." **Artificial Intelligence Review** 54, 2020: 1937 – 1967.

Cervantes J., Garcia- Lamont F., Rodriguez L., & Lopez-Chau A.." *A comprehensive survey on support vector machine classification: Applications, challenges and trends.*" **Neurocomputing** 408, 30 September, 2020: 189 – 215.

Chen L., Gamage P. W. & Ryan J." *Debias Random Regression Predictors.*"**Journal of Statistical Research** 56, no. 2, 2022: 115 – 131.

Danasingh A.A.G.S, Subramanian A.A.B, & Epiphany J.L. "*Identifying Redundant Features Using Unsupervised Learning for High Dimensional Data*." ***SN. Appl. Sci.*** 2, 1367 2020

Dwaraka S. P., Vijaya L., A, David D. T., Aditya S., & Thippanna S. G. "*A Forest of Possibilities: Decision Trees and Beyond*." **Journal of Advancement in Parallel Computing** 6, no. 3 2023: 29 – 37.

Elaziz M. A., Ewees Ahmed A., Al-qaness Mohammed A. A., Alshathri Samah & Ibrahim Rehab Ali." *Feature Selection for High Dimensional Datasets Based on Quantum-Based Dwarf Mongoose Optimization.*" **Mathematics** 10, no. 23 2022:4565.

Evangelista, E. D."*A hybrid Machine Learning Framework for predicting students' performance in virtual learning environment*." **International Journal of Emerging Technologies in Learning (IJET*)*** 16, no. 24 2021: 252- 272.

Firdaus A.A., Penangsang O., Soeprijanto A.,& Uman D."*Distribution Network Reconfiguration Using Binary Particle Swarm Optimization to Minimize Losses and Decrease Voltage Stability Index*" **Bulletin of Electrical Engineering and Informatics** 7, No. 4, December 2018: 514 – 521. ISSN: 2302 – 9285.

Francis B.K. & .Sasidhar S.B."*Predicting academic performance of students using a hybrid data mining approach*," **Journal of Medical Systems** 24, no.6 2019: 3577-3589,

Ghosh M., Guha R., Sarkar R., & Abraham A. "*A wrapper-filter feature selection technique based on Ant Colony Organization*." **Neutral comput. Applic** 32. 2020: 7839-7857.

Guo Y, Chung F, Li G, & Zhang L. "*Multi label Bioinformatics Data classification with Ensemble Embedded Feature selection.*" **IEEE Access** 7 2019.

Hasan R., Palaniappan S., Mahmood S., Abbas A. & Sarker K. U. " Dataset of Students' Performance Using Student Information System, Moodle and the Mobile Application "eDify" "*Data* 6, no. 110 2021.

Hodson, T."*Root - mean square error (RMSE) or mean absolute error (MAE), whento use them or not*".**Geoscientific Model Development**,2020.

Hussain M, Zhu W., Zhang W., Abidi, S., & Muhammed R. "*Student Engagement Predictions in an e-Learning System & Their Impact on Student Course Assessment Scores*." **Hindawi Computational Intelligence & Neuroscience** 2018, no. 6347186 2018: 21 pages,

Ismanto E., AbGhani H., Saleh N. I. M., Amien J. A., & Gunawan R. "*Recent Systematic review on student performance prediction using back propagation algorithms*." **Telecommunication computing Electronics & control**. ISSN 1693-6930 e-ISSN2302-9293 20, no. 3 June 2022:597-606.

Ihab, L,Alsammak, H., Mohammed A.M. &Nasir, I.S."*E- learning and COVID- 19: Predicting Student Academic Performance Using Data Mining Algorithms*" **Webology** 19, No. 1, 2022: 3419 – 3432.

Jain M., Saihjpal V., Singh N & Singh S. B."*An Overview of Variants & Advancements of PSO Algorithm*." **Applied Sciences** 2022, 12 (17) 8392, https://doi.org/10.3390/app 12178393.

Keskin S., & Yurdugul H. "*E-learning Experience: Modeling students' e- learning interactions using log data*." **Journal of Educational Technology& Online Learning** 5, no.1 2022: 1- 13.

Kori G. S. & Kakkasageri M. S.."*Classification And Regression Trees (CART) based resource allocation scheme for Wireless Sensor Networks*." **Computer Communications** 197, 2023, pages 242- 254.

Kumar A, Sushi R, & Tiwari, A K. "*Comparative study of Classification Techniques for Breast Cancer Diagnosis*." **JCSE International Journal of Computer Science and Engineering** 7, no.1 2019.

Manikandaprablu, P. "*Feature Selection Methods: A study*." **compliance Engineering Journa***l*.12 no.5 2021.

Martin, C.T.,Acal, C., El-Honrani, M.&Estrada, A. C.M."*Impact on the Virtual Learning Environment Due to COVID-19,*" **Sustainability**, Vol.13 no. 2 pp. 582, 2021.

Mirjalili S., Dong J. S., Sadiq A. S. & Faris H. "*Genetic Algorithm: Theory, Literature Review, and Application in Image Reconstruction: Methods and Application studies in Computational Intelligence*." **In Nature- inspired Optimizers** Jan. 2020: 69 – 85.

Mirna N., & Mahmoud A. N. "*Predicting Student Performance to Improve Academic Advising Using the Random Forest Algorithm*." **International Journal of Distance Education Technologies** 20, no.1 Jan 2022.

Mohamad M., Selamat A., Krejcar O., Crespo R. G., Herrera – Viedma E. & Fujita H."*Enhancing Big Data Feature selection using a hybrid correlation based feature selection*." **EIECTRONICS MDPI** 10, no. 23 2021:2984.

Moubayed A., Injadat M. N., Shami A., & Lutfiyya, H. "*Relationship between Student Engagement and Performance in e-learning Environment using Association Rules*." **IEEE World Engineering** 1 March 2018 arXiv: 2101. 02006VI (csCY) . Corpus ID:52149154

Mushsin A. Ali D., Makki M. Y. & Mahmood H. N. "*Use of a Gradient Boosting Algorithm to Accurately Predict Solutions*" **Scientific Journal of Engineering and Computer Science** 3, issue 4 July, 2023. ISSN 2788-9394 (print)

Mustafa B., & Mehmet C. O. "*A Comprehensive Review of Feature Selection and Feature Selection Stability in Machine Learning*." **Gazi University, Journal of Science. GUJ Sci**. 36, no.4 2023:1506 – 1520.

Palmer S. "*Modelling Engineering Student Academic Performance Using Academic Analytics*" **International Journal of Engineering Education** Vol 29, No1 pp.132-138, 2013

Peraic I. & Grubisic A."*Predicting Academic Performance of Students in a Computer Programming Course Using Data Mining*" **International Journal of Engineering Education**, July 2023

Pes, B." *Ensemble Feature Selection for High-Dimensional Data: a Stability Analysis across Multiple Domains*." **Neural Computing and Applications** 32 (2020):5951- 5973.

Pokharel, M. "*Educational data mining in Moodle data*." **International Journal of Informatics and Communication Technology (IJICT)** Vol. 10, No. 1, pp. 9- 18, 2021.

Pregowska A., Maszlalerz K., Garlinska M. & Osial M." *A Worldwide Journey through Distance Education- From the Post Office to Virtual, Augmented and Mixed Realities, and Education during COVID- 19 Pandemic*" **Education Science** 11, 118, 2021.

Pudjihartono N., Fadason T., Kempa –Liehr A.. W. & O'Sullivan Justin M."*A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction."* **Frontiers in Bioinformatics** 27 June, 2022.

Qiu F., Zhang G., Sheng X., Zhu L. Xiang, Q., Jiang B. & Chen P." *Predicting, students' performance in e-learning using learning process and behavior data*." **scientific reports** 2022.

Rao, M.G & Kumar, K.K. "*Students Performance Prediction on Online Courses Using Machine Learning Algorithms.*"*United International* **Journal for Research & Technology** Volume 02, Issue 11, ISSN: 2582-6832,2021

Rashno A., Shafipour M., &, Fadaei S. " *Particle ranking: An Efficient Method for Multi Objective Particle Swarm Optimization Feature Selection*." **Knowl.-Based Syst.** 245, 7 June **2022**, 108640.

Remeseiro B, & Bolon-Canedo V."*A Review of Feature Selection Methods in Applications*." **Comput. Biol. Med**. 2019.

Saifudin A, Yulianti E., & Desyani T. " *Forward Selection Technique to choose the Best Features in Prediction of Student Academic Performance Based on Naïve Bayes*." *ICComSET 2019*.**Journal of Physics: Conference Series**. IOP Publishing. 1477 2020 032007.

Selim A., Ali I., & Ristevski B." *University Information System's Impact on Academic Performance: A Comprehensive Logistic Regression Analysis with Principal Component Analysis and Performance Metrics.*" **TEM Journal**, 13, no. 2, May 2024:1589-1598.

Sokkhey P. & Okazaki T. "*Hybrid Machine Learning Algorithms for Predicting Academic Performance*." **International Journal of Advanced Computer Science and Applications** 11, no. 1, 2020.

Song, X. F. Zhang, Y. Gong, D. W. & Gao, X. Z. "*A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high dimensional Data.*" **IEEE Transactions on Cybernetics** 99,2021: 1–14.

Song, X.F. Zhang,Y. Guo,Y.N. Sun, X.Y. & Wang,Y.L. "*Variable-size cooperative convolutionary particle swarm optimization for feature selection on high-dimensional data*." **IEEE Transactions on Evolutionary Computation** 24, no. 5 2020: 882–895.

Suguna R., Devi S., Bagate R. A., & Joshi, A. S. "*Assessment of feature selection for student academic performance through machine learning classification*." **Journal of statistics and management systems** 22 no. 4 25 Jun 2019: 729-739, ISSN: 0972- 0510.

Tanweer A., Shamimul Q., Amit D. & Mohamed B. "*Genetic Algorithm: Reviews, Implementations and Applications*." **International Journal of Engineering Pedagogy (IJE.P***)* 2020.

Torres M. C., Acal, C., El Honrani, M., & Estrada M. A. C.,"*Impact on the Virtual Learning Environment Due to COVID- 19*." **Sustainability**, vol. 13, no. 2, pp. 582, 2021.

Tran B., Xue B., & Zhang, M. "*A new representation in Particle Swarm Optimization for discretization-based feature selection*." **IEEE Transactions on Cybernetics** 48, no. 6 2018: 1733–1746.

Tsai,C,,F., & Sung,Y., T. "*Ensemble Feature Selection in High Dimension, Low Size Sample Datasets:Paralleland Serial Combination Approaches*." **Knowledge-Based System** 203, 106097 2020.

Udding, S. K. A., Hossain M. E., & Ali, M. "*Comparing different supervised machine learning algorithms for disease prediction.*" **BMC Medical Informatics and Decision Making** 19, no. 281, 2019.

Usama J., Shahid M., Saba R., Muhammad W., Sabar A., Muhammad Z. & Muhammad J. I. "*Student Performance Prediction in E-learning Environment Using Machine*

Learning." Jilin, Daxue Xuebao (GongXueban)/**Journal of Jilin University (Engineering and Technology Edition)**. 41, no.12 2022: ISSN: 1671.

Verma, S., Yadav R.K. & Kholiya, K." *Prediction of Academic Performance of Engineering Students by using Data Mining Techniques*." **International Journal of Information and Education Technology**, Vol 12,No.11, November 2022.

Wang J., Wang X., Li X., & Yi J. "*A Hybrid Particle Swarm Optimization Algorithm with Dynamic Adjustments of Inertia Weight Based on a new Feature Selection Method to Optimize SVM Parameters*." **Entropy** 25, no. 531 2023.

Widowati F. U." *Application of C4.5 algorithm with PSO Feature Selection and Bagging Technique on Breast Cancer Classification*." **International journal of Management Science and Information Technique** 4 (2), 2024, pages 312 – 320.

Xiao W., Ji P. & Hu J.. "*RnkHEU: A Hybrid Feature Selection Method for Predicting Students' Performance.*" **Hindawi Scientific Programming** 2021, no.1 2021: 1670593.

Xiong H., Huang X., Yang M., Wang L., & Yu S. " *Unbounded and efficient revocable attribute-based encryption with adaptive security for cloud-assisted internet of things*." **IEEE Internet of Things Journal**, 9, no. 4, 2021:3097-3111.

Xu Z , Heidari A A, Kuang F, Khalil A , Mafarja M , Zhang S, Chen H, & Pan Z. "*Enhanced Gaussian Bare-Bones Grasshopper Optimization: Mitigating the Performance Concerns for Feature Selection*." **Expert Systems with Applications** 212, February **2022**, 118642.

Yagci, M. "*Educational Data Mining: Prediction of students' academic performance using machine learning algorithms*." **smart learning environments** 9, no.11 2022.

Yap B. W., Ibraham N., Abdul-Rahman S., Fong S. & Abdul H. H. "*Feature selection methods: case of filter and wrapper approaches for maximising classification accuracy,*" **Pertanika Journal of science and Technology** 26, no. 1 January, 2018: 329 – 340.

Ye L., Meng X., Hang Y. & Tayeb S. "*Reseach on Visual tracking method for students' browsing data in art literacy online education*." **Journal of Network Intelligence**, 5 2020.

Yue L., Hu P. & Zhu J. " *Binary Particle Swarm Optimization Predicting Students' Academic Performance in College English*." **Journal of Network Intelligence. Taiwan Ubiquitous Information** 9, no. 2, May 2024.

Zhang H., Zhou J., Armaghani D. J., Tahir, M.M, Pham B. T., & Huynh,V. V. "*A Combination of Feature and Random Forest Techniques to Solve a Problem Related to Blast-induced Ground Vibration*." **Applied Science** 10, no.3 27 January 2020.

Website

Al-shabandor R., Hussain A., Keight R. & Khan W."Students performance prediction in Online Course Using Machine Learning Algorithms" https://www.researchgate.net/publication/34701 9578 , 2020

Anisha C. & Arulanand N. "Tuned Homogenous Ensemble Regressor Model for Early Diagnosis of Parkinson Disorder Based on Voice Features Modality" September 2022. https://doi.org/10.36548/jaicn.2022.3.005

Hodson T." Root-mean-square (RMSE) or mean absolute error(MAE). When to use them or not". *Geoscientific Model Development*: https://doi.org/10.5194lgmd-15-5481-2020.

https://archive.ics.uci.edu/dataset/320/student+perfor***mance***

https://www.academia.edu/72282730/Dataset_of_Students_Performance_Using_Student_Information_System_Moodle_and_the_Mobile_Application_eDify

https://www.kaggle.com/datasets/aljarah/xAPI-Edu-Data/data

https://www.kaggle.com/datasets/devansodariya/student-performance-data

https://zenodo.org/record/5591907

https://www.tecton.ai>Blog.

Appendibx

```
#All models on No FS
s
import pandas as pd
import numpy as np
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.pipeline import Pipeline
import xgboost as xgb
import lightgbm as lgb
from sklearn.model_selection import train_test_split, KFold
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import ExtraTreesRegressor, AdaBoostRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import BayesianRidge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge, Lasso, ElasticNet
#from sklearn.ensemble import BaggingRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

# Load the data from 'scaled_data.csv'
df = pd.read_csv('scaled_data.csv')
X = df.drop(['ESE', 'ApplicantName'], axis=1)  # Features, dropping ApplicantName
y = df['ESE']  # Target variable

# Define models to evaluate
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Random Forest': RandomForestRegressor(random_state=42),
    'SVR (RBF)': SVR(kernel='rbf'),
    'KNN': KNeighborsRegressor(),
    'MLP': MLPRegressor(max_iter=1000, learning_rate='adaptive', random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42),
    'XGBoost': xgb.XGBRegressor(random_state=42),
    'LightGBM': lgb.LGBMRegressor(random_state=42),
    'Extra Trees': ExtraTreesRegressor(),
    'AdaBoost': AdaBoostRegressor(),
    'SVR (linear)': SVR(kernel='linear'),
    'SVR (poly)': SVR(kernel='poly'),
    'Gaussian Process': GaussianProcessRegressor(),
    'KNN (tuned)': KNeighborsRegressor(n_neighbors=7), # Example: Tune n_neighbors
    'Bayesian Ridge': BayesianRidge(),
```

```
    'Ridge Regression': Ridge(),
    'Lasso Regression': Lasso(),
    'ElasticNet Regression': ElasticNet(),
    #'Bagging (Decision Tree)':
BaggingRegressor(base_estimator=DecisionTreeRegressor(random_state=42)) # Optional
random_state within the DecisionTree
}

# K-Fold Cross-Validation
kfold = KFold(n_splits=10, shuffle=True, random_state=42)

# Storage for evaluation results (empty dictionary)
all_results = {}

# Evaluate each model
for model_name, model in models.items():
  # Create a list to store fold results for this model
  fold_results = []
  for fold_number, (train_index, test_index) in enumerate(kfold.split(X)):
      X_train, X_test = X.iloc[train_index], X.iloc[test_index]
      y_train, y_test = y.iloc[train_index], y.iloc[test_index]

      model.fit(X_train, y_train)
      y_pred = model.predict(X_test)

      mse = mean_squared_error(y_test, y_pred)
      r2 = r2_score(y_test, y_pred)
      mae = mean_absolute_error(y_test, y_pred)

      fold_results.append({
          'fold_number': fold_number + 1,  # Start fold numbers from 1
          'MSE': mse,
          'R-squared': r2,
          'MAE': mae
      })
  # Add fold results for this model to the main dictionary
  all_results[model_name] = fold_results

# Calculate average metrics and include model names
summary_results = {model_name: {
    'Average MSE': np.mean([fold['MSE'] for fold in fold_results]),
    'Average R-squared': np.mean([fold['R-squared'] for fold in fold_results]),
    'Average MAE': np.mean([fold['MAE'] for fold in fold_results]),
    } for model_name, fold_results in all_results.items()}

# Create the DataFrame directly and transpose
df_summary = pd.DataFrame(summary_results).transpose()  # Transpose here
```

```
# Save the DataFrame
writer = pd.ExcelWriter('model_comparison_results_exp-1.xlsx')
df_summary.to_excel(writer, sheet_name='Summary')

# Add sheets for individual fold results
for model_name, fold_results in all_results.items():
    df = pd.DataFrame(fold_results)
    df.to_excel(writer, sheet_name=model_name, index=False)

writer.close()  # Save the workbook
import pandas as pd

from sklearn.model_selection import cross_validate
import numpy as np

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
import lightgbm as lgb

from openpyxl import Workbook

# Load the preprocessed data (this is a different version - but i eventually used the OH
version)
data = pd.read_csv('c-scaled-dataset-2.csv')
X = data.drop('Class', axis=1)
y = data['Class']

#Define the models
seed = 42

models = {
    'Logistic Regression': LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=500, random_state=seed),
    'k-NN': KNeighborsClassifier(),
    'SVM': SVC(decision_function_shape='ovr', random_state=seed),
    'Random Forest': RandomForestClassifier(random_state=seed),
    'Gradient Boosting': GradientBoostingClassifier(random_state=seed),
    'XGBoost': xgb.XGBClassifier(random_state=seed, use_label_encoder=False,
eval_metric='mlogloss'),
    'LightGBM': lgb.LGBMClassifier(random_state=seed),
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(max_iter=500, random_state=seed)}
```

```
## Perform 10-fold CV and hold results
# Initialize a dictionary to hold results
results = {model_name: [] for model_name in models.keys()}

# Define scoring metrics
scoring = ['accuracy', 'f1_macro', 'recall_macro', 'precision_macro']

# Perform 10-fold cross-validation and collect detailed metrics
for model_name, model in models.items():
    cv_results = cross_validate(model, X, y, cv=10, scoring=scoring,
return_train_score=False)
    results[model_name] = cv_results

##Save results to Excel
# Create a workbook and sheets
wb = Workbook()
summary_sheet = wb.active
summary_sheet.title = 'Average Results'
details_sheet = wb.create_sheet(title='Fold Results')

# Write headers for summary
summary_headers = ['Model', 'Accuracy Mean', 'Accuracy Std', 'F1 Mean', 'F1 Std',
'Recall Mean', 'Recall Std', 'Precision Mean', 'Precision Std']
summary_sheet.append(summary_headers)

# Write headers for detailed fold results
details_headers = ['Model', 'Fold', 'Accuracy', 'F1', 'Recall', 'Precision']
details_sheet.append(details_headers)

# Fill in the summary sheet and detailed results sheet
for model_name, cv_results in results.items():
    # Compute means and standard deviations for the metrics
    accuracy_mean = np.mean(cv_results['test_accuracy'])
    accuracy_std = np.std(cv_results['test_accuracy'])
    f1_mean = np.mean(cv_results['test_f1_macro'])
    f1_std = np.std(cv_results['test_f1_macro'])
    recall_mean = np.mean(cv_results['test_recall_macro'])
    recall_std = np.std(cv_results['test_recall_macro'])
    precision_mean = np.mean(cv_results['test_precision_macro'])
    precision_std = np.std(cv_results['test_precision_macro'])

    # Write to summary sheet
    summary_row = [model_name, accuracy_mean, accuracy_std, f1_mean, f1_std,
recall_mean, recall_std, precision_mean, precision_std]
    summary_sheet.append(summary_row)

    # Write detailed results for each fold
```

```
    for fold_idx in range(10):
        details_row = [model_name, fold_idx+1,
                       cv_results['test_accuracy'][fold_idx],
                       cv_results['test_f1_macro'][fold_idx],
                       cv_results['test_recall_macro'][fold_idx],
                       cv_results['test_precision_macro'][fold_idx]]
        details_sheet.append(details_row)

# Save the workbook
wb.save('/mnt/data/experiment-set-7-results.xlsx')
import pandas as pd

from sklearn.model_selection import cross_validate
import numpy as np

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
import lightgbm as lgb

from openpyxl import Workbook

# Load the preprocessed data (this is a different version - but i eventually used the OH
version)
data = pd.read_csv('c-scaled-dataset-2.csv')
X = data.drop('Class', axis=1)
y = data['Class']

#Define the models
seed = 42

models = {
    'Logistic Regression': LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=500, random_state=seed),
    'k-NN': KNeighborsClassifier(),
    'SVM': SVC(decision_function_shape='ovr', random_state=seed),
    'Random Forest': RandomForestClassifier(random_state=seed),
    'Gradient Boosting': GradientBoostingClassifier(random_state=seed),
    'XGBoost': xgb.XGBClassifier(random_state=seed, use_label_encoder=False,
eval_metric='mlogloss'),
    'LightGBM': lgb.LGBMClassifier(random_state=seed),
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(max_iter=500, random_state=seed)
```

```
}


## Perform 10-fold CV and hold results
# Initialize a dictionary to hold results
results = {model_name: [] for model_name in models.keys()}

# Define scoring metrics
scoring = ['accuracy', 'f1_macro', 'recall_macro', 'precision_macro']

# Perform 10-fold cross-validation and collect detailed metrics
for model_name, model in models.items():
    cv_results = cross_validate(model, X, y, cv=10, scoring=scoring,
return_train_score=False)
    results[model_name] = cv_results

##Save results to Excel
# Create a workbook and sheets
wb = Workbook()
summary_sheet = wb.active
summary_sheet.title = 'Average Results'
details_sheet = wb.create_sheet(title='Fold Results')

# Write headers for summary
summary_headers = ['Model', 'Accuracy Mean', 'Accuracy Std', 'F1 Mean', 'F1 Std',
'Recall Mean', 'Recall Std', 'Precision Mean', 'Precision Std']
summary_sheet.append(summary_headers)

# Write headers for detailed fold results
details_headers = ['Model', 'Fold', 'Accuracy', 'F1', 'Recall', 'Precision']
details_sheet.append(details_headers)

# Fill in the summary sheet and detailed results sheet
for model_name, cv_results in results.items():
    # Compute means and standard deviations for the metrics
    accuracy_mean = np.mean(cv_results['test_accuracy'])
    accuracy_std = np.std(cv_results['test_accuracy'])
    f1_mean = np.mean(cv_results['test_f1_macro'])
    f1_std = np.std(cv_results['test_f1_macro'])
    recall_mean = np.mean(cv_results['test_recall_macro'])
    recall_std = np.std(cv_results['test_recall_macro'])
    precision_mean = np.mean(cv_results['test_precision_macro'])
    precision_std = np.std(cv_results['test_precision_macro'])

    # Write to summary sheet
    summary_row = [model_name, accuracy_mean, accuracy_std, f1_mean, f1_std,
recall_mean, recall_std, precision_mean, precision_std]
    summary_sheet.append(summary_row)
```

```
    # Write detailed results for each fold
    for fold_idx in range(10):
        details_row = [model_name, fold_idx+1,
                       cv_results['test_accuracy'][fold_idx],
                       cv_results['test_f1_macro'][fold_idx],
                       cv_results['test_recall_macro'][fold_idx],
                       cv_results['test_precision_macro'][fold_idx]]
        details_sheet.append(details_row)

# Save the workbook
wb.save('/mnt/data/experiment-set-7-results.xlsx')


import pandas as pd

from sklearn.model_selection import cross_validate
import numpy as np

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
import lightgbm as lgb

from openpyxl import Workbook

# Load the preprocessed data (this is a different version - but i eventually used the OH
version)
data = pd.read_csv('c-scaled-dataset-2.csv')
X = data.drop('Class', axis=1)
y = data['Class']

#Define the models
seed = 42

models = {
    'Logistic Regression': LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=500, random_state=seed),
    'k-NN': KNeighborsClassifier(),
    'SVM': SVC(decision_function_shape='ovr', random_state=seed),
    'Random Forest': RandomForestClassifier(random_state=seed),
    'Gradient Boosting': GradientBoostingClassifier(random_state=seed),
    'XGBoost': xgb.XGBClassifier(random_state=seed, use_label_encoder=False,
eval_metric='mlogloss'),
    'LightGBM': lgb.LGBMClassifier(random_state=seed),
```

```
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(max_iter=500, random_state=seed)}


## Perform 10-fold CV and hold results
# Initialize a dictionary to hold results
results = {model_name: [] for model_name in models.keys()}

# Define scoring metrics
scoring = ['accuracy', 'f1_macro', 'recall_macro', 'precision_macro']

# Perform 10-fold cross-validation and collect detailed metrics
for model_name, model in models.items():
    cv_results = cross_validate(model, X, y, cv=10, scoring=scoring,
return_train_score=False)
    results[model_name] = cv_results

##Save results to Excel
# Create a workbook and sheets
wb = Workbook()
summary_sheet = wb.active
summary_sheet.title = 'Average Results'
details_sheet = wb.create_sheet(title='Fold Results')

# Write headers for summary
summary_headers = ['Model', 'Accuracy Mean', 'Accuracy Std', 'F1 Mean', 'F1 Std',
'Recall Mean', 'Recall Std', 'Precision Mean', 'Precision Std']
summary_sheet.append(summary_headers)

# Write headers for detailed fold results
details_headers = ['Model', 'Fold', 'Accuracy', 'F1', 'Recall', 'Precision']
details_sheet.append(details_headers)

# Fill in the summary sheet and detailed results sheet
for model_name, cv_results in results.items():
    # Compute means and standard deviations for the metrics
    accuracy_mean = np.mean(cv_results['test_accuracy'])
    accuracy_std = np.std(cv_results['test_accuracy'])
    f1_mean = np.mean(cv_results['test_f1_macro'])
    f1_std = np.std(cv_results['test_f1_macro'])
    recall_mean = np.mean(cv_results['test_recall_macro'])
    recall_std = np.std(cv_results['test_recall_macro'])
    precision_mean = np.mean(cv_results['test_precision_macro'])
    precision_std = np.std(cv_results['test_precision_macro'])

    # Write to summary sheet
    summary_row = [model_name, accuracy_mean, accuracy_std, f1_mean, f1_std,
recall_mean, recall_std, precision_mean, precision_std]
    summary_sheet.append(summary_row)
```

```
    # Write detailed results for each fold
    for fold_idx in range(10):
        details_row = [model_name, fold_idx+1,
                       cv_results['test_accuracy'][fold_idx],
                       cv_results['test_f1_macro'][fold_idx],
                       cv_results['test_recall_macro'][fold_idx],
                       cv_results['test_precision_macro'][fold_idx]]
        details_sheet.append(details_row)

# Save the workbook
wb.save('/mnt/data/experiment-set-7-results.xlsx')
#All models on PCC

import pandas as pd
import numpy as np
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.pipeline import Pipeline
import xgboost as xgb
import lightgbm as lgb
from sklearn.model_selection import train_test_split, KFold
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import ExtraTreesRegressor, AdaBoostRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import BayesianRidge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge, Lasso, ElasticNet
#from sklearn.ensemble import BaggingRegressor - I left out bagging
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression

# Load the data from 'scaled_data.csv'
df = pd.read_csv('scaled_data.csv')
X = df.drop(['ESE', 'ApplicantName'], axis=1)  # Features, dropping ApplicantName
y = df['ESE']  # Target variable

# Feature Selection using Pearson Correlation
correlation_matrix = X.corr().abs()
upper_triangle = correlation_matrix.where(np.triu(np.ones(correlation_matrix.shape),
k=1).astype(bool))
```

```
top_correlated_features = [column for column in upper_triangle.columns if
any(upper_triangle[column] > 0.6)]

# Select the top features
selector = SelectKBest(f_regression, k=len(top_correlated_features))
selector.fit(X[top_correlated_features], y)

# Subset data for transformation
X_to_transform = X[top_correlated_features]
X_selected = selector.transform(X_to_transform)

# Print selected features
print(top_correlated_features)

# Define models to evaluate
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Random Forest': RandomForestRegressor(random_state=42),
    'SVR (RBF)': SVR(kernel='rbf'),
    'KNN': KNeighborsRegressor(),
    'MLP': MLPRegressor(max_iter=1000, learning_rate='adaptive', random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42),
    'XGBoost': xgb.XGBRegressor(random_state=42),
    'LightGBM': lgb.LGBMRegressor(random_state=42),
    'Extra Trees': ExtraTreesRegressor(),
    'AdaBoost': AdaBoostRegressor(),
    'SVR (linear)': SVR(kernel='linear'),
    'SVR (poly)': SVR(kernel='poly'),
    'Gaussian Process': GaussianProcessRegressor(),
    'KNN (tuned)': KNeighborsRegressor(n_neighbors=7), # Example: Tune n_neighbors
    'Bayesian Ridge': BayesianRidge(),
    'Ridge Regression': Ridge(),
    'Lasso Regression': Lasso(),
    'ElasticNet Regression': ElasticNet(),
}

# K-Fold Cross-Validation
kfold = KFold(n_splits=10, shuffle=True, random_state=42)

# Storage for evaluation results
all_results = {}

# Evaluate each model
for model_name, model in models.items():
    fold_results = []
    for fold_number, (train_index, test_index) in enumerate(kfold.split(X_selected)):
        X_train, X_test = X_selected[train_index], X_selected[test_index]
```

```
        y_train, y_test = y[train_index], y[test_index]

        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)
        mae = mean_absolute_error(y_test, y_pred)

        fold_results.append({
            'fold_number': fold_number + 1,
            'MSE': mse,
            'R-squared': r2,
            'MAE': mae
        })

    all_results[model_name] = fold_results

# Calculate average metrics
summary_results = {model_name: {
    'Average MSE': np.mean([fold['MSE'] for fold in fold_results]),
    'Average R-squared': np.mean([fold['R-squared'] for fold in fold_results]),
    'Average MAE': np.mean([fold['MAE'] for fold in fold_results]),
    } for model_name, fold_results in all_results.items()}

# Create the DataFrame directly and transpose
df_summary = pd.DataFrame(summary_results).transpose()  # Transpose here

# Save Results
writer = pd.ExcelWriter('model_comparison_results_exp-2.xlsx')

#Save Summary
df_summary.to_excel(writer, sheet_name='Summary')

# Add sheets for individual fold results
for model_name, fold_results in all_results.items():
    df = pd.DataFrame(fold_results)
    df.to_excel(writer, sheet_name=model_name, index=False)

writer.close()

print(top_correlated_features)
#All models on PCA

import pandas as pd
import numpy as np
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.pipeline import Pipeline
```

```
from sklearn.model_selection import train_test_split, KFold
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import ExtraTreesRegressor, AdaBoostRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import BayesianRidge, Ridge, Lasso, ElasticNet
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.decomposition import PCA
import xgboost as xgb
import lightgbm as lgb

# Load the data from 'scaled_data.csv'
df = pd.read_csv('scaled_data.csv')
X = df.drop(['ESE', 'ApplicantName'], axis=1)
y = df['ESE']

# Feature Selection using PCA
pca = PCA(n_components=0.95)  # Adjust 'n_components' as needed
X_pca = pca.fit_transform(X)

# Define models to evaluate
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Random Forest': RandomForestRegressor(random_state=42),
    'SVR (RBF)': SVR(kernel='rbf'),
    'KNN': KNeighborsRegressor(),
    'MLP': MLPRegressor(max_iter=1000, learning_rate='adaptive', random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42),
    'XGBoost': xgb.XGBRegressor(random_state=42),
    'LightGBM': lgb.LGBMRegressor(random_state=42),
    'Extra Trees': ExtraTreesRegressor(),
    'AdaBoost': AdaBoostRegressor(),
    'SVR (linear)': SVR(kernel='linear'),
    'SVR (poly)': SVR(kernel='poly'),
    'Gaussian Process': GaussianProcessRegressor(),
    'KNN (tuned)': KNeighborsRegressor(n_neighbors=7),
    'Bayesian Ridge': BayesianRidge(),
    'Ridge Regression': Ridge(),
    'Lasso Regression': Lasso(),
    'ElasticNet Regression': ElasticNet(),
}
```

```
# K-Fold Cross-Validation
kfold = KFold(n_splits=10, shuffle=True, random_state=42)

# Storage for evaluation results
all_results = {}

# Evaluate each model
for model_name, model in models.items():
    fold_results = []
    for fold_number, (train_index, test_index) in enumerate(kfold.split(X_pca)):
        X_train, X_test = X_pca[train_index], X_pca[test_index]
        y_train, y_test = y[train_index], y[test_index]

        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)
        mae = mean_absolute_error(y_test, y_pred)

        fold_results.append({
            'fold_number': fold_number + 1,
            'MSE': mse,
            'R-squared': r2,
            'MAE': mae
        })

    all_results[model_name] = fold_results

# Calculate average metrics
summary_results = {model_name: {
    'Average MSE': np.mean([fold['MSE'] for fold in fold_results]),
    'Average R-squared': np.mean([fold['R-squared'] for fold in fold_results]),
    'Average MAE': np.mean([fold['MAE'] for fold in fold_results]),
    } for model_name, fold_results in all_results.items()}

# Create the DataFrame directly and transpose
df_summary = pd.DataFrame(summary_results).transpose()  # Transpose here

# Save Results
writer = pd.ExcelWriter('model_comparison_results_exp-3.xlsx')

#Save Summary
df_summary.to_excel(writer, sheet_name='Summary')

# Add sheets for individual fold results
for model_name, fold_results in all_results.items():
    df = pd.DataFrame(fold_results)
```

```
    df.to_excel(writer, sheet_name=model_name, index=False)

writer.close()

print(X_pca)
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier,
AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
import lightgbm as lgb
from pyswarms.single.global_best import GlobalBestPSO
import matplotlib.pyplot as plt

# --- 1. Load and Inspect Data ---
df = pd.read_csv('oh-scaled-dataset-2.csv')
X = df.drop('Class', axis=1).values  # Convert to numpy array for compatibility
y = df['Class'].values  # Convert to numpy array for compatibility

# Inspect the Data
print(df.head())
print(df.shape)
print(df.dtypes)

# --- 2. Perform PCA Feature Selection ---
# Determine the number of components that explain at least 95% of the variance
pca_test = PCA().fit(X)
explained_variance_ratios = pca_test.explained_variance_ratio_
cumulative_variance = np.cumsum(explained_variance_ratios)

# Find the elbow point where variance plateaus
elbow_index = np.argmax(np.diff(cumulative_variance) < 0.03) + 1

# Plot the cumulative explained variance to find the elbow point
plt.figure(figsize=(8, 6))
plt.plot(np.cumsum(pca_test.explained_variance_ratio_))
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
```

```
plt.title('PCA Explained Variance (Elbow Method)')
plt.scatter(elbow_index, cumulative_variance[elbow_index], marker='x', s=100,
color='red')
plt.text(elbow_index + 0.5, cumulative_variance[elbow_index] - 0.05, f'Elbow Point
({elbow_index} Components)', color='red')
plt.savefig('pca_explained_variance_plot_exp11.png')
plt.show()

# Fit PCA to the data with the chosen number of components
#pca = PCA(n_components=elbow_index)  # Specify the number of components based
on graph results
pca = PCA(n_components=10)  # Force a number of components based on previous
experiment
X_pca = pca.fit_transform(X)  # Fit PCA to the data and transform the features

# Save the new features as a new dataframe
X_pca_select = pd.DataFrame(X_pca, columns=[f'PC{i+1}' for i in
range(X_pca.shape[1])])

# Display the shape of the new dataframe
print("Shape of X_pca_select:", X_pca_select.shape)

# Combine the PCA selected features with the target variable
X_pca_select['Class'] = y

# Save the new dataframe
X_pca_select.to_csv('X_pca_select_exp11.csv', index=False)

# --- 3. Use Forward Feature Selection (FFS) ---
# Define the model for FFS
estimator = RandomForestClassifier(random_state=42)
ffs = SequentialFeatureSelector(estimator, direction='forward', n_features_to_select=8,
cv=10)
X_ffs = ffs.fit_transform(X_pca, y)

# Save the new features into a dataframe
X_ffs_select = pd.DataFrame(X_ffs, columns=[f'Feature_{i+1}' for i in
range(X_ffs.shape[1])])

# Display the shape of the new dataframe
print("Shape of X_ffs_select:", X_ffs_select.shape)

# Combine with the target variable
X_ffs_select['Class'] = y

# Save the new dataframe
X_ffs_select.to_csv('X_ffs_select_exp11.csv', index=False)
```

```
# --- 4. Pass the new features into PSO ---

# --- PSO Feature Selection ---
def pso_objective_function(feature_mask, X, y):
    # Ensure feature_mask is one-dimensional
    feature_mask = feature_mask.astype(bool)
    if np.sum(feature_mask) == 0:
        return float('inf')  # Avoid selecting zero features
    X_subset = X

    best_cost = -np.inf  # Initialize best_cost

    for _ in range(1):
        model = RandomForestClassifier(random_state=42)
        scores = cross_val_score(model, X_subset, y, cv=5, scoring='accuracy')
        current_cost = scores.mean()
        if current_cost > best_cost:
            best_cost = current_cost
            best_pos = feature_mask
        print("Iteration:", _, "Best Cost:", best_cost)
    return best_cost

# Load the FFS selected data
X_ffs = X_ffs_select.drop('Class', axis=1).values
y_ffs = X_ffs_select['Class'].values

# Set up the PSO optimizer
dimensions = X_ffs.shape[1]  # Number of features after FFS
print(dimensions)
options = {'c1': 0.5, 'c2': 0.3, 'w': 0.8}
optimizer = GlobalBestPSO(n_particles=8, dimensions=dimensions, options=options)

# Perform PSO to find the best feature subset
best_cost, best_pos = optimizer.optimize(pso_objective_function, iters=10, X=X_ffs,
y=y_ffs)

# Get the selected features from PSO
best_feature_mask = best_pos > 0.5
selected_features = np.where(best_feature_mask)[0]  # Indices of selected features

# Apply mask to the FFS data
X_final = X_ffs[:, selected_features]  # Corrected to select the features based on PSO
results

# Combine with the target variable
feature_names = [f'Feature_{i+1}' for i in selected_features]  # Corrected to match the
number of selected features
X_final_df = pd.DataFrame(X_final, columns=feature_names)
```

```
X_final_df['Class'] = y_ffs

# Save the final dataframe
X_final_df.to_csv('X_final_select_exp11.csv', index=False)

# Display the shape of the final dataframe
print("Shape of X_final_df:", X_final_df.shape)


# --- 5. Evaluate Models with 10-fold Cross-Validation ---
X = X_final_df.drop('Class', axis=1)
y = X_final_df['Class']

# Define models to evaluate
models = {
    'Logistic Regression': LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=5000, random_state=42),
    'k-NN': KNeighborsClassifier(),
    'SVM': SVC(decision_function_shape='ovr', random_state=42),
    'Random Forest': RandomForestClassifier(random_state=42),
    'Gradient Boosting': GradientBoostingClassifier(random_state=42),
    'XGBoost': xgb.XGBClassifier(random_state=42, use_label_encoder=False,
eval_metric='mlogloss'),
    'LightGBM': lgb.LGBMClassifier(random_state=42),
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(max_iter=5000, random_state=42)
}

# K-Fold Cross-Validation
kf = KFold(n_splits=10, shuffle=True, random_state=42)

# Storage for evaluation results
all_results = {}

# Evaluate each model
for model_name, model in models.items():
    fold_results = []
    for fold_number, (train_index, test_index) in enumerate(kf.split(X)):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred, average='macro')
        recall = recall_score(y_test, y_pred, average='macro')
        f1 = f1_score(y_test, y_pred, average='macro')
```

```
        fold_results.append({
            'fold_number': fold_number + 1,
            'Accuracy': accuracy,
            'Precision': precision,
            'Recall': recall,
            'F1 Score': f1
        })

    all_results[model_name] = fold_results

# Calculate average metrics
summary_results = {model_name: {
    'Average Accuracy': np.mean([fold['Accuracy'] for fold in fold_results]),
    'Average Precision': np.mean([fold['Precision'] for fold in fold_results]),
    'Average Recall': np.mean([fold['Recall'] for fold in fold_results]),
    'Average F1 Score': np.mean([fold['F1 Score'] for fold in fold_results]),
    } for model_name, fold_results in all_results.items()}

# Create the DataFrame directly and transpose
df_summary = pd.DataFrame(summary_results).transpose()

# Save Results
writer = pd.ExcelWriter('exp-12-pcaforpos.xlsx')

# Save Summary
df_summary.to_excel(writer, sheet_name='Summary')

# Add sheets for individual fold results
for model_name, fold_results in all_results.items():
    df = pd.DataFrame(fold_results)
    df.to_excel(writer, sheet_name=model_name, index=False)

writer.close()
print("Selected Features:", selected_features)

import pandas as pd
import numpy as np
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression, BayesianRidge, Ridge, Lasso,
ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor,
AdaBoostRegressor, ExtraTreesRegressor
from sklearn.svm import SVR
```

```
import xgboost as xgb
import lightgbm as lgb
import time
import cProfile  # Import the cProfile module
import pstats  # Import pstats module for formatting cProfile results

# Function to run the main script
#Introduce profiling because this particular code ran for too long. It ran for days
#Eventually decided to run the code on the Cloud and then take the selected features and
use here
def main():
    # Load the data
    df = pd.read_csv('processed-dataset-3.csv')
    X = df.drop(['G3'], axis=1)  # Dropping target
    y = df['G3']  # Target variable

    # Selected features from PSO
    selected_features = ['Pstatus', 'Medu', 'traveltime', 'studytime', 'failures', 'schoolsup',
      'nursery', 'higher', 'romantic', 'famrel', 'freetime', 'Dalc',
      'absences', 'G2', 'Mjob_health', 'Mjob_other', 'Mjob_services',
      'Fjob_at_home', 'Fjob_health', 'reason_home', 'reason_other',
      'reason_reputation', 'guardian_other']

    # Use only the selected features
    X_selected = X[selected_features]

    # Initialize the KNN regressor
    knn = KNeighborsRegressor()

    # Define the parameter grid
    param_grid = {'n_neighbors': np.arange(1, 51)}

    # Initialize GridSearchCV
    grid_search = GridSearchCV(knn, param_grid, cv=5,
scoring='neg_mean_squared_error')

    # Fit the grid search to the data
    grid_search.fit(X_selected, y)

    # Get the best parameter
    best_n_neighbors = grid_search.best_params_['n_neighbors']
    print(f"The optimal number of neighbors is: {best_n_neighbors}")

    # Define models to evaluate
    models = {
        'Linear Regression': LinearRegression(),
        'Decision Tree': DecisionTreeRegressor(random_state=42),
        'Random Forest': RandomForestRegressor(random_state=42),
```

```
    'SVR (RBF)': SVR(kernel='rbf'),
    'KNN': KNeighborsRegressor(),
    'MLP': MLPRegressor(max_iter=10000, learning_rate='adaptive', random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42),
    'XGBoost': xgb.XGBRegressor(random_state=42),
    'LightGBM': lgb.LGBMRegressor(random_state=42),
    'Extra Trees': ExtraTreesRegressor(),
    'AdaBoost': AdaBoostRegressor(),
    'SVR (linear)': SVR(kernel='linear'),
    'SVR (poly)': SVR(kernel='poly'),
    'Gaussian Process': GaussianProcessRegressor(),
    'KNN (tuned)': KNeighborsRegressor(n_neighbors=best_n_neighbors),
    'Bayesian Ridge': BayesianRidge(),
    'Ridge Regression': Ridge(),
    'Lasso Regression': Lasso(),
    'ElasticNet Regression': ElasticNet(),
}

# K-Fold Cross-Validation
kfold = KFold(n_splits=10, shuffle=True, random_state=42)

# Storage for evaluation results (empty dictionary)
all_results = {}

# Evaluate each model
for model_name, model in models.items():
    fold_results = []
    for fold_number, (train_index, test_index) in enumerate(kfold.split(X_selected)):
        X_train, X_test = X_selected.iloc[train_index], X_selected.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)
        mae = mean_absolute_error(y_test, y_pred)

        fold_results.append({
            'fold_number': fold_number + 1,
            'MSE': mse,
            'R-squared': r2,
            'MAE': mae
        })

    # Add fold results for this model to the main dictionary
    all_results[model_name] = fold_results
```

```
    # Calculate average metrics and include model names
    summary_results = {model_name: {
        'Average MSE': np.mean([fold['MSE'] for fold in fold_results]),
        'Average R-squared': np.mean([fold['R-squared'] for fold in fold_results]),
        'Average MAE': np.mean([fold['MAE'] for fold in fold_results]),
    } for model_name, fold_results in all_results.items()}

    # Create the DataFrame directly and transpose
    df_summary = pd.DataFrame(summary_results).transpose()  # Transpose here

    # Save the DataFrame
    writer = pd.ExcelWriter('exp-17-PSO-final-results.xlsx')
    df_summary.to_excel(writer, sheet_name='Summary')

    # Add sheets for individual fold results
    for model_name, fold_results in all_results.items():
        df = pd.DataFrame(fold_results)
        df.to_excel(writer, sheet_name=model_name, index=False)

    writer.close()  # Save the workbook

    print(f"Selected features: {selected_features}")

# Profile the main function
cProfile.run('main()', 'profile_results')

# Print profiling results
p = pstats.Stats('profile_results')
p.sort_stats('cumulative').print_stats(10)

import pandas as pd
import numpy as np
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression, BayesianRidge, Ridge, Lasso,
ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor,
AdaBoostRegressor, ExtraTreesRegressor
from sklearn.svm import SVR
import xgboost as xgb
import lightgbm as lgb
import time
import cProfile  # Import the cProfile module
import pstats  # Import pstats module for formatting cProfile results
```

```
# Function to run the main script
#Introduce profiling because this particular code ran for too long. It ran for days
#Eventually decided to run the code on the Cloud and then take the selected features and
use here
def main():
    # Load the data
    df = pd.read_csv('processed-dataset-3.csv')
    X = df.drop(['G3'], axis=1)  # Dropping target
    y = df['G3']  # Target variable

    # Selected features from FFS
    selected_features = ['school', 'address', 'famsize', 'Pstatus', 'traveltime', 'studytime',
                         'failures', 'schoolsup', 'activities', 'nursery', 'higher', 'internet',
                         'romantic', 'Dalc', 'Walc', 'absences', 'G1', 'G2', 'Mjob_health',
                         'Mjob_other', 'Mjob_services', 'Fjob_at_home', 'Fjob_health',
                         'Fjob_other', 'Fjob_teacher', 'reason_course', 'reason_home',
                         'reason_other', 'reason_reputation', 'guardian_other']

    # Use only the selected features
    X_selected = X[selected_features]

    # Initialize the KNN regressor
    knn = KNeighborsRegressor()

    # Define the parameter grid
    param_grid = {'n_neighbors': np.arange(1, 51)}

    # Initialize GridSearchCV
    grid_search = GridSearchCV(knn, param_grid, cv=5,
scoring='neg_mean_squared_error')

    # Fit the grid search to the data
    grid_search.fit(X_selected, y)

    # Get the best parameter
    best_n_neighbors = grid_search.best_params_['n_neighbors']
    print(f"The optimal number of neighbors is: {best_n_neighbors}")

    # Define models to evaluate
    models = {
        'Linear Regression': LinearRegression(),
        'Decision Tree': DecisionTreeRegressor(random_state=42),
        'Random Forest': RandomForestRegressor(random_state=42),
        'SVR (RBF)': SVR(kernel='rbf'),
        'KNN': KNeighborsRegressor(),
        'MLP': MLPRegressor(max_iter=10000, learning_rate='adaptive', random_state=42),
        'Gradient Boosting': GradientBoostingRegressor(random_state=42),
```

```
    'XGBoost': xgb.XGBRegressor(random_state=42),
    'LightGBM': lgb.LGBMRegressor(random_state=42),
    'Extra Trees': ExtraTreesRegressor(),
    'AdaBoost': AdaBoostRegressor(),
    'SVR (linear)': SVR(kernel='linear'),
    'SVR (poly)': SVR(kernel='poly'),
    'Gaussian Process': GaussianProcessRegressor(),
    'KNN (tuned)': KNeighborsRegressor(n_neighbors=best_n_neighbors),
    'Bayesian Ridge': BayesianRidge(),
    'Ridge Regression': Ridge(),
    'Lasso Regression': Lasso(),
    'ElasticNet Regression': ElasticNet(),
}

# K-Fold Cross-Validation
kfold = KFold(n_splits=10, shuffle=True, random_state=42)

# Storage for evaluation results (empty dictionary)
all_results = {}

# Evaluate each model
for model_name, model in models.items():
    fold_results = []
    for fold_number, (train_index, test_index) in enumerate(kfold.split(X_selected)):
        X_train, X_test = X_selected.iloc[train_index], X_selected.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)
        mae = mean_absolute_error(y_test, y_pred)

        fold_results.append({
            'fold_number': fold_number + 1,
            'MSE': mse,
            'R-squared': r2,
            'MAE': mae
        })

    # Add fold results for this model to the main dictionary
    all_results[model_name] = fold_results

# Calculate average metrics and include model names
summary_results = {model_name: {
    'Average MSE': np.mean([fold['MSE'] for fold in fold_results]),
    'Average R-squared': np.mean([fold['R-squared'] for fold in fold_results]),
```

```
        'Average MAE': np.mean([fold['MAE'] for fold in fold_results]),
    } for model_name, fold_results in all_results.items()}

    # Create the DataFrame directly and transpose
    df_summary = pd.DataFrame(summary_results).transpose()  # Transpose here

    # Save the DataFrame
    writer = pd.ExcelWriter('exp-16-FFS-final-results.xlsx')
    df_summary.to_excel(writer, sheet_name='Summary')

    # Add sheets for individual fold results
    for model_name, fold_results in all_results.items():
        df = pd.DataFrame(fold_results)
        df.to_excel(writer, sheet_name=model_name, index=False)

    writer.close()  # Save the workbook

    print(f"Selected features: {selected_features}")

# Profile the main function
cProfile.run('main()', 'profile_results')

# Print profiling results
p = pstats.Stats('profile_results')
p.sort_stats('cumulative').print_stats(10)

import pandas as pd
import numpy as np
from sklearn.model_selection import cross_validate
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
import lightgbm as lgb
from skfeature.function.information_theoretical_based import CIFE
from sklearn.feature_selection import SelectKBest
from sklearn.preprocessing import StandardScaler
from openpyxl import Workbook
from pyswarm import pso

# Load the preprocessed data
data = pd.read_csv('oh-scaled-dataset-2.csv')
X = data.drop('Class', axis=1)
y = data['Class']
```

```
# Define the models
seed = 42
models = {
    'Logistic Regression': LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=5000, random_state=seed),
    'k-NN': KNeighborsClassifier(),
    'SVM': SVC(decision_function_shape='ovr', random_state=seed),
    'Random Forest': RandomForestClassifier(random_state=seed),
    'Gradient Boosting': GradientBoostingClassifier(random_state=seed),
    'XGBoost': xgb.XGBClassifier(random_state=seed, use_label_encoder=False,
eval_metric='mlogloss'),
    'LightGBM': lgb.LGBMClassifier(random_state=seed),
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(max_iter=5000, random_state=seed)
}

# Function to optimize using PSO
def optimize_feature_selection(X, y, model):
    def objective(features):
        # Convert features to indices
        selected_features = np.where(features)[0]
        if len(selected_features) == 0:
            return np.inf  # Return a large value if no feature is selected

        # Subset X with selected features
        X_selected = X.iloc[:, selected_features]

        # Perform 10-fold cross-validation and return negative accuracy (to maximize
accuracy)
        cv_results = cross_validate(model, X_selected, y, cv=10, scoring='accuracy',
return_train_score=False)
        accuracy_mean = np.mean(cv_results['test_score'])
        return -accuracy_mean

    # Number of features
    num_features = X.shape[1]

    # PSO optimization
    lb = np.zeros(num_features)  # Lower bounds for features (0 for unselected, 1 for
selected)
    ub = np.ones(num_features)   # Upper bounds for features (0 for unselected, 1 for
selected)
    xopt, fopt = pso(objective, lb, ub, swarmsize=100, maxiter=5, debug=True)

    # Convert the best solution to indices of selected features
    selected_features = np.where(xopt)[0]
    return selected_features
```

```
# Define the scoring metrics
scoring_metrics = ['accuracy', 'precision_macro', 'recall_macro', 'f1_macro']

# Perform Particle Swarm Optimization for feature selection
results = {model_name: [] for model_name in models.keys()}

# Perform 10-fold cross-validation with PSO feature selection and collect detailed metrics
for model_name, model in models.items():
    # PSO feature selection
    selected_features = optimize_feature_selection(X, y, model)

    # Get the selected features from X
    X_selected = X.iloc[:, selected_features]

    # Perform 10-fold cross-validation
    cv_results = cross_validate(model, X_selected, y, cv=10, scoring=scoring_metrics,
return_train_score=True)
    results[model_name] = cv_results

# Save results to Excel
wb = Workbook()
summary_sheet = wb.active
summary_sheet.title = 'Average Results'

# Write headers for summary
summary_headers = ['Model', 'Accuracy Mean', 'Accuracy Std', 'Precision Mean',
'Precision Std', 'Recall Mean', 'Recall Std', 'F1 Mean', 'F1 Std', 'Selected Features']
summary_sheet.append(summary_headers)

# Fill in the summary sheet and create detailed sheets for each model
for model_name, cv_results in results.items():
    # Compute means and standard deviations for each metric
    accuracy_mean = np.mean(cv_results['test_accuracy'])
    accuracy_std = np.std(cv_results['test_accuracy'])
    precision_mean = np.mean(cv_results['test_precision_macro'])
    precision_std = np.std(cv_results['test_precision_macro'])
    recall_mean = np.mean(cv_results['test_recall_macro'])
    recall_std = np.std(cv_results['test_recall_macro'])
    f1_mean = np.mean(cv_results['test_f1_macro'])
    f1_std = np.std(cv_results['test_f1_macro'])

    # Write to summary sheet
    summary_row = [model_name, accuracy_mean, accuracy_std, precision_mean,
precision_std, recall_mean, recall_std, f1_mean, f1_std, ', '.join(selected_features)]
    summary_sheet.append(summary_row)

    # Create a new sheet for each model's fold results
```

```
    model_sheet = wb.create_sheet(title=f'{model_name} Folds')
    model_headers = ['Fold', 'Test Accuracy', 'Train Accuracy', 'Test Precision', 'Train
Precision', 'Test Recall', 'Train Recall', 'Test F1', 'Train F1']
    model_sheet.append(model_headers)

    # Write detailed results for each fold
    for fold_idx in range(10):
        fold_row = [
            fold_idx + 1,
            cv_results['test_accuracy'][fold_idx], cv_results['train_accuracy'][fold_idx],
            cv_results['test_precision_macro'][fold_idx],
cv_results['train_precision_macro'][fold_idx],
            cv_results['test_recall_macro'][fold_idx],
cv_results['train_recall_macro'][fold_idx],
            cv_results['test_f1_macro'][fold_idx], cv_results['train_f1_macro'][fold_idx]
        ]
        model_sheet.append(fold_row)

# Save the workbook
wb.save('exp-11-pso.xlsx')
print("Selected Features:", selected_features)
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.pipeline import Pipeline
import xgboost as xgb
import lightgbm as lgb
from sklearn.model_selection import train_test_split, KFold
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import ExtraTreesRegressor, AdaBoostRegressor
from sklearn.linear_model import BayesianRidge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge, Lasso, ElasticNet
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.feature_selection import SelectKBest, f_regression

# Load the data
df = pd.read_csv('processed-dataset-3.csv')
X = df.drop(['G3'], axis=1)  # Dropping target
y = df['G3']  # Target variable
```

```
# Feature Selection using Pearson Correlation
correlation_matrix = X.corr().abs()

# Plot and save the correlation matrix
plt.figure(figsize=(16, 12))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Pearson Correlation Coefficient Matrix')
plt.savefig('PCC_matrix.png')
plt.show()

upper_triangle = correlation_matrix.where(np.triu(np.ones(correlation_matrix.shape),
k=1).astype(bool))
top_correlated_features = [column for column in upper_triangle.columns if
any(upper_triangle[column] > 0.6)]

# Select the top features
selector = SelectKBest(f_regression, k=len(top_correlated_features))
selector.fit(X[top_correlated_features], y)

# Subset data for transformation
X_to_transform = X[top_correlated_features]
X_selected = selector.transform(X_to_transform)

# Print selected features
print(top_correlated_features)

# Initialize the KNN regressor
knn = KNeighborsRegressor()

# Define the parameter grid
param_grid = {'n_neighbors': np.arange(1, 51)}

# Initialize GridSearchCV
grid_search = GridSearchCV(knn, param_grid, cv=5, scoring='neg_mean_squared_error')

# Fit the grid search to the data
grid_search.fit(X_selected, y)

# Get the best parameter
best_n_neighbors = grid_search.best_params_['n_neighbors']
print(f"The optimal number of neighbors is: {best_n_neighbors}")

# Define models to evaluate
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Random Forest': RandomForestRegressor(random_state=42),
```

```
    'SVR (RBF)': SVR(kernel='rbf'),
    'KNN': KNeighborsRegressor(),
    'MLP': MLPRegressor(max_iter=10000, learning_rate='adaptive', random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42),
    'XGBoost': xgb.XGBRegressor(random_state=42),
    'LightGBM': lgb.LGBMRegressor(random_state=42),
    'Extra Trees': ExtraTreesRegressor(),
    'AdaBoost': AdaBoostRegressor(),
    'SVR (linear)': SVR(kernel='linear'),
    'SVR (poly)': SVR(kernel='poly'),
    'Gaussian Process': GaussianProcessRegressor(),
    'KNN (tuned)': KNeighborsRegressor(n_neighbors=best_n_neighbors),
    'Bayesian Ridge': BayesianRidge(),
    'Ridge Regression': Ridge(),
    'Lasso Regression': Lasso(),
    'ElasticNet Regression': ElasticNet(),
}

# K-Fold Cross-Validation
kfold = KFold(n_splits=10, shuffle=True, random_state=42)

# Storage for evaluation results (empty dictionary)
all_results = {}

# Evaluate each model
for model_name, model in models.items():
    fold_results = []
    for fold_number, (train_index, test_index) in enumerate(kfold.split(X_selected)):
        X_train, X_test = X_selected[train_index], X_selected[test_index]
        y_train, y_test = y[train_index], y[test_index]

        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)
        mae = mean_absolute_error(y_test, y_pred)

        fold_results.append({
            'fold_number': fold_number + 1,
            'MSE': mse,
            'R-squared': r2,
            'MAE': mae
        })

    # Add fold results for this model to the main dictionary
    all_results[model_name] = fold_results
```

```
# Calculate average metrics and include model names
summary_results = {model_name: {
    'Average MSE': np.mean([fold['MSE'] for fold in fold_results]),
    'Average R-squared': np.mean([fold['R-squared'] for fold in fold_results]),
    'Average MAE': np.mean([fold['MAE'] for fold in fold_results]),
    } for model_name, fold_results in all_results.items()}

# Create the DataFrame directly and transpose
df_summary = pd.DataFrame(summary_results).transpose()  # Transpose here

# Save the DataFrame
writer = pd.ExcelWriter('exp-14-PCC-results.xlsx')
df_summary.to_excel(writer, sheet_name='Summary')

# Add sheets for individual fold results
for model_name, fold_results in all_results.items():
    df = pd.DataFrame(fold_results)
    df.to_excel(writer, sheet_name=model_name, index=False)

writer.close()  # Save the workbook

print(top_correlated_features)
#Exp 13: All models on No FS

import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.pipeline import Pipeline
import xgboost as xgb
import lightgbm as lgb
from sklearn.model_selection import train_test_split, KFold
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import ExtraTreesRegressor, AdaBoostRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import BayesianRidge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge, Lasso, ElasticNet
#from sklearn.ensemble import BaggingRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
```

```
# Load the data
df = pd.read_csv('processed-dataset-3.csv')
X = df.drop(['G3'], axis=1)  # Dropping target
y = df['G3']  # Target variable

# Initialize the KNN regressor
knn = KNeighborsRegressor()

# Define the parameter grid
param_grid = {'n_neighbors': np.arange(1, 51)}

# Initialize GridSearchCV
grid_search = GridSearchCV(knn, param_grid, cv=5, scoring='neg_mean_squared_error')

# Fit the grid search to the data
grid_search.fit(X, y)

# Get the best parameter
best_n_neighbors = grid_search.best_params_['n_neighbors']
print(f"The optimal number of neighbors is: {best_n_neighbors}")


# Define models to evaluate
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Random Forest': RandomForestRegressor(random_state=42),
    'SVR (RBF)': SVR(kernel='rbf'),
    'KNN': KNeighborsRegressor(),
    'MLP': MLPRegressor(max_iter=5000, learning_rate='adaptive', random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42),
    'XGBoost': xgb.XGBRegressor(random_state=42),
    'LightGBM': lgb.LGBMRegressor(random_state=42),
    'Extra Trees': ExtraTreesRegressor(),
    'AdaBoost': AdaBoostRegressor(),
    'SVR (linear)': SVR(kernel='linear'),
    'SVR (poly)': SVR(kernel='poly'),
    'Gaussian Process': GaussianProcessRegressor(),
    'KNN (tuned)': KNeighborsRegressor(n_neighbors=best_n_neighbors),
    'Bayesian Ridge': BayesianRidge(),
    'Ridge Regression': Ridge(),
    'Lasso Regression': Lasso(),
    'ElasticNet Regression': ElasticNet(),
    #'Bagging (Decision Tree)':
BaggingRegressor(base_estimator=DecisionTreeRegressor(random_state=42)) # Optional
random_state within the DecisionTree
}

# K-Fold Cross-Validation
```

```
kfold = KFold(n_splits=10, shuffle=True, random_state=42)

# Storage for evaluation results (empty dictionary)
all_results = {}

# Evaluate each model
for model_name, model in models.items():
  # Create a list to store fold results for this model
  fold_results = []
  for fold_number, (train_index, test_index) in enumerate(kfold.split(X)):
      X_train, X_test = X.iloc[train_index], X.iloc[test_index]
      y_train, y_test = y.iloc[train_index], y.iloc[test_index]

      model.fit(X_train, y_train)
      y_pred = model.predict(X_test)

      mse = mean_squared_error(y_test, y_pred)
      r2 = r2_score(y_test, y_pred)
      mae = mean_absolute_error(y_test, y_pred)

      fold_results.append({
          'fold_number': fold_number + 1,  # Start fold numbers from 1
          'MSE': mse,
          'R-squared': r2,
          'MAE': mae
      })
  # Add fold results for this model to the main dictionary
  all_results[model_name] = fold_results

# Calculate average metrics and include model names
summary_results = {model_name: {
    'Average MSE': np.mean([fold['MSE'] for fold in fold_results]),
    'Average R-squared': np.mean([fold['R-squared'] for fold in fold_results]),
    'Average MAE': np.mean([fold['MAE'] for fold in fold_results]),
    } for model_name, fold_results in all_results.items()}

# Create the DataFrame directly and transpose
df_summary = pd.DataFrame(summary_results).transpose()  # Transpose here

# Save the DataFrame
writer = pd.ExcelWriter('exp-13-noFS-results.xlsx')
df_summary.to_excel(writer, sheet_name='Summary')

# Add sheets for individual fold results
for model_name, fold_results in all_results.items():
    df = pd.DataFrame(fold_results)
    df.to_excel(writer, sheet_name=model_name, index=False)
```

```
writer.close()  # Save the workbook
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_validate
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
import lightgbm as lgb
from skfeature.function.information_theoretical_based import CIFE
from sklearn.feature_selection import SelectKBest
from sklearn.preprocessing import StandardScaler
from openpyxl import Workbook
from pyswarm import pso

# Load the preprocessed data
data = pd.read_csv('oh-scaled-dataset-2.csv')
X = data.drop('Class', axis=1)
y = data['Class']

# Define the models
seed = 42
models = {
    'Logistic Regression': LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=5000, random_state=seed),
    'k-NN': KNeighborsClassifier(),
    'SVM': SVC(decision_function_shape='ovr', random_state=seed),
    'Random Forest': RandomForestClassifier(random_state=seed),
    'Gradient Boosting': GradientBoostingClassifier(random_state=seed),
    'XGBoost': xgb.XGBClassifier(random_state=seed, use_label_encoder=False,
eval_metric='mlogloss'),
    'LightGBM': lgb.LGBMClassifier(random_state=seed),
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(max_iter=5000, random_state=seed)
}

# Function to optimize using PSO
def optimize_feature_selection(X, y, model):
    def objective(features):
        # Convert features to indices
        selected_features = np.where(features)[0]
        if len(selected_features) == 0:
            return np.inf  # Return a large value if no feature is selected
```

```
        # Subset X with selected features
        X_selected = X.iloc[:, selected_features]

        # Perform 10-fold cross-validation and return negative accuracy (to maximize
accuracy)
        cv_results = cross_validate(model, X_selected, y, cv=10, scoring='accuracy',
return_train_score=False)
        accuracy_mean = np.mean(cv_results['test_score'])
        return -accuracy_mean

    # Number of features
    num_features = X.shape[1]

    # PSO optimization
    lb = np.zeros(num_features)  # Lower bounds for features (0 for unselected, 1 for
selected)
    ub = np.ones(num_features)   # Upper bounds for features (0 for unselected, 1 for
selected)
    xopt, fopt = pso(objective, lb, ub, swarmsize=100, maxiter=5, debug=True)

    # Convert the best solution to indices of selected features
    selected_features = np.where(xopt)[0]
    return selected_features

# Define the scoring metrics
scoring_metrics = ['accuracy', 'precision_macro', 'recall_macro', 'f1_macro']

# Perform Particle Swarm Optimization for feature selection
results = {model_name: [] for model_name in models.keys()}

# Perform 10-fold cross-validation with PSO feature selection and collect detailed metrics
for model_name, model in models.items():
    # PSO feature selection
    selected_features = optimize_feature_selection(X, y, model)

    # Get the selected features from X
    X_selected = X.iloc[:, selected_features]

    # Perform 10-fold cross-validation
    cv_results = cross_validate(model, X_selected, y, cv=10, scoring=scoring_metrics,
return_train_score=True)
    results[model_name] = cv_results

# Save results to Excel
wb = Workbook()
summary_sheet = wb.active
summary_sheet.title = 'Average Results'
```

```
# Write headers for summary
summary_headers = ['Model', 'Accuracy Mean', 'Accuracy Std', 'Precision Mean',
'Precision Std', 'Recall Mean', 'Recall Std', 'F1 Mean', 'F1 Std', 'Selected Features']
summary_sheet.append(summary_headers)

# Fill in the summary sheet and create detailed sheets for each model
for model_name, cv_results in results.items():
    # Compute means and standard deviations for each metric
    accuracy_mean = np.mean(cv_results['test_accuracy'])
    accuracy_std = np.std(cv_results['test_accuracy'])
    precision_mean = np.mean(cv_results['test_precision_macro'])
    precision_std = np.std(cv_results['test_precision_macro'])
    recall_mean = np.mean(cv_results['test_recall_macro'])
    recall_std = np.std(cv_results['test_recall_macro'])
    f1_mean = np.mean(cv_results['test_f1_macro'])
    f1_std = np.std(cv_results['test_f1_macro'])

    # Write to summary sheet
    summary_row = [model_name, accuracy_mean, accuracy_std, precision_mean,
precision_std, recall_mean, recall_std, f1_mean, f1_std, ', '.join(selected_features)]
    summary_sheet.append(summary_row)

    # Create a new sheet for each model's fold results
    model_sheet = wb.create_sheet(title=f'{model_name} Folds')
    model_headers = ['Fold', 'Test Accuracy', 'Train Accuracy', 'Test Precision', 'Train
Precision', 'Test Recall', 'Train Recall', 'Test F1', 'Train F1']
    model_sheet.append(model_headers)

    # Write detailed results for each fold
    for fold_idx in range(10):
        fold_row = [
            fold_idx + 1,
            cv_results['test_accuracy'][fold_idx], cv_results['train_accuracy'][fold_idx],
            cv_results['test_precision_macro'][fold_idx],
cv_results['train_precision_macro'][fold_idx],
            cv_results['test_recall_macro'][fold_idx],
cv_results['train_recall_macro'][fold_idx],
            cv_results['test_f1_macro'][fold_idx], cv_results['train_f1_macro'][fold_idx]
        ]
        model_sheet.append(fold_row)

# Save the workbook
wb.save('exp-11-pso.xlsx')
print("Selected Features:", selected_features)
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_validate
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
import lightgbm as lgb
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from sklearn.ensemble import VotingClassifier
from openpyxl import Workbook

# Load the preprocessed data
data = pd.read_csv('oh-scaled-dataset-2.csv')
X = data.drop('Class', axis=1)
y = data['Class']

# Define the base models for ensemble
seed = 42
base_models = [
    ('logistic', LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=5000,
random_state=seed)),
    ('random_forest', RandomForestClassifier(random_state=seed))
]

# Create an ensemble model
ensemble_model = VotingClassifier(estimators=base_models, voting='soft', n_jobs=-1)

# Perform SFS with the ensemble model
sfs = SFS(ensemble_model,
          k_features='best',
          forward=True,
          scoring='accuracy',
          cv=5,
          n_jobs=-1)

sfs = sfs.fit(X, y)
selected_features = list(sfs.k_feature_names_)

# Reduce X to the selected features
X_selected = X[selected_features]

# Define the models to evaluate
models = {
    'Logistic Regression': LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=5000, random_state=seed),
    'k-NN': KNeighborsClassifier(),
    'SVM': SVC(decision_function_shape='ovr', random_state=seed),
```

```
    'Random Forest': RandomForestClassifier(random_state=seed),
    'Gradient Boosting': GradientBoostingClassifier(random_state=seed),
    'XGBoost': xgb.XGBClassifier(random_state=seed, use_label_encoder=False,
eval_metric='mlogloss'),
    'LightGBM': lgb.LGBMClassifier(random_state=seed),
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(max_iter=5000, random_state=seed)
}

# Define the scoring metrics
scoring_metrics = ['accuracy', 'precision_macro', 'recall_macro', 'f1_macro']

# Perform 10-fold cross-validation for each model and collect detailed metrics
results = {model_name: {} for model_name in models.keys()}

for model_name, model in models.items():
    cv_results = cross_validate(model, X_selected, y, cv=10, scoring=scoring_metrics,
return_train_score=True)
    results[model_name] = cv_results

# Save results to Excel
wb = Workbook()
summary_sheet = wb.active
summary_sheet.title = 'Average Results'

# Write headers for summary
summary_headers = ['Model', 'Accuracy Mean', 'Accuracy Std', 'Precision Mean',
'Precision Std', 'Recall Mean', 'Recall Std', 'F1 Mean', 'F1 Std', 'Selected Features']
summary_sheet.append(summary_headers)

# Fill in the summary sheet and create detailed sheets for each model
for model_name, cv_results in results.items():
    # Compute means and standard deviations for each metric
    accuracy_mean = np.mean(cv_results['test_accuracy'])
    accuracy_std = np.std(cv_results['test_accuracy'])
    precision_mean = np.mean(cv_results['test_precision_macro'])
    precision_std = np.std(cv_results['test_precision_macro'])
    recall_mean = np.mean(cv_results['test_recall_macro'])
    recall_std = np.std(cv_results['test_recall_macro'])
    f1_mean = np.mean(cv_results['test_f1_macro'])
    f1_std = np.std(cv_results['test_f1_macro'])

    # Write to summary sheet
    summary_row = [model_name, accuracy_mean, accuracy_std, precision_mean,
precision_std, recall_mean, recall_std, f1_mean, f1_std, ', '.join(selected_features)]
    summary_sheet.append(summary_row)

    # Create a new sheet for each model's fold results
```

```
    model_sheet = wb.create_sheet(title=f'{model_name} Folds')
    model_headers = ['Fold', 'Test Accuracy', 'Train Accuracy', 'Test Precision', 'Train
Precision', 'Test Recall', 'Train Recall', 'Test F1', 'Train F1']
    model_sheet.append(model_headers)

    # Write detailed results for each fold
    for fold_idx in range(10):
        fold_row = [
            fold_idx + 1,
            cv_results['test_accuracy'][fold_idx], cv_results['train_accuracy'][fold_idx],
            cv_results['test_precision_macro'][fold_idx],
cv_results['train_precision_macro'][fold_idx],
            cv_results['test_recall_macro'][fold_idx],
cv_results['train_recall_macro'][fold_idx],
            cv_results['test_f1_macro'][fold_idx], cv_results['train_f1_macro'][fold_idx]
        ]
        model_sheet.append(fold_row)

# Save the workbook
wb.save('exp-10-mlxSFS.xlsx')

import pandas as pd
import numpy as np
from sklearn.model_selection import cross_validate
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
import lightgbm as lgb
from openpyxl import Workbook

# Load the preprocessed data
data = pd.read_csv('oh-scaled-dataset-2.csv')
X = data.drop('Class', axis=1)
y = data['Class']

# Perform PCA for feature selection
pca = PCA(n_components=0.95)  # Retain 95% of the variance based on elbow analysis
X_pca = pca.fit_transform(X)

# Define the models
seed = 42
models = {
```

```
    'Logistic Regression': LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=5000, random_state=seed),
    'k-NN': KNeighborsClassifier(),
    'SVM': SVC(decision_function_shape='ovr', random_state=seed),
    'Random Forest': RandomForestClassifier(random_state=seed),
    'Gradient Boosting': GradientBoostingClassifier(random_state=seed),
    'XGBoost': xgb.XGBClassifier(random_state=seed, use_label_encoder=False,
eval_metric='mlogloss'),
    'LightGBM': lgb.LGBMClassifier(random_state=seed),
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(max_iter=5000, random_state=seed)
}

# Perform 10-fold CV and hold results
results = {model_name: [] for model_name in models.keys()}

# Define scoring metrics
scoring = ['accuracy', 'f1_macro', 'recall_macro', 'precision_macro']

# Perform 10-fold cross-validation and collect detailed metrics
for model_name, model in models.items():
    cv_results = cross_validate(model, X_pca, y, cv=10, scoring=scoring,
return_train_score=False)
    results[model_name] = cv_results

# Save results to Excel
wb = Workbook()
summary_sheet = wb.active
summary_sheet.title = 'Average Results'

# Write headers for summary
summary_headers = ['Model', 'Accuracy Mean', 'Accuracy Std', 'F1 Mean', 'F1 Std',
'Recall Mean', 'Recall Std', 'Precision Mean', 'Precision Std']
summary_sheet.append(summary_headers)

# Fill in the summary sheet and create detailed sheets for each model
for model_name, cv_results in results.items():
    # Compute means and standard deviations for the metrics
    accuracy_mean = np.mean(cv_results['test_accuracy'])
    accuracy_std = np.std(cv_results['test_accuracy'])
    f1_mean = np.mean(cv_results['test_f1_macro'])
    f1_std = np.std(cv_results['test_f1_macro'])
    recall_mean = np.mean(cv_results['test_recall_macro'])
    recall_std = np.std(cv_results['test_recall_macro'])
    precision_mean = np.mean(cv_results['test_precision_macro'])
    precision_std = np.std(cv_results['test_precision_macro'])

    # Write to summary sheet
```

```
    summary_row = [model_name, accuracy_mean, accuracy_std, f1_mean, f1_std,
recall_mean, recall_std, precision_mean, precision_std]
    summary_sheet.append(summary_row)

    # Create a new sheet for each model's fold results
    model_sheet = wb.create_sheet(title=f'{model_name} Folds')
    model_headers = ['Fold', 'Accuracy', 'F1', 'Recall', 'Precision']
    model_sheet.append(model_headers)

    # Write detailed results for each fold
    for fold_idx in range(10):
        fold_row = [fold_idx+1,
                    cv_results['test_accuracy'][fold_idx],
                    cv_results['test_f1_macro'][fold_idx],
                    cv_results['test_recall_macro'][fold_idx],
                    cv_results['test_precision_macro'][fold_idx]]
        model_sheet.append(fold_row)

# Add a sheet for selected features
features_sheet = wb.create_sheet(title='Selected Features')
features_sheet.append(['Selected Features'])
selected_features = [f'PC{i+1}' for i in range(X_pca.shape[1])]
for feature in selected_features:
    features_sheet.append([feature])

# Save the workbook
wb.save('exp-9-pca095.xlsx')

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.model_selection import cross_validate
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
import lightgbm as lgb
from openpyxl import Workbook

# Load the preprocessed data
data = pd.read_csv('oh-scaled-dataset-2.csv')
X = data.drop('Class', axis=1)
y = data['Class']
```

```
# Perform PCA to analyze explained variance
pca = PCA()
X_pca = pca.fit_transform(X)

# Calculate cumulative explained variance
cumulative_explained_variance = np.cumsum(pca.explained_variance_ratio_)

# Plot cumulative explained variance
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(cumulative_explained_variance) + 1),
cumulative_explained_variance, marker='o', linestyle='--')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Explained Variance Analysis')

# Find elbow point
diff = np.diff(cumulative_explained_variance)
elbow_point = np.argmax(diff < 0.05) + 1  # Adjust 0.05 threshold if needed
plt.axvline(x=elbow_point, color='r', linestyle='--', label=f'Elbow Point:
PC={elbow_point}')

# Annotate elbow point
plt.text(elbow_point + 1, cumulative_explained_variance[elbow_point - 1],
f'PC={elbow_point}', verticalalignment='bottom')

plt.grid()
plt.legend()
plt.tight_layout()

# Save the plot to a file
plt.savefig('explained_variance_analysis.png')

# Show the plot
plt.show()
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.model_selection import cross_validate
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
```

```
import lightgbm as lgb
from openpyxl import Workbook

# Load the preprocessed data
data = pd.read_csv('oh-scaled-dataset-2.csv')
X = data.drop('Class', axis=1)
y = data['Class']

# Perform PCA to analyze explained variance
pca = PCA()
X_pca = pca.fit_transform(X)

# Calculate cumulative explained variance
cumulative_explained_variance = np.cumsum(pca.explained_variance_ratio_)

# Plot cumulative explained variance
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(cumulative_explained_variance) + 1),
cumulative_explained_variance, marker='o', linestyle='--')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Explained Variance Analysis')

# Find elbow point
diff = np.diff(cumulative_explained_variance)
elbow_point = np.argmax(diff < 0.05) + 1  # Adjust 0.05 threshold if needed
plt.axvline(x=elbow_point, color='r', linestyle='--', label=f'Elbow Point:
PC={elbow_point}')

# Annotate elbow point
plt.text(elbow_point + 1, cumulative_explained_variance[elbow_point - 1],
f'PC={elbow_point}', verticalalignment='bottom')

plt.grid()
plt.legend()
plt.tight_layout()

# Save the plot to a file
plt.savefig('explained_variance_analysis.png')

# Show the plot
plt.show()

# Set n_components based on the elbow point from the explained variance plot
n_components = elbow_point  # Use the identified elbow point

# Perform PCA with the chosen number of components
pca = PCA(n_components=n_components)
```

```
X_pca_selected = pca.fit_transform(X)

# Define the models
seed = 42
models = {
    'Logistic Regression': LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=5000, random_state=seed),
    'k-NN': KNeighborsClassifier(),
    'SVM': SVC(decision_function_shape='ovr', random_state=seed),
    'Random Forest': RandomForestClassifier(random_state=seed),
    'Gradient Boosting': GradientBoostingClassifier(random_state=seed),
    'XGBoost': xgb.XGBClassifier(random_state=seed, use_label_encoder=False,
eval_metric='mlogloss'),
    'LightGBM': lgb.LGBMClassifier(random_state=seed),
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(max_iter=5000, random_state=seed)
}

# Perform 10-fold CV and hold results
results = {model_name: [] for model_name in models.keys()}

# Define scoring metrics
scoring = ['accuracy', 'f1_macro', 'recall_macro', 'precision_macro']

# Perform 10-fold cross-validation and collect detailed metrics
for model_name, model in models.items():
    cv_results = cross_validate(model, X_pca_selected, y, cv=10, scoring=scoring,
return_train_score=False)
    results[model_name] = cv_results

# Save results to Excel
wb = Workbook()
summary_sheet = wb.active
summary_sheet.title = 'Average Results'

# Write headers for summary
summary_headers = ['Model', 'Accuracy Mean', 'Accuracy Std', 'F1 Mean', 'F1 Std',
'Recall Mean', 'Recall Std', 'Precision Mean', 'Precision Std']
summary_sheet.append(summary_headers)

# Fill in the summary sheet and create detailed sheets for each model
for model_name, cv_results in results.items():
    # Compute means and standard deviations for the metrics
    accuracy_mean = np.mean(cv_results['test_accuracy'])
    accuracy_std = np.std(cv_results['test_accuracy'])
    f1_mean = np.mean(cv_results['test_f1_macro'])
    f1_std = np.std(cv_results['test_f1_macro'])
    recall_mean = np.mean(cv_results['test_recall_macro'])
```

```
    recall_std = np.std(cv_results['test_recall_macro'])
    precision_mean = np.mean(cv_results['test_precision_macro'])
    precision_std = np.std(cv_results['test_precision_macro'])

    # Write to summary sheet
    summary_row = [model_name, accuracy_mean, accuracy_std, f1_mean, f1_std,
recall_mean, recall_std, precision_mean, precision_std]
    summary_sheet.append(summary_row)

    # Create a new sheet for each model's fold results
    model_sheet = wb.create_sheet(title=f'{model_name} Folds')
    model_headers = ['Fold', 'Accuracy', 'F1', 'Recall', 'Precision']
    model_sheet.append(model_headers)

    # Write detailed results for each fold
    for fold_idx in range(10):
        fold_row = [fold_idx+1,
                    cv_results['test_accuracy'][fold_idx],
                    cv_results['test_f1_macro'][fold_idx],
                    cv_results['test_recall_macro'][fold_idx],
                    cv_results['test_precision_macro'][fold_idx]]
        model_sheet.append(fold_row)

# Add a sheet for selected features
features_sheet = wb.create_sheet(title='Selected Features')
features_sheet.append(['Selected Features'])
selected_features = [f'PC{i+1}' for i in range(X_pca_selected.shape[1])]
for feature in selected_features:
    features_sheet.append([feature])

# Save the workbook
wb.save('exp-9-pca-elbow.xlsx')

import pandas as pd
import numpy as np
from sklearn.model_selection import cross_validate
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
import lightgbm as lgb
from openpyxl import Workbook

# Load the preprocessed data
```

```
data = pd.read_csv('oh-scaled-dataset-2.csv')
X = data.drop('Class', axis=1)
y = data['Class']

# Perform Pearson Correlation Coefficient feature selection
correlation_threshold = 0.1  # We can adjust this threshold
correlation_with_target = X.apply(lambda x: x.corr(y)).abs()
selected_features = correlation_with_target[correlation_with_target >
correlation_threshold].index
X_selected = X[selected_features]

# Define the models
seed = 42
models = {
   'Logistic Regression': LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=5000, random_state=seed),
   'k-NN': KNeighborsClassifier(),
   'SVM': SVC(decision_function_shape='ovr', random_state=seed),
   'Random Forest': RandomForestClassifier(random_state=seed),
   'Gradient Boosting': GradientBoostingClassifier(random_state=seed),
   'XGBoost': xgb.XGBClassifier(random_state=seed, use_label_encoder=False,
eval_metric='mlogloss'),
   'LightGBM': lgb.LGBMClassifier(random_state=seed),
   'Naive Bayes': GaussianNB(),
   'Neural Network': MLPClassifier(max_iter=5000, random_state=seed)
}

# Perform 10-fold CV and hold results
results = {model_name: [] for model_name in models.keys()}

# Define scoring metrics
scoring = ['accuracy', 'f1_macro', 'recall_macro', 'precision_macro']

# Perform 10-fold cross-validation and collect detailed metrics
for model_name, model in models.items():
   cv_results = cross_validate(model, X_selected, y, cv=10, scoring=scoring,
return_train_score=False)
   results[model_name] = cv_results

# Save results to Excel
wb = Workbook()
summary_sheet = wb.active
summary_sheet.title = 'Average Results'

# Write headers for summary
summary_headers = ['Model', 'Accuracy Mean', 'Accuracy Std', 'F1 Mean', 'F1 Std',
'Recall Mean', 'Recall Std', 'Precision Mean', 'Precision Std']
summary_sheet.append(summary_headers)
```

```
# Fill in the summary sheet and create detailed sheets for each model
for model_name, cv_results in results.items():
    # Compute means and standard deviations for the metrics
    accuracy_mean = np.mean(cv_results['test_accuracy'])
    accuracy_std = np.std(cv_results['test_accuracy'])
    f1_mean = np.mean(cv_results['test_f1_macro'])
    f1_std = np.std(cv_results['test_f1_macro'])
    recall_mean = np.mean(cv_results['test_recall_macro'])
    recall_std = np.std(cv_results['test_recall_macro'])
    precision_mean = np.mean(cv_results['test_precision_macro'])
    precision_std = np.std(cv_results['test_precision_macro'])

    # Write to summary sheet
    summary_row = [model_name, accuracy_mean, accuracy_std, f1_mean, f1_std,
recall_mean, recall_std, precision_mean, precision_std]
    summary_sheet.append(summary_row)

    # Create a new sheet for each model's fold results
    model_sheet = wb.create_sheet(title=f'{model_name} Folds')
    model_headers = ['Fold', 'Accuracy', 'F1', 'Recall', 'Precision']
    model_sheet.append(model_headers)

    # Write detailed results for each fold
    for fold_idx in range(10):
        fold_row = [fold_idx+1,
                    cv_results['test_accuracy'][fold_idx],
                    cv_results['test_f1_macro'][fold_idx],
                    cv_results['test_recall_macro'][fold_idx],
                    cv_results['test_precision_macro'][fold_idx]]
        model_sheet.append(fold_row)

# Add a sheet for selected features
features_sheet = wb.create_sheet(title='Selected Features')
features_sheet.append(['Selected Features'])
for feature in selected_features:
    features_sheet.append([feature])

# Save the workbook
wb.save('exp-8c01.xlsx')

#Experiment 8: Drop those above threshold version
import pandas as pd
from sklearn.model_selection import cross_validate
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
import lightgbm as lgb
from openpyxl import Workbook

# Load the preprocessed data
data = pd.read_csv('oh-scaled-dataset-2.csv')
X = data.drop('Class', axis=1)
y = data['Class']

# Perform Pearson Correlation Coefficient feature selection
correlation_threshold = 0.1  # You can adjust this threshold
correlation_matrix = X.corr().abs()
upper_triangle = correlation_matrix.where(np.triu(np.ones(correlation_matrix.shape),
k=1).astype(bool))
to_drop = [column for column in upper_triangle.columns if any(upper_triangle[column] >
correlation_threshold)]
X_selected = X.drop(columns=to_drop)

# Define the models
seed = 42
models = {
    'Logistic Regression': LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=5000, random_state=seed),
    'k-NN': KNeighborsClassifier(),
    'SVM': SVC(decision_function_shape='ovr', random_state=seed),
    'Random Forest': RandomForestClassifier(random_state=seed),
    'Gradient Boosting': GradientBoostingClassifier(random_state=seed),
    'XGBoost': xgb.XGBClassifier(random_state=seed, use_label_encoder=False,
eval_metric='mlogloss'),
    'LightGBM': lgb.LGBMClassifier(random_state=seed),
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(max_iter=5000, random_state=seed)
}

# Perform 10-fold CV and hold results
results = {model_name: [] for model_name in models.keys()}

# Define scoring metrics
scoring = ['accuracy', 'f1_macro', 'recall_macro', 'precision_macro']

# Perform 10-fold cross-validation and collect detailed metrics
for model_name, model in models.items():
    cv_results = cross_validate(model, X_selected, y, cv=10, scoring=scoring,
return_train_score=False)
```

```
    results[model_name] = cv_results

# Save results to Excel
wb = Workbook()
summary_sheet = wb.active
summary_sheet.title = 'Average Results'

# Write headers for summary
summary_headers = ['Model', 'Accuracy Mean', 'Accuracy Std', 'F1 Mean', 'F1 Std',
'Recall Mean', 'Recall Std', 'Precision Mean', 'Precision Std']
summary_sheet.append(summary_headers)

# Fill in the summary sheet and create detailed sheets for each model
for model_name, cv_results in results.items():
    # Compute means and standard deviations for the metrics
    accuracy_mean = np.mean(cv_results['test_accuracy'])
    accuracy_std = np.std(cv_results['test_accuracy'])
    f1_mean = np.mean(cv_results['test_f1_macro'])
    f1_std = np.std(cv_results['test_f1_macro'])
    recall_mean = np.mean(cv_results['test_recall_macro'])
    recall_std = np.std(cv_results['test_recall_macro'])
    precision_mean = np.mean(cv_results['test_precision_macro'])
    precision_std = np.std(cv_results['test_precision_macro'])

    # Write to summary sheet
    summary_row = [model_name, accuracy_mean, accuracy_std, f1_mean, f1_std,
recall_mean, recall_std, precision_mean, precision_std]
    summary_sheet.append(summary_row)

    # Create a new sheet for each model's fold results
    model_sheet = wb.create_sheet(title=f'{model_name} Fold Results')
    model_headers = ['Fold', 'Accuracy', 'F1', 'Recall', 'Precision']
    model_sheet.append(model_headers)

    # Write detailed results for each fold
    for fold_idx in range(10):
        fold_row = [fold_idx+1,
                cv_results['test_accuracy'][fold_idx],
                cv_results['test_f1_macro'][fold_idx],
                cv_results['test_recall_macro'][fold_idx],
                cv_results['test_precision_macro'][fold_idx]]
        model_sheet.append(fold_row)

# Save the workbook
wb.save('experiment-set-8-results.xlsx')

import pandas as pd
```

```
from sklearn.model_selection import cross_validate
import numpy as np

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
import lightgbm as lgb

from openpyxl import Workbook

# Load the preprocessed data (This is the one-hot encoding version)
data = pd.read_csv('oh-scaled-dataset-2.csv')
X = data.drop('Class', axis=1)
y = data['Class']

#Define the models
seed = 42

models = {
    'Logistic Regression': LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=5000, random_state=seed),
    'k-NN': KNeighborsClassifier(),
    'SVM': SVC(decision_function_shape='ovr', random_state=seed),
    'Random Forest': RandomForestClassifier(random_state=seed),
    'Gradient Boosting': GradientBoostingClassifier(random_state=seed),
    'XGBoost': xgb.XGBClassifier(random_state=seed, use_label_encoder=False,
eval_metric='mlogloss'),
    'LightGBM': lgb.LGBMClassifier(random_state=seed),
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(max_iter=5000, random_state=seed)
}


## Perform 10-fold CV and hold results
# Initialize a dictionary to hold results
results = {model_name: [] for model_name in models.keys()}

# Define scoring metrics
scoring = ['accuracy', 'f1_macro', 'recall_macro', 'precision_macro']

# Perform 10-fold cross-validation and collect detailed metrics
for model_name, model in models.items():
    cv_results = cross_validate(model, X, y, cv=10, scoring=scoring,
return_train_score=False)
```

```
    results[model_name] = cv_results

##Save results to Excel
# Create a workbook and sheets
wb = Workbook()
summary_sheet = wb.active
summary_sheet.title = 'Average Results'
details_sheet = wb.create_sheet(title='Fold Results')

# Write headers for summary
summary_headers = ['Model', 'Accuracy Mean', 'Accuracy Std', 'F1 Mean', 'F1 Std',
'Recall Mean', 'Recall Std', 'Precision Mean', 'Precision Std']
summary_sheet.append(summary_headers)

# Write headers for detailed fold results
details_headers = ['Model', 'Fold', 'Accuracy', 'F1', 'Recall', 'Precision']
details_sheet.append(details_headers)

# Fill in the summary sheet and detailed results sheet
for model_name, cv_results in results.items():
    # Compute means and standard deviations for the metrics
    accuracy_mean = np.mean(cv_results['test_accuracy'])
    accuracy_std = np.std(cv_results['test_accuracy'])
    f1_mean = np.mean(cv_results['test_f1_macro'])
    f1_std = np.std(cv_results['test_f1_macro'])
    recall_mean = np.mean(cv_results['test_recall_macro'])
    recall_std = np.std(cv_results['test_recall_macro'])
    precision_mean = np.mean(cv_results['test_precision_macro'])
    precision_std = np.std(cv_results['test_precision_macro'])

    # Write to summary sheet
    summary_row = [model_name, accuracy_mean, accuracy_std, f1_mean, f1_std,
recall_mean, recall_std, precision_mean, precision_std]
    summary_sheet.append(summary_row)

    # Write detailed results for each fold
    for fold_idx in range(10):
        details_row = [model_name, fold_idx+1,
                       cv_results['test_accuracy'][fold_idx],
                       cv_results['test_f1_macro'][fold_idx],
                       cv_results['test_recall_macro'][fold_idx],
                       cv_results['test_precision_macro'][fold_idx]]
        details_sheet.append(details_row)

# Save the workbook
wb.save('experiment-set-7-results.xlsx')


import pandas as pd
```

```
from sklearn.model_selection import cross_validate
import numpy as np

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
import lightgbm as lgb

from openpyxl import Workbook

# Load the preprocessed data (this is a different version - but i eventually used the OH
version)
data = pd.read_csv('c-scaled-dataset-2.csv')
X = data.drop('Class', axis=1)
y = data['Class']

#Define the models
seed = 42

models = {
    'Logistic Regression': LogisticRegression(multi_class='multinomial', solver='lbfgs',
max_iter=500, random_state=seed),
    'k-NN': KNeighborsClassifier(),
    'SVM': SVC(decision_function_shape='ovr', random_state=seed),
    'Random Forest': RandomForestClassifier(random_state=seed),
    'Gradient Boosting': GradientBoostingClassifier(random_state=seed),
    'XGBoost': xgb.XGBClassifier(random_state=seed, use_label_encoder=False,
eval_metric='mlogloss'),
    'LightGBM': lgb.LGBMClassifier(random_state=seed),
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(max_iter=500, random_state=seed)
}
## Perform 10-fold CV and hold results
# Initialize a dictionary to hold results
results = {model_name: [] for model_name in models.keys()}

# Define scoring metrics
scoring = ['accuracy', 'f1_macro', 'recall_macro', 'precision_macro']

# Perform 10-fold cross-validation and collect detailed metrics
for model_name, model in models.items():
```

```
    cv_results = cross_validate(model, X, y, cv=10, scoring=scoring,
return_train_score=False)
    results[model_name] = cv_results

##Save results to Excel
# Create a workbook and sheets
wb = Workbook()
summary_sheet = wb.active
summary_sheet.title = 'Average Results'
details_sheet = wb.create_sheet(title='Fold Results')

# Write headers for summary
summary_headers = ['Model', 'Accuracy Mean', 'Accuracy Std', 'F1 Mean', 'F1 Std',
'Recall Mean', 'Recall Std', 'Precision Mean', 'Precision Std']
summary_sheet.append(summary_headers)

# Write headers for detailed fold results
details_headers = ['Model', 'Fold', 'Accuracy', 'F1', 'Recall', 'Precision']
details_sheet.append(details_headers)

# Fill in the summary sheet and detailed results sheet
for model_name, cv_results in results.items():
    # Compute means and standard deviations for the metrics
    accuracy_mean = np.mean(cv_results['test_accuracy'])
    accuracy_std = np.std(cv_results['test_accuracy'])
    f1_mean = np.mean(cv_results['test_f1_macro'])
    f1_std = np.std(cv_results['test_f1_macro'])
    recall_mean = np.mean(cv_results['test_recall_macro'])
    recall_std = np.std(cv_results['test_recall_macro'])
    precision_mean = np.mean(cv_results['test_precision_macro'])
    precision_std = np.std(cv_results['test_precision_macro'])

    # Write to summary sheet
    summary_row = [model_name, accuracy_mean, accuracy_std, f1_mean, f1_std,
recall_mean, recall_std, precision_mean, precision_std]
    summary_sheet.append(summary_row)

    # Write detailed results for each fold
    for fold_idx in range(10):
        details_row = [model_name, fold_idx+1,
                       cv_results['test_accuracy'][fold_idx],
                       cv_results['test_f1_macro'][fold_idx],
                       cv_results['test_recall_macro'][fold_idx],
                       cv_results['test_precision_macro'][fold_idx]]
        details_sheet.append(details_row)

# Save the workbook
wb.save('/mnt/data/experiment-set-7-results.xlsx')
```

```
#All models on PCA

import pandas as pd
import numpy as np
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, KFold
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import ExtraTreesRegressor, AdaBoostRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import BayesianRidge, Ridge, Lasso, ElasticNet
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.decomposition import PCA
import xgboost as xgb
import lightgbm as lgb

# Load the data from 'scaled_data.csv'
df = pd.read_csv('scaled_data.csv')
X = df.drop(['ESE', 'ApplicantName'], axis=1)
y = df['ESE']

# Feature Selection using PCA
pca = PCA(n_components=0.95)  # Adjust 'n_components' as needed
X_pca = pca.fit_transform(X)

# Define models to evaluate
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Random Forest': RandomForestRegressor(random_state=42),
    'SVR (RBF)': SVR(kernel='rbf'),
    'KNN': KNeighborsRegressor(),
    'MLP': MLPRegressor(max_iter=1000, learning_rate='adaptive', random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42),
    'XGBoost': xgb.XGBRegressor(random_state=42),
    'LightGBM': lgb.LGBMRegressor(random_state=42),
    'Extra Trees': ExtraTreesRegressor(),
    'AdaBoost': AdaBoostRegressor(),
    'SVR (linear)': SVR(kernel='linear'),
    'SVR (poly)': SVR(kernel='poly'),
    'Gaussian Process': GaussianProcessRegressor(),
```

```
    'KNN (tuned)': KNeighborsRegressor(n_neighbors=7),
    'Bayesian Ridge': BayesianRidge(),
    'Ridge Regression': Ridge(),
    'Lasso Regression': Lasso(),
    'ElasticNet Regression': ElasticNet(),
}

# K-Fold Cross-Validation
kfold = KFold(n_splits=10, shuffle=True, random_state=42)

# Storage for evaluation results
all_results = {}

# Evaluate each model
for model_name, model in models.items():
    fold_results = []
    for fold_number, (train_index, test_index) in enumerate(kfold.split(X_pca)):
        X_train, X_test = X_pca[train_index], X_pca[test_index]
        y_train, y_test = y[train_index], y[test_index]

        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)
        mae = mean_absolute_error(y_test, y_pred)

        fold_results.append({
            'fold_number': fold_number + 1,
            'MSE': mse,
            'R-squared': r2,
            'MAE': mae
        })

    all_results[model_name] = fold_results

# Calculate average metrics
summary_results = {model_name: {
    'Average MSE': np.mean([fold['MSE'] for fold in fold_results]),
    'Average R-squared': np.mean([fold['R-squared'] for fold in fold_results]),
    'Average MAE': np.mean([fold['MAE'] for fold in fold_results]),
    } for model_name, fold_results in all_results.items()}

# Create the DataFrame directly and transpose
df_summary = pd.DataFrame(summary_results).transpose()  # Transpose here

# Save Results
writer = pd.ExcelWriter('model_comparison_results_exp-3.xlsx')
```

```
#Save Summary
df_summary.to_excel(writer, sheet_name='Summary')

# Add sheets for individual fold results
for model_name, fold_results in all_results.items():
    df = pd.DataFrame(fold_results)
    df.to_excel(writer, sheet_name=model_name, index=False)

writer.close()

print(X_pca)
```

Bio-data

1.	**Name:**	Felicia Ojiyovwi Adelodun
2.	**Mobile N0:**	08033815562
3.	**Email Address:**	adelodunfelicia@gmail.com
4.	**Marital Status**:	Married
5.	**Number of Children**:	Two
6.	**Date and Places of Birth:**	3rd March, 1965, Benin, Edo State
7.	**Nationality/State of origin**:	Nigerian / Oyo State
8.	**Current Post Address:**	Department of computer Studies, Faculty of Science, The Polytechnic, Ibadan
9.	**Date and Grade of First Appointment**:	15th March 1994 / USS 8 Step 3
10	**Date of Confirmation**:	15th March 1997
11.	**Date and Grade of Last Promotion:**	11th October, 2022/CONCPAS 07 Step 1
12.	**Present Grade and Salary**:	CONCPAS 07 Step 2
13.	**Date and Grade of Current Appointment**	11th October, 2022 / CONCPAS 07 Step 02

Educational Institutions Attended with Dates:

i	Ladoke Akintola University of Technology, Ogbomosho	2014 -2020
ii.	Federal University of Technology, Akure	2000–2004
iii.	University of Benin, Edo State.	1984–1988
iv.	Abubakar Tafawa Balewa University, Bauchi	1990–1991
v.	Baptist Girls' High School, Agbor, Delta State	1977-1982

9. A. **Academic Qualifications and Distinctions with Dates**

i.	M. Tech Computer Science	2020
ii.	Post-graduate Diploma in Computer Science	2004
iii.	B.Sc Industrial Mathematics	1990
vi.	Certificate COBOL Programming	1991
v.	WAEC May/June	1982
vi.	WAEC Nov/Dec.	2012

B. **Professional Qualifications with Dates**

i.	Computer Professional Registration Council of Nig.	2022

C. **Research/Project/Invention/Innovation/Design**

Nil

D. **Publications**

Journals:

i. Sarumi, J.A. (PhD), Longe, O.B. & **Adelodun, F.O,** "An Empirical Evaluation of the Effectiveness of the Computer-Based Network Security and Firewall in Banking Systems"*Journal of Advances in Mathematics & Computations Science.* Vol. 9, No. 2, Pp 21 -31. 2022. DOI: dx.doi.org/10.22624/AIMS/MATHS/V10N1P3 Available online at www.isteams.net/mathematics-computationaljournal.

ii. Sarumi, J.A.., **Adelodun, F.O. &** Longe, O.B., "A Theoretical Perspective on the Effect of Communication on Cybercrimes in Nigeria" *Social Informatics, Business, Politics, Law, Environmental Sciences & Technology Journal.* Vol. 7. No 2, Pp 69-78. 2021 www.insteams/socialinformaticsjournal.

iii. Egbetade, S.A. & **Adelodun, F.O**, "Mathematical Analysis for the Dynamics of Broiler Production in Nigeria" *Journal of Engineering, Technology, Sciences and Environmental Management (R & D JETSEM ISSN: 2705-277X)*. Vol. 2, (1) 2021: 105-109.

iv. Egbetade, S.A., Olatunji O.O., **Adelodun F.O**., Oni. S.O., Olawoore, W.A., Sangotola, T.M., & Adebayo, W.A "Modification of Mathematical Model for the Dynamics of Cholera Transmission" *Journal of Engineering, Technology, Sciences and Environmental Management (R & D JETSEM ISSN: 2705-277X)*. Vol. 2, (1) 2021, 100-104.

v. Adebisi, A. Baale, Olawumi R. Olasunkanmi & **Felicia E. Adelodun,** "Opinion Analysis and Machine Learning Modeling for Depression Detection" *ULJRT/United International Journal of Research & Technology.* 2021, Vol. 02 (4): 38 – 43.

vi. Adebisi, Abimbola Baale, Roseline Olawumi Olasunkan,& **Ojiyovwi Felicia Adelodun**, "Mobile Students' Academic Record Manager" *Annals Computer Science Series.* 2018 Vol. 16 (2): 161 – 171.

vii. Baale, A.A and **Adelodun F.O.** , "Analyses of Students' Vocational Data Using Some Selected Classification Algorithms" *Advances in Mathematical & Computational Science* 2018Vol. 6(2): 51-62.

Conferences:

i. **Adelodun, Felicia Ojiyovwi,** Oguns, Oyetunde Josephine & Ganiyu AminatAbidemi: (2022): Big Data in the Industry and its Challenges: A Survey. Paper presented at The Polytechnic, Ibadan, Faculty of Science, 11th National Conference. 4th – 6th July, 2022.

ii Oguns, Yetunde Josephine, Akinlade, Yemisi Omolara, **Adelodun, Felicia Ojiyovwi** & Ganiyu, AminatAbidemi: (2022): Artificial Intelligence Based E-Learning System: A Review and Recommendation. Paper presented at The Polytechnic, Ibadan, Faculty of Science, 11th National Conference. 4th – 6th July, 2022.

iii. Oni, Oluwabunmi Ayankemi, Omotosho, Folorunsho Segun &**Adelodun, Felicia Ojiyovwi:** (2022): A Self Pace Expert System on Early Detection of Covid-19. Paper presented at 2nd International Conference, Faculty of Natural Science, Ajayi Crowther University, Oyo State, Nigeria. Monday 4th – Wednesday 6th April, 2022.

iv. Omotosho, Oluwabusayo, I. Baale Abimbola Adebisi, Oladejo, Olajide Ademola & **Adelodun, Felicia Ojiyovwi: (**2021): An Exploratory Study of Recurrent Neural Networks for Cybersecurity. Accra Bespoke Innovation Conference & The Africa AI Stakeholders Summit An iSTEAMS Event Series. Paper presented at Academic City University College, Acra Ghana, 12th – 14th December, 2021.

v. Egbetade, S.A., & **Adelodun, F.O.: (**2021): An Error Estimation Technique for Solution of Linear Differential Equations with the TAU Method. Paper presented at Federal Polytechnic, Ede, Osun State, 13th International Conference on Science, Engineering and Environmental Technology (ICONSEET 2021) (Physical and Virtual), 6th – 10th September, 2021

vi. Egbetade, S.A., Olatunji, O.O., Ganiyu, K.A & **Adelodun, F.O**.:(2021): Global Asympotic Stability of Endemic Equilibrium of SIR Model of Tuberculosis with Lyapunov Function Method. Paper presented at Faculty of Science, 10th National Conference held at the Assembly Hall, The Polytechnic, Ibadan.

vii. **Adelodun Felicia Ojiyovwi** & Baale Abimbola Adebisi (2018): Analyses of Students' Vocational Data Using Some Selected Classification Algorithms. Paper presented at the 13thiSTEAMS Multidisciplinary Conference, held at the University of Ghana, Legon, Accra, Ghana. 24th – 26th October, 2018. Pp 179 – 188.

viii Omotosho Folorunsho Segun, **Adelodun Felicia Ojiyovwi,** Oguns Yetunde Josephine, Sonubi Tajudeen Adegboyega & Adeyemo Timilehin Kehinde (2017): Vulnerability Assessment of Biometric Systems- An empirical study. Paper presented at 1st Annual National Conference held at The Ibarapa Polytechnic, Eruwa, 17th – 20th July, 2017.

ix. Adewole Olumide, Fadiora Babatunde, **Adelodun Felicia** &Oguns Yetunde (2013): Framework for the Investigation of the time efficiency of insertion and merge sort algorithms in GSM phone directory creation. Paper presented at Faculty of Science National Conference held at The North Campus Assembly Hall, The Polytechnic, Ibadan.

E. Work Experience:

i. **Employer**: The Polytechnic, Ibadan
Nature of Duty: Senior Lecturer
Date: 2022 to Date

ii. **Employer**: The Polytechnic, Ibadan
Nature of Duty: Chief Programmer (Department of Computer Studies)
Date: 2000 to 2022

iii. **Employer**: The Polytechnic, Ibadan
Nature of Duty: Systems Programmer Grade 1 (Department of Curriculum and Science)
Date: 1994 – 2000

iv. **Employer**: Continuing Education Centre, The Polytechnic, Ibadan.
Nature of Duty: Teaching
Date: 2005 to date

v. **Employer**: Centre for Preliminary Programme, The Polytechnic, Ibadan
Nature of Duty: Teaching
Date: 2007 – 2010

vi. **Employer**: The Federal Polytechnic, Bauchi
Nature of Duty: Lecturing (Lecturer Three)
Date: 1991 - 1993

vi. **Employer**: Command Day Secondary School, Bauchi
Nature of Duty: Teaching
Date: 1990 – 1991

MEMBERSHIP OF PROFESSIONAL BODY
Computer Professionals of Nigeria (CPN) 008015/2022

PROJECT SUPERVISED: 250 Including ND and HND Projects

Courses Taught in the Polytechnic, Ibadan

1. CSC 321: Numerical Methods, Department of Computer Science
2. CSC 322: Operation Research II, Department of Computer Science
3. COM 213: COBOL Programming Language, Department of Computer Science
4. COM 125: Introduction to System Analysis & Design, Department of Computer Science
5. COM 223: Management Information System, Department of Computer Science
6. COM 215: Computer Application Packages, Department of Computer Science
7. CSC 301: Cobol Programming Language, Department of Mathematics and Statistics

8. CSC 301: Computer Application (FORTRAN), Department of Physics and Electronics

9. COM 301: Scientific Programming Language, Department of Biology/Microbiology (Part-Time)

10. COM 303: Computer Application Package, Department of Biology/Microbiology

11. SUG 316: Fundamentals of Computer, Department of Survey and Geo-Informants

12. SUG 326: Computer Application, Department of Survey and Geo-Informants

13. CSC 113: Introduction to Computer for Statistics, Department of Mathematics and Statistics

14. MAA 324: Management Information System (MIS) Department of Accountancy

15. MTS 014: Mathematics CPP, The Polytechnic, Ibadan.

16. CSC 415: Data Communication (Part Time), CEC, The Polytechnic, Ibadan

17. CSC 311: Operation Research 1 (Part Time), CEC, The Polytechnic, Ibadan

18. CSC 312: Numerical Methods (Part Time), CEC, The Polytechnic, Ibadan

19. CSC 213: COBOL Programming (Part-Time), CEC, The Polytechnic, Ibadan

20. PAD 429: Computer Programming, Department of Geology

10. **Present Employment Status and Salary:**

The Polytechnic, Ibadan / Chief System Programmer/CONTEDISS 14 Step 4

11. **Extra Curriculum Activities:** Teaching, Reading, Travelling, Meeting and Interacting with people of interest.

12. **Names of References**

(i). Mrs. O.O. Salawu
Department of Mathematics and Statistics,
The Polytechnic, Ibadan, Oyo State.
08035793328

(ii). Dr. I.T. Ayorinde

Department of Computer Science,
University of Ibadan, Oyo State.
08035289814

(iii). Dr. M.F. Babalola
Department of Computer Studies,
The Polytechnic, Ibadan, Oyo State.
08052236814.

Mrs. Adelodun, Felicia Ojiyovwi **……**

University Compliance Form

This is to certify that this thesis is by Felicia Ojiyovwi ADELODUN with matriculation Number LCU/PG/002519 in the Department of Computer Science, Faculty of Natural and Applied Sciences, Lead City University, Ibadan is in full compliance with the approval vof the University's format and style.

…….. ……

Signature **Date**

14% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Small Matches (less than 10 words)
- Submitted works

Match Groups

197 Not Cited or Quoted 9%
Matches with neither in-text citation nor quotation marks

1 Missing Quotations 0%
Matches that are still very similar to source material

101 Missing Citation 5%
Matches that have quotation marks, but no in-text citation

1 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

13% Internet sources
11% Publications
0% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.