University of Hertfordshire UH

School of Physics, Engineering and Computer Science

MSc Data Science Project

7PAM2002-0901-2024

Department of Physics, Astronomy and Mathematics

Data Science FINAL PROJECT REPORT

Project Title:

Optimizing Employee Attrition Prediction with Feature Selection and Machine Learning

Student Name and SRN:

DHANA SRIVALLI.GOLUKONDA and 21087223

Supervisor: Mykola Gordovskyy

Date Submitted: 6th January 2025

Word Count: 6727

GitHub address: https://github.com/DhanaSrivalli/Optimizing-Employee-Attrition-Prediction-with-Feature-Selection-and-Machine-Learning.git

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science at the University of Hertfordshire.

I have read the guidance to students on academic integrity, misconduct and plagiarism information at Assessment Offences and Academic Misconduct and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project module or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6).

I have not used chatGPT, or any other generative AI tool, to write the report or code (other than where declared or referenced).

I did not use human participants or undertake a survey in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student SRN number: 21087223

Student Name printed: DHANA SRIVALLI.GOLUKONDA

Student signature: Dhana Srivalli.Golukonda

UNIVERSITY OF HERTFORDSHIRE
SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

Abstract

Employee Attrition is a major problem in organizations that results in monetary losses, disrupted interruption, loss of quality in human capital. The reasons for trying to accurately predict employee turnover are because such information directly contributes to the design of an effective retention plan and the management of problems associated with turnover. Dependent solely on the quality and relevance of the features used, both machine learning models couldn't attain certain success in this domain with certain level of success. The effect of using feature selection techniques is the main focus of this research in improving the efficiency of predictive models in identifying employee turnover. Based on the IBM HR Analytics Attrition dataset obtained from Kaggle, L1 regularization technique as well as Recursive Feature Elimination (RFE) and combined techniques (union and intersection) applied to select number of influential attributes affecting attrition. Experimental results of Gradient Boosting, Decision Tree and MLP are compared with results obtained with feature selection applied and without its application. Performance metrics including accuracy, precision, recall, F1 measure, False Positive Rate (FPR), and False Negative Rate (FNR) are used to assess the model. Overall, Gradient Boosting and Decision Tree models have relatively good result with 32 original features, the highest accuracy rate of 91% was achieved with XGB and HGB models. Feature selection techniques especially the union of L1 and RFE marginally enhanced the performance of these models selecting 29 important features. However, both over-reduction methods, which use L1 and RFE (11 features) and L2 (five features), decrease the predictive accuracy. Among the four models developed, Decision Tree and MLP models are significantly impacted by feature selection, and the use of RFE enhances model performance. The findings are consistent with the need to find the right compromise between feature set dimensionality and model performance and risk of overfitting and the excessive load on the computation resources on the other. The strategy used to select features in this research is the union of L1 and RFE and is suggested as the most effective for attrition prediction, revealing important factors influencing turnover. Hence, the research contributes for efficient predictive models and feature selection procedures in relation to data-driven management of workers.

Table of Contents

1 Introduction

1.1 Background

The problems of employee attrition, resulting in a loss of invaluable tacit knowledge and costing organizations in the form of hiring and training new staff becomes substantial. Attrition rates indicate that organizations are becoming more interested in deducing the causes of employee turnover to improve their retention strategies. The use of human resource practices needs to be optimised by predicting attrition and identifying key factors contributing to attrition (Mansor, Sani and Aliff, 2021). Therefore, employee attrition is of paramount importance in human resource management and hence it is essential for the organizations targeting to maximize on their workforce and maintaining competitive advantage (Nurhindarto et al, 2021). This interplay between understanding employee attrition and leveraging data-driven insights underlines the position of feature selection in ML, which can further boost HR strategies. This will eventually allow stakeholders to make decisions that retain and satisfy employees, and periodic feature selection ultimately makes them streamline the model with transition (Shafie et al*, 2024).*

1.2 Problem Statement

Predictive models based on traditional techniques frequently face problems with high dimensionality and feature noise resulting in overfitting and poor generalization on unseen data. In the arena of employee attrition, this is extremely relevant because there are many factors that can prompt an employee to resign, everything from personal issues to company culture. The organizations are trying to mine the data for better decision making, robust feature selection techniques are extremely imperative. However, models become hard to interpret without effective feature selection, leading to unwieldy and eventually worse models that HR professionals can't leverage for actionable insights.

1.3 Aim

The aim of this research is to optimize the employee attrition prediction by evaluating the efficiency of feature selection techniques (L1 regularization, Recursive Feature Elimination, union, and intersection of L1 and RFE) in combination with machine learning models. The research seeks to identify the most effective pair of feature selection method and model for improving accuracy while reducing overfitting, and computational inefficiency in employee attrition prediction.

1.4 Research Questions:

- How does a using different feature selection technique (L1 regularization, Recursive Feature Elimination, union of L1 and RFE, intersection of L1 and RFE)

affect the Gradient Boosting and Decision Tree model accuracy in predicting the attrition of employees?

- Which attributes are found to be most significant for calculating employee attrition as identified by the feature selection techniques?

1.5 Project Objectives

- To review current literature on analysis of employee attrition, feature selection, and machine learning techniques.
- To acquire the IBM HR Analytics Attrition Dataset from Kaggle and preprocess the data by dealing with the missing values, duplications, and ensuring the quality of dataset.
- To implement feature selection techniques such as L1 regularization and Recursive Feature Elimination (RFE) to choose the most relevant features and formulate feature subsets from the union and intersection of the features selected by L1 and RFE.
- To implement and train Gradient Boosting, Decision Tree models and MLP on the full dataset without applying any feature selection techniques.
- To apply L1, RFE, union of L1 and RFE, and intersection of L1 and RFE to choose feature subsets and then use these subsets to train Gradient Boosting, Decision Tree and MLP models.
- To evaluate the effectiveness of the proposed models based on metrics like accuracy, precision, recall, F1-Score, False Positive Rate (FPR), and False Negative Rate (FNR).
- To compare the performance of Gradient Boosting, Decision Tree models and MLP while using the L1, RFE, union of L1 and RFE, and intersection of L1 and RFE feature selection techniques.
- To determine the key factors that affect employee turnover and identify the best combination of feature selection technique and machine learning and deep learning algorithms for optimized employee attrition prediction.

1.6 Research outline

Figure 1 Research Outline

2 Literature Review

Recent research in attrition prediction using ML and feature selection approaches is reviewed and recent innovation and findings in the field are discussed.

2.1 Feature Selection Techniques for employee attrition prediction

Most studies on employee attrition prediction have employed feature selection methods in order to improve the forecasting accuracy of the chosen model by selecting the most effective variables. Most of these works employ Filter methods, including Chi-Square, ANOVA, and Pearson Correlation to quantitatively assess the correlation between individual features and the target variable. There are also wrapper methods like Recursive Feature Elimination (RFE) where features are removed and added to the model based on the impact they have on the model's accuracy. Some of the popular approaches are Lasso (L1) and Ridge (L2) regularization techniques that are used in the model training process to avoid over fitting (Van Dam, 2021). These techniques are crucial in the analysis of data to achieve dimensionality reduction, increased model understandability and increased model performance.

2.2 Recent research on Optimizing Employee Attrition with Feature Selection

In this study (Mansor, Sani and Aliff, 2021) ML classifiers had been compared to determine employee attrition prediction. IBM Human Resource Analytic Employee Attrition and Performance dataset was used. The techniques were implemented with ANN, SVM, and DT classifiers to evaluate the employee attrition. The accuracy of the SVM model was higher than ANN and DT classifiers by 88.87% as indicated in the table below.

Table 1 Performance Comparison (Mansor, Sani and Aliff, 2021)

Classifier/ Results	Dataset	Accuracy (%)	Error Rate (%)	RMSE	ROC
DT – J48	Training	84.40	15.60	0.3704	0.850
	Test	80.95	19.05	0.4038	0.633
SVM	**Training**	**88.87**	**11.13**	**0.3335**	**0.853**
	Test	**87.76**	**12.25**	**0.3499**	**0.990**
ANN	Training	87.98	12.02	0.3274	0.925
	Test	85.03	14.97	0.3571	0.88

Source: Mansor, N., Sani, N.S. and Aliff, M., (2021), 'Machine learning for predicting employee attrition', *International Journal of Advanced Computer Science and Applications*, 12(11). (Available at: https://publications.eai.eu/index.php/IoT/article/view/4762).

This goal (Shafie et al, 2024) aimed mainly to use ANN combined with clustering techniques to develop a predictive methodology for employee turnover. Data sets were collected from Kaggle. Hyperparameter tuning of ANN models, data augmentation using conditional generative adversarial networks (CTGAN) on imbalanced clusters and evaluation of various ANN and traditional ML models are the methods implemented. The accuracy of the optimized ANN models was 100%.

This study (Tanasescu et al, 2024) aimed at developing ML algorithms capable of predicting employee performance scores, thereby ensuring increased objectivity in appraisals. Kaggle website dataset was used for this study. DT, RF, SVM, KNN, ANN, XGB were implemented and trained on the dataset. The below table showed that the DT reduced the human bias in employee evaluation with the achieved result of 94%.

Table 2 Machine learning performance (Tanasescu et al, 2024)

Algorithm	Hyperparameter Tuning	Precision	Recall	Accuracy	F1 Score
Decision trees	No tuning	75%	85%	85%	80%
	Grid search	91%	90%	90%	90%
	Optuna	94%	92%	92%	91%
Random forest	No tuning	83%	89%	89%	85%
	Grid search	83%	89%	89%	85%
	Optuna	83%	89%	89%	85%
SVM	No tuning	83%	89%	89%	85%
	Grid search	73%	86%	86%	79%
	Optuna	83%	90%	90%	87%
KNN	No tuning	72%	78%	78%	75%
	Grid search	83%	89%	89%	85%
	Optuna	73%	86%	86%	79%
ANN	No tuning	51%	71%	71%	60%
	Grid search	63%	79%	79%	70%
	Optuna	77%	88%	88%	82%
XGB	No tuning	88%	86%	86%	86%
	Grid search	83%	89%	89%	85%
	Optuna	94%	92%	92%	91%

Source: Tanasescu, L.G., Vines, A., Bologa, A.R. and Vîrgolici, O., (2024), 'Data analytics for optimizing and predicting employee performance', *Applied Sciences*, 14(8), p. 3254. (Available at: https://www.mdpi.com/2076-3417/14/8/3254).

The achievement of accuracy of 88.87% SVM in the employee attrition prediction (Mansor et al,2021) and 100% with optimized ANN and clustering methods (Shafie et al, 2024) demonstrated the possibility of overfitting. According to Tanasescu et al, (2024), decision DT reduces bias with accuracy of 94%, but may be inflexible. Sources and methods of studies for each dataset possess different degrees of reliability and applicability (Mansor et al, 2021; Shafie et al, 2023; Tanasescu et al, 2023).

The aim of this work (Alshiddy and Aljaber, 2023) was to study the effect of applying the ensemble learning methods on improving the application of the employee prediction attrition task. The IBM HR Analytics Employee Attrition and Performance dataset was being used to collect the datasets. Baseline comparison algorithms were limited to NB, SVM and RF and a two-layer nested ensemble algorithm was applied. As shown in the table the result achieved the accuracy of 94.5255% of the projected model to enhance the application of employee prediction attrition.

Table 3 Classifications of Models (Alshiddy and Aljaber, 2023)

Algorithm	Accuracy	Precision	Recall	F1-Measure	AUC
NB	90.957%	91.9%	91%	90.9%	95.5%
SVM	92.7818%	92.8%	92.8%	92.8%	92.8%
RF	94.2417%	94.3%	94.2%	94.2%	98.4%
The proposed model	94.5255%	94.5%	94.5%	94.5%	98.5%*

Source: Alshiddy, M.S. and Aljaber, B.N., (2023), 'Employee attrition prediction using nested ensemble learning techniques', *International Journal of Advanced Computer Science and Applications*, 14(7). (Available at: https://pdfs.semanticscholar.org/2675/041bec6ebc91735b43063349ec57cd485c92.pdf).

The aim of this study (Tümen and Sunar, 2023) was to cluster workers according to turnover levels considering effort and work-life balance factors. The IBM HR Analytics Employee Attrition and Performance Dataset available on Kaggle, was used in the project. The non-ensemble learning methods DT and SVM were implemented together with the ensemble learning techniques KNN, RF, GB, XGBOOST, SVM had the greatest score of 96% overall, as illustrated in the table below.

Table 4 Results of algorithms (Tümen and Sunar, 2023)

Models	Precision	Recall	f-Score	Accuracy
Decision Tree	0.87	0.87	0.86	0.87
SVM	0.97	0.96	0.96	0.96
kNN	0.87	0.86	0.86	0.87
RF	0.96	0.95	0.95	0.95
GBM	0.91	0.91	0.91	0.91
LightGBM	0.91	0.91	0.91	0.91
Average	**0.92**	**0.91**	**0.91**	**0.91**

Source: Tümen, V. and Sunar, A.S., 'Predicting the work-life balance of employees based on the ensemble learning method', *Bitlis Eren Üniversitesi Fen BilimleriDergisi*, 12(2), pp. 344-353. (Available at: https://dergipark.org.tr/en/pub/bitlisfen/issue/78380/1196174).

The aim of this article (Pereira, 2024) was to analyze what factors influenced employee turnover at Willis Towers Watson and to understand the best predictive model. The human resource department of the LRDH of Willis towers Watson provided the datasets. Different performances of DT, RF, KNN, LR methodologies were implemented to the models. The result carried out also demonstrated that RF proved valid to predict 78% accuracy.

For employee attrition, two studies are considered from the work of Alshiddy and Aljaber (2023), Tümen and Sunar (2023), and Pereira (2024), primarily focused on the IBM HR Analytics dataset and various algorithms. A two-layer ensemble model used by Alshiddy and Aljaber, yielded an accuracy of 94.53%, while Tümen and Sunar's Random Forest averaged 95%. Conversely, Pereira's Random Forest produced 78%. Reliance on particular datasets and overfitted models in ensemble methods (Alshiddy and Aljaber, 2023; Tümen and Sunar, 2023; Pereira, 2024) are limitations.

The core objective of this (Farman et al, 2024) work was to determine important factors for churn of telecom customers and build predictive models for better customer retention. The study used a dataset sourced from Kaggle to generate predicted models for the customer attrition. The methods implemented as RF, DT, and LR, the classification model based on, selected the performance metrics. The result shows that RF model had the accuracy of 98% outperforming other models.

The aim of this study (Heuten, 2021) was to determine whether and to what extent neural networks outperformed classic ML methods in forecasting employee attrition. IBM Watson Analytics provided the dataset which was used to address the study question posed at Kaggle. RF, SVM, and KNN described belong to classic ML techniques. RF was the greatest performing baseline technique meant for this ML algorithm with accuracy range of 93% as illustrated in the below table.

Table 5 Classification Techniques of RF, SVM, KNN (Heuten, 2021)

Algorithm	Accuracy	F1	Precision	Recall
KNN	0.927	0.926	0.945	**0.907**
SVM	0.911	0.906	0.963	0.854
RF	**0.933**	**0.929**	**0.991**	0.874
ANN	0.905	0.899	0.955	0.850

Source: Heuten, D., (2021), *Predicting employee attrition: A machine learning comparison*. Doctoral dissertation, Tilburg University. (Available at: http://arno.uvt.nl/show.cgi?fid=158268).

2.3 Research Gap

Many papers have been published to predict employee turnover by applying different machine learning algorithms. Mansor, Sani and Aliff (2021) conducted a classification between ANN, SVM & DT and revealed that SVM outperformed the rest with an accuracy rate of 88.87%. Shafie et al, (2024) obtained a 100% accuracy in applying optimized artificial neural network models and clustering methods. Tanasescu et al, (2024) applied DT to minimize the effect of human subjectivity in the performance of the employees and got an accuracy of 94%. According to Alshiddy and Aljaber (2023) using ensemble learning to enhance the model's accuracy, the prediction accuracy was 94.5255%. Tümen and Sunar (2023) deployed a number of different algorithms, including the one based on support vector machines, which was the most accurate with 96%. Pereira (2024) employed DT, RF, KNN and LR, with RF presenting a 78% of accuracy. Farman et al. (2024) applied RF, DT and LR for telecom customer churn prediction with RF demonstrating 98% accuracy. Heuten (2021) compared the performance of ANN, KNN, RF as well as SVM and RF was selected as the best baseline method with 93% accuracy.

Although these works show that machine learning can be effectively applied to the task of employee attrition prediction, most of the research is based on certain datasets and algorithms. Alshiddy and Aljaber (2023), Tümen and Sunar (2023), and Pereira (2024) works can be prone to overfitting because they employ ensemble methods and specific datasets. Further, the validity of these findings may be restricted by the underlying properties of the datasets applied.

Although previous works have investigated the application of machine learning algorithms to predict employee turnover, little attention has been paid to the effect of feature selection methods on model accuracy. This research seeks to address this gap by establishing an empirical comparison of the performance of different feature selection techniques including L1 regularization and Recursive Feature Elimination in enhancing the predictive and explanatory power of models for employee turnover.

3 Methodology

3.1 Project Architecture

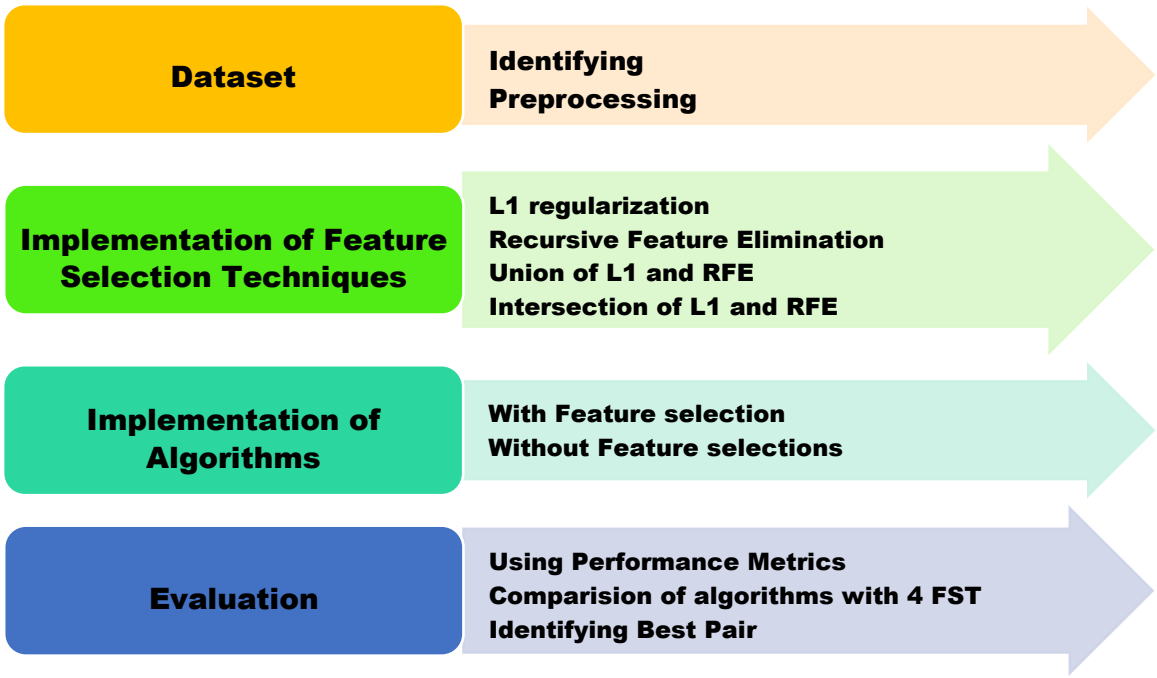


Figure 2 Project Workflow

This project presents the project architecture of machine learning and feature selection techniques on employee attrition prediction as follows. First the IBM HR Analytics dataset is obtained and improved with the handling of missing values, duplication, and feature that is no longer required. The next step is feature selection where three approaches are used, L1 regularization (Lasso), Recursive Feature Elimination (RFE) and union and intersection of features. When the relevant features are selected, five machine learning models, MLP, GB, XGB, HGB and DT are developed with and without the selected features to assess the usefulness of the chosen features to predict attrition. Features from the given baseline L1 combined with Recursive Feature Elimination (RFE) help to improve the models. The comparison metrics used are precision, recall, F1 rating, precision, FNR and FPR. Last of all, the results of each feature selection method and machine learning model combination are compared and the best approach for effective employee attrition prediction is determined as well as the key factors that affect employees' turnover.

3.2 Choice of methods

This project requires the selection of ML classifiers and feature selection methods that are ideal for predicting employee attrition. For the models to learn effectively from data selection of pertinent factors matters greatly. The project applies L1 Regularization (Lasso) as its initial method and this technique imposes a penalty on insignificant features to make their coefficients zero (Khan et al, 2024). By iteratively

fitting the model and discarding unimportant features the procedure of RFE enhances selection (Jafar and Lee, 2024). Investigation focuses on the combination of attributes chosen by L1 and RFE to bring together all elements found by both approaches (Pineda, 2021). The convergence of L1 and RFE uncovers overlapping traits chosen by both frameworks (Wu et al, 2020).

With each model added in Gradient Boosting, the preceding errors are addressed consequently improving accuracy (Zeleke et al, 2023). Data division in Decision Tree is driven by feature values allowing for clear understanding and strong capabilities when dealing with categorical or continuous information (Sultana and Islam, 2023). MLP is selected since it can learn complex non-linear relations in the dataset that might help to improve the prediction accuracy for employee attrition (Kuruneru and Kim, 2024). Research chooses XGBoost for its ability to work well with sparse datasets while providing strong performance through regularization techniques such as L1 and L2 among others (Nandal et al, 2024). The computational efficiency and scalability especially with large datasets makes HGB a choice. It uses histograms to bin continuous features of a problem which reduces the computation cost and helps the model to find the fine details of the data that is crucial for attrition analysis (Wang et al, 2023).

3.3 Dataset Description

This research data was obtained from Kaggle IBM HR Analytics Employee Attrition (Subhash, 2021) https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset. The data set consists of 1,470 records of employees' information, with 35 attributes including age, job satisfaction, performance rating, monthly income, years with the firm, and attrition (employee turnover). The data set contains both categorical and numerical variables including Employee details, experience, compensation etc., which can be used in the creation of predictive models. Interestingly, the last column, labeled as "Attrition", is the response variable that means the employee left or not.

3.4 Feature selection techniques

3.4.1 L1 Regularization (Lasso)

L1 regularisation works exceptionally in datasets containing numerous features by lessening overfitting and clarifying the model by stressing vital features (Khan et al, 2024).While predicting employee turnover some attributes might be unnecessary or repetitive, Lasso discards these unimportant qualities and maintains only the vital elements for model training.

3.4.2 Recursive Feature Elimination (RFE)

The objective of RFE is to identify elements that improve model performance and decrease the number of dimensions. Models that utilize a distinct method to rank

feature relevance benefit greatly from RFE methods such as decision trees or linear models (Priyatno and Widiyaningtyas, 2024).Workflow of RFE is shown below.

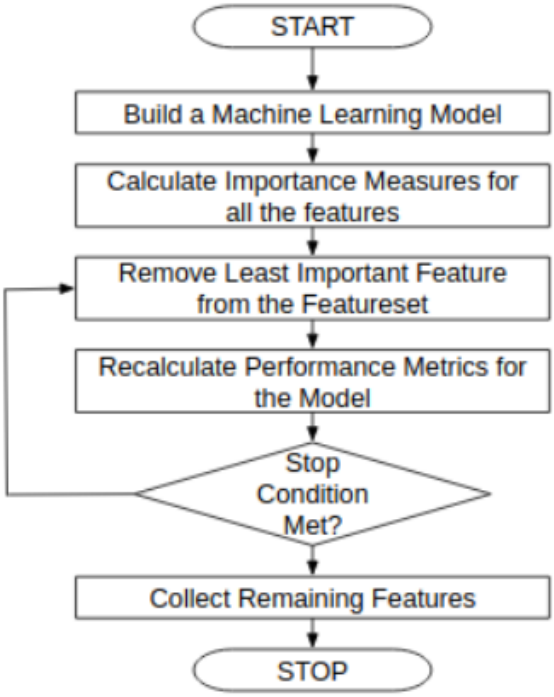


Figure 3 RFE (Ramakrishnaiah, Kuhlmann and Tyagi, 2021)

This technique supports L1 regularization and functions as a distinct way to select features that can be evaluated or merged with for discovering the most useful group.

3.4.3 Union of Selected Features

This method unites the features recognized as vital by various approaches to feature selection. All features picked by L1 and RFE get combined in the union to guarantee the detection of every potentially valuable feature (Pineda, 2021). A more inclusive set of features emerges for model training. When the feature set X1= {x1, x2, x3} is picked by L1 regularization and X2= {y1, y2, x2} by RFE then the sets together will feature all distinct attributes selected by each method. Mathematically, the union is represented as shown below.

$$\text{X1 Features} = \{x_1, x_2, x_3\}$$

$$\text{X2 Features} = \{y_1, y_2, x_2\}$$

$$\text{Union (X1} \cup \text{X2)} = \{x_1, x_2, x_3, y_1, y_2\}$$

The union includes all unique elements from both sets:

$$X_{\text{union}} = \{x_1, x_2, x_3, y_1, y_2\}$$

Figure 4 Mathematical Representation

3.4.4 Intersection of Selected Features

Identifying shared features among different selection methods happens through the intersection of selected features. When X1= {x1, x2, x3} is chosen by L1 regularization and X2= {y1, x2, x3} by RFE the intersection will consist of the shared features. Using the intersection highlights features that several selection techniques have consistently marked as significant. These shared features are probably the most significant for forecasting the result since they have been consistently chosen according to unique standards (Wu et al, 2020). As illustrated below is the representation of the intersection.

$$\text{X1 Features} = \{x_1, x_2, x_3\}$$

$$\text{X2 Features} = \{y_1, x_2, x_3\}$$

$$\text{Intersection (X1} \cap \text{X2)} = \{x_2, x_3\}$$

The intersection includes only the features common to both sets:

$$X_{\text{intersection}} = \{x_2, x_3\}$$

Figure 5 Mathematical Representation

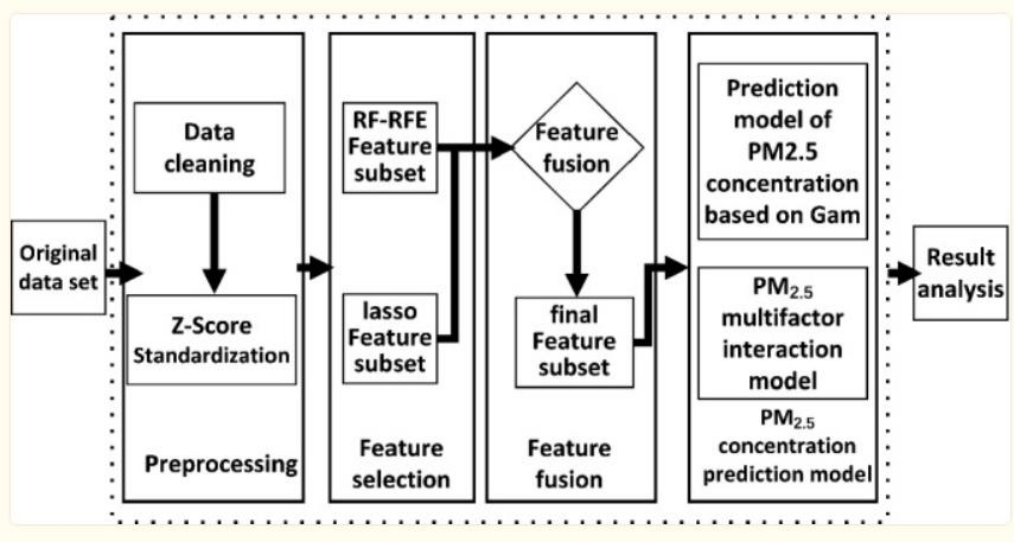


Figure 6 Intersection of Lasso and RFE (Wu et al, 2020)

3.5 ML algorithms

3.5.1 Gradient Boosting

The method of Gradient Boosting constructs powerful models through the sequential combination of weak learners like decision trees as shown below. New models in gradient boosting work to lower previous model errors by targeting the misclassified examples (Zeleke et al, 2023).

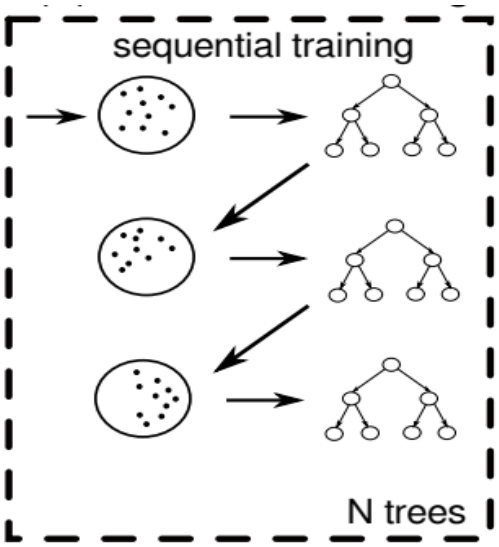


Figure 7 Gradient Boosting (Kowalek, Loch-Olszewska and Szwabiński, 2019)

The turnover of employees is frequently shaped by multiple factors that could have irregular interactions. In these circumstances Gradient Boosting performs well since it progressively develops models that refine forecasting and increase precision with each iteration.

3.5.2 Extreme Gradient Boosting

XGB is one of the most sophisticated ML models that consist of many weak decision trees, which are combined into a strong model. These trees are trained one at a time in sequential order until the least prediction error can be made and the most difficult data points are targeted and are shown below (Gulmeher and Aiman, 2023).

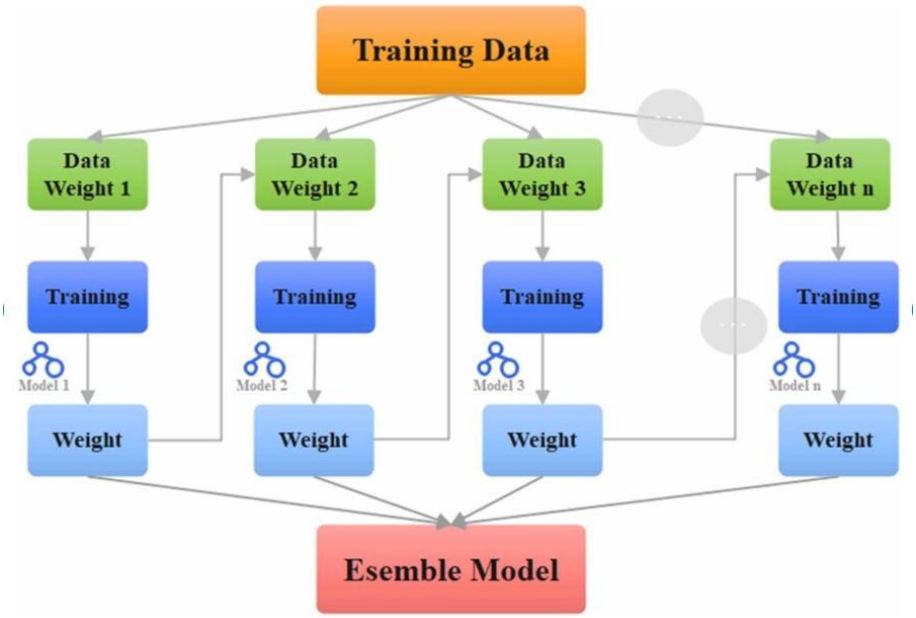


Figure 8 XGBoost (Xu et al, 2022)

Moreover, this research has built in feature importance metrics that closely matches the objectives of the research to discover important factors which impact employee attrition and as a result it suits best for feature selection and prediction tasks.

3.5.3 Histogram Based Gradient Boosting

In HGB, continuous features are grouped in bins (or histogram) as shown in figure, which reduces the computational overhead during training of the model. First numerical data is prepared, bins are defined and numerical data is sufficiently represented, and memory usage is less than in other algorithms. Second, this binning technique speeds up training by limiting the number of complex calculations during split evaluations (Al Adwan et al, 2023).

Figure 9 HGB (Zhao et al, 2023)

HGB is well suited in reducing computation time without sacrificing high predictive accuracy which makes it suitable for comparing performance across various feature selection techniques in predicting employee attrition.

3.5.4 Decision Tree Models

Based on a tree-like design the Decision Tree acts as a straightforward yet impactful machine learning algorithm making choices. Each node symbolizes a feature while each branch indicates a choice made from the feature value in the model. Creating a tree of decisions recursively separates the data into suitable groups until it precisely identifies employee retention or departure (Sultana and Islam, 2023). The structure of decision tree is outlined below.

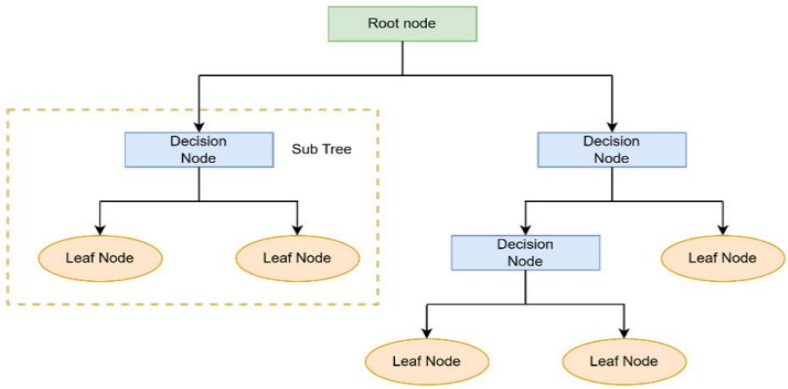


Figure 10 Decision Tree (Sultana and Islam, 2023)

3.5.5 MLP

Being one of the deep-learning models, Multilayer Perceptron (MLP) is build-up of interconnected nodes or artificial neurons that assemble in a formulated neural network. Network nodes are like biological neurons, they process input data using an activation function to create output that is the output of each layer will be input to the next, in-between layer is the hidden layers as shown in figure. The weighted connections between these neurons are adjusted by the network through the difference between expected and actual values (Nandal et al, 2024).

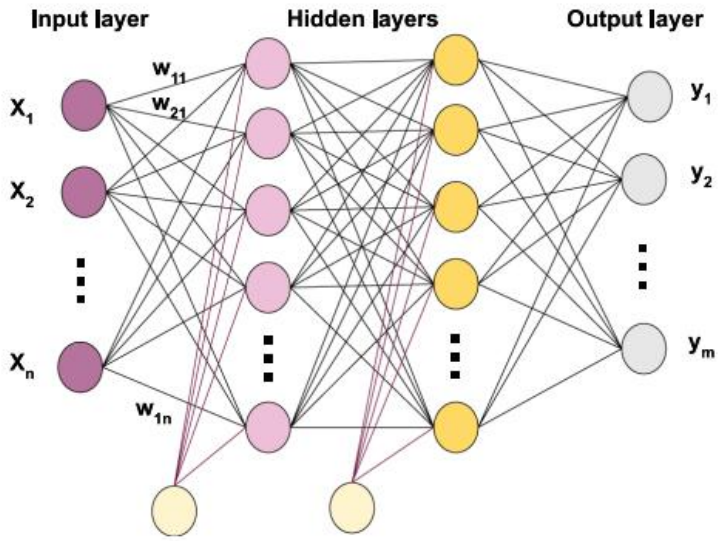


Figure 11 MLP (Chan et al, 2023)

Often employee attrition data contains many factors, for instance work experience, job satisfaction, performance, which potentially interacts in a complex and nonlinear way.

3.6 Evaluation Metrics

Through a Confusion Matrix representing performance metrics the model's effectiveness is highlighted by presenting the counts of true and false outcomes. The ratio of correct outcomes to overall predictions defines the accuracy in this matrix (Nandal et al, 2024). This model's precision reveals the ratio of actual attrition forecasts to the overall number of anticipated attritions. The effectiveness of the model in accurately detecting real attrition scenarios is evaluated by recall while pointing out its power to lower false negatives. This investigation presents a balanced analysis of model effectiveness by integrating accuracy with recall as part of the F1-score. FPR evaluates the ratio of misclassified attrition forecasts to genuine retention events and is defined in the below figure (Deviprasad et al, 2023).

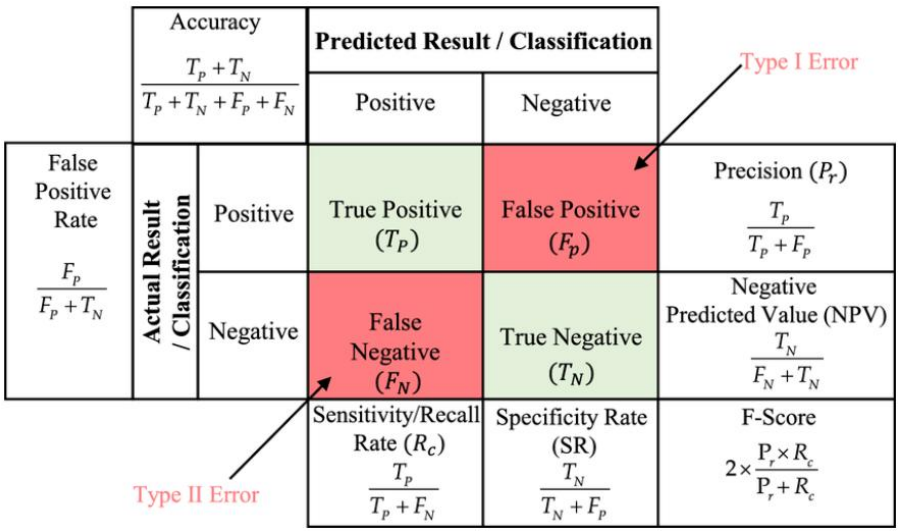


Figure 12 Evaluation Metrics (Kaur et al, 2023)

4 Experiment

4.1 Data collection

The dataset utilised for this employee attrition experiment is WA_Fn-UseC_-HR-Employee-Attrition.csv, which was obtained from Kaggle. This data set comprises 1,470 records and 35 features, rendering it appropriate for the examination of factors associated with employee attrition. Nine of the columns are categorical (object) data types, while 26 are numerical (int64). This dataset is notable for its completeness and reliability, as it contains no absent values, which guarantees the accuracy of the analysis.

Here is a screen shot of the Attrition column's value count:

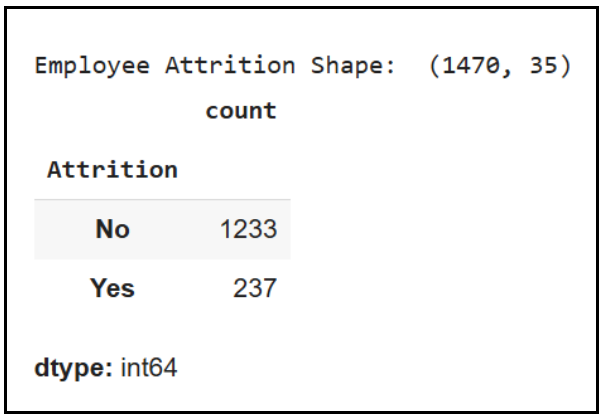


Figure 13 Distribution of samples in Attrition Column

In this investigation, the "Attrition" column serves as the objective variable, indicating whether an employee has departed the organisation. By employing the value_counts() function, it was determined that the attrition distribution was unbalanced: 1,233 employees are designated as "no" (indicating they remained), while 237 employees are designated as "yes" (indicating they departed). Before implementing the models, it is imperative to rectify the data imbalance.

4.2 Preprocessing the Data

Since EmployeeCount and StandardHours are invariant columns that do not contribute any predictive value, they are removed during the preprocessing of the dataset. The data is validated by checking for missing values and duplicate rows, but none were identified. The surviving dataset has 33 characteristics and 1,470 entries.

```
### Dropping Unwanted Features and Finding Missing Values, Duplicates
E_Attrition= E_Attrition.drop(['EmployeeCount','StandardHours'], axis=1) # deleting 2 columns
print("Missing Values in the E-Attrition: ",E_Attrition.isnull().values.sum())
print("Duplicates in the E-Attrition    : ",E_Attrition.duplicated().sum())
print("Employee Attrition Shape: ",E_Attrition.shape)
```

```
Missing Values in the E-Attrition:  0
Duplicates in the E-Attrition    :  0
Employee Attrition Shape:  (1470, 33)
```

Figure 14 Shape of the employee attrition data after cleaning

In order to predict employee attrition, this preprocessing phase optimizes the dataset by preserving pertinent features such as categorical attributes (attrition, business travel, etc.) and numerical data. This dataset has been thoroughly cleaned and organized, making it ideal for training models and conducting additional analyses.

4.3 Visualisation of Key Features

For better understanding of distributions, correlations, patterns, and stages in the dataset, charts are vital. This helps with analysis and decision-making.

In the first plot, a count plot, one can see how many employees have left (Attrition = "Yes") and how many have stayed (Attrition = "No"). This data helps one to understand the distribution of employee attrition. With 1,233 people still employed by the company ("no") and 237 people having departed ("yes"), the disparity is obvious. Modelling may suffer as a result of this imbalance, and so it has to be adjusted to account for the lower number of "yes" occurrences.

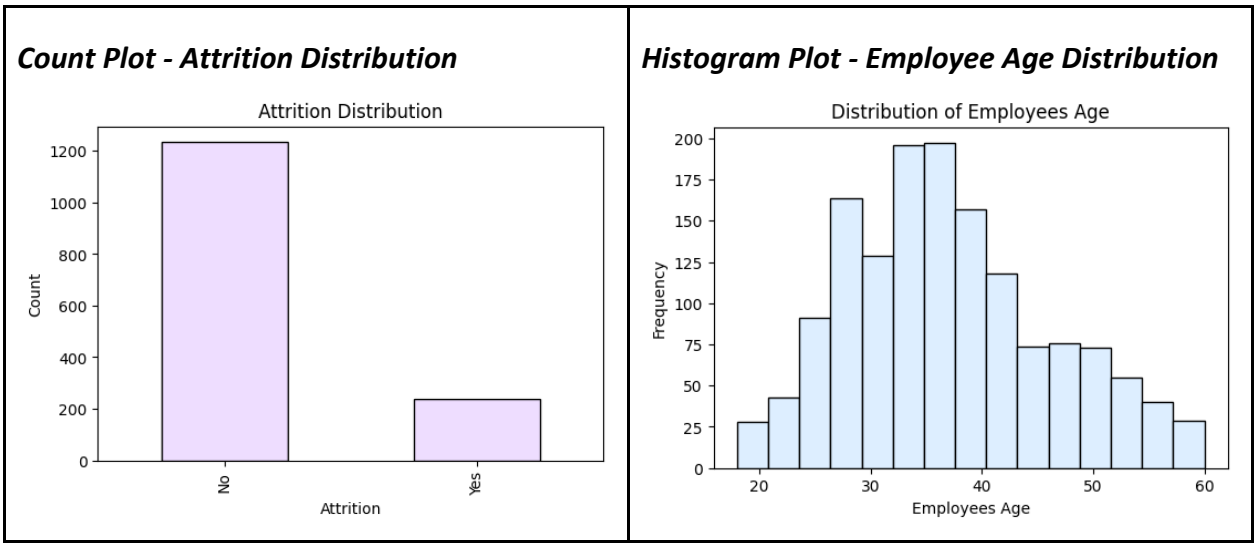


Figure 15 Plots showing Distribution of Employee Attrition and Age

The second plot shows the distribution of ages among employees, and it's a histogram. Across a wide age range, the histogram shows that most employees are concentrated in the middle to late stages of their careers, with a disproportionate number of workers in the 30–40 age bracket.

4.4 Analysis of Correlation Heatmap

The WA_Fn-UseC_-HR-Employee-Attrition dataset's numerical properties are shown in the correlation heatmap. Darker blues indicate positive correlations, and lighter blues indicate negative correlations, according to the YlGnBu colormap.

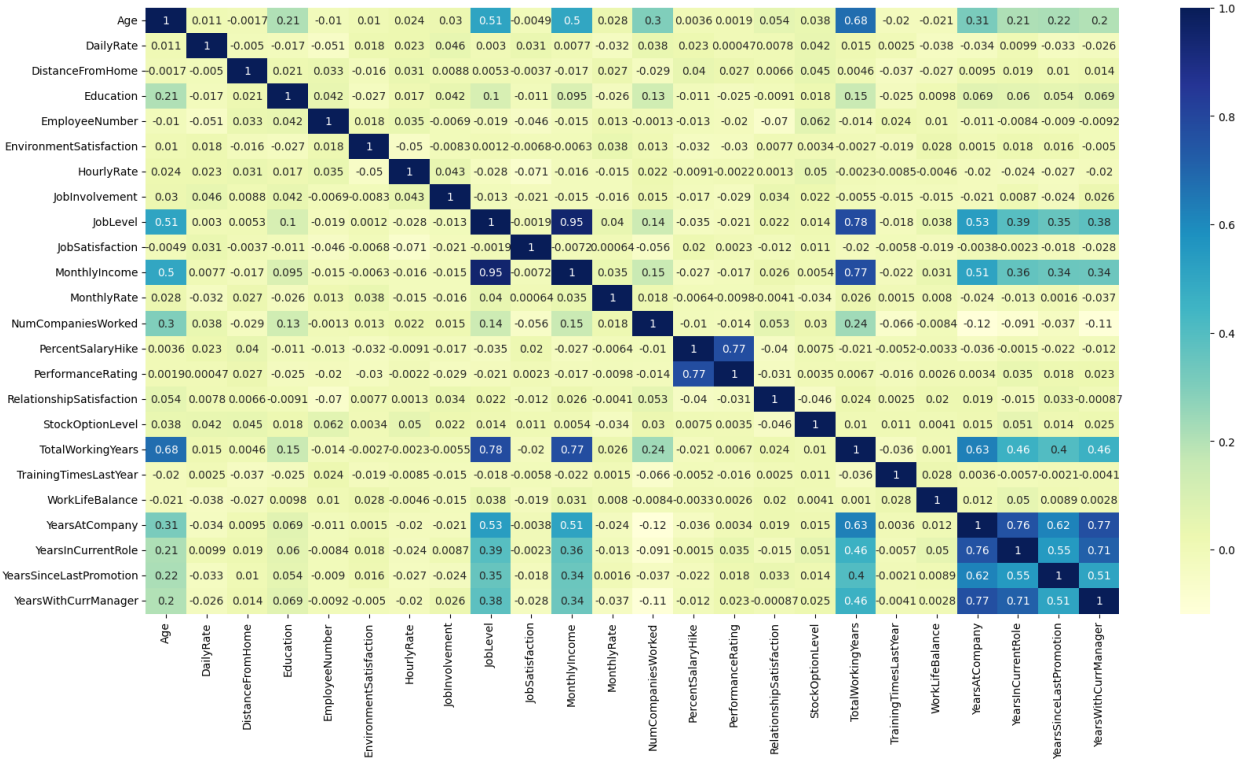


Figure 16 Correlation Heat Map

One important finding is that the variables JobLevel and MonthlyIncome have a very strong positive linear association; this is seen by the dark blue colour on the heatmap, which has a correlation of 0.95. This indicates that workers should expect a nearly proportional increase in their monthly income as their job levels rise. When one variable's value goes up, another goes down; this is known as an inverse relationship, and it's represented by a negative correlation value. There may be an inverse relationship between YearsAtCompany and NumCompaniesWorked, for instance.

4.5 Encoding Categorical Features

For the categorical data in the dataset, encoding is done by converting them into numerical form using LabelEncoder from sklearn.preprocessing. Every single unique value in a given column is given a numerical value in the form of a label, which translates the object data types to integers and the format is shown below.

```
from sklearn import preprocessing as E_AttPProcess
E_Ob_to_num = E_AttPProcess.LabelEncoder()
E_Attrition['Attrition']= E_Ob_to_num.fit_transform(E_Attrition['Attrition'])
E_Attrition['BusinessTravel']= E_Ob_to_num.fit_transform(E_Attrition['BusinessTravel'])
E_Attrition['Department']= E_Ob_to_num.fit_transform(E_Attrition['Department'])
E_Attrition['EducationField']= E_Ob_to_num.fit_transform(E_Attrition['EducationField'])
E_Attrition['Gender']= E_Ob_to_num.fit_transform(E_Attrition['Gender'])
E_Attrition['JobRole']= E_Ob_to_num.fit_transform(E_Attrition['JobRole'])
E_Attrition['MaritalStatus']= E_Ob_to_num.fit_transform(E_Attrition['MaritalStatus'])
E_Attrition['Over18']= E_Ob_to_num.fit_transform(E_Attrition['Over18'])
E_Attrition['OverTime']= E_Ob_to_num.fit_transform(E_Attrition['OverTime'])
```

Figure 17 Encoding the nine object type features

This transformation was done on attributes like Attrition, BusinessTravel, Department, EducationField, Gender, JobRole, MaritalStatus, Over18, and OverTime. All the object-type features have been converted to numerical values for the dataset to be suitable for model use after encoding.

4.6 Balancing the Data Using SMOTE

Out of 1,233 employees, 1,233 were marked as "Class 0" (no attrition) and 237 as "Class 1" (attrition), creating an imbalance in the original employee attrition data. A synthetic sample is generated for the minority class ("Class 1") using the SMOTE approach, which is used to solve this issue.

```
Attrition Categories  Counter({0: 1233, 1: 237})
Balanced Attrition Categories Counter({1: 1233, 0: 1233})

Employee Attrition Shape:  (2466, 33)
```

Attrition	count
1	1233
0	1233

dtype: int64

Figure 18 Balancing the Employee Attrition dataset

The attrition classes are balanced after SMOTE, thus the numbers for both the classes are 1,233 each. The total shape of the dataset is improved to 2,466 rows, making sure that both classes are more evenly represented. This makes machine learning models better at forecasting employee attrition. E_Attrition.csv is the final destination for the cleaned and encoded data, which will be used for model testing and analysis.

4.7 Data Splitting

The dataset, which comprises 33 columns, is partitioned into the target variable and features. Additionally, the Attrition column is designated as the target variable for prediction, while all 32 columns are utilised as features, with the exception of the Attrition column.

Now, the dataset with 2466 rows and 32 features, pertaining to the employee attrition is partitioned in a manner to facilitate model training, validation, and testing. Firstly, a train_test_split with a test size of 0.3 (30%) split the data into 70% for training purposes and 30% for further splitting the content.

```
print("\nTraining Samples for E_Attrition: ",E_AttritionXtrn.shape)
print("Validation Samples for E_Attrition: ",E_AttritionXvld.shape)
print("Testing Samples for E_Attrition: ",E_AttritiontXtst.shape)
```

```
Training Samples for E_Attrition:  (1726, 32)
Validation Samples for E_Attrition:  (370, 32)
Testing Samples for E_Attrition:  (370, 32)
```

Figure 19 Screenshot of the splitted data with all the initial 32 features

Next, the 30% split created above was further divided into two equal parts, being 15% each for validation and for testing sets, respectively, with a test size of 0.5. The samples in the three sets are: training containing 1,726 samples, 370 for validation, and 370 for testing. This is to ensure there is enough data to evaluate the model at each stage.

4.8 Experimental Overview

This initiative is intended to improve the accuracy of employee attrition prediction by minimising overfitting and utilising machine learning models in conjunction with feature selection techniques. The dataset that contains 32 features is considered the baseline.

The table below is a list of all the 32 features.

Table 6 The original 32 features after preprocessing the employee attrition dataset

Age	**EnvironmentSatisfaction**	**MonthlyIncome**	**StockOptionLevel**
BusinessTravel	**Gender**	**MonthlyRate**	**TotalWorkingYears**
DailyRate	**HourlyRate**	**NumCompaniesWorked**	**TrainingTimesLastYear**
Department	**JobInvolvement**	**Over18**	**WorkLifeBalance**
DistanceFromHome	**JobLevel**	**OverTime**	**YearsAtCompany**
Education	**JobRole**	**PercentSalaryHike**	**YearsInCurrentRole**
EducationField	**JobSatisfaction**	**PerformanceRating**	**YearsSinceLastPromotion**
EmployeeNumber	**MaritalStatus**	**RelationshipSatisfaction**	**YearsWithCurrManager**

Methods of feature selection, such as RFE and L1, as well as their combinations (union and intersection), are implemented to mitigate dimensionality. The models are trained using GridSearchCV and 2-fold cross-validation, which involves hyperparameter optimization. Accuracy, recall, precision, F1 score, FPR, and FNR are metrics that are assessed, and the results are represented by confusion matrices. Learning curves are used to evaluate variance and bias, which is helpful in the identification of overfitting or underfitting. This strategy identifies the most effective model and feature selection method for predicting employee turnover.

4.9 Feature Selection Techniques Applied

The following are the phases of the investigation that involve the selected features:

1. **L1 Feature Selection**
2. **Recursive Feature Elimination (RFE)**
3. **Union of Features from L1 and RFE**
4. **Intersection of Features from L1 and RFE**

Please refer to the table below for the selected features and their respective counts.

Table 7 Selected using feature selection techniques

Number of Selected features	**Selected Features**
Selected 20 features using L1 Feature Selection	`Index(['Age', 'BusinessTravel', 'DailyRate', 'Department', 'DistanceFromHome', 'Education', 'EducationField', 'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction', 'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked', 'Over18'], dtype='object')`

Selected 20 features using RFE Feature Selection	`Index(['BusinessTravel', 'Department', 'Education', 'EnvironmentSatisfaction', 'Gender', 'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction', 'MaritalStatus', 'OverTime', 'PerformanceRating', 'RelationshipSatisfaction', 'StockOptionLevel', 'TrainingTimesLastYear', 'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion', 'YearsWithCurrManager'], dtype='object')`
Number of Features in Union of L1 and RFE = 29	['EmployeeNumber', 'RelationshipSatisfaction', 'WorkLifeBalance', 'MaritalStatus', 'TrainingTimesLastYear', 'YearsInCurrentRole', 'Gender', 'NumCompaniesWorked', 'Department', 'YearsSinceLastPromotion', 'HourlyRate', 'DistanceFromHome', 'DailyRate', 'OverTime', 'Education', 'YearsWithCurrManager', 'EducationField', 'PerformanceRating', 'StockOptionLevel', 'JobSatisfaction', 'MonthlyIncome', 'Age', 'JobLevel', 'MonthlyRate', 'JobRole', 'Over18', 'JobInvolvement', 'BusinessTravel', 'EnvironmentSatisfaction']
Number of Features in Intersection of L1 and RFE = 11	'BusinessTravel', 'Department', 'JobLevel', 'EnvironmentSatisfaction', 'EducationField', 'Gender', 'JobSatisfaction', 'JobInvolvement', 'JobRole', 'MaritalStatus', 'Education']

5 Results

The results of this experiment illustrate the influence of various feature selection techniques on the predictive capabilities of machine learning models in the context of employee attrition.

The following table indicates the performance metrics of predictive models that were executed with and without the selected features.

Table 8 Metrics of Predictive Models using the dataset with and without feature selection techniques

Techniques Applied	ML	Val. Acc.	Test. Pre.	Test. Rec.	Test. F1	Test. Acc.	Test. FPR	Test. FNR
Without Feature Selection	**GB**	0.89	0.87	0.86	0.87	0.86	0.11	0.15
	HGB	0.91	0.91	0.91	0.91	0.91	0.07	0.11
	XGB	0.92	0.91	0.91	0.91	0.91	0.06	0.12
	DT	0.79	0.81	0.80	0.80	0.80	0.19	0.20
	MLP	0.56	0.57	0.56	0.56	0.56	0.43	0.45
L1	**GB**	0.86	0.88	0.88	0.88	0.88	0.10	0.13
	HGB	0.88	0.88	0.87	0.87	0.87	0.12	0.14
	XGB	0.85	0.86	0.86	0.86	0.86	0.11	0.17
	DT	0.77	0.76	0.76	0.76	0.76	0.29	0.20
	MLP	0.54	0.62	0.61	0.52	0.61	0.87	0.05
RFE	**GB**	0.89	0.88	0.87	0.87	0.87	0.12	0.14
	HGB	0.89	0.87	0.86	0.86	0.86	0.01	0.18
	XGB	0.90	0.87	0.86	0.87	0.86	0.10	0.16
	DT	0.81	0.83	0.83	0.83	0.83	0.14	0.2

	MLP	0.85	0.83	0.82	0.83	0.82	0.14	0.20
Union of L1 and RFE	**GB**	0.89	0.89	0.88	0.88	0.88	0.12	0.12
	HGB	0.93	0.89	0.89	0.89	0.89	0.10	0.12
	XGB	0.91	0.89	0.88	0.88	0.88	0.11	0.12
	DT	0.79	0.80	0.79	0.79	0.79	0.23	0.19
	MLP	0.55	0.57	0.55	0.55	0.55	0.38	0.50
Intersection of L1 and RFE	**GB**	0.83	0.82	0.81	0.81	0.81	0.13	0.23
	HGB	0.83	0.83	0.81	0.81	0.81	0.10	0.25
	XGB	0.84	0.82	0.81	0.81	0.81	0.12	0.24
	DT	0.77	0.77	0.76	0.76	0.76	0.22	0.25
	MLP	0.76	0.76	0.75	0.75	0.75	0.22	0.28

5.1 Observations from the Results

5.1.1 Without Feature Selection

Validation and testing accuracies were highest for XGB and HGB, respectively, at 0.92 and 0.91, demonstrating their proficiency with high-dimensional datasets. Their 0.91 F1 scores show that they are equally proficient at spotting cases of attrition and non-attrition. The low false positive rate (FPR: 0.06-0.07, FNR: 0.11-0.12) and false negative rate (FNR: 0.12-0.12) for XGB and HGB demonstrate their excellent resistance to misclassification. In the absence of feature selection, MLP failed miserably, achieving the worst FPR (0.43) and FNR (0.45).

5.1.2 L1 Regularization

XGB and HGB exhibit robust performance, with high testing accuracies (0.86 and 0.87) and low false positive rates (FPR of 0.11 and 0.12), thereby establishing their reliability for the identification of employees who are at risk of departing. GB also exhibits a low FPR (0.10) and excellent accuracy (0.88), whereas MLP performs poorly with the lowest accuracy (0.61) and a high FPR (0.87), indicating that it is less suitable for this task. DT's

accuracy is moderate (0.76), but its FPR and FNR are high (0.29) and 0.20, respectively, suggesting that there is room for advancement.

5.1.3 Recursive Feature Elimination (RFE)

Compared to L1, MLP's test accuracy (0.82) and F1 score (0.83) were enhanced by RFE-selected features, which also resulted in a reduction in FPR (0.14) and FNR (0.20), indicating improved generalisation. Boosting models (HGB, XGB) maintained robust performance (test accuracy of 0.86, F1 scores: 0.86-0.87). It is worth noting that HGB exhibited the lowest FPR of 0.01, while XGB exhibited an FPR of 0.10. DT demonstrated enhanced performance (test accuracy: 0.83, F1 score: 0.83) at a lower FPR (0.14) in comparison to L1.

5.1.4 Union of L1 and RFE

With a 0.89 test accuracy, low FPR (0.10), and moderate FNR (0.12), HGB exhibits the highest performance when the union of L1 and RFE-selected features is employed, effectively balancing false positives and negatives. XGB is closely followed by a 0.88 test accuracy, while the FPR and FNR are similar at 0.11 and 0.12. GB exhibits a 0.88 accuracy and a slightly higher FPR (0.12), providing satisfactory performance. While MLP exhibits poor performance with low accuracy (0.55) and high FPR (0.38), DT has a lesser testing accuracy (0.70) and a higher FPR (0.23), suggesting that accurate predictions are a challenge.

5.1.5 Intersection of L1 and RFE

The over-reduction of features in this feature set led to substantial decreases in model performance. The F1 scores and testing accuracies of the boosting models (GB, HGB, XGB) were reduced by 0.81, while the FPR and FNR were increased by 0.10-0.13 and 0.23-0.25, respectively. The performance of DT and MLP was subpar, with MLP obtaining a test accuracy of only 0.75 and a very high FNR of 0.28.

5.2 Learning Curve of HGB and XGB models using data with original 32 features

The learning curve illustrates the model's performance on both training and validation datasets at varied training set sizes. The training score is represented by the blue line, while the validation score is represented by the green line. The standard deviation is denoted by the shaded regions surrounding the lines, which reflect the variability of the scores.

The graphs below show the learning curve of the HGB and XGB models using the dataset with all the original 32 features.

HGB without feature selection	XGB without feature selection

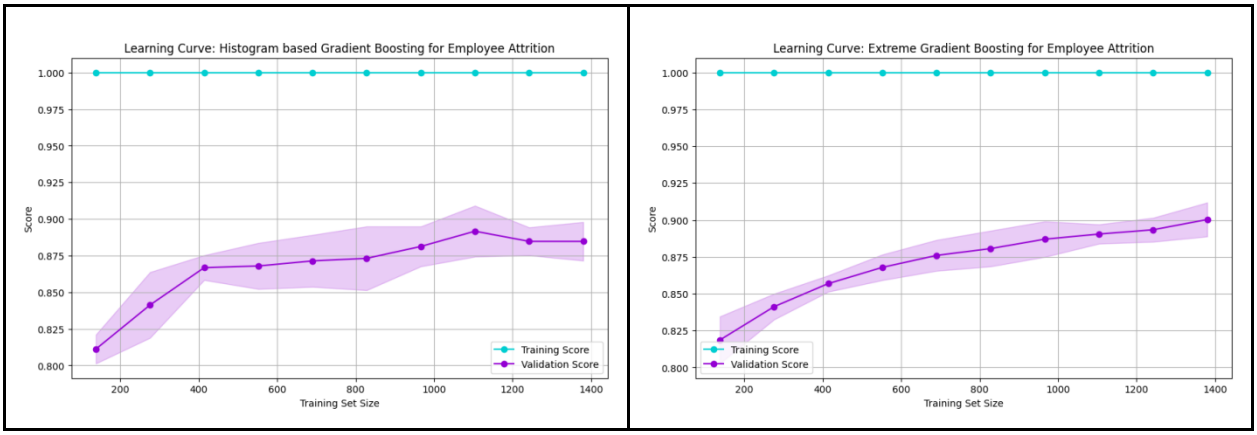


Figure 20 Learning Curves

Overfitting is reduced when the disparity between training and validation scores is narrowing, whereas underfitting or overfitting is indicated by a large gap. Ideally, the curves should converge near the top, which would indicate that they are well-generalised.

5.2.1 Confusion Matrix of HGB and XGB models using data with original 32 features

With the first set of 32 features, both the HGB and XGB models were 91% accurate; however, HGB somewhat outperformed XGB in terms of TP and TN. While XGB recorded 189 TP and 146 TN, HGB recorded 191 TP. The low rates of false positives (FP) and false negatives (FN) demonstrated by both models suggest that they are highly predictive. With a slightly higher total probability (TP) count, HGB appears to have been the superior model when it came to accurately predicting attrition.

The table below shows the confusion matrix of the HGB and XGB models using the dataset with all the original 32 features.

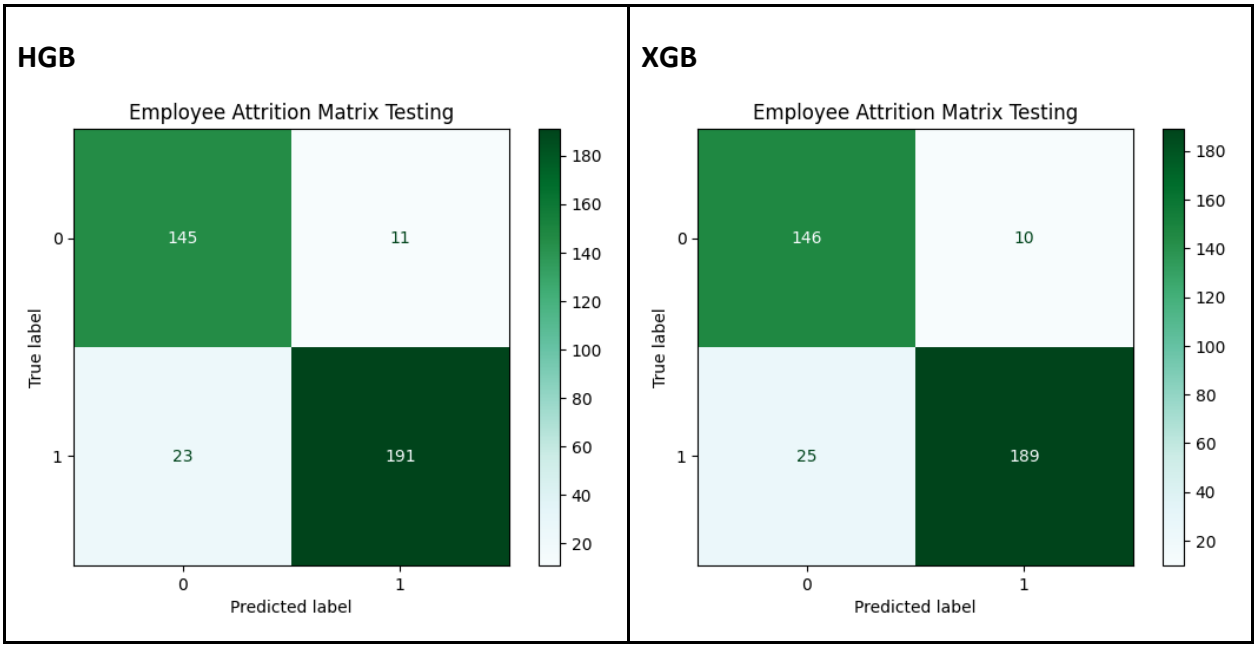


Figure 21 Confusion matrix during testing data without feature selection

The results indicate that the boosting models (HGB and XGB) are well-suited for real-world applications such as HR analytics, as they are able to accurately predict employee turnover without feature selection. Businesses may use these models to better plan for the future of their staff and implement retention efforts by pinpointing which employees are most likely to leave.

5.3 Addressing Research Questions

1. **How does a using different feature selection technique (L1 regularization, Recursive Feature Elimination, union of L1 and RFE, intersection of L1 and RFE) affect the Gradient Boosting and Decision Tree model accuracy in predicting the attrition of employees?**

   Using different feature selection techniques affected the model accuracy in predicting employee attrition. The boosting models (GB, HGB, and XGB) consistently outperformed the Decision Tree (DT), with XGB and HGB achieving the highest testing accuracy of 0.91 without any feature selection. Feature selection techniques did not improve their accuracy, with performance ranging from 0.81 to 0.89 after applying L1, RFE, or their union. GB showed slight improvement from 0.86 to 0.88 with L1 and the union of L1 and RFE. DT exhibited moderate improvement with RFE but showed reduced accuracy with other feature selection methods.

2. **Which attributes are found to be most significant for calculating employee attrition as identified by the feature selection techniques?**

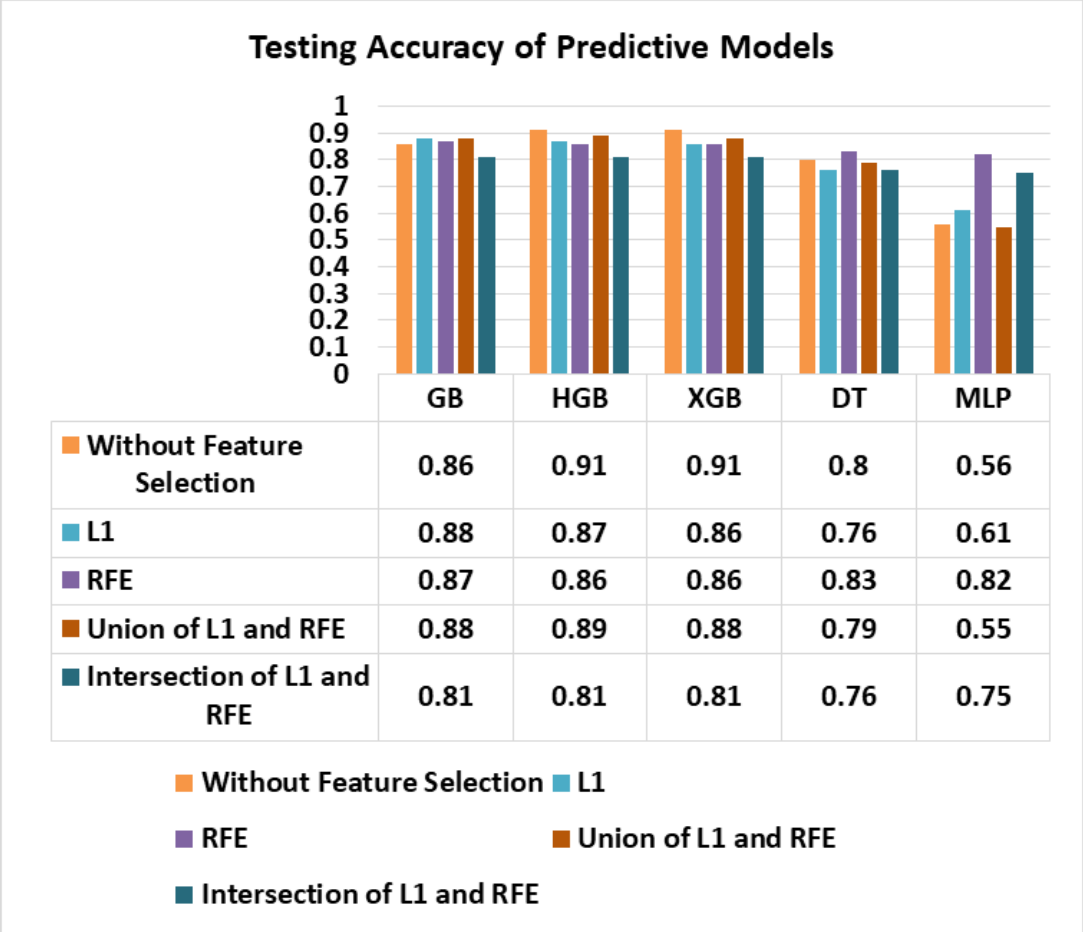
   While the highest accuracy of 91% was achieved with all 32 features for both HGB and XGB, feature selection did lead to some improvements. GB showed 88% accuracy with L1 regularization (20 features) and the union of L1 and RFE (29 features). For boosting models, in case of feature selection, the union of L1 and RFE was the most effective, outperforming other feature selection methods. RFE also showed good performance across all models (accuracy ranging 0.82 to 0.87), but none of the feature-selected models outperformed the original 91%. Notably, the intersection of L1 and RFE (with just 11 features) performed poorly, suggesting that a smaller feature set may not be optimal. Overall, boosting models benefitted from feature selection, with the union of L1 and RFE being the most effective for improving performance, while RFE was the best option for DT and MLP models.

6 Analysis and Discussion

6.1 Interpretation of Results

Predicting employee attrition is best handled by XGB and HGB, according to the results. These models get the highest accuracy of 91% with all features and continue to perform well even when features are selected. Feature selection generally did not improve performance for these models, with the best results from the union of L1 and RFE, which kept 29 features. However, DT and MLP were more affected by feature selection, and RFE enhanced their accuracy. Reducing the feature set through the intersection of L1 and RFE had a devastating effect on model performance, demonstrating how removing too many features can have a severe effect on prediction accuracy.

The graph below shows the testing accuracy of all the predictive models.



	GB	HGB	XGB	DT	MLP
Without Feature Selection	0.86	0.91	0.91	0.8	0.56
L1	0.88	0.87	0.86	0.76	0.61
RFE	0.87	0.86	0.86	0.83	0.82
Union of L1 and RFE	0.88	0.89	0.88	0.79	0.55
Intersection of L1 and RFE	0.81	0.81	0.81	0.76	0.75

Figure 22 Testing Accuracy of predictive Models

6.2 Best Model: HGB (Histogram-Based Gradient Boosting)

HGB retains a 91% accuracy rate even without feature selection, making it the best option for dealing with high-dimensional data. Multiple weak decision trees are joined to improve prediction accuracy through their ensemble learning approach, which is responsible for their robust performance. Additionally, HGB's quick computation, reduced overfitting, and ability to handle both linear and non-linear interactions are its strong points. While XGB and HGB had comparable performance, HGB was more accurate in predicting employee attrition because it marginally outperformed in detecting true positives. It is particularly adept at accommodating a variety of feature sets, particularly when the union of L1 and RFE is implemented.

The graph below shows the testing accuracy of HGB and XGB Models.

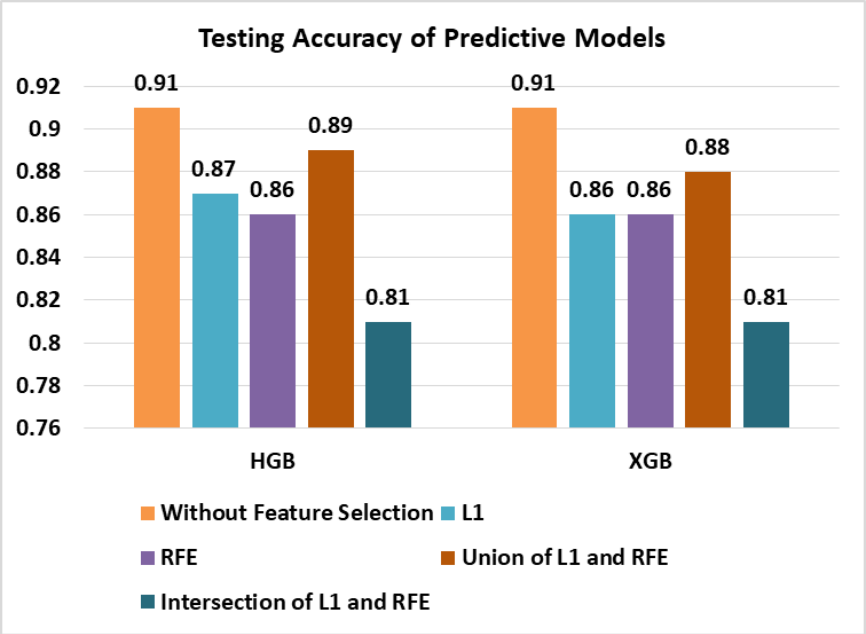


Figure 23 Testing Accuracy of HGB vs. XGB

7 Conclusion

7.1 Summary of Key Results

This research aimed at creating machine learning models for the purpose of predicting employee turnover so that the best algorithms and features could be determined. It is observed that the models that performed well are XGB and HGB, with an accuracy of 91% when using the original 32 features. XGB and HGB both had good performance without feature selection and only slightly improved with L1 and RFE feature selection. The highest performance of XGB and HGB in terms of feature selection was achieved when using the union of L1 and RFE, which selected 29 features. On the other hand, the DT and MLP were more dependent on the feature selection, and the RFE increased the performance. The intersection of L1 and RFE, which resulted in 11 features, led to a decrease in the predictive accuracy for all models, thus showing that over feature reduction is counterproductive.

7.2 Justified Conclusions

The results from this research show that the XGB and HGB are the best models for predicting employee turnover, as they provide a high level of efficacy and stability in performance regardless of the features used. The fact that they do not require much feature selection to work with complex data makes them useful in practical organisational situations. DT and MLP was the least stable, and the performance was very sensitive to the number of features to be included. This shows that when training the model, correct feature selection is crucial since wrong selection of features is likely to reduce the model's accuracy. The results also provide evidence that using feature selection techniques can be useful for some models, but over-reduction of the features could be detrimental to the prediction accuracy.

References

Adwan, J., Alzubi, Y., Alkhdour, A. and Alqawasmeh, H., (2023), 'Predicting compressive strength of concrete using histogram-based gradient boosting approach for rapid design of mixtures', *Civil Engineering Infrastructures Journal*, 56(1), pp. 159-172. (Available at: https://ceij.ut.ac.ir/article_88087_4f9d413c376d34905f465b11b1426c73.pdf).

Chan, K.Y., Abu-Salih, B., Qaddoura, R., Ala'M, A.Z., Palade, V., Pham, D.S., Del Ser, J. and Muhammad, K., (2023), 'Deep neural networks in the cloud: review, applications, challenges and research directions', *Neurocomputing*, 545, p. 126327. (Available at: https://doi.org/10.1016/j.neucom.2023.126327).

Deviprasad, S., Madhumithaa, N., Vikas, I.W., Yadav, A. and Manoharan, G., (2023), 'The machine learning-based task automation framework for human resource management in MNC companies', *Engineering Proceedings*, 59(1), p. 63. (Available at: https://doi.org/10.3390/engproc2023059063).

Farman, H., Khan, A.W., Ahmed, S., Khan, D., Imran, M. and Bajaj, P., (2024), 'An analysis of supervised machine learning techniques for churn forecasting and component identification in the telecom sector', *Journal of Computing & Biomedical Informatics*, 7(01), pp. 264-280. (Available at: https://jcbi.org/index.php/Main/article/view/478/408).

Gulmeher, R. and Aiman, U., (2023), 'A novel approach to unveiling employee attrition patterns using machine learning algorithms', *Journal of Scientific Research and Technology*, pp. 234-241. (Available at: https://jsrtjournal.com/index.php/JSRT/article/view/58).

Jafar, A. and Lee, M., (2024), 'High-accuracy COVID-19 prediction using optimized union ensemble feature selection approach', *IEEE Access*. (Available at: https://ieeexplore.ieee.org/iel8/6287639/10380310/10586989.pdf).

Kato, R., Zeng, W., Siramshetty, V.B., Williams, J., Kabir, M., Hagen, N., Padilha, E.C., Wang, A.Q., Mathé, E.A., Xu, X. and Shah, P., (2023), 'Development and validation of PAMPA-BBB QSAR model to predict brain penetration potential of novel drug candidates', *Frontiers in Pharmacology*, 14, p. 1291246. (Available at: https://www.frontiersin.org/journals/pharmacology/articles/10.3389/fphar.2023.1291246/full).

Kaur, M., Singh, D., Jabarulla, M.Y., Kumar, V., Kang, J. and Lee, H.N., (2023), 'Computational deep air quality prediction techniques: a systematic review', *Artificial Intelligence Review*, 56(Suppl 2), pp. 2053-2098. (Available at: https://www.semanticscholar.org/paper/Computational-deep-air-quality-prediction-a-review-Kaur-Singh/2759de37364b8ce450b87ef6f2a2e9da7bb01f7c).

Khan, M.A., Azim, A., Liscano, R., Smith, K., Chang, Y.K., Seferi, G. and Tauseef, Q., (2024), 'On the effectiveness of feature selection techniques in the context of ML-based regression test prioritization', *IEEE Access*. (Available at: https://ieeexplore.ieee.org/iel8/6287639/10380310/10679125.pdf).

Kowalek, P., Loch-Olszewska, H. and Szwabiński, J., (2019), 'Classification of diffusion modes in single-particle tracking data: feature-based versus deep-learning approach', *Physical Review E*, 100(3), p. 032410. (Available at: https://www.nature.com/articles/s41598-022-13446-0).

Kuruneru, S.T.W. and Kim, J.S., (2024), 'Erosive wear and particle attrition in multi-stage solar particle receivers and screw conveyors: a CFD-DEM approach with machine learning and artificial neural networks', *Chemical Engineering Science*, 300, p. 120585. (Available at: https://doi.org/10.1016/j.ces.2024.120585).

Nandal, M., Grover, V., Sahu, D. and Dogra, M., (2024), 'Employee attrition: analysis of data driven models', *EAI Endorsed Transactions on Internet of Things*, 10. (Available at: https://publications.eai.eu/index.php/IoT/article/view/4762).

Nurhindarto, A., Andriansyah, E.W., Alzami, F., Purwanto, P., Soeleman, M.A. and Prabowo, D.P., (2021), 'Employee attrition and performance prediction using univariate ROC feature selection and random forest', *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*. (Available at: https://kinetik.umm.ac.id/index.php/kinetik/article/view/1345).

Pereira, L.A.S., (2024), *Using machine learning to predict employee turnover: A case study of the Willis Towers Watson Lisbon Hub*. Doctoral dissertation. (Available at: https://repositorio.ucp.pt/bitstream/10400.14/44817/1/203590686.pdf).

Pineda, F., (2021), 'Selection of characteristics by hybrid method: RFE, ridge, lasso, and Bayesian for the power forecast for a photovoltaic system', *Soft Computing and its Engineering Applications: Second International Conference*, pp. 75. (Available at: https://cris.pucp.edu.pe/en/publications/selection-of-characteristics-by-hybrid-method-rfe-ridge-lasso-and-2).

Priyatno, A.M. and Widiyaningtyas, T., (2024), 'A systematic literature review: Recursive feature elimination algorithms', *JITK (JurnalIlmuPengetahuan dan TeknologiKomputer)*, 9(2), pp. 196-207. (Available at: https://ejournal.nusamandiri.ac.id/index.php/jitk/article/view/5015).

Ramakrishnaiah, Y., Kuhlmann, L. and Tyagi, S., (2021), 'linc2function: A deep learning model to identify and assign function to long noncoding RNA (lncRNA)', *bioRxiv*, pp. 2021-01. (Available at: https://www.biorxiv.org/content/10.1101/2021.01.29.428785.full).

Shafie, M.R., Khosravi, H., Farhadpour, S., Das, S. and Ahmed, I., (2024), 'A cluster-based human resources analytics for predicting employee turnover using optimized

artificial neural networks and data augmentation', *Decision Analytics Journal*, 11, p. 100461. (Available at: https://colab.ws/articles/10.1016%2Fj.dajour.2024.100461).

Subhash, P., (2021), *IBM HR Analytics Employee Attrition dataset*. (Available at: https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset).

Sultana, A. and Islam, R., (2023), 'Machine learning framework with feature selection approaches for thyroid disease classification and associated risk factors identification', *Journal of Electrical Systems and Information Technology*, 10(1), p. 32. (Available at: https://jesit.springeropen.com/articles/10.1186/s43067-023-00101-5).

Taşkıran, N., (2023), *A recommendation approach for employee retention by using a new feature selection strategy*. Master's thesis, Middle East Technical University. (Available at: https://open.metu.edu.tr/bitstream/handle/11511/105412/Nagihan%20Ta%C5%9Fk%C4%B1ran%20Thesis.pdf).

Van Dam, R., (2021), *Predicting employee attrition*. Doctoral dissertation, Tilburg University. (Available at: http://arno.uvt.nl/show.cgi?fid=158268).

Wu, T., Zhao, Z., Wei, H. and Peng, Y., (2020), 'Research on PM 2.5 integrated prediction model based on Lasso-RF-GAM', *Data Mining and Big Data: 5th International Conference, DMBD 2020*, pp. 83-94. (Available at: https://pmc.ncbi.nlm.nih.gov/articles/PMC7351685/).

Xu, Y., Shi, Q., Zhou, Z., Xu, K., Lin, Y., Li, Y., Zhang, Z. and Wu, J., (2022), 'Machine learning assisted insights into the mechanical strength of nanocrystalline graphene oxide', *2D Materials*, 9(3), p. 035002. (Available at: https://ui.adsabs.harvard.edu/abs/2022TDM.....9c5002X/abstract).

Zeleke, A.J., Palumbo, P., Tubertini, P., Miglio, R. and Chiari, L., (2023), 'Machine learning-based prediction of hospital prolonged length of stay admission at emergency department: a gradient boosting algorithm analysis', *Frontiers in Artificial Intelligence*, 6, p. 1179226. (Available at: https://doi.org/10.3389/frai.2023.1179226).

Zhao, Y., Wang, X., Li, J., Li, W., Sun, Z., Jiang, M., Zhang, W., Wang, Z., Chen, M. and Li, W.J., (2023), 'Using IoT smart basketball and wristband motion data to quantitatively evaluate action indicators for basketball shooting', *Advanced Intelligent Systems*, 5(12), p. 2300239. (Available at: https://scholars.cityu.edu.hk/en/publications/using-iot-smart-basketball-and-wristband-motion-data-to-quantitatively-evaluate-action-indicators-for-basketball-shooting(fed17e91-0d21-4213-8655-929641fad9c4).html).

Appendix

Technical Feasibility

Based on available tools and data sets, technical feasibility of the proposed research on optimizing employee attrition prediction through feature selection and machine learning is demonstrated. The IBM HR Analytics Attrition Dataset is definitely a robust dataset to use to gain enough experience than other datasets. The data cleaning and preprocessing are performed using libraries like Pandas. Scikit learn makes it relatively easy to implement feature selection techniques like L1 regularization as well as RFE, and to train gradient boosting and decision tree models. Specifically, can use built in functions in Scikit-Learn to easily compute precision, correctness, recall and F1 score, making it easy to compare different representations and feature subsets. Matplotlib and Seaborn can be used to generate visualizations to understand results, making both accuracy and interpretability easy to address. All these components enable a systematic approach to determining how employee attrition is affected by major factors using a computational technique.

Significance of the study

It is essential to address employee attrition because this can create wide ranging consequences, both in financial and operational terms, on the organization. This work evaluates feature selection techniques alongside machine learning models to improve predictive accuracy and interpretability, which can then help organizations to build a future proactive retention strategy. This gives a real time application like optimizing talent management processes, informing the succession planning and enhancing the organizational development initiatives using these data driven insights. HR professionals, organizational leaders, and employees are among major beneficiaries as a result of effective attrition prediction because this will enable them to create more effective workplace environments, reduce turnover costs, and align compensation and benefits (compensation and benefits) with the needs of employees, thereby fostering a more engaged and productive workforce.

Significant points from several studies related to predict Employee Attrition

Citations and year	Objective of the paper	Dataset	Algorithms	Result
Mansor, Sani and Aliff, 2021	To forecast the loss of employees	the IBM Human Resource Analytic	DT, SVM, ANN	SVM= 88.87%

		Employee Attrition and Performance Dataset		
Shafie et al,2024	To predict Employee attrition	Kaggle	ANN, KNN, LR, SVM, NB	ANN=100%
Tanasescu et al,(2024)	To predict employee performance	Kaggle	RF, SVM, KNN, ANN, XGB, DT	DT=94%
Alshiddy and Aljaber, 2023	To predict Employee Turnover	IBM HR Analytics Employee Attrition and Performance dataset	NB, SVM, RF	Projected model=94.525%
Tümen and Sunar, 2023	To predict Employee attrition	Kaggle	DT, SVM, KNN, RF, GBM, LIGHT BGM	SVM has the high accuracy 96%
Pereira, 2024	To predict the employee attrition	LRDH of Willis towers Watson	LR, KNN, DT, RF	RF has high value 78%
Farman et al,2024	To evaluate Employee attrition	Kaggle	LR, RF, DT	RF shows of high accuracy 98%
Heuten, 2021	To predict Employee attrition	Kaggle	ANN, KNN, RF, SVM	RF value is 93%

Detailed Dataset Description

Overview of the Dataset: The IBM HR Analytics Employee Attrition dataset which is obtained from Kaggle and it is a fictional dataset of employees to analyse the employee attrition. It contains attributes like age, job position, education, and attrition details.

Data Collection: The dataset will be downloaded from Kaggle's repository (https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset).

Metadata: The dataset is in CSV format and contains approximately 1,470 records, amounting to around 500KB in size. The code files, written in Python, are expected to be around 1-2MB.

Details of Input Variables:

Below table gives a more detailed and structured view of the independent variables, their data types, descriptions, and possible values or ranges.

Table 9 Dataset Variables

Variable Name	Data Type	Description	Possible Values/Range
Age	Numeric	Age of the employee	Integer (e.g., 18-60)
BusinessTravel	Categorical	Frequency of business-related travel	Travel_Rarely, Travel_Frequently, Non-Travel
DailyRate	Numeric	Employee's daily income rate	Integer (e.g., 102 - 2068)
Department	Categorical	Department where the employee works	Sales, Research & Development, Human Resources
DistanceFromHome	Numeric	Distance from employee's home to work (miles/km)	Integer (e.g., 1-29)
Education	Categorical	Level of education	1 = Below College, 2 = College, 3 = Bachelor, 4 = Master, 5 = Doctor
EducationField	Categorical	Field of study or education area	Life Sciences, Medical, Marketing, Technical Degree, Other
EnvironmentSatisfaction	Categorical	Satisfaction with work environment	1 = Low, 2 = Medium, 3 = High, 4 = Very High
Gender	Categorical	Gender of the employee	Male, Female
JobInvolvement	Categorical	Involvement level of the	1 = Low, 2 = Medium, 3 = High, 4

		employee in their job	= Very High
JobLevel	Numeric	Job position level of the employee	Integer (1-5)
JobRole	Categorical	The employee's job role	Sales Executive, Research Scientist, Laboratory Technician, etc.
JobSatisfaction	Categorical	Satisfaction level with the job	1 = Low, 2 = Medium, 3 = High, 4 = Very High
MaritalStatus	Categorical	Marital status of the employee	Single, Married, Divorced
MonthlyIncome	Numeric	Employee's monthly income	Integer (e.g., 1000 - 20,000)
NumCompaniesWorked	Numeric	Number of companies the employee has worked for	Integer (e.g., 1-10)
OverTime	Categorical	Whether the employee works overtime	Yes, No
PercentSalaryHike	Numeric	Percent increase in salary since last hike	Integer (e.g., 11-25)
PerformanceRating	Categorical	Rating of the employee's performance	1 = Low, 2 = Good, 3 = Excellent, 4 = Outstanding
RelationshipSatisfaction	Categorical	Satisfaction with interpersonal relationships at work	1 = Low, 2 = Medium, 3 = High, 4 = Very High
StockOptionLevel	Numeric	Stock option level of the employee	Integer (0-3)
TotalWorkingYears	Numeric	Total number	Integer (e.g., 0-40)

		of years the employee has worked	
TrainingTimesLastYear	Numeric	Number of times the employee was trained in the last year	Integer (e.g., 0-6)
WorkLifeBalance	Categorical	Employee's perception of their work-life balance	1 = Bad, 2 = Good, 3 = Better, 4 = Best
YearsAtCompany	Numeric	Number of years the employee has spent at the company	Integer (e.g., 0-40)
YearsInCurrentRole	Numeric	Number of years the employee has been in their current role	Integer (e.g., 0-20)
YearsSinceLastPromotion	Numeric	Number of years since the employee's last promotion	Integer (e.g., 0-15)
YearsWithCurrManager	Numeric	Number of years the employee has worked with their current manager	Integer (e.g., 0-17)

Ethical Requirements: This dataset does not fall under the scope of the GDPR as it does not include any personal data or data that can be used to identify a person. The research does not pose any ethical issues and is in compliance with the University of Hertfordshire's ethical principles. The dataset is obtained from Kaggle, and there are no restrictions to using the dataset for research, hence, the necessary permission has been obtained. As the dataset used in this research was a fictional dataset and no people were involved in the data collection process and therefore adherence to ethical considerations was not an issue.

Tools and Technology used

The tools and technology which are used during this project include Google Colab which is a powerful interactive web-based platform for notebook-based programming with integrated support for collaborative editing and execution of the code for data analysis and machine learning. This feature includes superior computing ability and the characteristic of compatibility with almost all libraries. The principal libraries used in this research include Scikit-Learn to design the Boosting and Decision Tree models and to apply feature selection strategies involving L1 regularization and Recursive Feature Elimination. For data management and mugging of the dataset, the Pandas library has been used, for the visualization of the data, Matplotlib and Seaborn have been employed for the creation of informative and appealing plots regarding the parameters of feature and model performance. These tools in combination provide all the required tools for data management, model building and evaluation of the models and as such suitable for use in the present research on employee attrition prediction.

Considering Ethical, Social, Legal and Professional Issues

Ethical Issues

This project's data is accessible to everyone and employee turnover predictions consider ethical practices. This system highlights dependability and clarity to prevent unfair forecasts and ensures accurate assessments for each community. Project creates systems to avoid misusing technology and to clearly accept the model's restrictions. When sensitive details exist, they receive cautious handling and are made public while adhering to ethical guidelines within tech innovation.

Social Issues

The aim of the project is to benefit society by offering knowledge about worker turnover that helps companies enhance their retention methods. The research carefully analyses the dangers of automated bias and puts measures in place to stop abnormal results. This work supports organisations and individuals by supplying approaches to deepen their insight into employee loss reasons and promote fairness. This work elevates social responsibility by concentrating on ethical concepts in machine learning and points out their significance in workforce management.

Legal Issues

The dataset is anonymised and meant for research within this project which follows GDPR and CCPA data protection rules. Such an initiative does not include personal information to address privacy issues. During the project’s duration the data is kept safe and intact thanks to the license that enables its application. When the research wraps up, the data will undergo secure deletion while the research will confirm that there are no security breaches or unauthorised access.

Professional Issues

This project intends to build a solid system for anticipating staff turnover. Clear procedures for assessing and verifying the model are outlined by the strong strategy framework. Data collection and storing are both secured and done in accordance with the university's guidelines and best practice standards. This project timeline can be achieved while managing all resources carefully. To confront possible issues of bias or errors in the model extensive testing and analysis is used. This project examines the capabilities of the model for further development and its use in actual environments.

Comparison of Results with Existing Literature on Employee Attrition Prediction

The results are consistent with the findings of specific studies in the literature, including Shafie et al. (2024) and Mansor et al. (2021), in which models such as SVM and ANN exhibited robust performance and achieved high accuracy. Nevertheless, the results suggest that XGB and HGB outperform other models, including DT and MLP, with an accuracy of 91%. These models demonstrated greater resilience to feature selection, maintaining their robust performance even after the implementation of feature reduction techniques. Conversely, DT and MLP demonstrated a higher degree of sensitivity to feature selection, and RFE improved their accuracy. The predictive performance of XGB and HGB was not substantially enhanced by the union of L1 and RFE, in contrast to certain studies that demonstrated that feature selection improved predictive results.

Table 10 Comparison of results with existing literature

Citations and year	**Algorithms**	**Best Result**
Current Research	**GB, XGB, HGB, DT, MLP**	**XGB = 91%** **HGB = 91%**
Mansor, Sani and Aliff, 2021	DT, SVM, ANN	SVM = 88.87%
Shafie et al,2024	ANN, KNN, LR, SVM, NB	ANN = 100%
Tanasescu et al, 2024)	RF, SVM, KNN, ANN, XGB, DT	DT = 94%
Alshiddy and Aljaber, 2023	NB, SVM, RF	Projected model = 94.525%
Tümen and Sunar, 2023	DT, SVM, KNN, RF, GBM, LIGHT BGM	SVM = 96%
Pereira, 2024	LR, KNN, DT, RF	RF = 78%
Farman et al,2024	LR, RF, DT	RF = 98%

Heuten, 2021	ANN, KNN, RF, SVM	RF = 93%

Results and Their Relevance to the Project Objectives

The results are the direct reflection of the project objectives to predict the employee attrition with high accuracy. XGB and HGB models showed the highest performance with an accuracy of 91%, proving their capacity to capture the patterns of employee turnover. This directly speaks to the research question of the project, which is to predict employee turnover. As for the applicability, models such as XGB and HGB can be integrated into HR systems to predict the likelihood of turnover and address the issue before it becomes a problem. They are flexible, perform well in working with large data sets, and can be incorporated in organisational decision-making systems. The findings are also consistent with the observation that feature selection has a relatively small effect on the performance of these models; nevertheless, using it can help models such as MLP and DT to generalize better. The present research questions have been addressed, and the most suitable models for attrition prediction are outlined with the focus on their practical implications for HR management.

Limitations

The limitations of this research include the use of SMOTE for dealing with class imbalance, which can not necessarily mimic real-life distributions and therefore affect the generalisation of the model. It is possible that due to the use of grid search with two-fold cross-validation, the range of hyperparameter values was not fully optimized. Among the selected features, L1 and RFE had a negative impact on model performance, showing that rigorous feature elimination can harm the model's predictive power. Moreover, the study was based on a single data set, which makes the conclusions valid only for this specific data set, while data distributions may be quite different in real applications. The results could be further validated with the help of different datasets and different feature selection and hyperparameter tuning methods in order to increase the generalizability of the results.

Applications and Real-World Situations

In the field of human resource management, this research has real-world implications for retention tactics. By examining aspects including job satisfaction, performance on the job, and work-life balance, predictive models such as XGB and HGB can assist organisations in identifying individuals who may be in danger of leaving. Businesses can save money on staff turnover, boost morale, and keep production levels high by taking a proactive approach to these problems. Personalised retention strategies, better personnel planning, and optimised resource allocation are all possible with the help of these models. The concept has potential applications outside of human resources in fields where attrition prediction is important, such as customer retention and educational institutions' dropout analyses.

Future Work

Further work could be conducted in a number of areas to improve the effectiveness of employee turnover prediction models. First, a higher level of addressing class imbalance by using techniques such as ADASYN or borderline SMOTE may enhance the model's reliability. In addition, application of more complex hyperparameter tuning strategies, such as Bayesian optimization or random search, can improve the model performance that is being limited at the moment by the grid search CV. As for the feature selection, other methods could be used, for example, principal component analysis (PCA), forward feature selection and backward feature elimination, and ElasticNet feature selection to get more detailed information about the most significant features for the attrition prediction. Another way in which the approach could be enhanced is in the application of deep learning models that can reveal intricate relationships in large datasets. Extending the current models and applying them to more heterogeneous and larger sets of data from different fields will reveal their transferability. Lastly, incorporating the employee sentiment analysis from surveys or feedback could be a useful addition to improve the models' performance and make them more applicable to real-world settings.

CODE

PREPROCESS, VISUALIZATION AND WITHOUT FEATURE SELECTION OF E_ATTRITION

```
pip install matplotlib==3.8.0

pip install seaborn==0.13.2

pip install xgboost==2.1.2

pip install scikit-learn==1.3.0

from google.colab import drive

drive.mount('/content/drive')

import pandas as E_AttPan

import seaborn as E_AttSea

import matplotlib.pyplot as E_AttPypl

E_Attrition = E_AttPan.read_csv('/content/drive/MyDrive/WA_Fn-UseC_-HR-Employee-Attrition.csv')

E_Attrition

E_Attrition.info()

print("Employee Attrition Shape: ",E_Attrition.shape)

E_Attrition['Attrition'].value_counts()

E_Attrition.select_dtypes(include=['object']).columns

E_Attrition['BusinessTravel'].nunique()

E_Attrition['EmployeeCount'].nunique()

E_Attrition['StandardHours'].nunique()

### Removing Non-Essential Features, Inspecting for Missing Entries, and Resolving Duplicates

E_Attrition= E_Attrition.drop(['EmployeeCount','StandardHours'], axis=1) ### Removing Two Columns from the DataFrame to Manage Duplicates

print("Missing Values in the E-Attrition: ",E_Attrition.isnull().values.sum())

print("Duplicates in the E-Attrition    : ",E_Attrition.duplicated().sum())

print("Employee Attrition Shape: ",E_Attrition.shape)

E_AttPypl.figure(figsize=(6, 4))

E_Attrition['Attrition'].value_counts().plot(kind='bar', color='#eeddff', edgecolor='black')

E_AttPypl.title('Attrition Distribution')

E_AttPypl.xlabel('Attrition')

E_AttPypl.ylabel('Count')
```

```
E_AttPypl.show()
print("\n\n")
E_AttPypl.figure(figsize=(6, 4))
E_AttPypl.hist(E_Attrition['Age'], bins=15, color='#ddeeff', edgecolor='black')
E_AttPypl.title('Distribution of Employees Age')
E_AttPypl.xlabel('Employees Age')
E_AttPypl.ylabel('Frequency')
E_AttPypl.show()
### Visualizing Feature Correlations Using a Heatmap
E_Attrition.corr(numeric_only=True)
E_AttPypl.figure(figsize=(20, 10))
dataplot = E_AttSea.heatmap(E_Attrition.corr(numeric_only=True), cmap="YlGnBu",
annot=True)
E_AttPypl.show()
E_Attrition.select_dtypes(include=['object']).columns
from sklearn import preprocessing as E_AttPProcess
E_Ob_to_num = E_AttPProcess.LabelEncoder()
E_Attrition['Attrition']= E_Ob_to_num.fit_transform(E_Attrition['Attrition'])
E_Attrition['BusinessTravel']=
E_Ob_to_num.fit_transform(E_Attrition['BusinessTravel'])
E_Attrition['Department']= E_Ob_to_num.fit_transform(E_Attrition['Department'])
E_Attrition['EducationField']=
E_Ob_to_num.fit_transform(E_Attrition['EducationField'])
E_Attrition['Gender']= E_Ob_to_num.fit_transform(E_Attrition['Gender'])
E_Attrition['JobRole']= E_Ob_to_num.fit_transform(E_Attrition['JobRole'])
E_Attrition['MaritalStatus']= E_Ob_to_num.fit_transform(E_Attrition['MaritalStatus'])
E_Attrition['Over18']= E_Ob_to_num.fit_transform(E_Attrition['Over18'])
E_Attrition['OverTime']= E_Ob_to_num.fit_transform(E_Attrition['OverTime'])
from collections import Counter as E_AttCount
from imblearn.over_sampling import SMOTE as E_AttSTE
E_AttritionX = E_Attrition.drop('Attrition',axis=1)
E_AttritionY = E_Attrition['Attrition']
print('Attrition Categories  %s' % E_AttCount(E_AttritionY))
Att_EMod = E_AttSTE()
```

```
E_AttritionX, E_AttritionY = Att_EMod.fit_resample(E_AttritionX, E_AttritionY)
print('Balanced Attrition Categories %s' % E_AttCount(E_AttritionY))
E_Attrition = E_AttPan.concat([E_AttritionX, E_AttritionY], axis=1)
print("\nEmployee Attrition Shape: ",E_Attrition.shape)
E_Attrition['Attrition'].value_counts()
E_Attrition.select_dtypes(include=['object']).columns
E_Attrition.to_csv('E_Attrition.csv', index=False)
E_Attrition
from sklearn.model_selection import train_test_split as E_AttTSpl
E_AttritionX = E_Attrition.drop('Attrition',axis=1)
E_AttritionY = E_Attrition['Attrition']
### Splitting the Data into 70% for Training, 15% for Validation, and 15% for Testing
E_AttritionXtrn, E_AttritiontXtst, E_AttritionYtrn, E_AttritionYtst =
E_AttTSpl(E_AttritionX,E_AttritionY,test_size=0.3,random_state= 7)
E_AttritionXvld, E_AttritiontXtst, E_AttritionYvld, E_AttritionYtst =
E_AttTSpl(E_AttritiontXtst,E_AttritionYtst,test_size=0.5,random_state= 7)
print("\nTraining Samples for E_Attrition: ",E_AttritionXtrn.shape)
print("Validation Samples for E_Attrition: ",E_AttritionXvld.shape)
print("Testing Samples for E_Attrition: ",E_AttritiontXtst.shape)
from sklearn.model_selection import GridSearchCV as E_AttGsss
from sklearn import metrics as support_mtr
from sklearn.metrics import confusion_matrix as E_AttMx
from sklearn.metrics import ConfusionMatrixDisplay as Att_EMDS
from sklearn.metrics import classification_report as E_AttCR
import warnings as Att_EWARN
Att_EWARN.filterwarnings("ignore")
```

Gradient Boosting without Feature Selection

```
from sklearn.ensemble import GradientBoostingClassifier as E_AttGrb
Att_Eprm = { 'loss':['log_loss', 'exponential'],'learning_rate': [0.1, 0.3,
0.8],'n_estimators':[100, 140, 150, 200]}
Att_EMod = E_AttGrb(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
```

```
Att_EMod = E_AttGrb(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn)  ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
from sklearn.model_selection import learning_curve as AttLearnCurve
import numpy as E_Attnu
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="GB Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):
    train_sizes, train_scores, val_scores = AttLearnCurve(
```

```
    estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs,
train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
  )
  ### Assessing the Mean and Standard Deviation for Data Analysis
  train_mean = E_Attnu.mean(train_scores, axis=1)
  train_std = E_Attnu.std(train_scores, axis=1)
  val_mean = E_Attnu.mean(val_scores, axis=1)
  val_std = E_Attnu.std(val_scores, axis=1)
  ### Visualizing Model Training with a Learning Curve
  E_AttPypl.figure(figsize=(10, 6))
  E_AttPypl.plot(train_sizes, train_mean, label="Training Score", color="blue",
marker='o')
  E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
color="blue", alpha=0.2)
  E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="green",
marker='o')
  E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std,
color="green", alpha=0.2)
  E_AttPypl.title(title)
  E_AttPypl.xlabel("Training Set Size")
  E_AttPypl.ylabel("Score")
  E_AttPypl.legend(loc="best")
  E_AttPypl.grid()
  E_AttPypl.show()
Att_EMod = E_AttGrb(learning_rate= 0.1, loss= 'log_loss', n_estimators= 200) ###
Tuning the Model with the Best-Performing Parameters
Emp_learningCurve(
  estimator=Att_EMod,
  E_AttritionX=E_AttritionXtrn,
  E_AttritionY=E_AttritionYtrn,
  title="Learning Curve: Gradient Boosting for Employee Attrition",
  cv=5
)
```

Hist.Gradient Boosting Without Feature Selection

```
from sklearn.ensemble import HistGradientBoostingClassifier as E_AttHisGrb
```

```
Att_Eprm = {'learning_rate': [0.1, 0.3,
0.8],'max_iter':[100,200,350],'max_depth':[2,5,8,9]}

Att_EMod = E_AttHisGrb(random_state=7)

Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)

Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))

print(Att_EMod.best_params_)

Att_EMod = E_AttHisGrb(**Att_EMod.best_params_)

Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data

Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition
Data

print(E_AttCR(E_AttritionYvld, Att_YpredE))

E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)

Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')

E_AttPypl.title("Employee Attrition Matrix Validation")

E_AttPypl.show()

### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")

print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")

Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ###  Testing Model on E_Attrition
Data

print(E_AttCR(E_AttritionYtst, Att_YpredE))

E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)

Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')

E_AttPypl.title("Employee Attrition Matrix Testing")

E_AttPypl.show()

### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
```

```
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")

def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="HistGB Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):
    train_sizes, train_scores, val_scores = AttLearnCurve(
        estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
    )
    ### Assessing the Mean and Standard Deviation for Data Analysis
    train_mean = E_Attnu.mean(train_scores, axis=1)
    train_std = E_Attnu.std(train_scores, axis=1)
    val_mean = E_Attnu.mean(val_scores, axis=1)
    val_std = E_Attnu.std(val_scores, axis=1)
    ### Visualizing Model Training with a Learning Curve
    E_AttPypl.figure(figsize=(10, 6))
    E_AttPypl.plot(train_sizes, train_mean, label="Training Score", color="darkturquoise", marker='o')
    E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="darkturquoise", alpha=0.2)
    E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet", marker='o')
    E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, color="darkviolet", alpha=0.2)
    E_AttPypl.title(title)
    E_AttPypl.xlabel("Training Set Size")
    E_AttPypl.ylabel("Score")
    E_AttPypl.legend(loc="best")
    E_AttPypl.grid()
    E_AttPypl.show()
Att_EMod = E_AttHisGrb(learning_rate= 0.8, max_depth= 2, max_iter= 200) ### Tuning the Model with the Best-Performing Parameters
Emp_learningCurve(
    estimator=Att_EMod,
    E_AttritionX=E_AttritionXtrn,
    E_AttritionY=E_AttritionYtrn,
    title="Learning Curve: Histogram based Gradient Boosting for Employee Attrition",
    cv=5
```

)

Extreme Gradient Boosting Without Feature Selection

```
from xgboost import XGBClassifier as E_AttExGrb

Att_Eprm = { 'max_depth':[2,5,7,9],'learning_rate': [0.1, 0.3, 0.8],'n_estimators':[100, 140, 150, 200]}

Att_EMod = E_AttExGrb(random_state=7)

Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)

Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7), E_AttritionYtrn.sample(400, random_state=7))

print(Att_EMod.best_params_)

Att_EMod = E_AttExGrb(**Att_EMod.best_params_)

Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data

Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data

print(E_AttCR(E_AttritionYvld, Att_YpredE))

E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)

Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')

E_AttPypl.title("Employee Attrition Matrix Validation")

E_AttPypl.show()

### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")

print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")

Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data

print(E_AttCR(E_AttritionYtst, Att_YpredE))

E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)

Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')

E_AttPypl.title("Employee Attrition Matrix Testing")

E_AttPypl.show()

### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
```

```
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")

print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")

def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="XGB Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):

   train_sizes, train_scores, val_scores = AttLearnCurve(

      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)

   )

   ### Assessing the Mean and Standard Deviation for Data Analysis

   train_mean = E_Attnu.mean(train_scores, axis=1)

   train_std = E_Attnu.std(train_scores, axis=1)

   val_mean = E_Attnu.mean(val_scores, axis=1)

   val_std = E_Attnu.std(val_scores, axis=1)

   ### Visualizing Model Training with a Learning Curve

   E_AttPypl.figure(figsize=(10, 6))

   E_AttPypl.plot(train_sizes, train_mean, label="Training Score", color="darkturquoise", marker='o')

   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="darkturquoise", alpha=0.2)

   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet", marker='o')

   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, color="darkviolet", alpha=0.2)

   E_AttPypl.title(title)

   E_AttPypl.xlabel("Training Set Size")

   E_AttPypl.ylabel("Score")

   E_AttPypl.legend(loc="best")

   E_AttPypl.grid()

   E_AttPypl.show()

Att_EMod = E_AttExGrb(learning_rate= 0.1, max_depth= 9, n_estimators= 200) ### Tuning the Model with the Best-Performing Parameters

Emp_learningCurve(

   estimator=Att_EMod,

   E_AttritionX=E_AttritionXtrn,

   E_AttritionY=E_AttritionYtrn,
```

```
    title="Learning Curve: Extreme Gradient Boosting for Employee Attrition",
    cv=5
)
```

Decision Tree Without Feature Selection

```
from sklearn.tree import DecisionTreeClassifier as E_AttDEC
Att_Eprm = {'criterion':['gini', 'entropy', 'log_loss'],'max_depth': [10, 5, 8, 3],'min_samples_split':[2, 4, 5, 7]}
Att_EMod = E_AttDEC(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7), E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttDEC(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
### Assessing FPR and FNR for the E_Attrition Model
```

```
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")

print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")

def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="DT Learning
Curve", cv=5, scoring="accuracy", n_jobs=-1):

   train_sizes, train_scores, val_scores = AttLearnCurve(

      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs,
train_sizes=E_Attnu.linspace(0.1, 1.0, 10)

   )

   ### Assessing the Mean and Standard Deviation for Data Analysis

   train_mean = E_Attnu.mean(train_scores, axis=1)

   train_std = E_Attnu.std(train_scores, axis=1)

   val_mean = E_Attnu.mean(val_scores, axis=1)

   val_std = E_Attnu.std(val_scores, axis=1)

   ### Visualizing Model Training with a Learning Curve

   E_AttPypl.figure(figsize=(10, 6))

   E_AttPypl.plot(train_sizes, train_mean, label="Training Score",
color="darkturquoise", marker='o')

   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
color="darkturquoise", alpha=0.2)

   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet",
marker='o')

   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std,
color="darkviolet", alpha=0.2)

   E_AttPypl.title(title)

   E_AttPypl.xlabel("Training Set Size")

   E_AttPypl.ylabel("Score")

   E_AttPypl.legend(loc="best")

   E_AttPypl.grid()

   E_AttPypl.show()

Att_EMod = E_AttDEC(criterion= 'gini', max_depth= 8, min_samples_split= 7) ###
Tuning the Model with the Best-Performing Parameters

Emp_learningCurve(

   estimator=Att_EMod,
```

```
    E_AttritionX=E_AttritionXtrn,
    E_AttritionY=E_AttritionYtrn,
    title="Learning Curve: Decision Tree for Employee Attrition",
    cv=5
)
```

MLP(Multi-Layered Preciption) without Feature Selection

```
from sklearn.neural_network import MLPClassifier as E_AttMult
Att_Eprm = {'hidden_layer_sizes':[10, 20, 30],'activation':
['relu','tanh','logistic'],'solver':['lbfgs','adam','sgd']}
Att_EMod = E_AttMult(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttMult(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition
Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
```

```
E_AttPypl.show()

#### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")

print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")

def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="MLP Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):

   train_sizes, train_scores, val_scores = AttLearnCurve(

      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)

   )

   ### Assessing the Mean and Standard Deviation for Data Analysis

   train_mean = E_Attnu.mean(train_scores, axis=1)

   train_std = E_Attnu.std(train_scores, axis=1)

   val_mean = E_Attnu.mean(val_scores, axis=1)

   val_std = E_Attnu.std(val_scores, axis=1)

   ### Visualizing Model Training with a Learning Curve

   E_AttPypl.figure(figsize=(10, 6))

   E_AttPypl.plot(train_sizes, train_mean, label="Training Score", color="darkturquoise", marker='o')

   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="darkturquoise", alpha=0.2)

   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet", marker='o')

   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, color="darkviolet", alpha=0.2)

   E_AttPypl.title(title)

   E_AttPypl.xlabel("Training Set Size")

   E_AttPypl.ylabel("Score")

   E_AttPypl.legend(loc="best")

   E_AttPypl.grid()

   E_AttPypl.show()
```

```
Att_EMod = E_AttMult(activation= 'tanh', hidden_layer_sizes= 10, solver= 'lbfgs') ### Tuning the Model with the Best-Performing Parameters

Emp_learningCurve(
   estimator=Att_EMod,
   E_AttritionX=E_AttritionXtrn,
   E_AttritionY=E_AttritionYtrn,
   title="Learning Curve: Multi Layer Perceptron for Employee Attrition",
   cv=5
)
```

L1 FEATURE SELECTION FOR E_ATTRITION

L1 Feature Selection

Applying L1 Feature Selection

```
from sklearn.linear_model import Lasso as E_AttritionL1

E_AttritionX = E_Attrition.drop('Attrition',axis=1)

E_AttritionY = E_Attrition['Attrition']

EAttMod = E_AttritionL1(alpha=0.01)

EAttMod.fit(E_AttritionX, E_AttritionY)

E_AttCoef = EAttMod.coef_      #### Determining the Best Features for Improved Accuracy

E_L1Feat = [e for e in range(len(E_AttCoef)) if E_AttCoef[e] != 1]

E_L1NFeatures = E_L1Feat[:20]   #### Choosing the Optimal Number of Features for Model Efficiency

E_SelFeatures = E_AttritionX.columns[E_L1NFeatures]

print('L1 Features for E_Attrition:\n', E_SelFeatures)

print('\nNumber of Features selected by L1 Feature Selection for E_Attrition:', len(E_SelFeatures))

E_AttritionX = E_AttritionX[E_SelFeatures]

E_AttritionX.shape
```

Gradient Boosting Using L1 Feature Selection

```
from sklearn.ensemble import GradientBoostingClassifier as E_AttGrb

Att_Eprm = { 'loss':['log_loss', 'exponential'],'learning_rate': [0.1, 0.3, 0.8],'n_estimators':[100, 140, 150, 200]}

Att_EMod = E_AttGrb(random_state=7)

Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
```

```
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttGrb(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition
Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
from sklearn.model_selection import learning_curve as AttLearnCurve
import numpy as E_Attnu
```

```
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="GB Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):
    train_sizes, train_scores, val_scores = AttLearnCurve(
        estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
    )
    ### Assessing the Mean and Standard Deviation for Data Analysis
    train_mean = E_Attnu.mean(train_scores, axis=1)
    train_std = E_Attnu.std(train_scores, axis=1)
    val_mean = E_Attnu.mean(val_scores, axis=1)
    val_std = E_Attnu.std(val_scores, axis=1)
    ### Visualizing Model Training with a Learning Curve
    E_AttPypl.figure(figsize=(10, 6))
    E_AttPypl.plot(train_sizes, train_mean, label="Training Score", color="darkturquoise", marker='o')
    E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="darkturquoise", alpha=0.2)
    E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet", marker='o')
    E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, color="darkviolet", alpha=0.2)
    E_AttPypl.title(title)
    E_AttPypl.xlabel("Training Set Size")
    E_AttPypl.ylabel("Score")
    E_AttPypl.legend(loc="best")
    E_AttPypl.grid()
    E_AttPypl.show()
Att_EMod = E_AttGrb(learning_rate= 0.3, loss= 'log_loss', n_estimators= 140) ### Tuning the Model with the Best-Performing Parameters
Emp_learningCurve(
    estimator=Att_EMod,
    E_AttritionX=E_AttritionXtrn,
    E_AttritionY=E_AttritionYtrn,
    title="Learning Curve: Gradient Boosting for Employee Attrition",
    cv=5
)
```

Hist.Gradient Boosting L1 Feature Selection

```
from sklearn.ensemble import HistGradientBoostingClassifier as E_AttHisGrb

Att_Eprm = {'learning_rate': [0.1, 0.3,
0.8],'max_iter':[100,200,350],'max_depth':[2,5,8,9]}

Att_EMod = E_AttHisGrb(random_state=7)

Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)

Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))

print(Att_EMod.best_params_)

Att_EMod = E_AttHisGrb(**Att_EMod.best_params_)

Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data

Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition
Data

print(E_AttCR(E_AttritionYvld, Att_YpredE))

E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)

Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')

E_AttPypl.title("Employee Attrition Matrix Validation")

E_AttPypl.show()

#### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")

print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")

Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data

print(E_AttCR(E_AttritionYtst, Att_YpredE))

E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)

Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')

E_AttPypl.title("Employee Attrition Matrix Testing")

E_AttPypl.show()

#### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
```

```
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")

print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")

def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="HistGB Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):

   train_sizes, train_scores, val_scores = AttLearnCurve(

      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)

   )

   ### Assessing the Mean and Standard Deviation for Data Analysis

   train_mean = E_Attnu.mean(train_scores, axis=1)

   train_std = E_Attnu.std(train_scores, axis=1)

   val_mean = E_Attnu.mean(val_scores, axis=1)

   val_std = E_Attnu.std(val_scores, axis=1)

   ### Visualizing Model Training with a Learning Curve

   E_AttPypl.figure(figsize=(10, 6))

   E_AttPypl.plot(train_sizes, train_mean, label="Training Score", color="darkturquoise", marker='o')

   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="darkturquoise", alpha=0.2)

   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet", marker='o')

   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, color="darkviolet", alpha=0.2)

   E_AttPypl.title(title)

   E_AttPypl.xlabel("Training Set Size")

   E_AttPypl.ylabel("Score")

   E_AttPypl.legend(loc="best")

   E_AttPypl.grid()

   E_AttPypl.show()

Att_EMod = E_AttHisGrb(learning_rate= 0.8, max_depth= 5, max_iter= 200) ### Tuning the Model with the Best-Performing Parameters

Emp_learningCurve(

   estimator=Att_EMod,

   E_AttritionX=E_AttritionXtrn,

   E_AttritionY=E_AttritionYtrn,

   title="Learning Curve: Histogram based Gradient Boosting for Employee Attrition",
```

```
    cv=5
)
```

Extreme Gradient Boosting L1 Feature Selection

```
from xgboost import XGBClassifier as E_AttExGrb
Att_Eprm = { 'max_depth':[2,5,7,9],'learning_rate': [0.1, 0.3, 0.8],'n_estimators':[100, 140, 150, 200]}
Att_EMod = E_AttExGrb(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7), E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttExGrb(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
```

```
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")

print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")

def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="XGB Learning
Curve", cv=5, scoring="accuracy", n_jobs=-1):

   train_sizes, train_scores, val_scores = AttLearnCurve(

      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs,
train_sizes=E_Attnu.linspace(0.1, 1.0, 10)

   )

   ### Assessing the Mean and Standard Deviation for Data Analysis

   train_mean = E_Attnu.mean(train_scores, axis=1)

   train_std = E_Attnu.std(train_scores, axis=1)

   val_mean = E_Attnu.mean(val_scores, axis=1)

   val_std = E_Attnu.std(val_scores, axis=1)

   ### Visualizing Model Training with a Learning Curve

   E_AttPypl.figure(figsize=(10, 6))

   E_AttPypl.plot(train_sizes, train_mean, label="Training Score",
color="darkturquoise", marker='o')

   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
color="darkturquoise", alpha=0.2)

   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet",
marker='o')

   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std,
color="darkviolet", alpha=0.2)

 E_AttPypl.title(title)

   E_AttPypl.xlabel("Training Set Size")

   E_AttPypl.ylabel("Score")

   E_AttPypl.legend(loc="best")

   E_AttPypl.grid()

   E_AttPypl.show()

Att_EMod = E_AttExGrb(learning_rate= 0.1, max_depth= 2, n_estimators= 200) ###
Tuning the Model with the Best-Performing Parameters

Emp_learningCurve(

   estimator=Att_EMod,

   E_AttritionX=E_AttritionXtrn,
```

```
    E_AttritionY=E_AttritionYtrn,
    title="Learning Curve: Extreme Gradient Boosting for Employee Attrition",
    cv=5
)
```

Decision Tree Using L1 Feature Selection

```
from sklearn.tree import DecisionTreeClassifier as E_AttDEC
Att_Eprm = {'criterion':['gini', 'entropy', 'log_loss'],'max_depth': [10, 5, 8, 3],'min_samples_split':[2, 4, 5, 7]}
Att_EMod = E_AttDEC(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7), E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttDEC(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
```

```
#### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")

print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")

def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="DT Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):

   train_sizes, train_scores, val_scores = AttLearnCurve(

      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)

   )

   ### Assessing the Mean and Standard Deviation for Data Analysis

   train_mean = E_Attnu.mean(train_scores, axis=1)

   train_std = E_Attnu.std(train_scores, axis=1)

   val_mean = E_Attnu.mean(val_scores, axis=1)

   val_std = E_Attnu.std(val_scores, axis=1)

   ### Visualizing Model Training with a Learning Curve

   E_AttPypl.figure(figsize=(10, 6))

   E_AttPypl.plot(train_sizes, train_mean, label="Training Score", color="darkturquoise", marker='o')

   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="darkturquoise", alpha=0.2)

   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet", marker='o')

   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, color="darkviolet", alpha=0.2)

   E_AttPypl.title(title)

   E_AttPypl.xlabel("Training Set Size")

   E_AttPypl.ylabel("Score")

   E_AttPypl.legend(loc="best")

   E_AttPypl.grid()

   E_AttPypl.show()

Att_EMod = E_AttDEC(criterion= 'gini', max_depth= 10, min_samples_split= 4) ### Tuning the Model with the Best-Performing Parameters
```

```
Emp_learningCurve(
   estimator=Att_EMod,
   E_AttritionX=E_AttritionXtrn,
   E_AttritionY=E_AttritionYtrn,
   title="Learning Curve: Decision Tree for Employee Attrition",
   cv=5
)
```

MLP Using L1 Feature Selection

```
from sklearn.neural_network import MLPClassifier as E_AttMult
Att_Eprm = {'hidden_layer_sizes':[10, 20, 30],'activation':
['relu','tanh','logistic'],'solver':['lbfgs','adam','sgd']}
Att_EMod = E_AttMult(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttMult(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition
Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
```

```
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="MLP Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):
   train_sizes, train_scores, val_scores = AttLearnCurve(
      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
   )
   ### Assessing the Mean and Standard Deviation for Data Analysis
   train_mean = E_Attnu.mean(train_scores, axis=1)
   train_std = E_Attnu.std(train_scores, axis=1)
   val_mean = E_Attnu.mean(val_scores, axis=1)
   val_std = E_Attnu.std(val_scores, axis=1)
   ### Visualizing Model Training with a Learning Curve
   E_AttPypl.figure(figsize=(10, 6))
   E_AttPypl.plot(train_sizes, train_mean, label="Training Score", color="darkturquoise", marker='o')
   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="darkturquoise", alpha=0.2)
   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet", marker='o')
   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, color="darkviolet", alpha=0.2)
   E_AttPypl.title(title)
   E_AttPypl.xlabel("Training Set Size")
   E_AttPypl.ylabel("Score")
   E_AttPypl.legend(loc="best")
```

```
    E_AttPypl.grid()
    E_AttPypl.show()
Att_EMod = E_AttMult(activation= 'logistic', hidden_layer_sizes= 10, solver= 'lbfgs') ### using the best parameters
Emp_learningCurve(
    estimator=Att_EMod,
    E_AttritionX=E_AttritionXtrn,
    E_AttritionY=E_AttritionYtrn,
    title="Learning Curve: Multi Layer Perceptron for Employee Attrition",
    cv=5
)
```

RECURSIVE FEATURE ELIMINATION(RFE) FOR E_ATTRITION

Recursive Feature Elimination (RFE Features)

Gradient Boosting Using RFE(Recursive Feature Elimination)

```
from sklearn.ensemble import GradientBoostingClassifier as E_AttGrb
Att_Eprm = { 'loss':['log_loss', 'exponential'],'learning_rate': [0.1, 0.3, 0.8],'n_estimators':[100, 140, 150, 200]}
Att_EMod = E_AttGrb(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7), E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttGrb(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
```

```
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
from sklearn.model_selection import learning_curve as AttLearnCurve
import numpy as E_Attnu
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="GB Learning
Curve", cv=5, scoring="accuracy", n_jobs=-1):
   train_sizes, train_scores, val_scores = AttLearnCurve(
      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs,
train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
   )
   ### Assessing the Mean and Standard Deviation for Data Analysis
   train_mean = E_Attnu.mean(train_scores, axis=1)
   train_std = E_Attnu.std(train_scores, axis=1)
   val_mean = E_Attnu.mean(val_scores, axis=1)
   val_std = E_Attnu.std(val_scores, axis=1)
   ### Visualizing Model Training with a Learning Curve
   E_AttPypl.figure(figsize=(10, 6))
   E_AttPypl.plot(train_sizes, train_mean, label="Training Score",
color="darkturquoise", marker='o')
   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
color="darkturquoise", alpha=0.2)
```

```
    E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet", marker='o')
    E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, color="darkviolet", alpha=0.2)
    E_AttPypl.title(title)
    E_AttPypl.xlabel("Training Set Size")
    E_AttPypl.ylabel("Score")
    E_AttPypl.legend(loc="best")
    E_AttPypl.grid()
    E_AttPypl.show()
Att_EMod = E_AttGrb(learning_rate= 0.8, loss= 'log_loss', n_estimators= 150) ### Tuning the Model with the Best-Performing Parameters
Emp_learningCurve(
    estimator=Att_EMod,
    E_AttritionX=E_AttritionXtrn,
    E_AttritionY=E_AttritionYtrn,
    title="Learning Curve: Gradient Boosting for Employee Attrition",
    cv=5
)
```

Hist.Gradient Boosting using RFE(Recursive Feature Elimination)

```
from sklearn.ensemble import HistGradientBoostingClassifier as E_AttHisGrb
Att_Eprm = {'learning_rate': [0.1, 0.3, 0.8],'max_iter':[100,200,350],'max_depth':[2,5,8,9]}
Att_EMod = E_AttHisGrb(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7), E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttHisGrb(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
```

```
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="HistGB Learning
Curve", cv=5, scoring="accuracy", n_jobs=-1):
  train_sizes, train_scores, val_scores = AttLearnCurve(
    estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs,
train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
  )
  ### Assessing the Mean and Standard Deviation for Data Analysis
  train_mean = E_Attnu.mean(train_scores, axis=1)
  train_std = E_Attnu.std(train_scores, axis=1)
  val_mean = E_Attnu.mean(val_scores, axis=1)
  val_std = E_Attnu.std(val_scores, axis=1)
  ### Visualizing Model Training with a Learning Curve
  E_AttPypl.figure(figsize=(10, 6))
```

```
    E_AttPypl.plot(train_sizes, train_mean, label="Training Score",
color="darkturquoise", marker='o')
    E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
color="darkturquoise", alpha=0.2)
    E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet",
marker='o')
    E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std,
color="darkviolet", alpha=0.2)
    E_AttPypl.title(title)
    E_AttPypl.xlabel("Training Set Size")
    E_AttPypl.ylabel("Score")
    E_AttPypl.legend(loc="best")
    E_AttPypl.grid()
    E_AttPypl.show()
Att_EMod = E_AttHisGrb(learning_rate= 0.1, max_depth= 2, max_iter= 350) ### Tuning
the Model with the Best-Performing Parameters
Emp_learningCurve(
    estimator=Att_EMod,
    E_AttritionX=E_AttritionXtrn,
    E_AttritionY=E_AttritionYtrn,
    title="Learning Curve: Histogram based Gradient Boosting for Employee Attrition",
    cv=5
)
```

Extreme Gradient Boosting using RFE(Recursive Feature Elimination)

```
from xgboost import XGBClassifier as E_AttExGrb
Att_Eprm = { 'max_depth':[2,5,7,9],'learning_rate': [0.1, 0.3, 0.8],'n_estimators':[100,
140, 150, 200]}
Att_EMod = E_AttExGrb(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttExGrb(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition
Data
```

```
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="XGB Learning
Curve", cv=5, scoring="accuracy", n_jobs=-1):
  train_sizes, train_scores, val_scores = AttLearnCurve(
    estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs,
train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
  )
  ### Assessing the Mean and Standard Deviation for Data Analysis
  train_mean = E_Attnu.mean(train_scores, axis=1)
  train_std = E_Attnu.std(train_scores, axis=1)
  val_mean = E_Attnu.mean(val_scores, axis=1)
  val_std = E_Attnu.std(val_scores, axis=1)
```

```
   ### Visualizing Model Training with a Learning Curve
   E_AttPypl.figure(figsize=(10, 6))
   E_AttPypl.plot(train_sizes, train_mean, label="Training Score",
color="darkturquoise", marker='o')
   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
color="darkturquoise", alpha=0.2)
   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet",
marker='o')
   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std,
color="darkviolet", alpha=0.2)
   E_AttPypl.title(title)
   E_AttPypl.xlabel("Training Set Size")
   E_AttPypl.ylabel("Score")
   E_AttPypl.legend(loc="best")
   E_AttPypl.grid()
   E_AttPypl.show()
Att_EMod = E_AttExGrb(learning_rate= 0.3, max_depth= 2, n_estimators= 200) ###
Tuning the Model with the Best-Performing Parameters
Emp_learningCurve(
   estimator=Att_EMod,
   E_AttritionX=E_AttritionXtrn,
   E_AttritionY=E_AttritionYtrn,
   title="Learning Curve: Extreme Gradient Boosting for Employee Attrition",
   cv=5
)
```

Decision Tree Using RFE(Recursive Feature Elimination)

```
from sklearn.tree import DecisionTreeClassifier as E_AttDEC
Att_Eprm = {'criterion':['gini', 'entropy', 'log_loss'],'max_depth': [10, 5, 8,
3],'min_samples_split':[2, 4, 5, 7]}
Att_EMod = E_AttDEC(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
```

```
Att_EMod = E_AttDEC(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="DT Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):
   train_sizes, train_scores, val_scores = AttLearnCurve(
      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
   )
```

```
   ### Assessing the Mean and Standard Deviation for Data Analysis
   train_mean = E_Attnu.mean(train_scores, axis=1)
   train_std = E_Attnu.std(train_scores, axis=1)
   val_mean = E_Attnu.mean(val_scores, axis=1)
   val_std = E_Attnu.std(val_scores, axis=1)
   ### Visualizing Model Training with a Learning Curve
   E_AttPypl.figure(figsize=(10, 6))
   E_AttPypl.plot(train_sizes, train_mean, label="Training Score",
color="darkturquoise", marker='o')
   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
color="darkturquoise", alpha=0.2)
   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet",
marker='o')
   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std,
color="darkviolet", alpha=0.2)
   E_AttPypl.title(title)
   E_AttPypl.xlabel("Training Set Size")
   E_AttPypl.ylabel("Score")
   E_AttPypl.legend(loc="best")
   E_AttPypl.grid()
   E_AttPypl.show()
Att_EMod = E_AttDEC(criterion= 'entropy', max_depth= 10, min_samples_split= 5) ###
Tuning the Model with the Best-Performing Parameters
Emp_learningCurve(
   estimator=Att_EMod,
   E_AttritionX=E_AttritionXtrn,
   E_AttritionY=E_AttritionYtrn,
   title="Learning Curve: Decision Tree for Employee Attrition",
   cv=5
)
```

Multi-Layer Perceptron (MLP) using RFE(Recursive Feature Elimination)

```
from sklearn.neural_network import MLPClassifier as E_AttMult
Att_Eprm = {'hidden_layer_sizes':[10, 20, 30],'activation':
['relu','tanh','logistic'],'solver':['lbfgs','adam','sgd']}
Att_EMod = E_AttMult(random_state=7)
```

```
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttMult(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition
Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="MLP Learning
Curve", cv=5, scoring="accuracy", n_jobs=-1):
    train_sizes, train_scores, val_scores = AttLearnCurve(
```

```
    estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs,
train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
  )
  ### Assessing the Mean and Standard Deviation for Data Analysis
  train_mean = E_Attnu.mean(train_scores, axis=1)
  train_std = E_Attnu.std(train_scores, axis=1)
  val_mean = E_Attnu.mean(val_scores, axis=1)
  val_std = E_Attnu.std(val_scores, axis=1)
  ### Visualizing Model Training with a Learning Curve
  E_AttPypl.figure(figsize=(10, 6))
  E_AttPypl.plot(train_sizes, train_mean, label="Training Score",
color="darkturquoise", marker='o')
  E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
color="darkturquoise", alpha=0.2)
  E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet",
marker='o')
  E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std,
color="darkviolet", alpha=0.2)
  E_AttPypl.title(title)
  E_AttPypl.xlabel("Training Set Size")
  E_AttPypl.ylabel("Score")
  E_AttPypl.legend(loc="best")
  E_AttPypl.grid()
  E_AttPypl.show()
Att_EMod = E_AttMult(activation= 'relu', hidden_layer_sizes= 30, solver= 'lbfgs') ###
Tuning the Model with the Best-Performing Parameters
Emp_learningCurve(
  estimator=Att_EMod,
  E_AttritionX=E_AttritionXtrn,
  E_AttritionY=E_AttritionYtrn,
  title="Learning Curve: Multi Layer Perceptron for Employee Attrition",
  cv=5
)
```

UNION OF L1 AND RFE FEATURES FOR ATTRITION

Union of L1 and RFE Features

Gradient Boosting Using Union Features

```
from sklearn.ensemble import GradientBoostingClassifier as E_AttGrb

Att_Eprm = { 'loss':['log_loss', 'exponential'],'learning_rate': [0.1, 0.3,
0.8],'n_estimators':[100, 140, 150, 200]}

Att_EMod = E_AttGrb(random_state=7)

Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)

Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))

print(Att_EMod.best_params_)

Att_EMod = E_AttGrb(**Att_EMod.best_params_)

Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data

Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition
Data

print(E_AttCR(E_AttritionYvld, Att_YpredE))

E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)

Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')

E_AttPypl.title("Employee Attrition Matrix Validation")

E_AttPypl.show()

#### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")

print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")

Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data

print(E_AttCR(E_AttritionYtst, Att_YpredE))

E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)

Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')

E_AttPypl.title("Employee Attrition Matrix Testing")

E_AttPypl.show()

#### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
```

```
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
from sklearn.model_selection import learning_curve as AttLearnCurve
import numpy as E_Attnu
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="GB Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):
   train_sizes, train_scores, val_scores = AttLearnCurve(
      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
   )
   ### Assessing the Mean and Standard Deviation for Data Analysis
   train_mean = E_Attnu.mean(train_scores, axis=1)
   train_std = E_Attnu.std(train_scores, axis=1)
   val_mean = E_Attnu.mean(val_scores, axis=1)
   val_std = E_Attnu.std(val_scores, axis=1)
   ### Visualizing Model Training with a Learning Curve
   E_AttPypl.figure(figsize=(10, 6))
   E_AttPypl.plot(train_sizes, train_mean, label="Training Score", color="darkturquoise", marker='o')
   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="darkturquoise", alpha=0.2)
   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet", marker='o')
   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, color="darkviolet", alpha=0.2)
   E_AttPypl.title(title)
   E_AttPypl.xlabel("Training Set Size")
   E_AttPypl.ylabel("Score")
   E_AttPypl.legend(loc="best")
   E_AttPypl.grid()
   E_AttPypl.show()
Att_EMod = E_AttGrb(learning_rate= 0.8, loss= 'log_loss', n_estimators= 140) ### Tuning the Model with the Best-Performing Parameters
Emp_learningCurve(
   estimator=Att_EMod,
   E_AttritionX=E_AttritionXtrn,
```

```
    E_AttritionY=E_AttritionYtrn,
    title="Learning Curve: Gradient Boosting for Employee Attrition",
    cv=5
)
```

Hist.Gradient Boosting Using Union Features

```
from sklearn.ensemble import HistGradientBoostingClassifier as E_AttHisGrb
Att_Eprm = {'learning_rate': [0.1, 0.3,
0.8],'max_iter':[100,200,350],'max_depth':[2,5,8,9]}
Att_EMod = E_AttHisGrb(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttHisGrb(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition
Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
```

```
#### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")

print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")

def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="HistGB Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):

  train_sizes, train_scores, val_scores = AttLearnCurve(

    estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)

  )

  ### Assessing the Mean and Standard Deviation for Data Analysis

  train_mean = E_Attnu.mean(train_scores, axis=1)

  train_std = E_Attnu.std(train_scores, axis=1)

  val_mean = E_Attnu.mean(val_scores, axis=1)

  val_std = E_Attnu.std(val_scores, axis=1)

  ### Visualizing Model Training with a Learning Curve

  E_AttPypl.figure(figsize=(10, 6))

  E_AttPypl.plot(train_sizes, train_mean, label="Training Score", color="darkturquoise", marker='o')

  E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="darkturquoise", alpha=0.2)

  E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet", marker='o')

  E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, color="darkviolet", alpha=0.2)

  E_AttPypl.title(title)

  E_AttPypl.xlabel("Training Set Size")

  E_AttPypl.ylabel("Score")

  E_AttPypl.legend(loc="best")

  E_AttPypl.grid()

  E_AttPypl.show()

Att_EMod = E_AttHisGrb(learning_rate= 0.8, max_depth= 8, max_iter= 350) ### Tuning the Model with the Best-Performing Parameters

Emp_learningCurve(
```

```
    estimator=Att_EMod,
    E_AttritionX=E_AttritionXtrn,
    E_AttritionY=E_AttritionYtrn,
    title="Learning Curve: Histogram based Gradient Boosting for Employee Attrition",
    cv=5
)
```

Extreme Gradient Boosting Using Union Features

```
from xgboost import XGBClassifier as E_AttExGrb
Att_Eprm = { 'max_depth':[2,5,7,9],'learning_rate': [0.1, 0.3, 0.8],'n_estimators':[100, 140, 150, 200]}
Att_EMod = E_AttExGrb(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7), E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttExGrb(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
```

```
E_AttPypl.title("Employee Attrition Matrix Testing")

E_AttPypl.show()

#### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")

print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")

def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="XGB Learning
Curve", cv=5, scoring="accuracy", n_jobs=-1):

   train_sizes, train_scores, val_scores = AttLearnCurve(

      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs,
train_sizes=E_Attnu.linspace(0.1, 1.0, 10)

   )

   ### Assessing the Mean and Standard Deviation for Data Analysis

   train_mean = E_Attnu.mean(train_scores, axis=1)

   train_std = E_Attnu.std(train_scores, axis=1)

   val_mean = E_Attnu.mean(val_scores, axis=1)

   val_std = E_Attnu.std(val_scores, axis=1)

   ### Visualizing Model Training with a Learning Curve

   E_AttPypl.figure(figsize=(10, 6))

   E_AttPypl.plot(train_sizes, train_mean, label="Training Score",
color="darkturquoise", marker='o')

   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
color="darkturquoise", alpha=0.2)

   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet",
marker='o')

   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std,
color="darkviolet", alpha=0.2)

   E_AttPypl.title(title)

   E_AttPypl.xlabel("Training Set Size")

   E_AttPypl.ylabel("Score")

   E_AttPypl.legend(loc="best")

   E_AttPypl.grid()

   E_AttPypl.show()
```

```
Att_EMod = E_AttExGrb(learning_rate= 0.8, max_depth= 7, n_estimators= 100) ### Tuning the Model with the Best-Performing Parameters

Emp_learningCurve(
   estimator=Att_EMod,
   E_AttritionX=E_AttritionXtrn,
   E_AttritionY=E_AttritionYtrn,
   title="Learning Curve: Extreme Gradient Boosting for Employee Attrition",
   cv=5
)
```

Decision Tree Using Union Features

```
from sklearn.tree import DecisionTreeClassifier as E_AttDEC

Att_Eprm = {'criterion':['gini', 'entropy', 'log_loss'],'max_depth': [10, 5, 8, 3],'min_samples_split':[2, 4, 5, 7]}

Att_EMod = E_AttDEC(random_state=7)

Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)

Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7), E_AttritionYtrn.sample(400, random_state=7))

print(Att_EMod.best_params_)

Att_EMod = E_AttDEC(**Att_EMod.best_params_)

Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data

Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data

print(E_AttCR(E_AttritionYvld, Att_YpredE))

E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)

Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')

E_AttPypl.title("Employee Attrition Matrix Validation")

E_AttPypl.show()

#### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")

print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")

Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data

print(E_AttCR(E_AttritionYtst, Att_YpredE))
```

```
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="DT Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):
   train_sizes, train_scores, val_scores = AttLearnCurve(
      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
   )
   ### Assessing the Mean and Standard Deviation for Data Analysis
   train_mean = E_Attnu.mean(train_scores, axis=1)
   train_std = E_Attnu.std(train_scores, axis=1)
   val_mean = E_Attnu.mean(val_scores, axis=1)
   val_std = E_Attnu.std(val_scores, axis=1)
   ### Visualizing Model Training with a Learning Curve
   E_AttPypl.figure(figsize=(10, 6))
   E_AttPypl.plot(train_sizes, train_mean, label="Training Score", color="darkturquoise", marker='o')
   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="darkturquoise", alpha=0.2)
   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet", marker='o')
   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, color="darkviolet", alpha=0.2)
   E_AttPypl.title(title)
   E_AttPypl.xlabel("Training Set Size")
   E_AttPypl.ylabel("Score")
   E_AttPypl.legend(loc="best")
```

```
    E_AttPypl.grid()
    E_AttPypl.show()
Att_EMod = E_AttDEC(criterion= 'gini', max_depth= 8, min_samples_split= 2) ### Tuning the Model with the Best-Performing Parameters
Emp_learningCurve(
    estimator=Att_EMod,
    E_AttritionX=E_AttritionXtrn,
    E_AttritionY=E_AttritionYtrn,
    title="Learning Curve: Decision Tree for Employee Attrition",
    cv=5
)
```

Multi-Layer Perceptron (MLP) Using Union Features

```
from sklearn.neural_network import MLPClassifier as E_AttMult
Att_Eprm = {'hidden_layer_sizes':[10, 20, 30],'activation': ['relu','tanh','logistic'],'solver':['lbfgs','adam','sgd']}
Att_EMod = E_AttMult(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7), E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttMult(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
```

```
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) #### Testing Model on E_Attrition Data

print(E_AttCR(E_AttritionYtst, Att_YpredE))

E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)

Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')

E_AttPypl.title("Employee Attrition Matrix Testing")

E_AttPypl.show()

#### Assessing FPR and FNR for the E_Attrition Model

EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()

EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)

EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)

print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")

print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")

def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="MLP Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):

   train_sizes, train_scores, val_scores = AttLearnCurve(

      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)

   )

   #### Assessing the Mean and Standard Deviation for Data Analysis

   train_mean = E_Attnu.mean(train_scores, axis=1)

   train_std = E_Attnu.std(train_scores, axis=1)

   val_mean = E_Attnu.mean(val_scores, axis=1)

   val_std = E_Attnu.std(val_scores, axis=1)

   ### Visualizing Model Training with a Learning Curve

   E_AttPypl.figure(figsize=(10, 6))

   E_AttPypl.plot(train_sizes, train_mean, label="Training Score", color="darkturquoise", marker='o')

   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="darkturquoise", alpha=0.2)

   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet", marker='o')

   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, color="darkviolet", alpha=0.2)

   E_AttPypl.title(title)

   E_AttPypl.xlabel("Training Set Size")
```

```
   E_AttPypl.ylabel("Score")
   E_AttPypl.legend(loc="best")
   E_AttPypl.grid()
   E_AttPypl.show()
Att_EMod = E_AttMult(activation= 'relu', hidden_layer_sizes= 20, solver= 'lbfgs') ### Tuning the Model with the Best-Performing Parameters
Emp_learningCurve(
   estimator=Att_EMod,
   E_AttritionX=E_AttritionXtrn,
   E_AttritionY=E_AttritionYtrn,
   title="Learning Curve: Multi Layer Perceptron for Employee Attrition",
   cv=5
)
```

INTERSECTION OF L1 AND RFE FOR E_ATTRITION

Intersection of L1 and RFE(Recursive Feature Elimination) Features

Gradient Boosting Using Intersection Features

```
from sklearn.ensemble import GradientBoostingClassifier as E_AttGrb
Att_Eprm = { 'loss':['log_loss', 'exponential'],'learning_rate': [0.1, 0.3, 0.8],'n_estimators':[100, 140, 150, 200]}
Att_EMod = E_AttGrb(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7), E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttGrb(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
```

```
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
from sklearn.model_selection import learning_curve as AttLearnCurve
import numpy as E_Attnu
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="GB Learning
Curve", cv=5, scoring="accuracy", n_jobs=-1):
   train_sizes, train_scores, val_scores = AttLearnCurve(
      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs,
train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
   )
   ### Assessing the Mean and Standard Deviation for Data Analysis
   train_mean = E_Attnu.mean(train_scores, axis=1)
   train_std = E_Attnu.std(train_scores, axis=1)
   val_mean = E_Attnu.mean(val_scores, axis=1)
   val_std = E_Attnu.std(val_scores, axis=1)
   ### Visualizing Model Training with a Learning Curve
   E_AttPypl.figure(figsize=(10, 6))
   E_AttPypl.plot(train_sizes, train_mean, label="Training Score",
color="darkturquoise", marker='o')
```

```
   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="darkturquoise", alpha=0.2)

   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet", marker='o')

   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std, color="darkviolet", alpha=0.2)

   E_AttPypl.title(title)

   E_AttPypl.xlabel("Training Set Size")

   E_AttPypl.ylabel("Score")

   E_AttPypl.legend(loc="best")

   E_AttPypl.grid()

   E_AttPypl.show()

Att_EMod = E_AttGrb(learning_rate= 0.1, loss= 'log_loss', n_estimators= 200) ### Tuning the Model with the Best-Performing Parameters

Emp_learningCurve(

   estimator=Att_EMod,

   E_AttritionX=E_AttritionXtrn,

   E_AttritionY=E_AttritionYtrn,

   title="Learning Curve: Gradient Boosting for Employee Attrition",

   cv=5

)
```

Hist.Gradient Boosting Using Intersection Features

```
from sklearn.ensemble import HistGradientBoostingClassifier as E_AttHisGrb

Att_Eprm = {'learning_rate': [0.1, 0.3, 0.8],'max_iter':[100,200,350],'max_depth':[2,5,8,9]}

Att_EMod = E_AttHisGrb(random_state=7)

Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)

Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7), E_AttritionYtrn.sample(400, random_state=7))

print(Att_EMod.best_params_)

Att_EMod = E_AttHisGrb(**Att_EMod.best_params_)

Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data

Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data

print(E_AttCR(E_AttritionYvld, Att_YpredE))

E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
```

```
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="HistGB Learning
Curve", cv=5, scoring="accuracy", n_jobs=-1):
  train_sizes, train_scores, val_scores = AttLearnCurve(
    estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs,
train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
  )
  ### Assessing the Mean and Standard Deviation for Data Analysis
  train_mean = E_Attnu.mean(train_scores, axis=1)
  train_std = E_Attnu.std(train_scores, axis=1)
  val_mean = E_Attnu.mean(val_scores, axis=1)
  val_std = E_Attnu.std(val_scores, axis=1)
  ### Visualizing Model Training with a Learning Curve
  E_AttPypl.figure(figsize=(10, 6))
```

```
    E_AttPypl.plot(train_sizes, train_mean, label="Training Score",
color="darkturquoise", marker='o')
    E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
color="darkturquoise", alpha=0.2)
    E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet",
marker='o')
    E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std,
color="darkviolet", alpha=0.2)
    E_AttPypl.title(title)
    E_AttPypl.xlabel("Training Set Size")
    E_AttPypl.ylabel("Score")
    E_AttPypl.legend(loc="best")
    E_AttPypl.grid()
    E_AttPypl.show()
Att_EMod = E_AttHisGrb(learning_rate= 0.1, max_depth= 5, max_iter= 100) ### Tuning
the Model with the Best-Performing Parameters
Emp_learningCurve(
    estimator=Att_EMod,
    E_AttritionX=E_AttritionXtrn,
    E_AttritionY=E_AttritionYtrn,
    title="Learning Curve: Histogram based Gradient Boosting for Employee Attrition",
    cv=5
)
```

Extreme Gradient Boosting Using Intersection Features

```
from xgboost import XGBClassifier as E_AttExGrb
Att_Eprm = { 'max_depth':[2,5,7,9],'learning_rate': [0.1, 0.3, 0.8],'n_estimators':[100,
140, 150, 200]}
Att_EMod = E_AttExGrb(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
Att_EMod = E_AttExGrb(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition
Data
```

```
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="XGB Learning
Curve", cv=5, scoring="accuracy", n_jobs=-1):
  train_sizes, train_scores, val_scores = AttLearnCurve(
    estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs,
train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
  )
  ### Assessing the Mean and Standard Deviation for Data Analysis
  train_mean = E_Attnu.mean(train_scores, axis=1)
  train_std = E_Attnu.std(train_scores, axis=1)
  val_mean = E_Attnu.mean(val_scores, axis=1)
  val_std = E_Attnu.std(val_scores, axis=1)
```

```
   ### Visualizing Model Training with a Learning Curve

   E_AttPypl.figure(figsize=(10, 6))

   E_AttPypl.plot(train_sizes, train_mean, label="Training Score",
color="darkturquoise", marker='o')

   E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
color="darkturquoise", alpha=0.2)

   E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet",
marker='o')

   E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std,
color="darkviolet", alpha=0.2)

   E_AttPypl.title(title)

   E_AttPypl.xlabel("Training Set Size")

   E_AttPypl.ylabel("Score")

   E_AttPypl.legend(loc="best")

   E_AttPypl.grid()

   E_AttPypl.show()

Att_EMod = E_AttExGrb(learning_rate= 0.1, max_depth= 7, n_estimators= 100) ###
Tuning the Model with the Best-Performing Parameters

Emp_learningCurve(

   estimator=Att_EMod,

   E_AttritionX=E_AttritionXtrn,

   E_AttritionY=E_AttritionYtrn,

   title="Learning Curve: Extreme Gradient Boosting for Employee Attrition",

   cv=5

)
```

Decision Tree Using Intersection Features

```
from sklearn.tree import DecisionTreeClassifier as E_AttDEC

Att_Eprm = {'criterion':['gini', 'entropy', 'log_loss'],'max_depth': [10, 5, 8,
3],'min_samples_split':[2, 4, 5, 7]}

Att_EMod = E_AttDEC(random_state=7)

Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)

Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))

print(Att_EMod.best_params_)

Att_EMod = E_AttDEC(**Att_EMod.best_params_)

Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
```

```
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="DT Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):
    train_sizes, train_scores, val_scores = AttLearnCurve(
        estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
    )
    ### Assessing the Mean and Standard Deviation for Data Analysis
    train_mean = E_Attnu.mean(train_scores, axis=1)
    train_std = E_Attnu.std(train_scores, axis=1)
```

```
    val_mean = E_Attnu.mean(val_scores, axis=1)
    val_std = E_Attnu.std(val_scores, axis=1)
    ### Visualizing Model Training with a Learning Curve
    E_AttPypl.figure(figsize=(10, 6))
    E_AttPypl.plot(train_sizes, train_mean, label="Training Score",
color="darkturquoise", marker='o')
    E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
color="darkturquoise", alpha=0.2)
    E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet",
marker='o')
    E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std,
color="darkviolet", alpha=0.2)
    E_AttPypl.title(title)
    E_AttPypl.xlabel("Training Set Size")
    E_AttPypl.ylabel("Score")
    E_AttPypl.legend(loc="best")
    E_AttPypl.grid()
    E_AttPypl.show()
Att_EMod = E_AttDEC(criterion= 'entropy', max_depth= 10, min_samples_split= 2) ###
Tuning the Model with the Best-Performing Parameters
Emp_learningCurve(
    estimator=Att_EMod,
    E_AttritionX=E_AttritionXtrn,
    E_AttritionY=E_AttritionYtrn,
    title="Learning Curve: Decision Tree for Employee Attrition",
    cv=5
)
```

Multi-Layer Perceptron (MLP) Using Intersection Features

```
from sklearn.neural_network import MLPClassifier as E_AttMult
Att_Eprm = {'hidden_layer_sizes':[10, 20, 30],'activation':
['relu','tanh','logistic'],'solver':['lbfgs','adam','sgd']}
Att_EMod = E_AttMult(random_state=7)
Att_EMod = E_AttGsss(Att_EMod, Att_Eprm, cv=2)
Att_EMod.fit(E_AttritionXtrn.sample(400, random_state=7),
E_AttritionYtrn.sample(400, random_state=7))
print(Att_EMod.best_params_)
```

```
Att_EMod = E_AttMult(**Att_EMod.best_params_)
Att_EMod.fit(E_AttritionXtrn, E_AttritionYtrn) ### Training Model on E_Attrition Data
Att_YpredE = Att_EMod.predict(E_AttritionXvld) ### Validating Model on E_Attrition Data
print(E_AttCR(E_AttritionYvld, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYvld, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Validation")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Validation E_Attrition: {EAtt_flsprt}")
print(f"FNR for Validation E_Attrition: {EAtt_flsnrt}")
Att_YpredE = Att_EMod.predict(E_AttritiontXtst) ### Testing Model on E_Attrition Data
print(E_AttCR(E_AttritionYtst, Att_YpredE))
E_MxxAtt = E_AttMx(E_AttritionYtst, Att_YpredE)
Att_EMDS(confusion_matrix = E_MxxAtt, display_labels = [0, 1]).plot(cmap='BuGn')
E_AttPypl.title("Employee Attrition Matrix Testing")
E_AttPypl.show()
#### Assessing FPR and FNR for the E_Attrition Model
EAtt_truN, EAtt_flsP, EAtt_flsN, EAtt_truP = E_MxxAtt.ravel()
EAtt_flsprt = EAtt_flsP / (EAtt_flsP + EAtt_truN)
EAtt_flsnrt = EAtt_flsN / (EAtt_flsN + EAtt_truP)
print(f"FPR for Testing E_Attrition: {EAtt_flsprt}")
print(f"FNR for Testing E_Attrition: {EAtt_flsnrt}")
def Emp_learningCurve(estimator, E_AttritionX, E_AttritionY, title="MLP Learning Curve", cv=5, scoring="accuracy", n_jobs=-1):
   train_sizes, train_scores, val_scores = AttLearnCurve(
      estimator, E_AttritionX, E_AttritionY, cv=cv, scoring=scoring, n_jobs=n_jobs, train_sizes=E_Attnu.linspace(0.1, 1.0, 10)
   )
   ### Assessing the Mean and Standard Deviation for Data Analysis
```

```
    train_mean = E_Attnu.mean(train_scores, axis=1)
    train_std = E_Attnu.std(train_scores, axis=1)
    val_mean = E_Attnu.mean(val_scores, axis=1)
    val_std = E_Attnu.std(val_scores, axis=1)
    ### Visualizing Model Training with a Learning Curve
    E_AttPypl.figure(figsize=(10, 6))
    E_AttPypl.plot(train_sizes, train_mean, label="Training Score",
color="darkturquoise", marker='o')
    E_AttPypl.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
color="darkturquoise", alpha=0.2)
    E_AttPypl.plot(train_sizes, val_mean, label="Validation Score", color="darkviolet",
marker='o')
    E_AttPypl.fill_between(train_sizes, val_mean - val_std, val_mean + val_std,
color="darkviolet", alpha=0.2)
    E_AttPypl.title(title)
    E_AttPypl.xlabel("Training Set Size")
    E_AttPypl.ylabel("Score")
    E_AttPypl.legend(loc="best")
    E_AttPypl.grid()
    E_AttPypl.show()
Att_EMod = E_AttMult(activation= 'relu', hidden_layer_sizes= 10, solver= 'lbfgs') ###
Tuning the Model with the Best-Performing Parameters
Emp_learningCurve(
    estimator=Att_EMod,
    E_AttritionX=E_AttritionXtrn,
    E_AttritionY=E_AttritionYtrn,
    title="Learning Curve: Multi Layer Perceptron for Employee Attrition",
    cv=5
)
```