NEOMA BUSINESS SCHOOL

Machine Learning & Artificial Intelligence in Finance

CHALLENGE : FRAUD DETECTION

Realized by :

Hafsa Bouhout

Ikram El Yaziji

Ikram Dannoune

Intro :
Rapid advances in technology and digitization have led to an increase in the use of online financial transactions. These advancements have certainly improved the speed and convenience of transactions, but they also pose the problem of fraudulent activity. Fraud detection in financial transactions is a major challenge for financial institutions, and traditional rule-based systems often struggle to keep up with sophisticated fraud techniques.

In this report, we investigate the effectiveness of various classification algorithms in identifying fraudulent transactions.The main objective is to find an accurate fraud detection model that can help reduce losses caused by fraudulent activities.

We conducted a comprehensive comparative study using several popular machine-learning classification algorithms, including Random Forest, XGBoost, Logistic Regression, and Support Vector Machines (SVM). These algorithms are chosen based on their ability to handle complex data patterns and their wide usability in various machine-learning applications. Our study focused on evaluating the performance of these models on a the data set provided, comparing their accuracy, precision, recall, and F1 scores.

Throughout our analysis, we used rigorous preprocessing techniques, such as resampling to correct class imbalance, hot coding for classification and division features. feature ratio to ensure a fair comparison between models.We also used cross-validation to optimize the model`s hyperparameters and reduce redundancy.

This report is structured as follows: First, we provide an overview of the data set and the preprocessing steps applied to prepare the data for modeling. Then, we present the method and results obtained for each classification algorithm. We then compare the performance metrics of the different models and discuss their strengths and weaknesses in the context of fraud detection. Finally, we conclude with a summary of our findings and recommendations for future work in this area.

Exploring and preprocessing our data:

In order to understand our data, we need to describe it and pre processed it. Our database contains a list of purchases made from a partner. The information exclusively describes the contents of each basket. For each observation, there are 147 columns, of which 144 can be grouped into 6 categories:

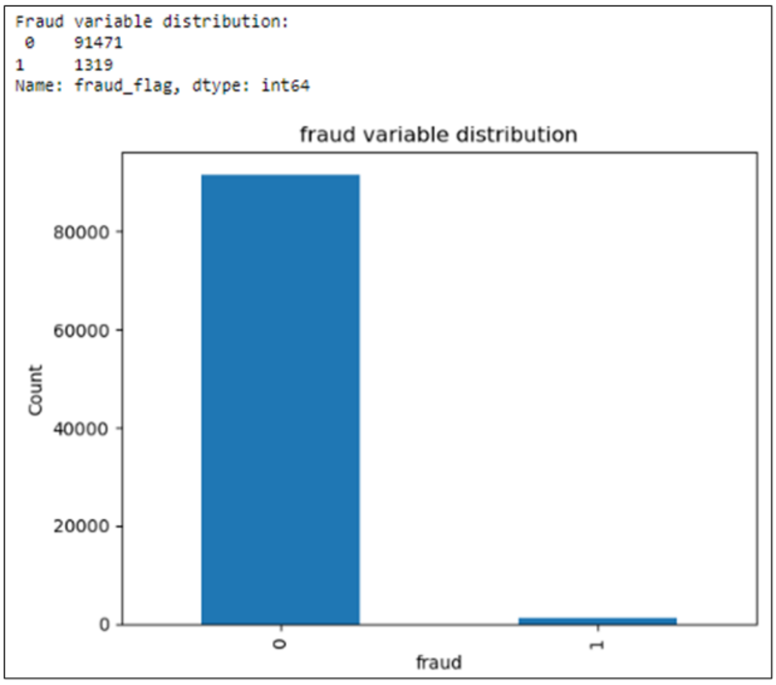
- item,
- cash_price,
- make,
- model,
- goods_code,
- Nbr_of_prod_purchas.

The basket is composed of a maximum of 24 items, where an item corresponds to a product or a grouping of equivalent products. The variable Nb_of_items corresponds to the number of items in the basket, while the sum of the Nbr_of_prod_purchas variables corresponds to the number of products. The fraud_flag indicator allows one to know if the observation has been identified as fraudulent or not.

To analyze our data and have some insights into it, we proceeded through some steps:

- Visualizing the distribution of the target variable (fraud_flag) to check if the data is balanced or imbalanced.

According to our distribution, our fraud variable is highly imbalanced.



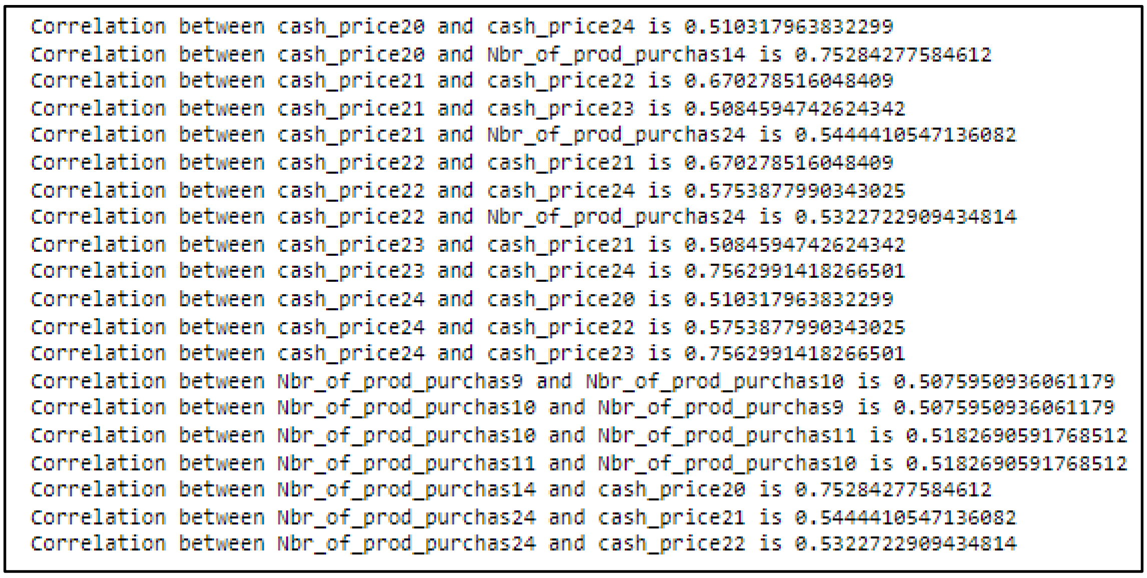
```
Fraud variable distribution:
 0    91471
1     1319
Name: fraud_flag, dtype: int64
```

- Checking for missing values and fixing a strategy to handle them:

By a simple look to our data, we notice the huge amount of present Nan values. To deal with this problem, we proceed by replacing categorical Nan values with a void while replacing numerical Nan values with their mean.

- Exploring the correlation between the features to identify possible multicollinearity:

Considering the huge amount of data we have, we got a dense correlation matrix. Most of our variables are slightly correlated, while we have some variables with a correlation higher than 0.5. However, we didn't take any actions regarding this since we judge that this high correlation is not expressive enough since most of columns have similar values.



```
Correlation between cash_price20 and cash_price24 is 0.510317963832299
Correlation between cash_price20 and Nbr_of_prod_purchas14 is 0.75284277584612
Correlation between cash_price21 and cash_price22 is 0.670278516048409
Correlation between cash_price21 and cash_price23 is 0.5084594742624342
Correlation between cash_price21 and Nbr_of_prod_purchas24 is 0.5444410547136082
Correlation between cash_price22 and cash_price21 is 0.670278516048409
Correlation between cash_price22 and cash_price24 is 0.5753877990343025
Correlation between cash_price22 and Nbr_of_prod_purchas24 is 0.5322722909434814
Correlation between cash_price23 and cash_price21 is 0.5084594742624342
Correlation between cash_price23 and cash_price24 is 0.7562991418266501
Correlation between cash_price24 and cash_price20 is 0.510317963832299
Correlation between cash_price24 and cash_price22 is 0.5753877990343025
Correlation between cash_price24 and cash_price23 is 0.7562991418266501
Correlation between Nbr_of_prod_purchas9 and Nbr_of_prod_purchas10 is 0.5075950936061179
Correlation between Nbr_of_prod_purchas10 and Nbr_of_prod_purchas9 is 0.5075950936061179
Correlation between Nbr_of_prod_purchas10 and Nbr_of_prod_purchas11 is 0.5182690591768512
Correlation between Nbr_of_prod_purchas11 and Nbr_of_prod_purchas10 is 0.5182690591768512
Correlation between Nbr_of_prod_purchas14 and cash_price20 is 0.75284277584612
Correlation between Nbr_of_prod_purchas24 and cash_price21 is 0.5444410547136082
Correlation between Nbr_of_prod_purchas24 and cash_price22 is 0.5322722909434814
```

- Balancing our imbalanced data:

Out of the total observations in the dataset, there are 91471 non-fraudulent transactions and 1319 fraudulent transactions. This indicates that the dataset is highly imbalanced with a majority of non-fraudulent transactions.

To fix this problem, we used the RandomUnderSampler object to undersample the dataset in order to deal with class imbalance. Specifically, it randomly selects a subset of the majority class observations to match the number of observations in the minority class. The resulting undersampled dataset is stored in X_r and y_r.

Next, the features in df_r that will be used for modeling are standardized using the StandardScaler. Standardization involves subtracting the mean and dividing by the standard deviation of each feature, which makes the features have zero mean and unit variance. This can improve the performance of our machine learning model.

- Encoding our categorical values:

Our code performs one-hot encoding on the categorical features of to convert categorical variables into a binary numerical representation that can be used by our models. The function "pd.get_dummies()" creates new binary columns for each unique value in the categorical columns specified in the columns parameter.

This code performs also a feature selection using Ridge regularization in combination with cross-validation. Feature selection allows to select the most relevant and informative features from a larger set of features in order to reduce the dimensionality of the dataset, which can improve the performance of the model and prevent overfitting.

It imports the necessary modules for Ridge and Lasso regression models, respectively, and SelectFromModel for feature selection. A RidgeCV model is initiated with a 5-fold cross-validation (cv=5) strategy to estimate the regularization parameter.
Finally, it creates a SelectFromModel object with the RidgeCV estimator, which means that the feature selection will be based on the coefficients of the Ridge regression model then it

applies feature selection by fitting the SelectFromModel object on the input data X and output variable y.

- Implementing our ML models:

To start with, we choose the **XGBoost** model for binary classification. The model is trained on a training set (X_train, y_train) and evaluated on a validation set (X_val, y_val).

The train_test_split function is used to split the original dataset (X and y) into a training set and a validation set with a test size of 20% of the data.

The XGBoost model is defined with a set of hyperparameters (params) that are used to optimize the model's performance. The hyperparameters used in this model are max_depth, objective, eval_metric, eta, subsample, and colsample_bytree.

After training the model, predictions are made on the validation set, and the performance of the model is evaluated using several metrics.

From the results, we can see that the model achieved an accuracy of 76.33% on the validation set. The confusion matrix shows that the model correctly identified 205 true positives and 198 true negatives, while misclassifying 61 false positives and 64 false negatives.

The precision of the model is 0.764, which means that out of all the positive predictions, 76.4% were correct. The recall of the model is 0.756, which means that out of all the actual positive cases, 75.6% were correctly identified by the model. Finally, the F1 score of the model is 0.760, which is a weighted average of the precision and recall scores.

Overall, the results show that the model is moderately accurate and performs reasonably well in identifying both positive and negative cases.

Random Forest Model

The model in our code is a Random Forest classifier, which is a type of ensemble learning method used for classification and regression tasks.

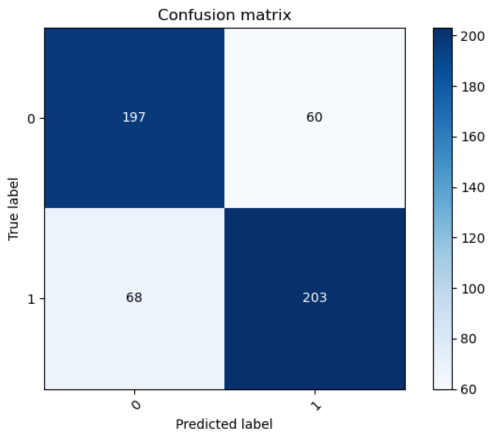
A Random Forest classifier consists of a collection of decision trees, where each tree is constructed using a random subset of the available features and a random subset of the training data. The final prediction is then determined by combining the predictions of all the trees.

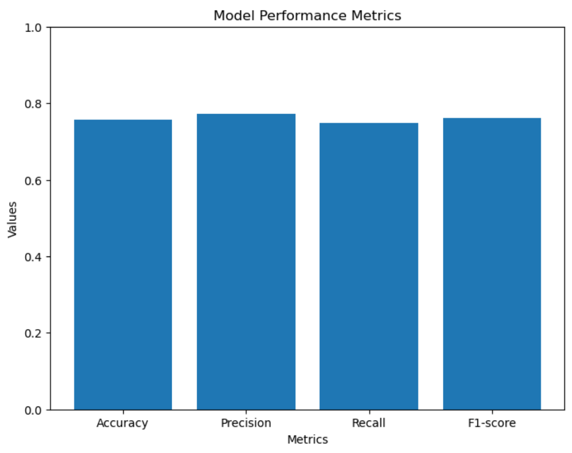
This technique helps to improve the accuracy and robustness of the model by reducing overfitting and increasing the generalization ability. It is particularly useful for datasets with a large number of features and complex relationships between the features and the target variable.

In our code, the Random Forest classifier is trained on a resampled and normalized dataset, where the class imbalance is addressed using the RandomOverSampler and RandomUnderSampler methods from the imblearn library, respectively. This approach is commonly used to handle imbalanced datasets where one class is significantly more prevalent than the other(s).

Results :

The results of our Random Forest model show that the overall accuracy of the model is 0.7576, indicating that it correctly predicted the target variable around 76% of the time on the validation set. The precision score is 0.7719, which means that when the model predicted the positive class, it was correct about 77% of the time. The recall score is 0.7491, indicating that the model correctly identified about 75% of the positive instances in the validation set. Finally, the F1-score is 0.7603, which is a harmonic mean of precision and recall, indicating the overall performance of the model. Overall, these results suggest that the Random Forest model has performed reasonably well on the validation set, but there may be some room for improvement in terms of recall and F1-score.





This indicates that :

There are 197 true positives, meaning the model correctly identified 197 instances of the positive class.

There are 203 true negatives, meaning the model correctly identified 203 instances of the negative class.

There are 60 false positives, meaning the model incorrectly predicted 60 instances as positive when they were actually negative.

There are 68 false negatives, meaning the model incorrectly predicted 68 instances as negative when they were actually positive.

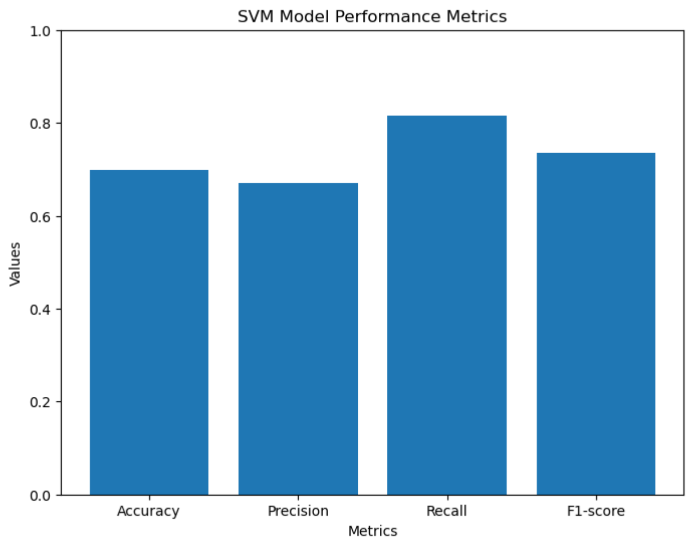
Overall, the model seems to have performed reasonably well, with an accuracy of 0.7576 and an F1-score of 0.7603. However, the precision and recall values indicate that the model may be slightly biased towards the positive class, as the precision is higher than the recall. This means that the model may be correctly predicting a large number of positives, but may be missing some instances of the positive class.

The SVM Model:

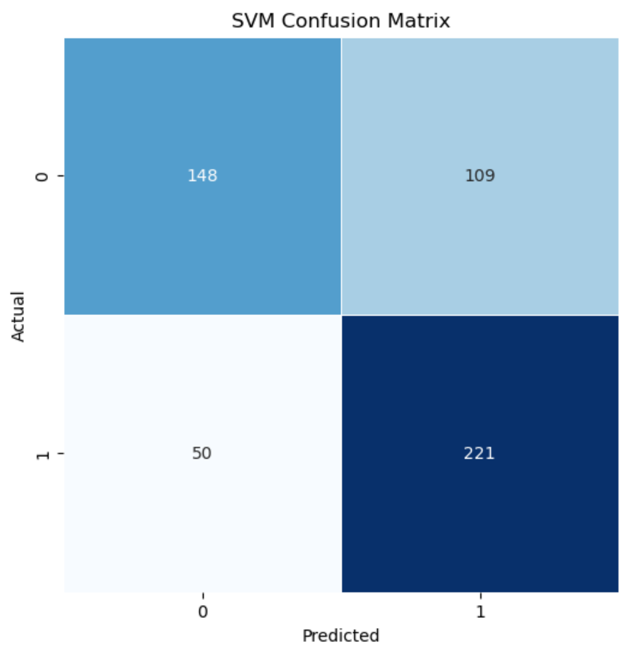
A Support Vector Machine, or large margin separator is a set of supervised learning techniques that address discrimination and regression problems. SVM is a generalization of linear classifiers.

Support vector machines (SVMs) can be a good model for our fraud detection problem due to their ability to efficiently process multidimensional data and define complex decision boundaries. These characteristics make SVM well-suited to classifying complex patterns in data, which often occur in fraud detection situations. Additionally, SVMs are relatively robust against overfitting, especially when using kernel functions, allowing them to generalize well to unseen data.This is an important aspect of fraud detection because having a model that can adapt to new and evolving fraud is essential.

results :



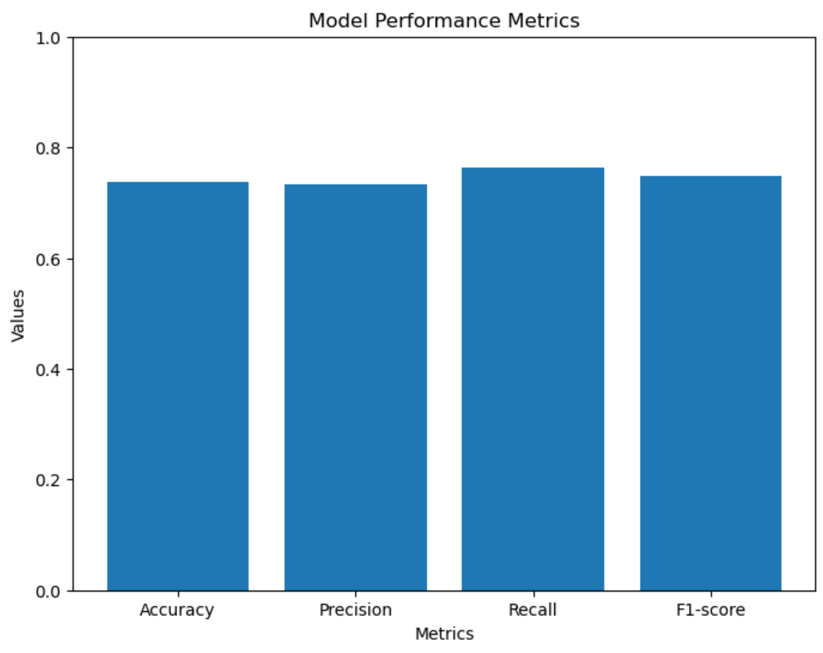
The results of the SVM model are as follows: 69.89% accuracy, 66.97% precision, 81.55% recall and an F1 score of 73.54%.These metrics show good performance in classifying fraud cases. Although the accuracy is relatively low, indicating a higher number of false positives, the recall is significantly high, which means that the model was successful in identifying the majority of real fraud cases. economic. The F1 score balances both accuracy and recall, and its value implies that the sample offers a reasonable compromise between these two measures. Overall, the SVM model exhibits good potential for detecting fraud cases.



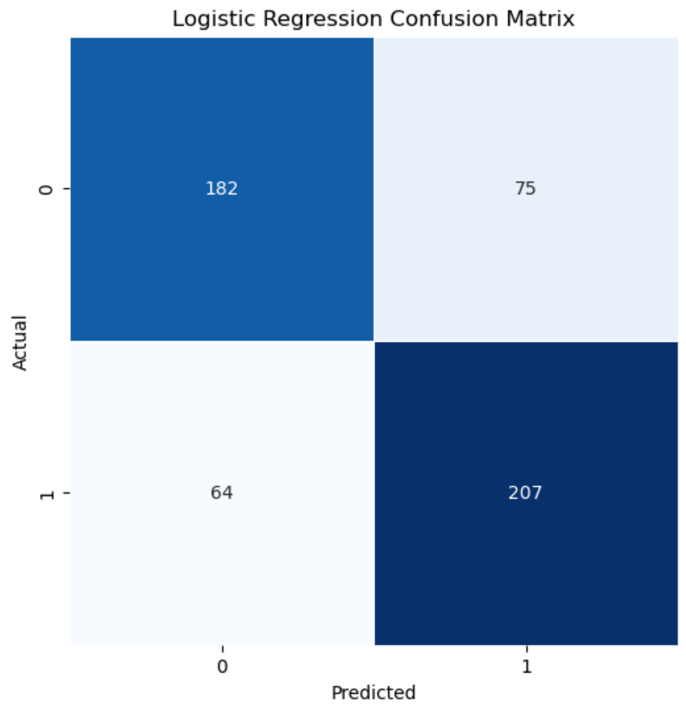
The confusion matrix for the SVM model can be interpreted as follows: True Positives (TP) are 221 cases that were correctly classified as fraud, True Negatives (TN) are 148 cases that were correctly classified as non-fraud, False Positives (FP) are 109 cases that were incorrectly classified as fraud when they were actually non-fraud (Type I error), and False Negatives (FN) are 50 cases that were incorrectly classified as non-fraud when they were actually fraud (Type II error). From the confusion matrix, we can observe that the SVM model has a higher number of false positives compared to false negatives, which means that the model tends to classify more non-fraud cases as fraud, leading to a higher recall but a lower precision.

The logistic regression Model :

Logistic regression is a machine learning algorithm used for classification problems, it is a predictive analysis algorithm and is based on the concept of probability.
One of the main reasons for its effectiveness in this context is its simplicity and interpretability. Logistic regression models the probability that an instance belongs to a particular class, such as fraudulent or non-fraudulent, by fitting a logistic function to input characteristics. This allows us to not only predict the class but also quantify the likelihood that an instance is cheating. Furthermore, its computational efficiency and ease of implementation make logistic regression an attractive choice for large-scale fraud detection problems where real-time transaction processing is a necessity.



The results of the logistic regression model are as follows: the accuracy is 73.67%, the precision is 73.40%, the recall is 76.38% and the F1 score is 74.86%.These metrics show good performance in classifying fraud cases. Accuracy and recall values were relatively balanced, showing that the model was capable of accurately identifying fraud cases while maintaining a moderate false-positive rate. The F1 score, which balances both accuracy and recall, demonstrates that the model offers a satisfactory compromise between these two metrics. Compared with the SVM model, the logistic regression model achieved a slightly higher accuracy and precision, indicating the potential for overall performance improvement.



From the confusion matrix, we can observe that the Logistic Regression model has a lower number of false positives compared to false negatives, meaning that the model is more conservative in classifying cases as fraud, resulting in a higher precision but a slightly lower recall. This balance between precision and recall is reflected in the overall performance metrics, suggesting that the Logistic Regression model provides a reasonable compromise between identifying fraudulent cases and minimizing false alarms.

Conclusion

The report starts by explaining the importance of fraud detection in financial transactions, highlighting the risks and potential losses associated with fraudulent activities. It then introduces the concept of machine learning and its potential applications in fraud detection.

Next, the report describes the dataset used for the study, which consists of financial transaction records with labeled data indicating whether the transaction was fraudulent or not. The report explains that the dataset is highly imbalanced, with a small percentage of fraudulent transactions compared to legitimate ones.

The report then goes into detail on the machine learning algorithms used in the study, including Random Forest, XGBoost, Logistic Regression, and Support Vector Machines (SVM). For each algorithm, the report explains how it works and its strengths and weaknesses. It also explains the preprocessing techniques used to prepare the data for analysis, such as resampling, hot coding, and feature selection.

The report then presents the results of the study, comparing the accuracy, precision, recall, and F1 scores of each algorithm. It concludes that the XGBoost algorithm performed the best overall, with the highest accuracy, precision, and F1 score. However, it also notes that each algorithm has its strengths and weaknesses, and that the best algorithm may vary depending on the specific dataset and problem.

Finally, the report concludes with a summary of the findings and recommendations for future work in this area. It suggests that more research is needed to explore the use of other machine learning algorithms and preprocessing techniques, as well as the impact of different parameter settings on algorithm performance. Overall, the report provides a detailed analysis of the effectiveness of different machine learning algorithms for fraud detection in financial transactions, highlighting the importance of careful data preprocessing and algorithm selection in achieving accurate and reliable results.