Credit Card Fraud Detection Using Machine Learning Pipeline with Transformers and Ensemble Learning Estimator

Amogh Ramagiri
Department of Computer Science,
20201CAI0114, Presidency University,
Bangalore, Karnataka, India

Pramod Krishnachari
Department of Computer Science,
20201CAI0099, Presidency University,
Bangalore, Karnataka, India

Abstract
The detection of fraudulent Credit Card Transactions is a critical problem that affects the profitability and sustainability of Finance (Banking) companies. This study focuses on developing an effective Credit Card fraud detection model using supervised machine learning algorithms with ensemble learning methods. The objective is to build a model that can accurately identify fraudulent Credit Card Transactions while minimizing false positives. The model employs various supervised Machine Learning algorithms such as Logistic Regression, Support Vector Machine, Decision Trees, Random Forest, and Ensemble Learning Methods such as Bagging and Boosting. This study uses a real-world Credit Card Fraud Transactions dataset to evaluate using metrics such as accuracy, precision, recall, and F1-Score. The results demonstrate that the proposed model performs better than individual Machine Learning Algorithms. This study contributes to the development of effective fraud detection models that can assist Financial companies in detecting fraudulent Transactions, improving profitability and sustainability, and safeguarding the interests of honest Credit Card Holders.

Keywords:

1. INTRODUCTION

Credit card fraud is a serious concern that occurs when unauthorized individuals gain access to our information and use it for financial gain. The various ways fraudsters obtain our information include lost or stolen credit cards, skimming at gas stations, hacking into computers, and phishing attempts through fake emails. They may also retrieve sensitive information through keystroke capturing or by stealing our mail. To avoid becoming a victim of credit card fraud, it is essential to take preventive measures. These include being cautious while using credit cards and avoiding swiping them through suspicious machines, using chip-based credit cards, shredding bills and documents containing sensitive information, being vigilant while replying to emails and

SMS, using virtual keyboards while entering personal information, and having reliable antivirus software to protect our system. In addition, it is crucial to keep track of when our IDs and other documents are used and to be cautious while performing online transactions, avoiding suspicious websites and refraining from clicking on every link received. By taking these precautions, we can protect ourselves from falling prey to the various types of credit card fraud. Despite all these precautions people fall prey to credit card frauds or con artists somehow manage to fraud common people.[1][2]

1.1 Significance

Credit card fraud detection is crucial in the era of digital payments as it helps prevent fraudulent activities and ensures secure transactions. As more and more consumers are using credit and debit cards for payments, merchants must implement fraud detection systems to protect their client's financial data. Machine learning models offer faster detection, higher accuracy, and improved efficiency with larger datasets, making them a valuable tool in the fight against credit card fraud. By understanding fraudulent patterns and behaviors, financial institutions can develop effective solutions to mitigate the risk of fraud and maintain the trust of their customers[3][4][5].

1.2 Types of Algorithms:

Supervised Machine Learning is a type of machine learning where labeled data is used to train algorithms to accurately classify or predict outcomes. The algorithm is fed a known set of input data and corresponding output data (known as training dataset) to learn how to generate a model for the prediction of new data. The model is fitted through a process called cross-validation, and the algorithm adjusts its weights until the error has been minimized. Supervised learning can solve a variety of problems at scale, such as spam classification in emails. It can be divided into two types of problems: classification and regression. Classification algorithms accurately assign test data to specific categories, while regression algorithms are used to understand the relationship between dependent and independent variables to make predictions. Some common Supervised learning algorithms include[6]:

- **Logistic Regression**
- **Decision Tree**
- **Random Forest**

2. CLASSIFICATION ALGORITHMS

2.1 Logistic Regression

Logistic regression is a supervised learning algorithm that is used for classification problems where the dependent variable is binary or categorical in nature. The model describes the relationship between the input features and the binary output variable using the sigmoid function. Logistic regression predicts the probability that a given input belongs to a specific category, and this probability is then used to make a binary classification decision by applying a threshold. Logistic regression is easy to

implement and achieves good performance in linearly separable classification problems. It is widely used for classification tasks in various industries, and sci-kit-learn provides a highly optimized implementation of the logistic regression algorithm that supports multiclass classification[6][7][8]. The Linear Regression Model is a linear line in which the dependent variables are obtained with a binomial distribution, it is expressed as follows: [16]

$$\pi(x) = \frac{\alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p}{1 + e^{\alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p}}$$

Where:
α Denote the intercept
β$_1$ to **β**□ Represent the coefficient for each input variable
x$_1$ to **x**□ Represent the individual input variables

2.2 Decision Tree Classifier

A Decision Tree Classifier is a machine-learning algorithm that uses Decision trees for predictive modeling. It is a nonparametric and supervised learning method used for classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. The algorithm conducts a top-down, recursive search to identify the optimal split classified under specific class labels. Decision tree learners can be simple to understand and interpret and require little data preparation. However, they can create over-complex trees that do not generalize the data well, resulting in overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node, or setting the maximum depth of the tree, are necessary to avoid this problem. Decision tree learners can also create biased trees if some classes dominate, making it important to balance the dataset prior to fitting with the decision tree. [10][11][16]

Initially, the entropy is calculated for 1 attribute, as in the below equation:

$$H(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Then, the entropy is calculated for different dependent features entropies, as in the below equations:

$$H(X_k) = -\sum_{i=1}^{n} \frac{|S_i|}{|X_k|} \log \frac{|S_i|}{|X_k|}$$

$$H(X,S) = -\sum_{k=1}^{n} \frac{|X_k|}{|X|} H(X_k)$$

Finally, the entropy is calculated for several attributes, as in the below equation:

$$IG(X,S) = H(S) - H(X,S)$$

These equations were defined by Abdullah Elen.[12][13]

2.3 Random Forest

Random forest is a popular ensemble learning method in supervised machine learning. It is used for both classification and regression tasks. The algorithm creates multiple decision trees, each based on a random subset of training data and a random subset of the input features. These decision trees are then combined to form a “forest” that makes predictions based on the output of the individual trees. The main advantage of the random forest algorithm is that it reduces overfitting and improves the accuracy of predictions compared to a single decision tree. By using a collection of decision trees, the random forest can capture more complex relationships between the input features and the target variable. Another advantage of the random forest algorithm is that it can handle missing data and outliers in the input features. It also provides estimates of feature importance, which can be useful in feature selection. One of the drawbacks of random forests is that they can be computationally expensive and may require a lot of memory to train. Additionally, the resulting model can be difficult to interpret compared to a single decision tree. Random forest is widely used in various applications such as image classification, speech recognition, and bioinformatics. It is implemented in many popular machine learning libraries such as sci-kit Learn and TensorFlow.

For classification data, we often use Gini Index used to decide the number of nodes on a decision tree branch.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

We can also use entropy to determine the arrangement of nodes:

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

2.4 Ensemble Learning:

Ensemble learning is a powerful technique in machine learning that involves combining multiple models to improve predictive performance. It is a broad category of algorithms that includes methods such as bagging, boosting, and stacking. In Bagging, multiple models are trained on different subsets of training data, and their predictions are combined to produce the final result. This approach helps to reduce overfitting and increase model stability. Boosting is another popular ensemble method, which involves

training multiple weak learners sequentially. Each subsequent model focuses on improving the mistakes of the previous model. This leads to a strong learner that is better able to generalize the new data. Stacking is a more advanced ensemble method that involves combing the predictions of multiple models using a meta-model. The meta-model is trained on the outputs of the base models and learns to combine their predictions in a way that optimized that optimizes performance. In our research, we use AdaBoost and XGBoost which have gained popularity recently due to their high predictive accuracy and robustness. Hyperparameter tuning using randomized search is often used to optimize the performance of these models. By searching over a range of hyperparameters, the randomized search can help to find the optimal configuration of a model, leading to better performance. [14][17][18]

2.4.1 AdaBoost

Adaboost(Adaptive Boosting) is an ensemble learning algorithm that can be used for both classification and regression problems. It was proposed by Yoav Freund and Robert Schapire in 1996. The basic idea behind Adaboost is to combine several weak learners(classifiers with accuracy only slightly better than random guessing) to form a strong classifier with high accuracy. This algorithm works by sequentially training a set of weak classifiers on weighted versions of the training data. In each iteration, the weights of misclassified samples are increased and the weight of correctly classified samples is decreased. This allows subsequent weak classifiers to focus on the previously misclassified samples. Finally, the output of all weak classifiers is combined using a weighted majority vote to obtain the final prediction. Adaboost has several advantages over other classification algorithms. It is simple to implement, has high accuracy, and can handle complex data sets with high dimensionality. Additionally, Adaboost can be used with a variety of weak classifiers, including decision trees, neural networks, and support vector machines. One potential drawback of Adaboost is that it is sensitive to noisy data and outliers. However, this can be addressed by using robust weak classifiers or by preprocessing the data to remove the outliers and the noise in the data. Another limitation of Adaboost is that it can be computationally expensive, especially when training a large number of weak classifiers. Overall, Adaboost is a powerful ensemble learning algorithm that has been successfully applied to a wide range of classification problems in various domains, including image classification, face recognition, and bioinformatics.[15][18][19]

2.4.2 XGBoost

XGBoost is an optimized and scalable gradient-boosting library that is used for solving data science problems, including classification, regression, and ranking tasks. It was developed by Tianqu Chen in C++ and is known for its computational speed and accuracy. The XGBoost algorithm uses a gradient-boosting framework that trains and combines multiple decision trees to make predictions. It starts by building a simple decision tree model and then iteratively adds more trees, with each tree attempting to

correct the errors made by the previous trees. The final model is a weighted combination of these trees. One key advantage of XGBoost is that it allows for parallel processing and can handle large datasets with millions of examples and thousands of features. It also provides regularization techniques to prevent over fittings, such as L1 and L2 regularization. Hyperparameter tuning is an important step in optimizing the performance of XGBoost models. The most common hyperparameters tweaked include the learning rate, maximum depth of trees, number of trees, and the subsample ratio of training instances. Randomized search is often used to efficiently search through the hyperparameter space to find the optimal values.[20][21][22]

3. MACHINE LEARNING PIPELINE

Machine learning pipelining is a systematic approach to building and deploying a machine learning model. It consists of several stages, including Data collection, Data preprocessing, feature engineering, model training, model evaluation, and model deployment.

3.1 Data Collection:

Data collection is the first step in the machine learning pipeline, and it involves gathering data from various sources. The type of data collected can vary depending on the problem being solved. In some cases, the data may be structured and stored in a database, while in other cases it may be unstructured and come from sources such as sensors, social media, or web scraping. Once the data is collected, it needs to be cleaned and preprocessed to remove any irrelevant or noisy information. This stage is critical as the quality of the data will directly impact the performance of the final model. The primary goal of this step is to gather high-quality data that is representative of the problem being solved and is suitable for training machine learning models.

3.1.1 Methods of Data Collection:

Data collection can be performed using various methods, such as surveys, experiments, observations, and data mining. Surveys involve collecting data by asking questions to a sample of individuals or groups. Experiments involve manipulating variables to observe their effect on the outcome of interest. Observation involves collecting data by observing events or behaviors. Data mining involves extracting patterns and knowledge from large datasets using statistical and machine-learning techniques.

3.1.2 Challenges faced:

Data collection can be challenging due to various factors such as the size and complexity of the data, privacy concerns, and biases in the data. To ensure high-quality data, it is essential to define clear research questions, establish appropriate data collection protocols, and carefully select the sample population

3.2 Data Preprocessing:

Data preprocessing is the stage where the collected data is cleaned and transformed into a format suitable for machine learning algorithms. This stage includes tasks such as removing missing values, scaling the data, and encoding categorical variables. Data preprocessing can also involve handling missing data and dealing with class imbalances. The goal is to create a dataset that is consistent, accurate, and complete. It is a crucial step in the machine learning pipeline as it can greatly impact the accuracy and efficiency of the model. By performing data preprocessing effectively, we can ensure that the machine learning model is trained on high-quality data and is capable of making accurate predictions on new and transformed data. Data preprocessing involves tasks such as data cleaning, data transformation, and feature selection.

3.2.1 Data cleaning:

Data cleaning involves handling missing values, dealing with outliers, and removing irrelevant data. Missing values can be imputed or removed depending on the amount of missing data and the nature of the problem being solved. Outliers can be detected and removed using statistical methods such as z-score or interquartile range. Irrelevant data can be removed by selecting only the relevant variables or by filtering out observations that do not contribute to the problem being solved.

3.2.2 Data transformation:

Data transformation involves transforming the data into a format that is more suitable for machine learning algorithms. This can include tasks such as scaling, normalization, and encoding categorical variables. Scaling involves transforming the data to a common scale to avoid bias toward variables with larger values. Normalization involves transforming the data to have Normal distribution. Encoding categorical variables involves transforming categorical variables into a numerical format that can be preprocessed by machine learning algorithms.

3.2.3 Feature Selection:

Feature selection involves selecting the most important variables that are relevant to the problem being solved. This can be done using statistical methods such as correlation analysis, or by using machine learning algorithms such as Decision trees or LASSO. Feature selection can help reduce the dimensionality of the dataset. Which can improve the performance of the model and reduce the risk of overfitting.

3.3 Feature Engineering:

Feature engineering is a critical step in the machine learning pipeline as it involves transforming raw data into features that can be used to train machine learning models effectively. It is the process of selecting or creating features that are relevant to the problem being solved. This stage involves selecting the most important variables or

transforming the data into a more meaningful representation that can improve the performance of the model. By carefully selecting and transforming features, we can ensure that machine learning models are trained on representative data and are capable of making accurate predictions on new data. Feature engineering is a critical step as it can greatly impact the accuracy and efficiency of the model.

3.3.1 Types of Features:

Features can be broadly classified into three types: Numerical features, categorical features, and text features. Numerical features are features that represent continuous or discrete values, such as age, income, or height. Categorical features are features that represent non-numeric values, such as gender, occupation, or country of origin. Text features are features that represent textual data, such as customer reviews or product descriptions

3.3.2 Techniques for Feature Engineering:

Feature engineering involves several techniques, such as feature scaling, feature extraction, and feature encoding. Feature scaling involves transforming the values of numerical features to a common scale. Feature extraction involves transforming raw data into a set of relevant features, through the use of principal component analysis or clustering. Feature encoding involves converting categorical features into numerical features that can be used in machine learning models.

3.3.3 Challenges in feature engineering:

Feature engineering can be challenging due to various factors such as the high dimensionality of the data, the sparsity of the data, and the presence of noise and outliers in the data. To address these challenges, it is essential to carefully select features and apply appropriate techniques for feature scaling, feature extraction, and feature encoding.

3.4 Model Training:

Model training is the stage where the machine learning algorithm is trained on the preprocessed data to find the optimal set of model parameters that minimize the error between the predicted output and the actual output. There are various algorithms and techniques used for model training such as Regressing, Decision Trees, AdaBoost, and XGBoost with hyperparameter tuning. By carefully selecting the appropriate algorithm, defining the model architecture, tuning hyperparameters, and training the model using an optimization algorithm, we can ensure that the machine learning models are trained effectively and can make accurate predictions on new data.

3.4.1 Data Splitting:

Once the evaluation metrics are selected, the next step is to split the data into training, validation, and test sets. The training set is used to train the model, the

validation set is used to tune hyperparameters and evaluate the performance of the model during training, and the test set is used to evaluate the final performance of the model on unseen data. The split can be performed in different ways, such as using a fixed percentage of the data for each set or using cross-validation techniques.

3.4.2 Algorithms Selection:

The first step in model training is selecting a suitable algorithm that can solve the problem at hand. The choice of algorithm depends on the nature of the data, the type of problem, and the desired output. Some popular algorithms include Logistic Regression, Decision trees, AdaBoost Technique, and Random Forests.

3.4.3 Model Architecture :

Once the algorithm is selected, the next step is to define the model architecture. The model architecture is defined by the set of rules or Decision trees used to make predictions. The Architecture for Linear Regression involves a single output node that computes a weighted sum of the input feature, followed by an activation function. For Decision Trees, the architecture involves a tree-like structure where each node represents a decision based on one of the input features. The tree is constructed by recursively splitting the data into smaller subsets based on the feature values until a stopping criterion is met. The model architecture for Random Forest involves training multiple decision trees on different subsets of the data and averaging their predictions to make the final prediction. For Adaboost it involves combining multiple weak classifiers into a strong classifier. Each weak classifier is trained on a subset of the training data, and the final prediction is made by combining the predictions of all weak classifiers. For XGBoost the architecture involves combining multiple decision trees into a single model, using a combination of additive training and regularization techniques to prevent overfitting and improve performance.

3.4.4 Hyperparamters Selection:

The next step is selecting appropriate hyperparameters for the model. Hyperparameters are parameters that are not learned during training and are set prior to training. These parameters control the behavior of the optimization algorithm used during training, such as the learning rate, the number of iterations, or maximum depth of the tree, minimum child weight, and gamma. The hyperparameters are usually tuned using a validation set to find the best combination of hyperparameters that optimize the performance of the model on the validation set.

3.4.4 Training the model:

Once the hyperparameters are selected, the model is trained using an optimization algorithm, such as GridSearchCV or RandomizedSearchCV, that minimizes a loss function between the predicted output and the true output. During training, the model parameters are updated iteratively using the training data until convergence or a

stopping criterion is reached. The model is then evaluated on a test set to assess its performance on unseen data.

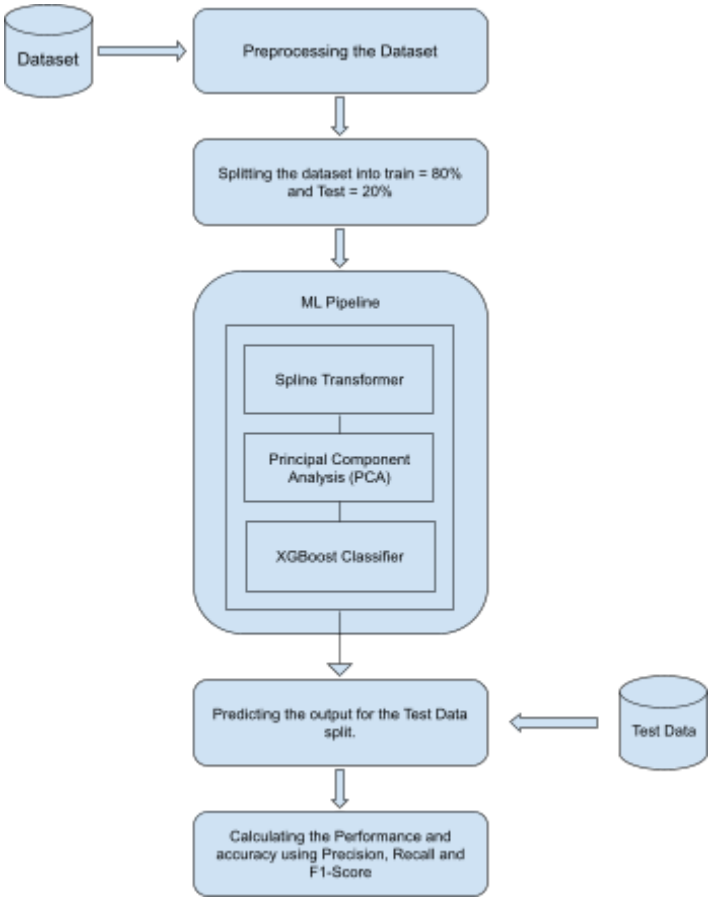
3.5 Model Evaluation:

Model Evaluation is the stage where the trained model is tested on a new set of data to assess its performance. The goal is to determine how well the model can generalize the unseen data. Model Evaluation involves using various metrics such as accuracy, precision, recall, and F1-score, Cross Validation Score, ROC-AUC Score and Learning Curve to measure the performance of the model. The model may need to be retrained and fine-tuned based on the evaluation results. The goal of the model evaluation is to determine how well the model is generalized to new, unseen data and to identify any issues with the model, such as overfitting or underfitting. Model evaluation involves several steps, including selecting appropriate evaluation metrics, splitting the data into training, validation, and test sets, and using the evaluation metrics to assess the performance of the model.

3.5.1 Evaluation Metrics:

The first step in model evaluation is selecting appropriate evaluation metrics that measure the performance of the model on the task at hand. The choice of evaluation metrics depends on the type of problem and the desired output. Some common evaluation metrics for classification tasks include accuracy, precision, recall, and F1-score, and are under the ROC curve.

4. METHODOLOGY



5. Data Pre-Processing

Data Pre-processing is a critical step in any Machine Learning pipeline, as it can have a significant impact on the accuracy and reliability of the model. In this experiment, we explored a range of data pre-processing techniques, including data cleaning, Transformation, Reduction.

5.1 Dataset

This experiment was implemented using the Kaggle dataset of Credit Card Fraud Detection. The dataset consisted of 284807 rows and 31columns. In which, the y_train and y_test was the last column “Class”.

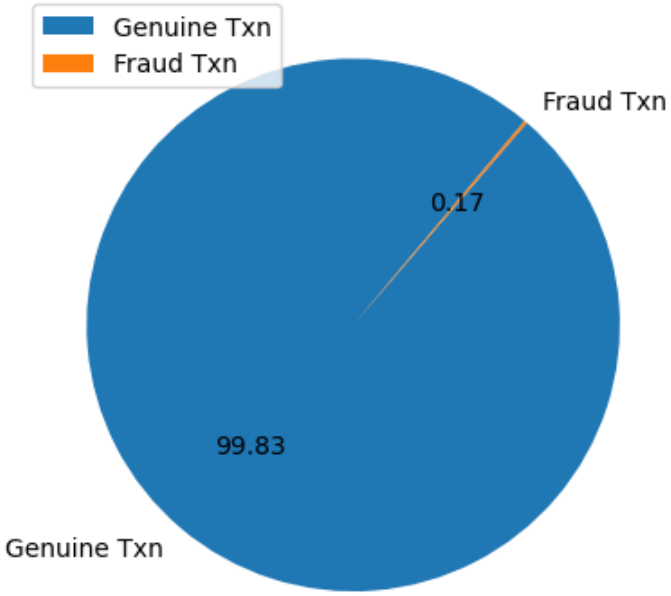
Transformer techniques were applied to the different feature columns due to privacy and security concerns.

Dataset : https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

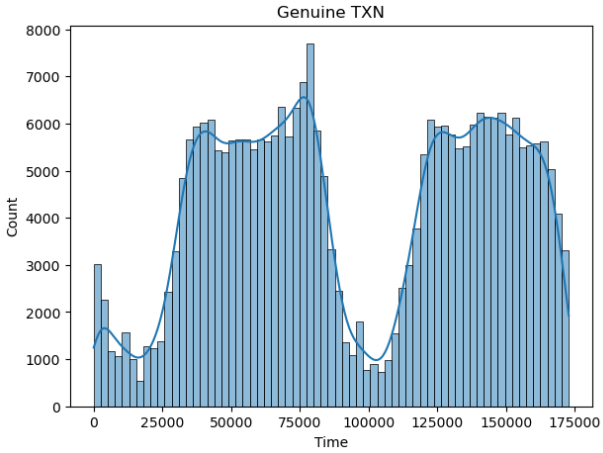
5.2 Data Visualization

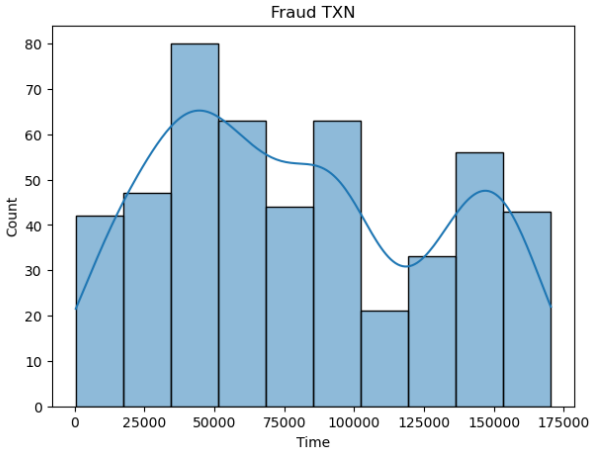
Visualizing the data can help to identify patterns, trends, and outliers in the dataset. This can help to identify potential issues with the data and inform the selection of appropriate machine learning algorithms.

By separating the dataset based on class labels and assigning them to different variables and plotting the counts using pie-chart.



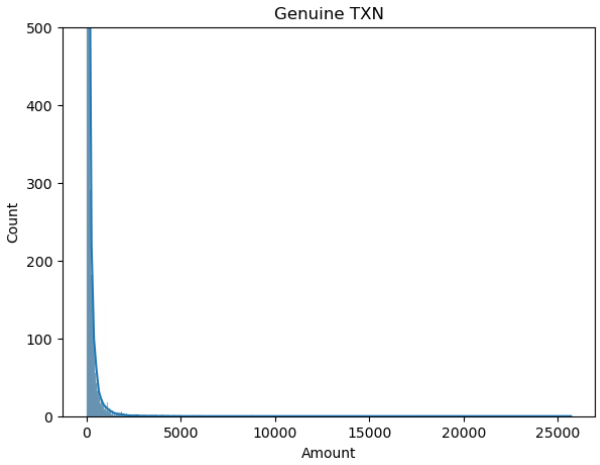
Now, by plotting the time feature on two different class labels. Basically on Genuine Transaction and Fraud Transaction.

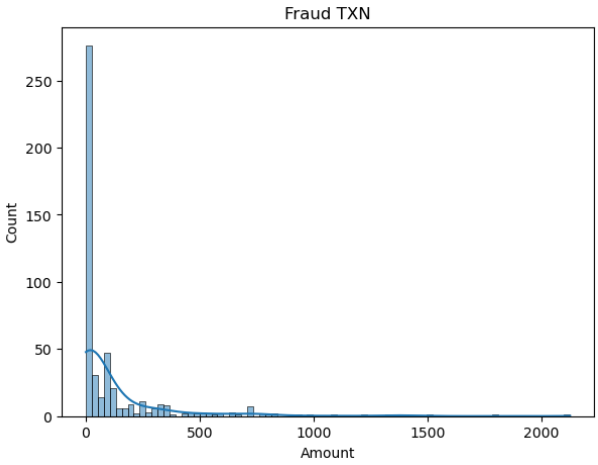




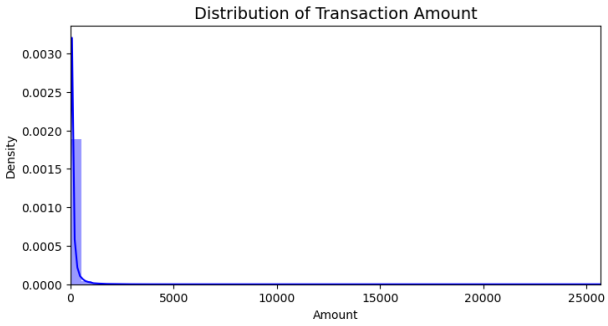
The above graph visulaization shows us that the fraud transaction are in the low amounts **0-400**.

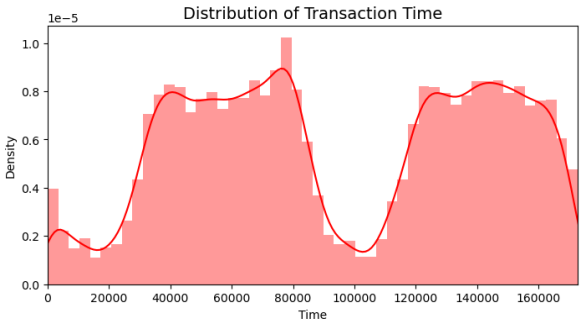
Then, by plotting the transactional amount feature on two different class labels. Mainly on Genuine Transaction and Fraud Transaction.



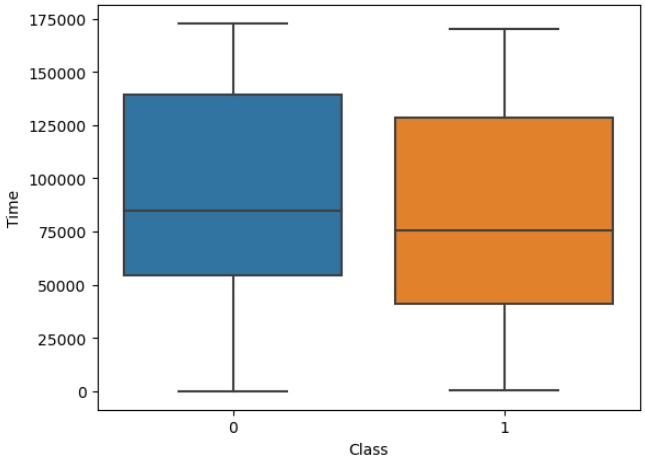


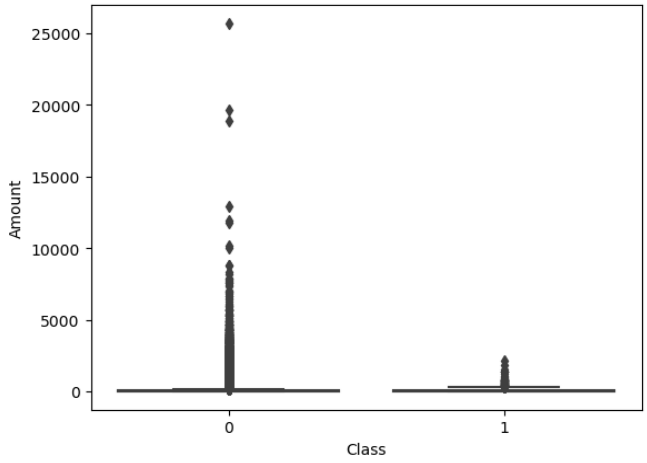
After carful visualization and understanding that the data is imbalanced, we explored the distribution graph. In which, the distribution of transaction Amount and Time was compared.





We performed box plot visualization to find if there is any difference across how the transactions took place based on time and amount.



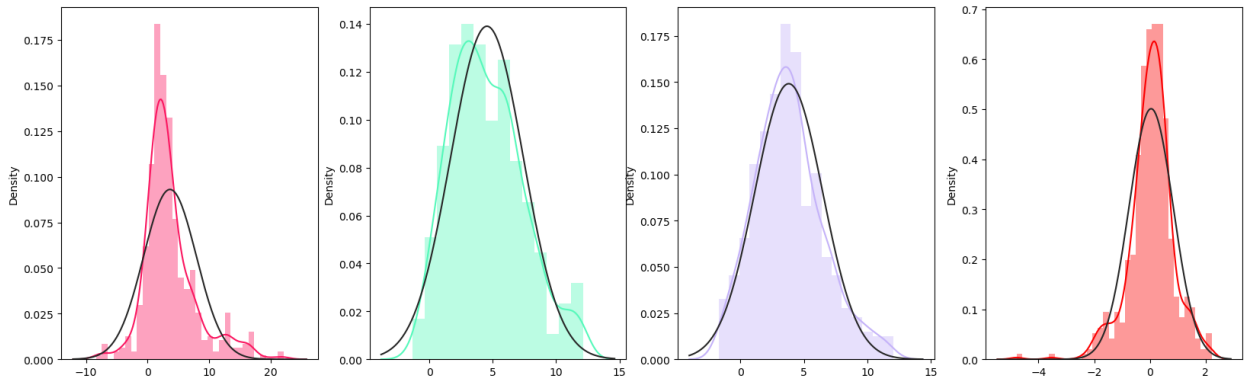


5.3 Data Cleaning

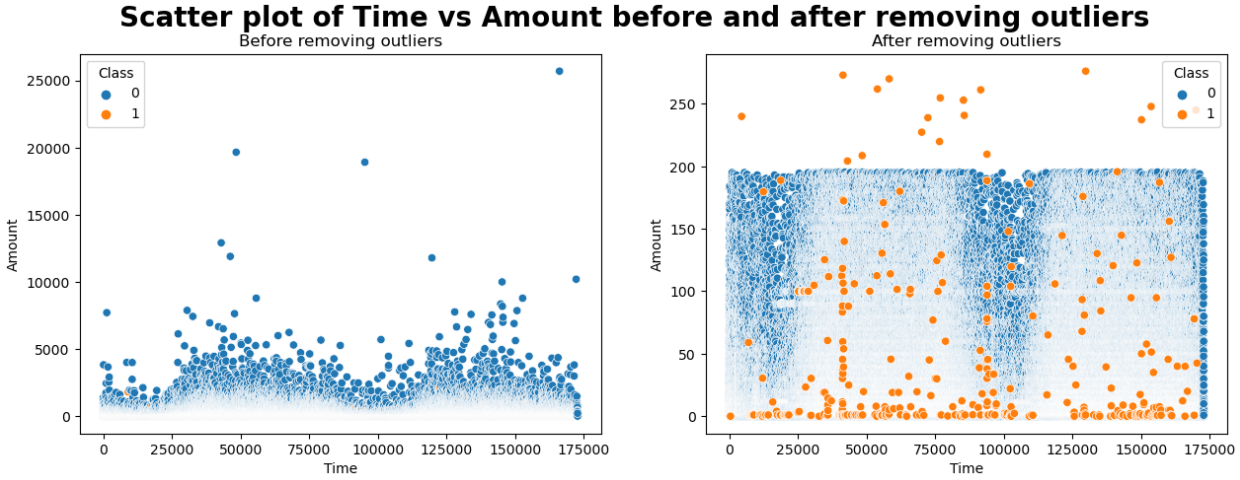
We explored the dataset for any missing values, we found out that there wasn't any missing values since the features was already applied the transformer technique.

The dataset was identified for any duplicates, the data had no duplicates.

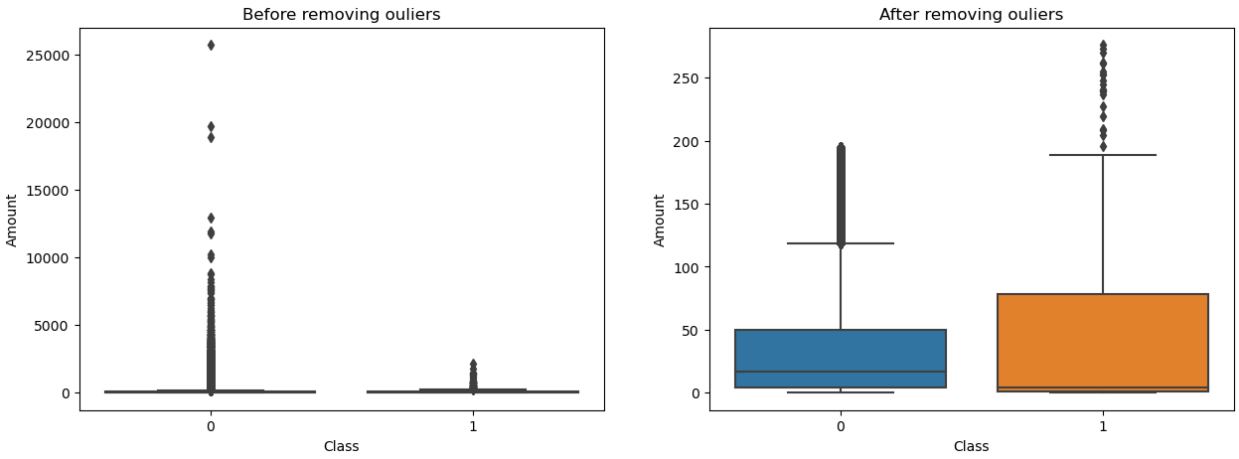
The dataset was identified for having outliers and We visualized few statistics graph of 4 of the features. Which helped us to understand the norm of the dataset features.



we then decided to handle them by finding the upper fence , lower fence (using mean and Interquartile Range) and the count of outliers and then concatenating them to the dataset of both genuine and fraud. We then used scatter plot to visualize the before and after removal of the outliers.



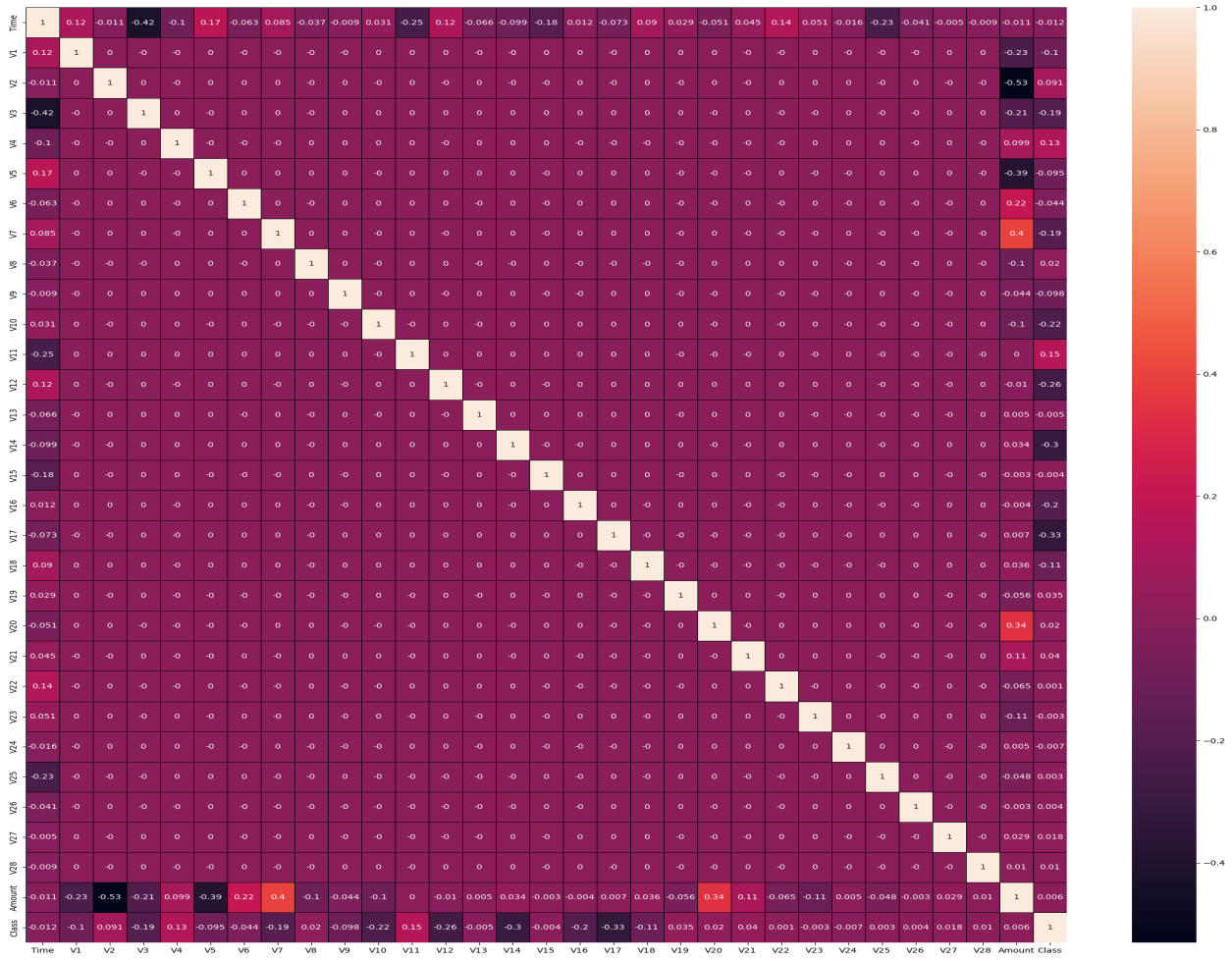
We also visualized the removal of outliers using box-plot.



Now, to understand the relationship between variables in a dataset. Correlation analysis can help us to identify the strength and direction of the relationship between variables.

By performing correlation analysis using heatmap, we can identify variables that are highly correlated with each other. This is important because highly correlated variables can introduce redundancy into the model, which will reduce the accuracy of the model complexity.

It also helps us to identify variables that are not correlated with the target variables. These variables may not be useful for predicting the target variable and can be removed from the dataset to simplify the model.



6. RESULTS

6.1 Performance Evaluation Metrics

The results of this study demonstrate the effectiveness of a well-designed Machine Learning Pipeline in achieving a high-performance model with optimal efficiency. Our experiments show that each step of the pipeline, from data preprocessing to model selection and optimization, plays a critical role in determining the overall performance of the system.

We found that careful feature engineering and data cleaning significantly improved the performance of our models, particularly when dealing with noisy data. Additionally, our experiments show that choosing the right algorithm and hyperparameters for a given problem can greatly enhance model accuracy.

We also evaluated the impact of various performance metrics on the final model selection. While some metrics such as accuracy, precision, F1-Score, and Area Under the Curve (AUC) were more appropriate for imbalance datasets.

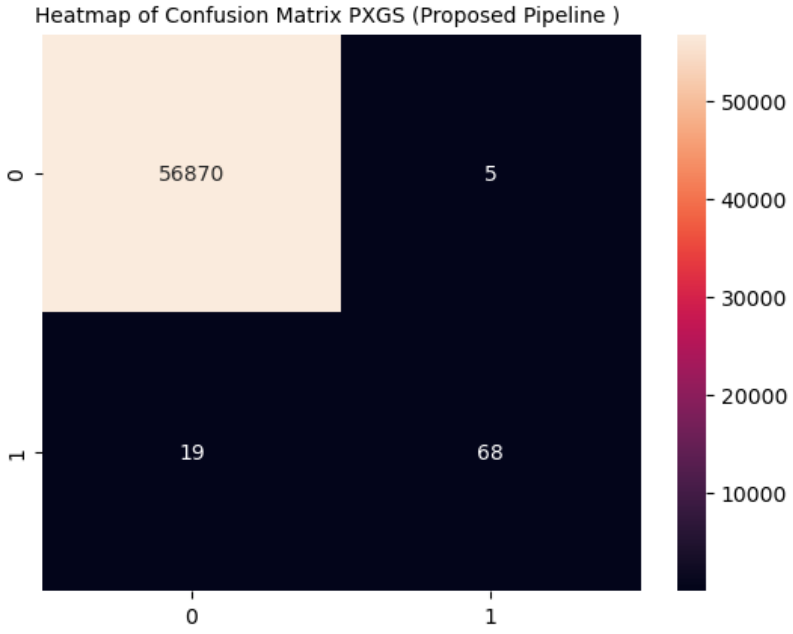
Technique	Accuracy (in %)	Cross-Validation Score (in %)	Precision (in %)	Recall (in %)	F1-Score (in %)	ROC-AUC Score (in %)
***PXGS**	***99.958***	***99.954***	***93.151***	***78.161***	***85***	***97.167***

***PXGS: Pipeline-XGBoost-GridsearchCV**

6.2 Confusion Matrix

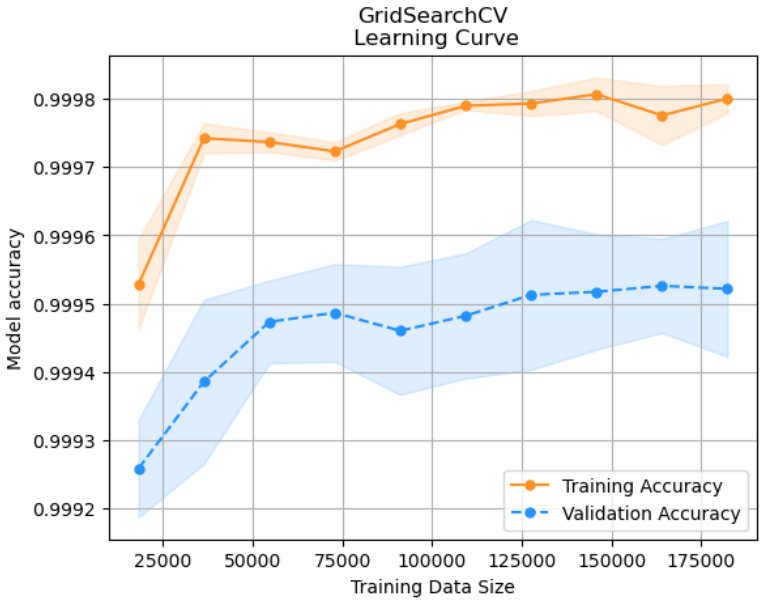
We also conducted experiments on several benchmark datasets to evaluate the effectiveness of the proposed Machine Learning Pipeline model and used the confusion matrix to assess their performance. Our results indicate that the confusion matrix provides a comprehensive and intuitive view of the model's accuracy, precision, recall, and other key metrics.

In particular, we found that the confusion matrix can help identify and diagnose common errors made by the model, such as misclassifying certain classes or confusing similar instances. By examining the false positives and false negatives in the matrix, we were able to identify patterns and trends in the data that could inform feature engineering and model selection.



6.3 Learning Curve

Learning is a powerful tool for analyzing the performance of machine learning models as a function of the size of the training dataset. Our experiments demonstrate the importance of analyzing learning curves in order to optimize model performance, avoid overfitting and reduce training time and costs.



Our results indicate that learning curves can provide valuable insights into the performance of the model and help guide improvements in its design and implementation.

Our experiment highlights the importance of learning curve as a key component of the machine learning evaluation process.

6.4 Comparative Study

6.4.1 Performance Evaluation Metrics

Comparing the performance of different machine learning models is a crucial step in selecting the best approach for a given problem statement. The choice of performance evaluation metric can have a significant impact on the results and conclusions of the comparisons. In our experiments, we evaluated several commonly used metrics for comparative performance evaluation and analyzed their strengths and weaknesses.

Our experiments are focused on two main categories of metrics: accuracy-based and ranking-based. For accuracy-based metrics, we used Accuracy, Precision, Recall, and F1-Score which is basically used for binary classification problems. For ranking-based metrics, we used Receiver Operating Characteristic (ROC) curve which is mostly used for imbalanced data. These evaluation metrics are a simple and intuitive measure of model performance.

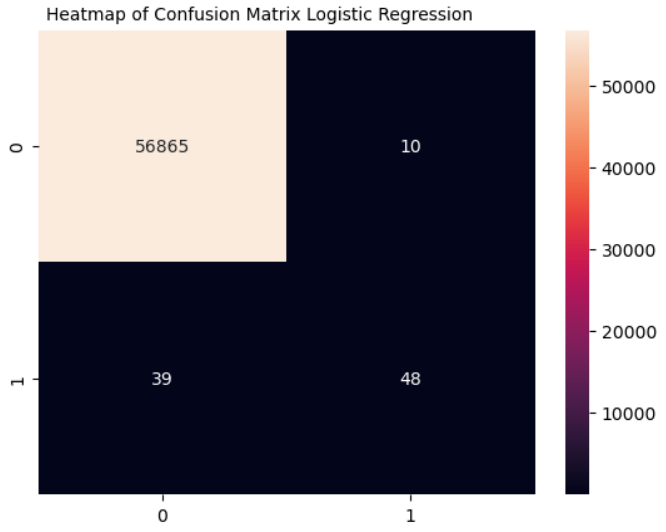
Classifier	Accuracy (in %)	Cross-Validation Score (in %)	Precision (in %)	Recall (in %)	F1-Score (in %)	ROC-AUC Score (in %)
LR	99.914	99.898	82.759	55.172	66.207	49.556
DT	99.916	99.919	72.414	72.414	72.414	87.898
AB (AdaBoost)	99.925	99.92	76.414	73.563	74.854	87.901
RF	99.956	99.953	94.286	75.862	84.076	92.18
PXGS (Proposed)	***99.958***	***99.954***	***93.151***	***78.161***	***85***	***97.167***

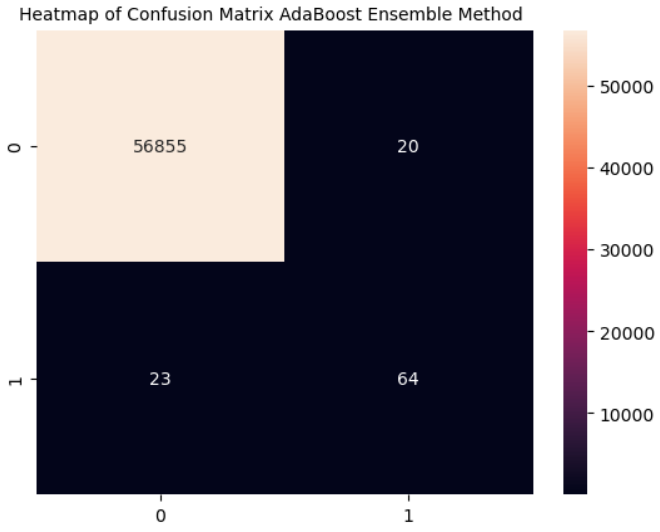
Overall, our experiments show that the proposed methodology **(PXGS)** has higher accuracy, Cross-Validation, Recall, F1, and ROC Scores. We did observe that our precision is lower compared to Random Forest Classifier.

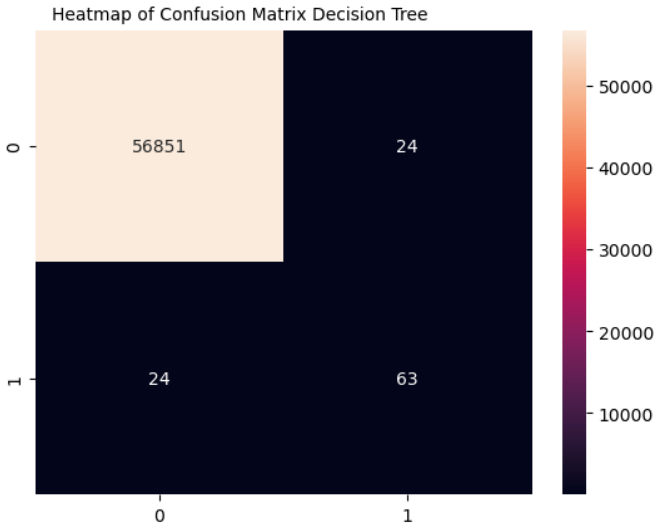
6.4.2 Confusion Matrix

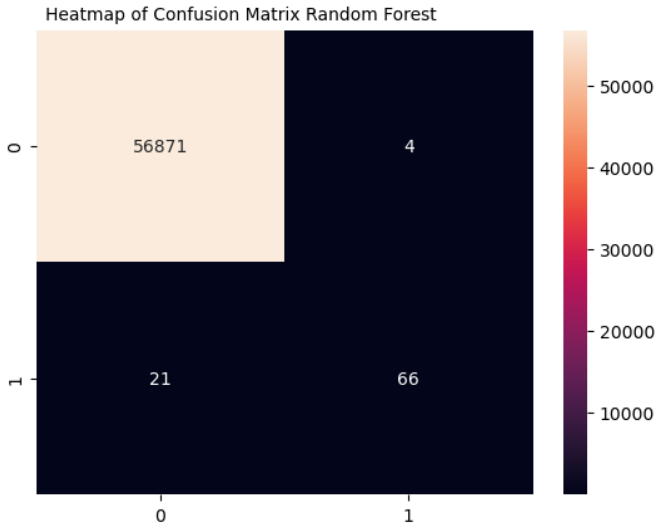
In our experiments, we used confusion matrix to compare the performance of several ML algorithms on a range of binary classification problems, using confusion matrices to compare their accuracy, precision, recall, and F1-score.. Our experiments focused on four different machine learning algorithms.

Our results demonstrates that the choice of algorithm can have a significant impact on the accuracy and reliability of the classification system. In particular,

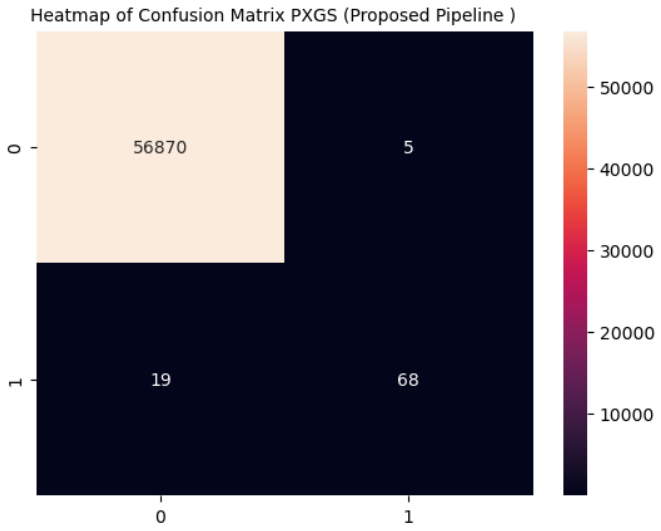








we found that XGBoost with hyperparameter tuning tended to perform better on datasets with complex decision boundaries, while Logistic Regression and AdaBoost were better suited to linearly separable datasets.



Overall, our study highlights the importance of using confusion matrices to compare and evaluate the performance of different ML algorithms on a variety of classification taska.

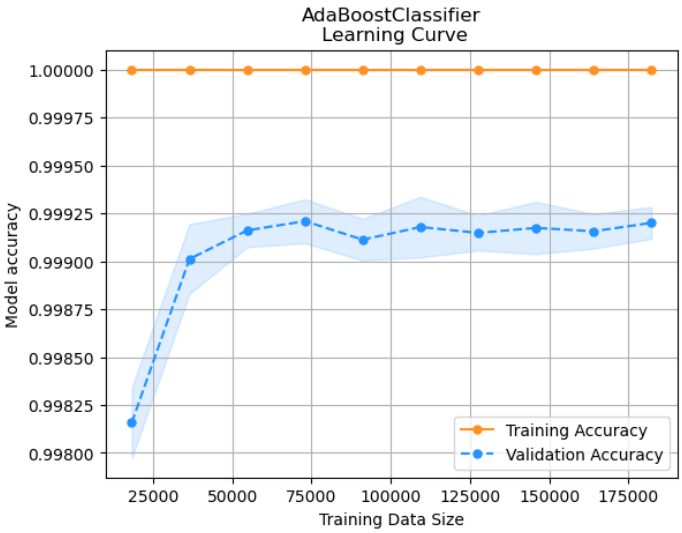
6.4.3 Learning Curve

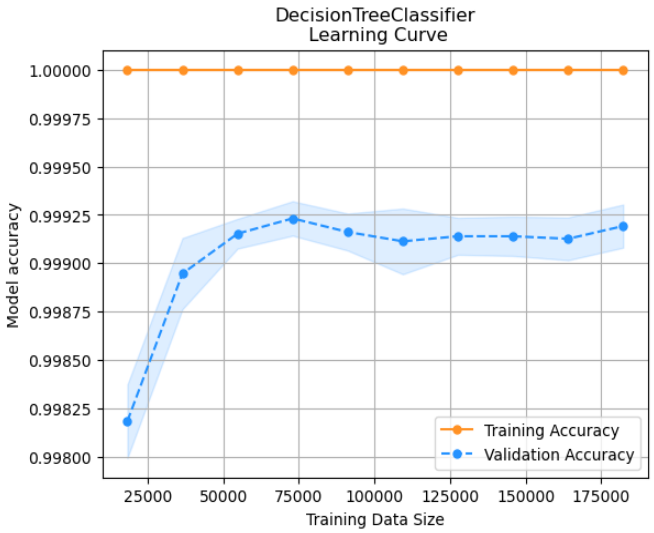
In our experiments, we used learning curves to compare the performance of several machine learning algorithms on a range of benchmark datasets.

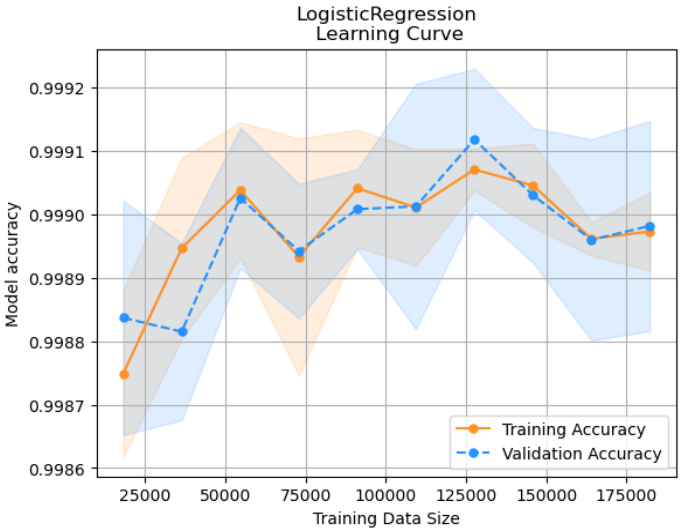
Our experiments focused on four popular machine learning algorithms: Logistic Regression, Decision Tree, AdaBoost, Random Forest. We evaluated the performance of each algorithm on a variety of binary classification problems, using learning curves to analyze the impact of training set size on performance.

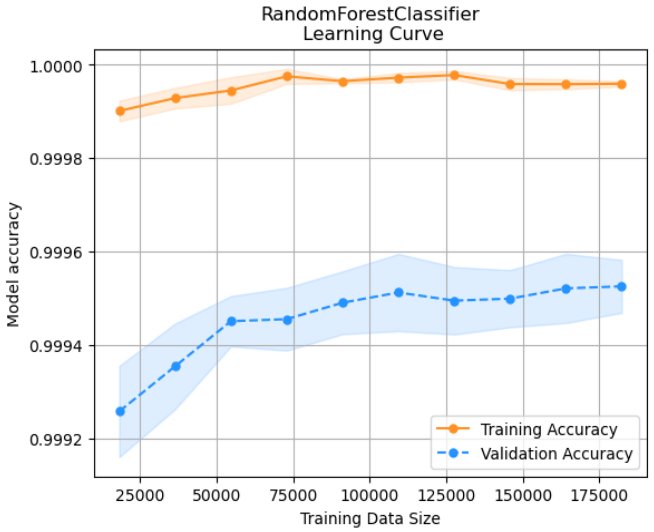
Our results indicate that the shape and convergence of the learning curve can vary significantly between algorithms and datasets. In particular, we found that decision trees and random forest tend to converge more quickly than XGBoost with Hyperparameter tuning, but may suffer from overfitting if the dataset is too small or noisy.

We also observed that the choice of hyperparameters and model architecture can have a significant impact on the shape and convergence of the learning curve.









In conclusionl, Our experiments highlights the importance of using learning curves to compare the performance of different ML algorithms and to optimize their design and implementation. By analyzing the shape and convergence of the learning curve, we were able to identify the optimal size of the training set and to diagnose problems such as overfitting and high variance.

7. CONCLUSIONS AND FUTURE WORK

In this experiment, we have explored the use of machine learning pipelines and algorithms for a variety of classifications tasks and evaluated their performance using a range of comparative and quantitative metrics. Our results demonstrates the importance of careful pipeline design and parameter tuning in order to optimize the accuracy and efficiency of the system.

Our experiments, highlights the need for a comprehensive and iterative approach to machine learning, which involves continuous evaluation and refinement of the pipeline and algorithm over time. We have shown that the use of techniques such as hyperparameter optimization, feature selection, and learning curves can significantly improve the performance of the system and reduce training time and costs.

In addition to this practical implications of our work, there are also a number of avenues for future research in this area. One important area for future work is the development of more efficient and scalable machine learning algorithms that can handle increasingly large and complex datasets. This may involve the use of techniques such as distributed computing, Cloud Computing and deep learning architectures.

Another important area for future work is the development of more robust and interpretable Machine Learning Pipelines, which can provide insight into the underlying data and the decision-making process of the algorithm.

7.1 Classification Report

```
Logistic Regression:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56875
           1       0.83      0.55      0.66        87

    accuracy                           1.00     56962
   macro avg       0.91      0.78      0.83     56962
weighted avg       1.00      1.00      1.00     56962

Decision Tree:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56875
           1       0.72      0.72      0.72        87

    accuracy                           1.00     56962
   macro avg       0.86      0.86      0.86     56962
```

8. REFERENCES

[1]. https://www.cnbc.com/select/credit-card-fraud/

[2].https://www.godigit.com/finance/identity-theft-and-fraud/types-of-credit-card-frauds

[3]. https://rapidcents.com/the-importance-of-credit-card-fraud-detection-in-the-era-of-digital-payments/

[4]. https://arxiv.org/abs/2204.05265

[5].
https://www.infosysbpm.com/blogs/bpm-analytics/machine-learning-for-credit-card-fraud-detection.html

[6]. https://www.ibm.com/topics/supervised-learning

[7]. https://www.sciencedirect.com/topics/computer-science/logistic-regression

[8].https://www.ibm.com/topics/logistic-regression#:~:text=Resources-,What%20is%20logistic%20regression%3F,given%20dataset%20of%20independent%20variables.

[9] Andhavarapu Bhanusri, K.Ratna Sree Valli, P.Jyothi, G.Varun Sai, R.Rohith Sai Subash: Credit card fraud detection using Machine learning algorithms

[10]. https://www.ibm.com/topics/decision-trees

[11]. http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[12]. Abdullah Elen, M. Kamil Turan**:** **"**Classifying White Blood Cells Using Machine Learning Algorithms",2019, 498372, 41-152.

[13] Amogh Ramagiri: "Leukemia cancer cells detection and image classification using machine learning and deep learning: A systematic review

[14] R Natras, B Soja, M Schmidit - Remote Sensing,2022: "Ensemble Machine Learning of Random Forest, AdaBoost and XGBoost for Vertical Total Electron Content Forecasting"

[15] S Demir, EK Sahin - Neural Computing and Applications, 2023 - Springer: "An overview of boosting decision tree algorithms utilizing AdaBoost and XGBoost boosting strategies"

[16] Pratap Chandra Sen, Mahimarnab Hajra, Mitadru Ghosh: "Supervised Classification Algorithms in Machine Learning: A Survey and Review"

[17] Robi Polikar: "Ensemble Learning"

[18] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, Qianli Ma: "A survey on ensemble learning"

[19] Shraboni Rudra, Minhaz Uddin, Mohammed Minhajul Alam: "Forecasting of Breast Cancer and Diabetes Using Ensemble Learning"

[20] Tianqi Chen, Tong He: "xgboost: eXtreme Gradient Boosting"

[21] Tianqi Chen, Carlos Guestrin: "XGBoost: A Scalable Tree Boosting System"

[22] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, Mu Li, Junyuan Xie, Min Lin, Yifeng Geng, Yutian Li: " Package 'xgboost' "