Bank Fraud Detection:

Machine Learning Final Project Report

Mumuksha Pant, Dhruv Saini
Khoury College of Computer Sciences
Northeastern University, Boston, USA
pant.mu@northeastern.edu, saini.dh@northeastern.edu

***Abstract* — The present project report encompasses the execution and examination of the final project for the course 'CS6140 Machine Learning'. The main objective of this project is to address the issue of Bank Fraud Detection by employing a range of classifiers. The report aims to offer an outline of the problem statement and a comprehensive analysis of our implementation and utilization of diverse machine learning classification models. Additionally, it includes a discussion of the outcomes obtained through various metrics.**

Keywords: Support Vector Machines(SVM), GBC - Gradient Boosting Classifier, False Negatives, Principal Component Analysis

I. INTRODUCTION

With the advent of technology there has been a significant rise in the Banking Frauds which poses a significant threat to both- financial institutions and customers. Detecting fraudulent activities in timely manner is essential for reducing financial losses and for the customers to maintain their trust in the banking sector. We aim at solving the fraud detection on a publicly available dataset using Supervised Machine learning techniques like Logistic Regression, Decision Trees, Random Forests and GBC.

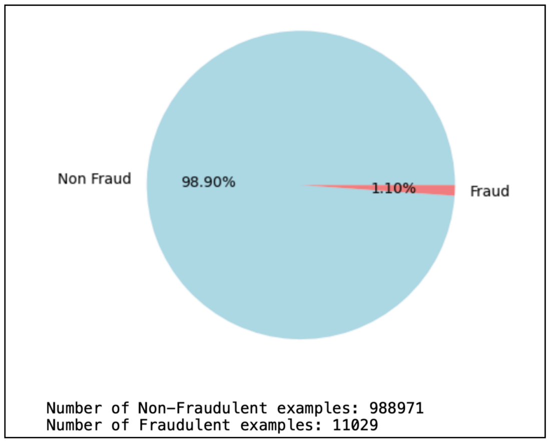


Fig 1 Pie Chart for Distribution of Target Variable

II. Data

The dataset used was found on Kaggle **[1]** and is published at NeurIPS 2022 **[2]**, which provides a valuable resource for training and evaluating our

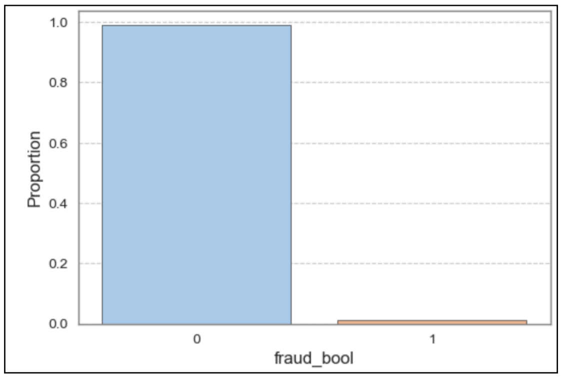


Fig 2 Count-plot for Distribution of Target Variable

model. The datasets on the suite were generated by leveraging Generative Adversarial Network models. The dataset contains 1 million records out of which 11029 transactions are labeled as fraud. It is a highly imbalanced data with only 1.10 % fraud transactions and this setting presents a extremely low prevalence of positive class. (Figure 1 and Figure 2)

The target variable is 'fraud_bool' which corresponds to fraudulent activities represented by 1 and non fraudulent activities represented by 0. There are no missing Values in the Dataset and the model will take 32 features as input and output value of either 0 (non fraud) or 1 (fraud).

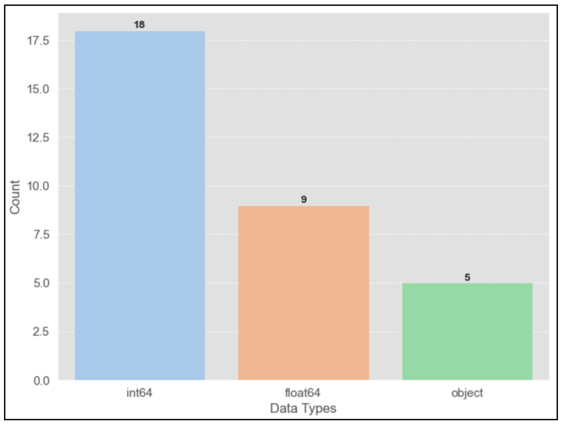


Fig 3 Distribution of Datatypes

There are 5 categorical values in the data (Figure 3) which have been appropriately handled using Hot Encoding Technique **[3]** to facilitate their integration into the model.

III. IMPLEMENTATION

In this project, all the classifier models were implemented using the scikit-learn libraries. The main goal is to distinguish between fraudulent and non-fraudulent transactions by establishing a decision boundary in the feature space. This decision boundary enables the classifiers to predict and classify new transactions as either fraudulent or non-fraudulent based on their feature characteristics.

For our problem statement, our ***primary focus is on minimizing False Negatives***. These are the instances where a transaction is fraudulent but is incorrectly classified as a legitimate transaction. Reducing False Negatives is crucial because if potential fraud goes undetected, it can jeopardize the bank's reputation and result in losses for both customers and financial institutions.

III . I) **Feature Engineering**

i) Our dataset contains 5 categorical attributes. To handle these categorical features, we applied the One Hot Encoding method. By using the pd.get_dummies() function, new columns were generated for each category found in the original categorical column **[4] .**

The choice between 'One Hot Encoding' and 'Label Encoding' depends on factors such as nature of the categorical variable and the intended use of algorithms. In our problem statement, One Hot Encoding proved to be the appropriate technique.

ii) The feature 'fraud_device_count' remained constant and had no impact on the target variable. As a result, we decided to remove this feature from the analysis.

iii) We decided to drop 3 more features from our dataset based on correlation threshold. For features having correlation greater than 0.7, we decided to drop them from our analysis.

III. II) **Model Implementation**

To achieve our objective, we have employed multiple algorithms and calculated various metrics to identify the best model suitable for this problem. Using multiple algorithms helps enhance the robustness of our solution and reduces the risk of a single algorithm failing to detect certain types of fraud. This approach allows us to create a more reliable and effective system for detecting and preventing fraudulent activities in the banking sector.

A. Logistic Regression

Logistic Regression is a linear algorithm commonly employed for various classification tasks. In our project, we utilized Principal Component Analysis (PCA) along with Logistic Regression to reduce the dimensions of our dataset, retaining only the most essential information.

By applying PCA to our dataset before using Logistic Regression, we aimed to enhance the efficiency of our model and improve its ability to discern relevant patterns and features for the classification task. This combination allows us to create a more streamlined and effective system for distinguishing between fraudulent and non-fraudulent transactions.

B. Decision Tree

The Decision Tree model is highly effective in capturing complex decision boundaries and solving intricate decision problems. However, in our project, we faced issues with overfitting the data, as evident from the results. (See Results)

To address this overfitting problem, we attempted hyper-parameter tuning. Unfortunately, despite achieving an accuracy of **0.70**, the model continued to suffer from **overfitting**. Additionally, the precision and recall values for both classes (fraud and non-fraud) were nearly identical, suggesting the model's limitations in effectively distinguishing between the two categories.

C. Random Forests

After extensive experimentation with different tree depths and splitting criteria (Gini Index and Entropy) on Decision Trees, we opted to utilize Random Forests, an ensemble technique comprising multiple Decision Trees. While we expected Random Forests to generalize better to our data, we still encountered signs of **overfitting** in the model.

Despite the overfitting concerns, the Random Forest model displayed a higher accuracy of **0.80** compared to the Logistic Regression Model's accuracy of 0.68. Moreover, the Random Forest approach proved to be more effective in **minimizing false negatives**.

D. Support Vector Machines

Support Vector Machines (SVMs) utilize various kernels to identify different types of hyperplanes in the dataset. The objective of SVMs is to find a hyperplane to maximizes the margin between both classes, which aids in better classification.

In our dataset, SVMs yielded a relatively low accuracy of 0.62. The similarity between the train and test accuracies (0.61 to 0.64) suggests that the model is not overfitting, which is encouraging.

Despite these efforts, the overall accuracy is still not significantly high. Therefore, further exploration and experimentation with different kernels are essential to determine the cause of this limitation. By delving deeply into SVMs and fine-tuning the kernel selection, we can better understand the shortcomings and work towards improving the model's performance.

E. Gradient Boosting Classifier (GBC)

GBC is a robust model that is well-known for its ability to mitigate overfitting. In our project, the GBC model delivered exceptional results, achieving an accuracy of **0.81** on both the training and testing datasets. Through careful hyper-parameter tuning, we further improved the model's performance, boosting the accuracy to 0.82. The False Negative instances were significantly minimized, with only **601** misclassifications.

F. Multi Layer Perceptron (MLP)

MLP is an Artificial Neural Network model that automatically learns relevant features from the data, reducing the need for manual feature engineering. Our MLP model initially achieved an accuracy of 0.52. However, after implementing k-fold cross-validation with 6-folds, the accuracy increased to 0.73. We further fine-tuned various hyper-parameters, including the number of layers, number of nodes in each layer, and activation functions within different specified ranges **[5]**.

IV. RESULTS

To get reliable performance metrics, we employed k folds cross validation techniques as it uses every part of the dataset in folds for testing and training. We used the 4,610-folds approach.

A. Logistic Regression

The model yielded an accuracy of 0.68 which means it correctly classified instances around 68% of the time. There were 2142 misclassified instances.

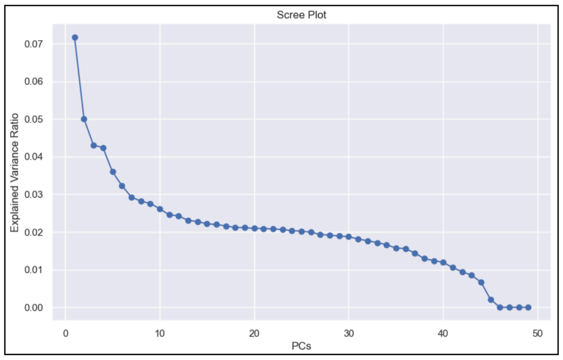
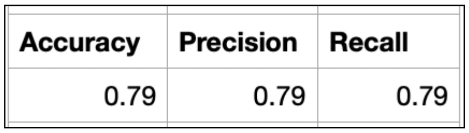


Fig 4 Logistic Regression using PCA Scree Plot



Accuracy	Precision	Recall
0.79	0.79	0.79

Fig 5 Metrics after PCA

Additionally, Logistic Regression using PCA (Figure 4) is giving the best accuracy of 0.7878 (Figure 5) . The precision is approximately 0.7855, indicating that out of the total instances predicted as 'positive' by the model, about 78.55% of them are actually correct. The recall of 0.7860, implies that the model correctly identifies about 78.60% of the actual positive instances in the dataset.

B. *Decision Tree*

Our decision tree model exhibited signs of Overfitting on the data which is indicated from a significant difference between the training accuracy and testing accuracy (Figure 6) .

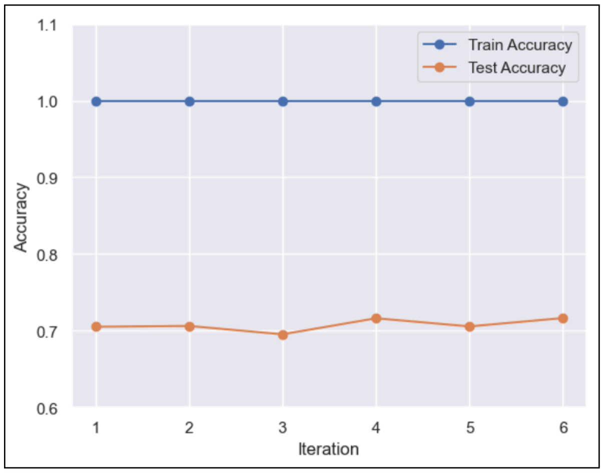


Fig 6 Training and Testing Accuracy of Decision Tree

Overfitting implies that the model is performing well on the training data but not generalizing effectively to unseen data. Despite multiple attempted runs and hyper-parameter tuning and evaluating on different accuracy metrics, the model still exhibited overfitting. This suggests that the nature of the dataset maybe very complex for the tree split.

C. *Random Forests*

The model achieved a high training accuracy (1.00) but the overall test prediction accuracy (0.80) indicates that the model is possibly overfitting the training data (Figure 7). The

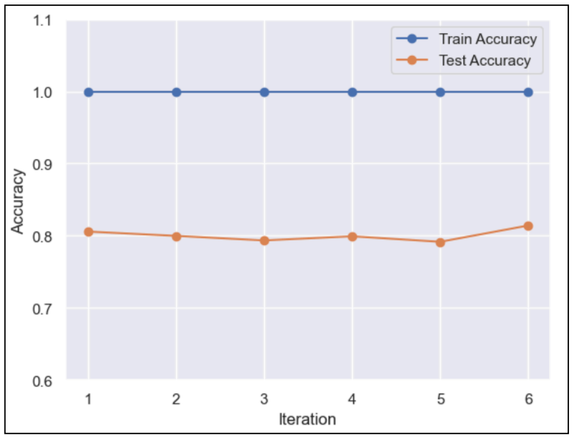


Fig 7 Training and Testing Accuracy of Random Forests

values in the confusion matrix suggest that the model is correctly predicting a substantial number of instances (TP and TN), while making relatively fewer false predictions (FP and FN) (Figure 8) . Even after Hyper-parameter tuning, the Random Forests Algorithm is overfitting.

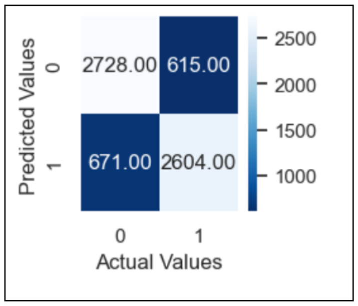


Fig 8 Confusion Matrix for Random Forests

D. *Support Vector Machine*

Although we wanted to explore more with SVMs, experimenting with different kernels, and regularization parameters. But, our hyper-parameter tuning for Support Vector Machines did not work well on our systems. Despite allowing the process to run for over

10 hours, we did not get any satisfactory results using the hyper parameter tuning. This could be a potential limitation of our computational resources [6,7] .

E. *Gradient Boosting Classifier*

Gradient Boosting Machine (GBM) model is performing consistently across all iterations of -fold cross validation. The test prediction

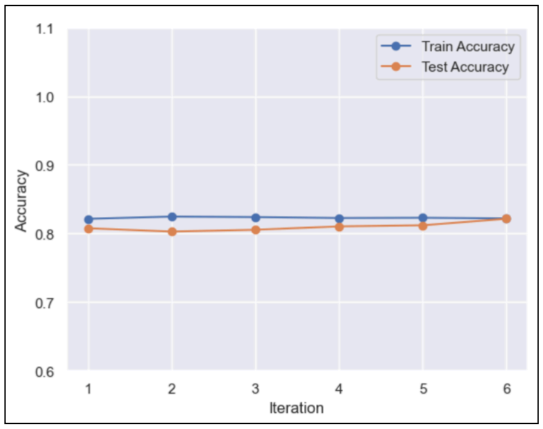


Fig 9 GBM Training and Testing Accuracy

accuracy ranges from 0.80 to 0.82(Figure 9), with an overall accuracy of 0.81. About 18.7% is the rate of misclassification (error rate) (Figure 10) .
After hyper-parameter tuning , using the best parameters to fit our training data yielded an accuracy of 0.82. Alongside the accuracy, we aim at minimizing the False Negatives (FN), ie , minimizing the 'incorrect classification of a Fraudulent transaction as a legit transaction'.

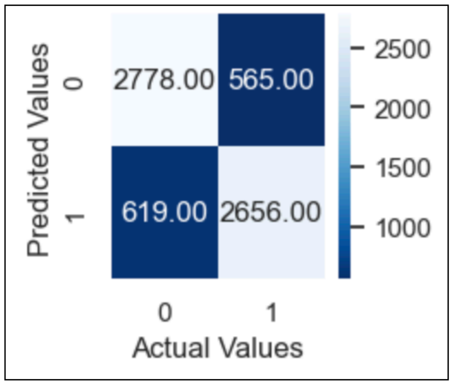


Fig 10 Confusion Matrix for GBM

F. *Multilayer Perceptron*

MLP with an accuracy of 0.73 overall appears to be performing relatively well . There is not much significant difference between the training

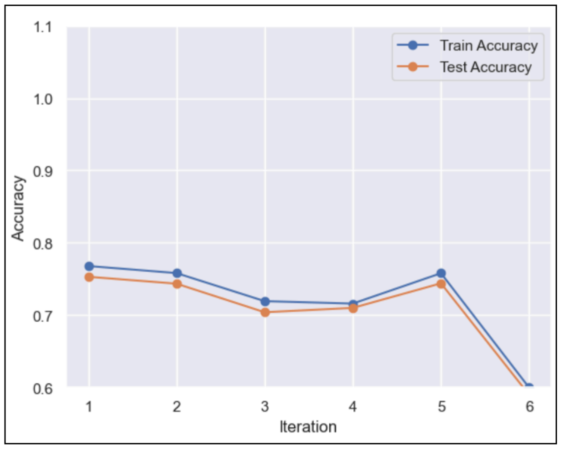


Fig 11 Training and Testing Accuracy MLP

and testing accuracies measured during k-folds cross validation (Figure 11) . The number of False Negatives are lowest in case of MLP , even though the accuracy is lesser than all models (Figure 12)

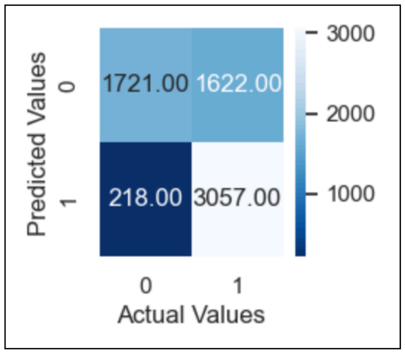


Fig 12 Confusion Matrix

V. CONCLUSION

Gradient Boosting Machine is giving the best accuracy of **0.82** after Hyper-parameter set at *{'learning_rate': 0.09473095986778095, 'max_depth': 5, 'max_features': 'sqrt'}* with a False Negative count of 619 instances. It correctly classifies 82% of the instances in the dataset. The precision and recall for both classes (0 and 1) are fairly balanced.

The Logistic Regression model when hyper-tuned is giving an accuracy of 80%. The Accuracy and F1-scores suggest that the model is generalizing reasonably well to the unseen test data.

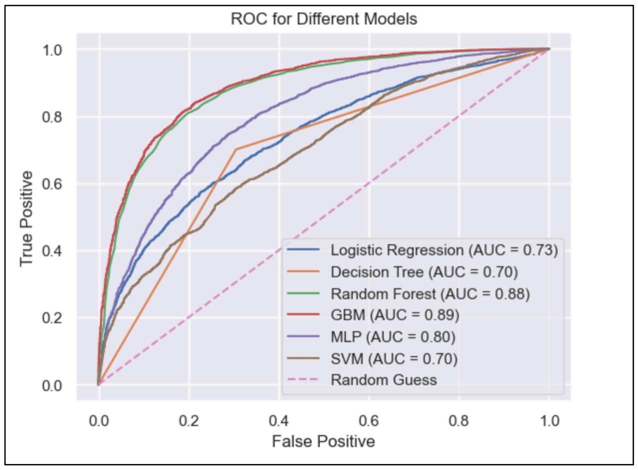


Fig 13 ROC Curves for different classifiers

Clearly, the ROC curve, which illustrates the trade-off between true positive rate and false positive rate, plotted for all the classification models used, suggests that both the GBM and Random Forest have exhibited commendable performance in classifying the dataset. However, as discussed earlier, the Random Forest model lags behind GBM due to its susceptibility to overfitting. Therefore, a confident conclusion can be drawn that the **Gradient Boosting Machine (GBM)** algorithm outperformed others and emerged as the optimal choice for our dataset.

VI. FUTURE WORK

1. Given the availability of time and computational resources, we would work on SVM hyper parameter tuning on both under sampled and over sampled datasets.
2. We would utilize more time on training and hyper parameter tuning MLP Models to understand in depth about its hidden layer structure and activation function.
3. To enhance accuracy and achieve superior predictive capabilities, we would delve into utilizing more sophisticated classifiers which we had not covered in the scope of this course.
4. Currently we have used under sampling to balance the dataset, we can implement over sampling using SMOTE and then analyze various models to predict results.
5. We would employ PyTorch for Neural Network implementation facilitating more efficient analysis.

VII . REFERENCES

[1] Kaggle
[2] NeurIPS Paper
[3] https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/#
[4] https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html
[5] https://saturncloud.io/blog/sklearn-mlp-classifier-hidden-layers-optimization-randomizedsearchcv/
[6] https://datascience.stackexchange.com/questions/989/svm-using-scikit-learn-runs-endlessly-and-never-completes-execution
[7] https://ai.stackexchange.com/questions/7202/why-does-training-an-svm-take-so-long-how-can-i-speed-it-up
[8] https://www.researchgate.net/profile/Rafael-De-Sousa-Junior/publication/47619350_Identifying_Bank_Frauds_Using_CRISP-DM_and_Decision_Trees/links/00463519d41d289455000000/Identifying-Bank-Frauds-Using-CRISP-DM-and-Decision-Trees.pdf
[9] https://cdn.techscience.cn/ueditor/files/csse/TSP_CSSE-45-1/TSP_CSSE_26508/TSP_CSSE_26508.pdf