Fraud Detection in Credit Card Transactions Using Machine Learning

Team Members: Akhil Bhargava, Ankit Kumar, Ankur Asthana

Course: ISyE 6740 – Fall 2024

1. Executive Summary

Fraudulent financial transactions pose a significant challenge to financial institutions, leading to substantial monetary losses and reputational damage. The increasing volume of digital transactions has intensified the need for sophisticated fraud detection systems. This project, titled *Fraud Detection in Credit card Transactions Using Machine Learning*, aims to address these challenges by developing an accurate and efficient fraud detection model.

Using a publicly available dataset from Kaggle, which contains both legitimate and fraudulent credit card transactions, the project explores various machine learning and traditional algorithms to detect fraudulent activity. The dataset presents an inherent challenge due to its highly imbalanced nature, with fraudulent transactions representing a small fraction of the total data.

The project involves comprehensive data exploration, preprocessing, and feature engineering to optimize the input for machine learning models. Techniques such as Synthetic Minority Oversampling (SMOTE) and class-weight adjustments are used to handle data imbalance. Multiple models, including Logistic Regression, Random Forest, Gradient Boosting Machines (XGBoost), Support Vector Machines (SVM), Neural Networks, K-Nearest Neighbor classification and Anomaly Detection techniques, are evaluated.

Key findings indicate that the **K-Nearest Neighbors (KNN)** model achieved the highest recall (98%) and ROC-AUC (99%), demonstrating its exceptional capability to detect fraudulent transactions while maintaining strong precision. The **Neural Network** model also performed well with a high recall (91.8%) and ROC-AUC (99.1%), effectively identifying fraud with high precision (87.6%). Additionally, **XGBoost** and **SVM (RBF)** showed good performance, balancing precision and recall effectively.

KNN performs better in fraud detection because it captures localized patterns and is less prone to overfitting, especially with imbalanced datasets. It excels at detecting fraud clusters based on proximity in the feature space. In contrast, Neural Networks may struggle with overfitting, require more data, and face challenges in learning subtle fraud patterns, particularly when data is imbalanced.

In conclusion, the project emphasizes the importance of selecting models based on business objectives, such as maximizing fraud detection or minimizing false positives. The findings offer valuable insights for implementing scalable fraud detection systems that can be applied to high-volume transaction environments.

2. Introduction

Fraud detection in financial transactions has become a critical issue for financial institutions due to the significant increase in digital transactions. With the growing reliance on online payments, detecting fraudulent activities has become both more important and more challenging. Fraudulent transactions can lead to severe financial losses and reputational damage for organizations. This project aims to develop a robust machine learning-based fraud detection system that accurately identifies fraudulent transactions while minimizing false positives. The system will be designed to handle the complexities of imbalanced data, a common issue in fraud detection datasets.

3. Problem Statement and Objectives

Financial institutions face significant financial losses and reputational risks due to fraudulent transactions, which can erode customer trust and cause substantial monetary damage. Although fraudulent activities represent only a small fraction of all transactions, their rarity creates a highly imbalanced dataset that complicates accurate classification. The primary challenge is to distinguish fraudulent transactions from legitimate ones with high precision and recall, ensuring early detection while minimizing false positives. This project addresses these challenges by building and comparing various machine learning and traditional models to identify the most efficient fraud detection approach.

This project aims to address these challenges by:

- Analyzing and preprocessing financial transaction data to uncover patterns indicative of fraud.
- Building and comparing multiple machine learning models to identify the most effective approach for fraud detection.
- Developing an efficient fraud detection model that minimizes false positives while maintaining high accuracy.
- Evaluating model performance using key metrics such as accuracy, precision, recall, F1-score, and Area Under the Curve (AUC) to ensure robust and reliable fraud detection.

By addressing these challenges, the project seeks to contribute to the development of a practical and scalable solution for real-time fraud detection in financial transactions.

4. Dataset Overview

Source: Kaggle

Data description:

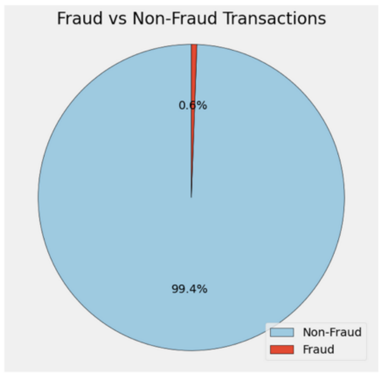
- The Fraud Detection dataset is a simulated credit card transaction dataset, available on Kaggle, that records legitimate and fraudulent transactions spanning from **January 1, 2019, to December 31, 2020**. The dataset includes credit card transaction details for **1,000 customers** interacting with **800 merchants**. It features a diverse set of attributes capturing various transactional and demographic aspects to support detailed fraud analysis.
- The dataset was generated using the **Sparkov Data Generation Tool** by Brandon Harris. To facilitate analysis, the files were combined and converted into a structured format, resulting in two synthetic datasets: `fraudTrain.csv` and `fraudTest.csv`.

Key Features and Their Descriptions:

- **Transaction Details**:
    - trans_date_trans_time: Date and time of the transaction.
    - cc_num: Credit card number.
    - trans_num: Unique transaction identifier.
    - unix_time: Unix timestamp of the transaction.
    - amt: Transaction amount.
    - merchant: Merchant where the transaction occurred.
    - category: Purchase category (e.g., travel, personal care).
- **Cardholder Information**:
    - first, last: First and last names of the cardholder.
    - gender: Gender of the cardholder.
    - dob: Date of birth.
    - street, city, state, zip: Address details of the cardholder.
    - lat, long: Latitude and longitude of the cardholder's location.
    - city_pop: Population of the cardholder's city.
    - job: Cardholder's occupation.
- **Merchant Information**:
    - merch_lat, merch_long: Latitude and longitude of the merchant's location.
- **Fraud Indicator**:
    - is_fraud: Binary label indicating if the transaction is fraudulent (1 for fraud, 0 for legitimate).

Dataset Structure

- The data contains a mix of categorical, numerical, and geospatial features, offering a comprehensive view of transactional behavior. The presence of labels (`is_fraud`) makes the dataset suitable for supervised machine learning tasks focused on fraud detection.
- This diverse and well-documented dataset provides a robust foundation for analyzing fraudulent activities, building predictive models, and evaluating performance across different machine learning approaches.
- The dataset is highly imbalanced, with fraudulent transactions making up a small proportion of the total data, requiring specialized techniques for accurate fraud detection.



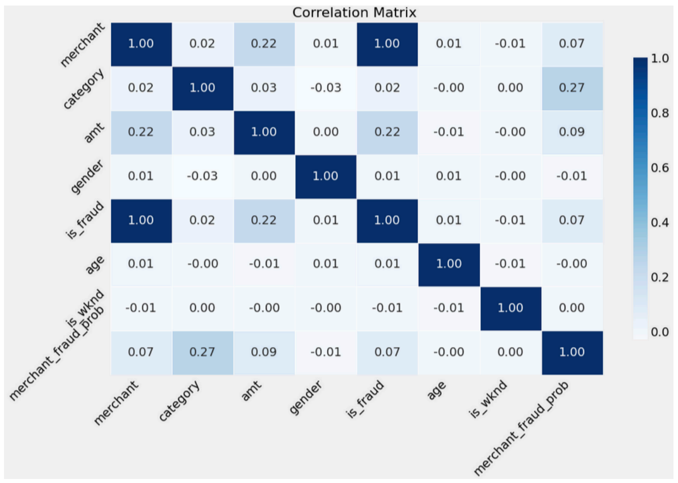
5. Methodology

5.1 Data Exploration and Preprocessing

We started with a credit card transaction dataset with 1.8 million records and 23 features (numeric and categorical). Before we could run our classification models, we converted some of these features into more meaningful features.

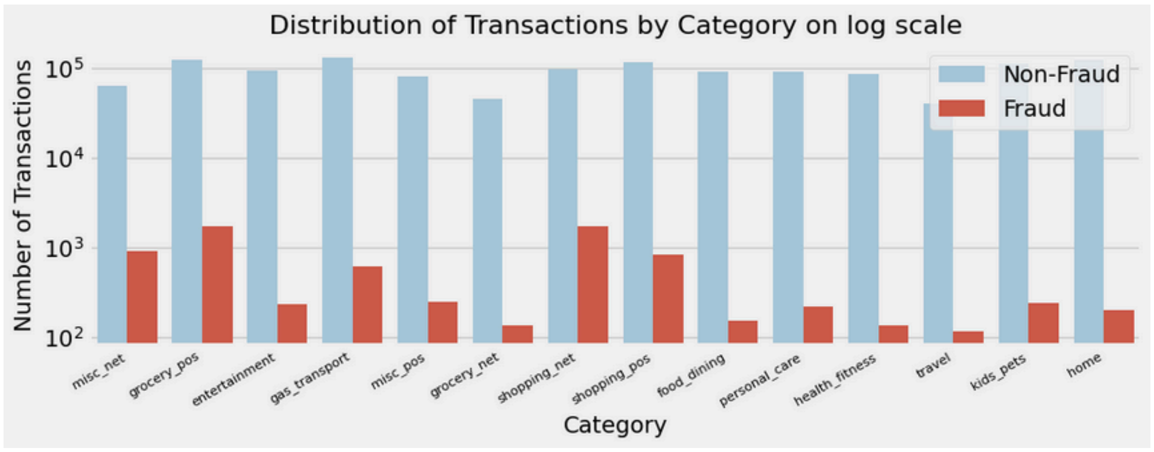
- ‘latitude’ and ‘longitude’ of card holder and merchant were used to create a distance measure and then these four features were dropped.
- Features like credit card number, ‘first’ name , ‘last’ name, ‘street’ address were dropped because this was personal data.
- Feature like ‘unix time’ and ‘transaction number’ were also dropped as they were redundant or not needed.
- The ‘job’ feature had around 500 unique values, so they were mapped to 14 concise job categories. These categories were encoded to numeric.
- We converted the feature ‘state’ to state fraud probability by taking a ratio of fraud transaction and total transaction by state converting this feature to numeric.
- Similarly, we had around 700 unique values for the feature ‘merchant’ and we converted that to merchant fraud probability by taking a ratio of fraud transaction and total transaction by merchant converting this feature to numeric as well.
- The feature ‘gender’ was converted to numeric(0: Female and 1: Male).
- A new feature card holder ‘age’ was created by calculating the difference between their date of birth and transaction date, ‘dob’ was dropped.
- Transaction time was used to calculate the time of day(Night, Morning, Afternoon and Evening) and were encoded to numeric field as well.
- The feature transaction ‘category’ as encoded to numeric as well.
- Redundant features were dropped.

Below is a correlation matrix of all the numeric columns of our dataset. Here we find that only the ‘amt’ field has a significant correlation with the ‘is_fraud’ our response variable.

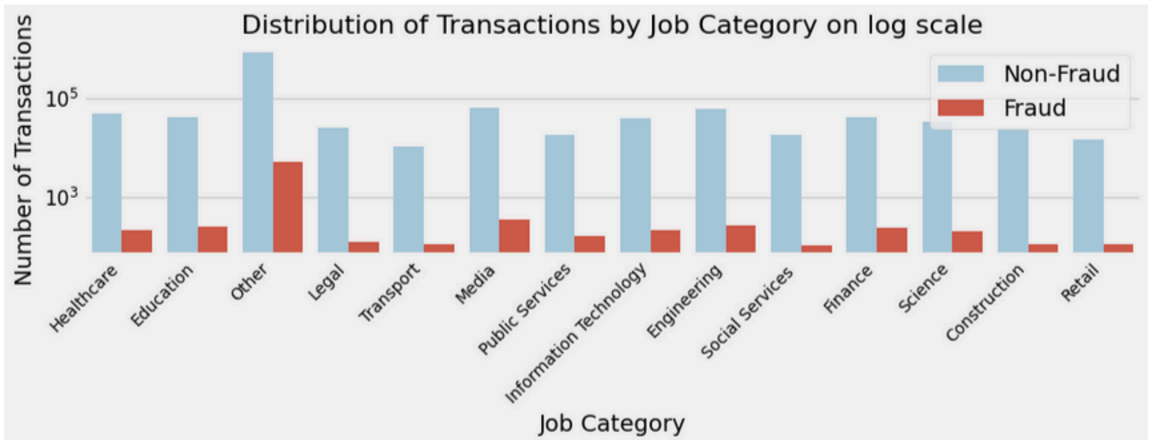


	merchant	category	amt	gender	is_fraud	age	is_wknd	merchant_fraud_prob
merchant	1.00	0.02	0.22	0.01	1.00	0.01	-0.01	0.07
category	0.02	1.00	0.03	-0.03	0.02	-0.00	0.00	0.27
amt	0.22	0.03	1.00	0.00	0.22	-0.01	-0.00	0.09
gender	0.01	-0.03	0.00	1.00	0.01	0.01	-0.00	-0.01
is_fraud	1.00	0.02	0.22	0.01	1.00	0.01	-0.01	0.07
age	0.01	-0.00	-0.01	0.01	0.01	1.00	-0.01	-0.00
is_wknd	-0.01	0.00	-0.00	-0.00	-0.01	-0.01	1.00	0.00
merchant_fraud_prob	0.07	0.27	0.09	-0.01	0.07	-0.00	0.00	1.00

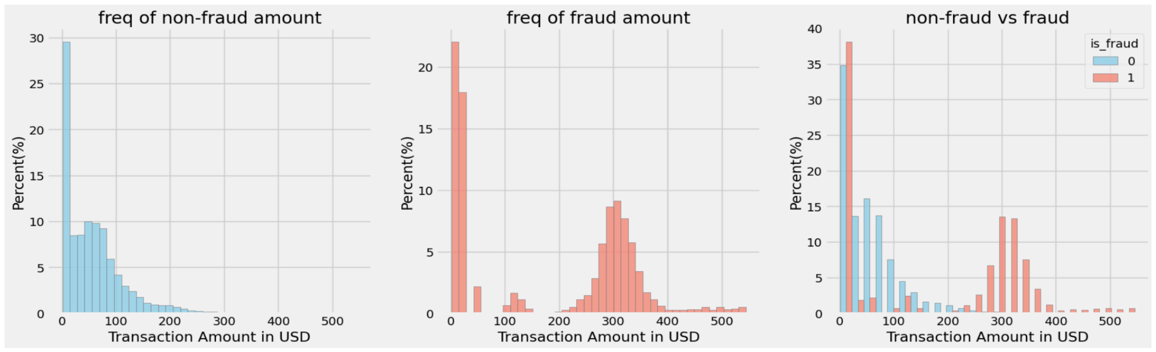
- Below is a histogram of transactions and fraud transaction by transaction category, we find that point of sales grocery and online shopping categories account for the greatest number of fraud transactions.



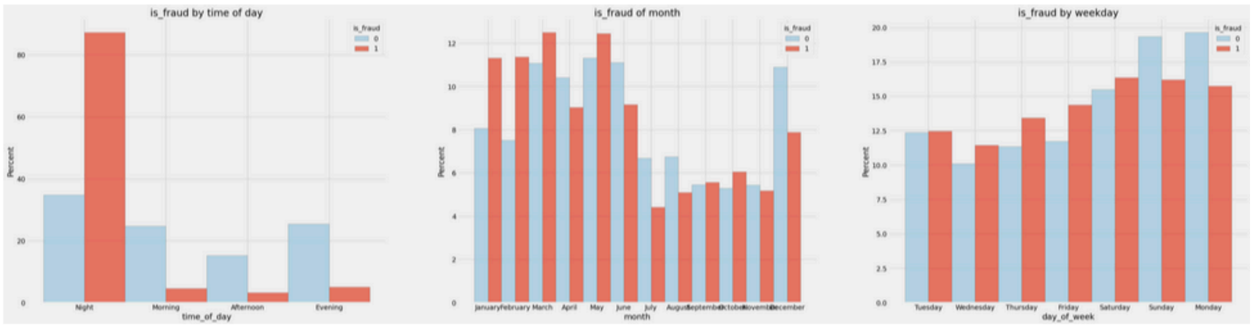
- Similar histogram for job category does not tell us much, the 'other' category account for most of the fraud transactions, however this category involves lot of non-conventional jobs.



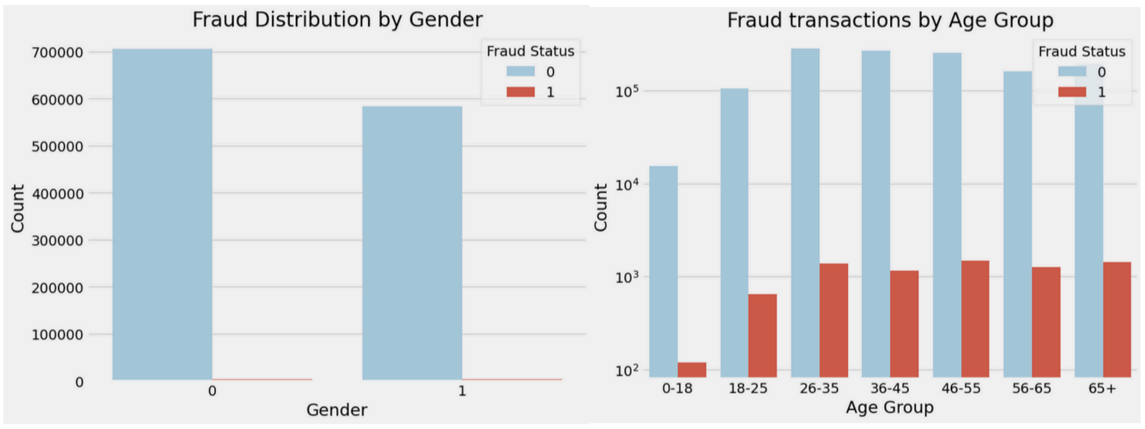
- Below graph shows that most of the fraud transactions amounts are between \$250-\$350 range or very low amounts like less than \$10.



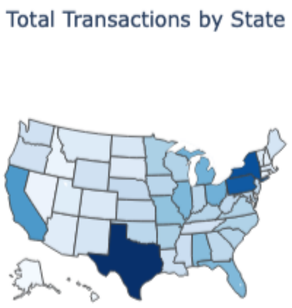
- In the below graph, we can see that most of the fraud transactions occurred during night time and Saturday, Sunday and Monday accounts for a lot of fraud transactions.

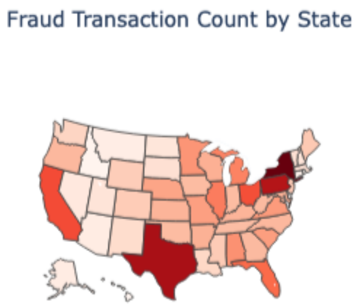


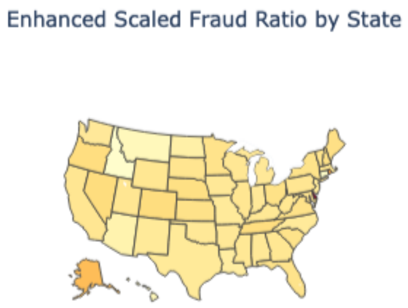
- In the below graph, we can see that there is no significant difference between the number of fraud victims with respect to gender. However, females were involved in more transactions than males. Based on age, we can deduce that middle aged and older people were targeted more as compared to teenagers, this can be explained by that fact that teenagers are less likely to have access to credit cards as compared to older individuals.



- Below graph shows total transactions, fraud transactions and fraud ration by state, we can see that Texas, Philedelphia and New york account for the most number of fraud transactions. This can be explained by the fact that these states also account for the most number of total credit card transactions. The fraud ratio graphs somewhat shows this as we can see very little variation in fraud ration by state.







5.2 Handling imbalanced data and using SMOTE

- In real-world datasets, such as the credit card fraud detection dataset, class imbalance is a common challenge. Class imbalance occurs when one class (fraudulent transactions) is significantly underrepresented compared to the other (legitimate transactions). This imbalance can negatively impact the performance of machine learning models, leading to biased predictions that favor the majority class.
- To address this issue, Synthetic Minority Over-sampling Technique (SMOTE) is commonly used. SMOTE works by generating synthetic samples for the minority class based on the existing data points, helping balance the class distribution. While SMOTE can improve model performance by providing more balanced data, its effectiveness varies across models.
- For this project, SMOTE was tested on two models: Naive Bayes and XGBoost.
    - **Naive Bayes:** Techniques like Naïve Bayes are not able to handle imbalanced data well. SMOTE improved recall a bit helping the model better identify fraud cases. However, it showed a significant deterioration in the Accuracy, Precision and AUC score, indicating that the model's overall ability to distinguish between fraud and non-fraud transactions remained similar or got worse.

Model	**Accuracy**	**Precision**	**Recall**	**ROC-AUC**
Naïve Bayes with SMOTE	97.7%	14.7%	70.0%	81.7%
Naïve Bayes w/o SMOTE	**99.1%**	**27.2%**	**46.8%**	**82.3%**

    - XGBoost: Applying SMOTE did not improve the model's performance. It did not add value in terms of recall, precision, or AUC, suggesting that SMOTE is not beneficial. It is also known that techniques like XGboost can handle imbalanced data well and hence SMOTE is not adding any significant benefit.

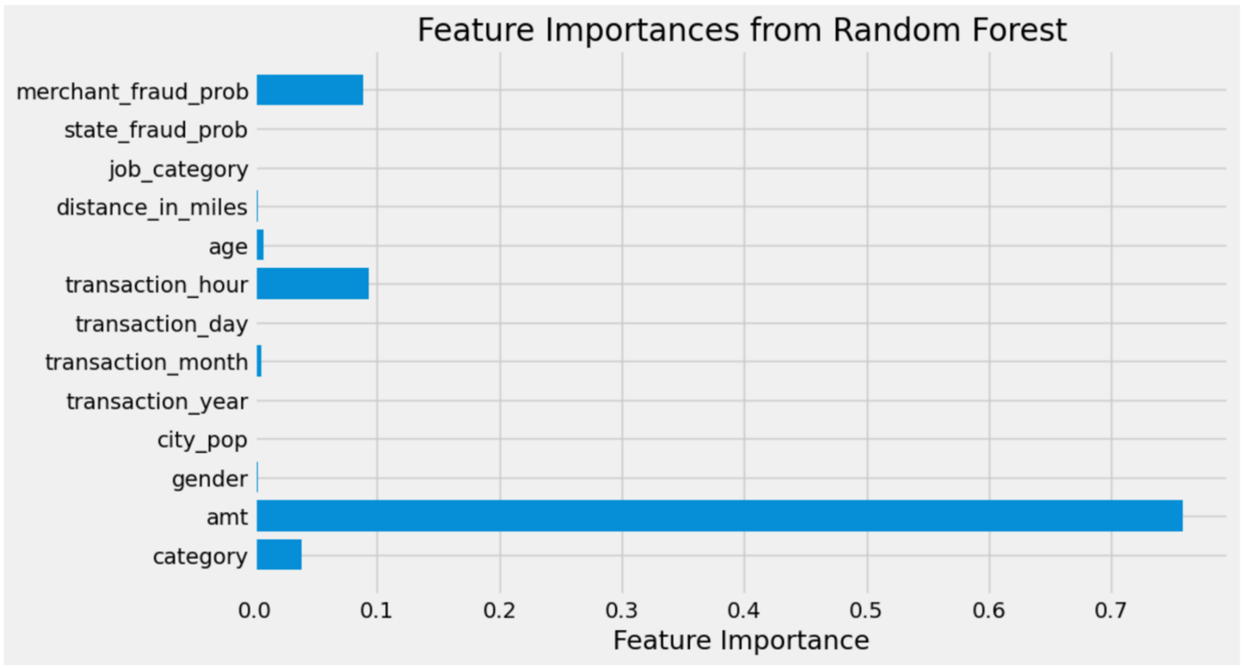
Model	Accuracy	Precision	Recall	ROC-AUC
Xgboost with SMOTE	98.7%	26.5%	80.3%	97.2%
Xgboost w/o SMOTE	**99.4%**	**46.0%**	**81.3%**	**98.8%**

- Given the results does not show any significant improvement , SMOTE was not recommended to be used for rest of the techniques which we initially proposed.

5.3 Feature Selection and Engineering

Feature selection and engineering are critical steps in preparing data for machine learning models, especially when working with complex datasets like credit card fraud detection. In this project, several transformations were applied to the dataset to enhance its predictive power, including the calculation of a **distance metric** and the encoding of **categorical variables**.

Below features importance plot from Random forest model shows that 'amt', 'transaction_hour', 'merchant_fraud_prob' and 'category' are important features.



- **Creation of Distance Metric Between Cardholder's Address and Transaction Location**

One of the key insights for fraud detection in this dataset was the potential link between the cardholder's location (address) and the location of the transaction. Fraudulent transactions may exhibit patterns such as large distances between the cardholder's usual location and the location where the transaction occurred. To quantify this, we created a **distance metric** that calculates the distance between the cardholder's address and the transaction's location using **longitude and latitude**.

The **Haversine distance** method was used to compute this distance, as it accurately calculates the distance between two points on the Earth, taking into account the curvature of the globe. The formula for the Haversine distance between two points (lat1,lon1) and (lat2,lon2) is:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cdot\cos(\phi_2)\cdot\sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2\cdot \operatorname{atan2}\left(\sqrt{a}, \sqrt{1-a}\right)$$

$$d = R\cdot c$$

Where:

- ϕ_1, ϕ_2 are the latitudes of the two points in radians,
- λ_1, λ_2 are the longitudes of the two points in radians,
- R is the Earth's radius (mean radius = 6,371 km),
- d is the distance between the two points.

This distance metric was calculated for each transaction and included as a new feature in the dataset, **distance_in_miles**. The hypothesis behind creating this variable was that a high distance indicates chances of fraud as the transaction location is far away from residential location.

- **Encoding Categorical Variables**

Some of the variables in the dataset are categorical, including **Merchant names, Spend categories, Card holder's Job categories**. To prepare these categorical features for machine learning models, they need to be transformed into a numerical format. Several encoding techniques were used based on exploratory data analysis (EDA).

- **Merchant Names (Category-based Encoding)**: Instead of using traditional one-hot encoding (which can be computationally expensive for a large number of categories), we applied a more sophisticated approach: we replaced each merchant name with the **average fraud rate** for that merchant. This method leverages the idea that certain merchants may have a higher propensity for fraud, and encoding them with their average fraud rate provides useful information for the model.
    - The fraud rate for each merchant was calculated as

$$Fraud\ Rate = (Number\ of\ Fraud\ Transactions) / (Total\ number\ of\ Transactions)$$

- **Spend and Job Categories (Label Encoding)**: The **Spend and Job category** variables represent the purchase category and cardholder's profession respectively, which are categorical features. Label encoding was applied to these features, where each unique job category was assigned, a numerical value based on its frequency of occurrence in the dataset.

5.4 Model Selection

Algorithms evaluated:

- *Logistic Regression*

Our paper compares two logistic regression models used for fraud detection: a standard logistic regression model (Model 1) and a logistic regression model with class weighting to handle imbalanced data (Model 2). The objective is to assess the performance of these models at different decision thresholds while also calculating the fraud detection savings and the cost associated with false positives. The savings are calculated based on the average fraud amount of ~525 USD for fraud cases, while the cost of a false positive is set as 1% of the fraud amount. Both models uses standard threshold of 50% for classifying fraud cases but we evaluated impact of the different thresholds.

Methodology:

- The thresholds for classification are varied from 0.01 to 0.5.
- Accuracy and Recall for fraud cases are tracked at each threshold.
- Fraud Savings are computed as the average fraud amount of 524.73 USD, which reflects the savings from detecting fraud cases.
  Fraud Savings=True Positives × Average Fraud Amount
- Cost of False Positives is calculated as 1% of the fraud amount, based on the assumption that investigating a false positive costs a fraction of the average fraud amount
  False Positive Cost=False Positives×0.01×Average Fraud Amount
- Net Savings: The difference between fraud savings and the cost of false positives

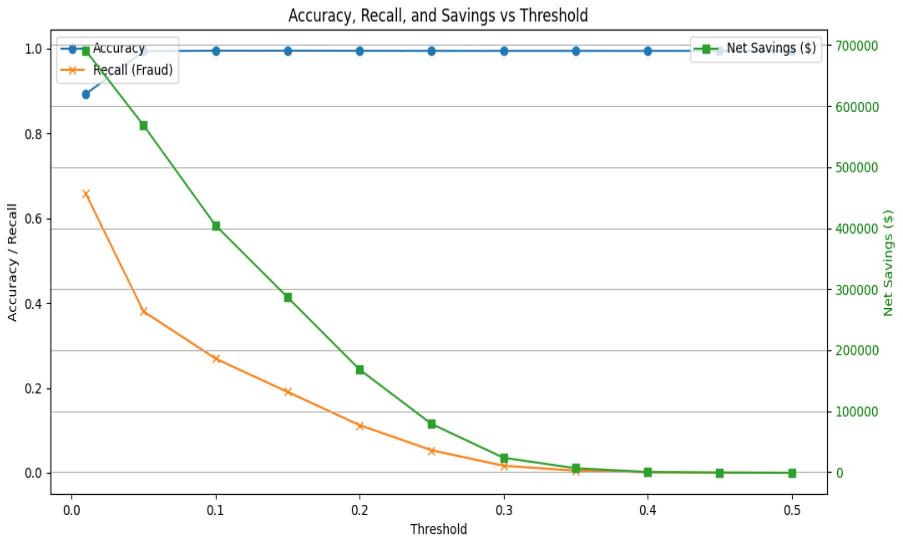


Figure 1 Model 1 Logistic Regression

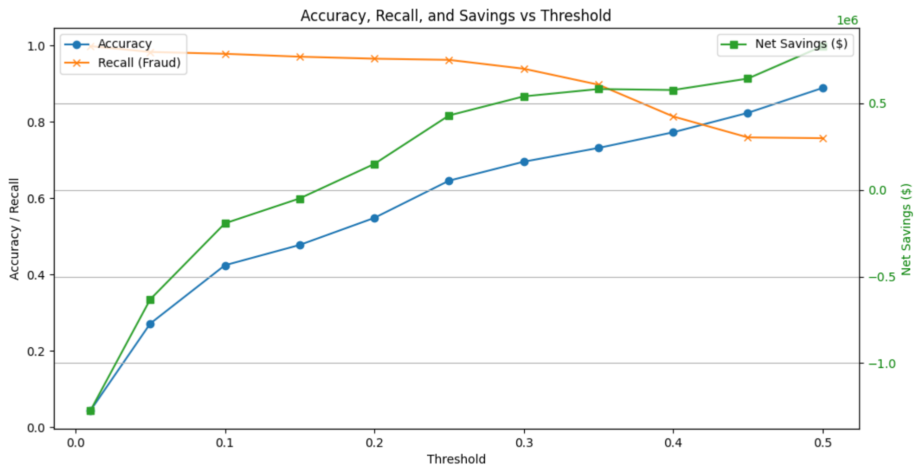


Figure 2 Model 2 Logistic Regression

Results:

Model 1 (No Class Weighting):

- Threshold 0.01: High accuracy (89.27%) and significant fraud savings (690,800.23 USD), but high false positive costs.
- Threshold 0.05: Improved accuracy (99.37%) with a slight decrease in fraud savings (568,815.59 USD). Good balance between fraud detection and accuracy.
- Threshold 0.5: High accuracy (99.44%) but negative savings (-1,107.19 USD), indicating the model is too conservative, missing fraud cases and incurring higher false positive costs.

Model 2 (With Class Weighting):

- Threshold 0.01: Low accuracy (4.33%) and negative savings (-1,273,678.66 USD) due to over-prediction of fraud cases (many false positives).
- Threshold 0.05: Accuracy improves to 27.12%, but savings remain negative (-632,045.90 USD).
- Threshold 0.5: Accuracy reaches 88.90%, and savings become positive (829,623.53 USD), indicating better fraud detection with fewer false positives.

Summary:

Model 1, which does not use class weighting, performs well at lower thresholds (0.01-0.05), achieving high accuracy and substantial fraud savings. However, as the threshold increases, while accuracy remains high, fraud savings diminish due to the growing cost of false positives. At higher thresholds (0.5), the model becomes overly conservative, resulting in negative net savings. Model 2, with class weighting, initially struggles with low accuracy and negative savings due to an over-prediction of fraud cases, but as the threshold increases, it shows improved performance with positive savings and higher accuracy. Model 2's effectiveness in managing false positives improves with higher thresholds, making it more suitable for cost-sensitive environments.

Conclusion/Recommendation:

For optimal fraud detection and savings, Model 1 is ideal when the priority is maximizing fraud detection, especially at lower thresholds like 0.05 to 0.1. However, if managing false positive costs is critical, Model 2 with class weighting is a better choice at higher thresholds (0.25-0.5), as it provides better precision and minimizes false positive costs, albeit with a slight trade-off in fraud detection. Ultimately, the choice of model and threshold depends on the balance between fraud detection accuracy and investigation costs.

- *Naive Bayes*

Naive Bayes is a probabilistic classifier that assumes the features are independent. It is based on Bayes' theorem and is particularly useful when dealing with categorical variables. Despite the strong assumption of feature independence, Naive Bayes often performs well in practical scenarios, especially with large datasets.

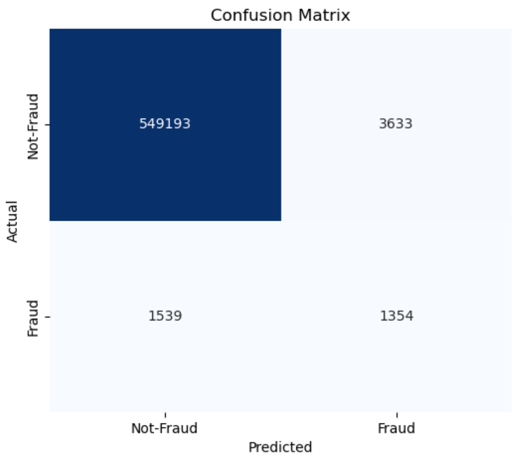
In fraud detection, Naive Bayes helps identify fraudulent transactions by calculating the probability of a transaction belonging to each class (fraud or not fraud). Its simplicity and efficiency make it a good starting point for classification tasks, although it may struggle with highly correlated features.

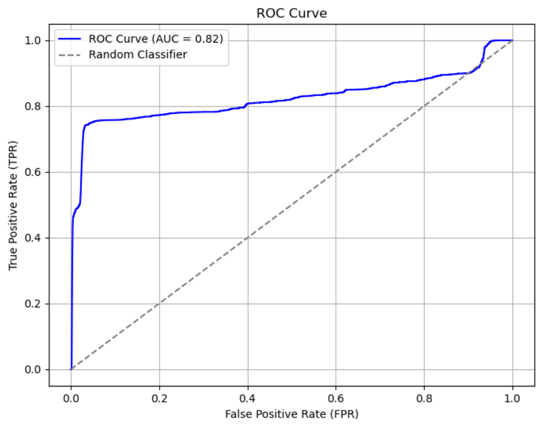
Gaussian Naïve Bayes was used in the model given all the variables are converted to numeric.

Performance:

- Classification metrics: A high accuracy is showing up but with a poor recall. Hence, the model with or without SMOTE does not show an ability to predict Fraud cases.
- AUC Curve: AUC score of 82% does show fair ability of the model to separate out Fraud from Non-Fraud cases.
- It will be advisable to compare this to some of the other conventional and machine learning techniques.

Model	Accuracy	Precision	Recall	ROC-AUC
Naïve Bayes with SMOTE	97.7%	14.7%	70.0%	81.7%
Naïve Bayes w/o SMOTE	**99.1%**	**27.2%**	**46.8%**	**82.3%**





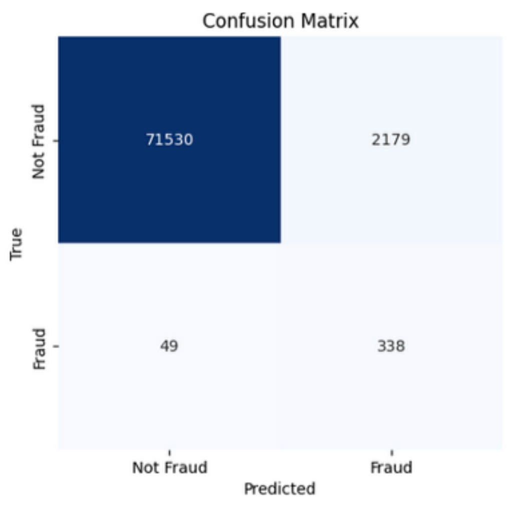
- *Random Forest*

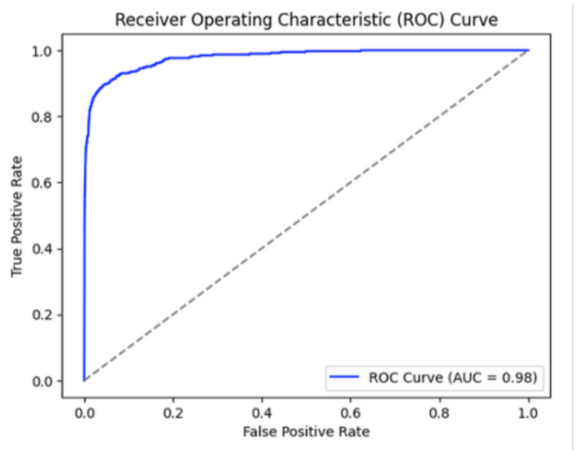
Random Forest is an ensemble learning technique used for both classification and regression tasks. Here multiple decision trees are trained on a random subset of the data with random subset of features. Prediction is made by taking majority vote(classification) or average(regression) of all the trees in the random forest model. Random Forests are robust to overfitting and can capture complex relationships in the data.

For fraud transaction detection we used oversampled data with around 17% fraud transactions to train a random forest model. During hyperparameter training we tried different values of max_depth, number of trees(or n_estimators) and min_samples_leaf and found that we got the best results at n_estimators = 50, max_depth=10, min_samples_split=10 and min_samples_leaf=10.

Performance: The model achieved a high accuracy score of 97% suggesting that it classifies 97% percent of the transactions correctly into fraud and non-fraud. The model also has a good recall of 87.3% which implies that it was able to identify 87% of the all the fraud cases. However, it struggled with a low precision score of 13.4% meaning that it is labelling many non-fraudulent transactions as fraud resulting in many false positives. ROC-AUC of 98.0% suggests that overall, the model performs well to distinguish between fraud and non-fraud transactions.

Model	Accuracy	Precision	Recall	ROC-AUC
Random Forest	99.0%	13.4.0%	87.3%	98.0%





- *Gradient Boosting Machines*

  XGBoost (Extreme Gradient Boosting) is an advanced machine learning algorithm based on gradient boosting, a technique where multiple weak learners (decision trees) are combined to create a strong predictive model. Unlike traditional decision trees, which build one tree at a time, XGBoost builds trees sequentially, with each tree learning from the errors of the previous one. This approach makes XGBoost highly effective for classification tasks, especially when dealing with imbalanced datasets, as it can focus on learning from difficult-to-predict cases (like fraud).

  Key features of XGBoost:

  - Gradient Boosting: XGBoost works by iteratively building decision trees where each subsequent tree attempts to correct the errors made by the previous trees.
  - Regularization: XGBoost includes a regularization term to penalize overly complex models and reduce the risk of overfitting.
  - Efficiency: XGBoost is optimized for both speed and memory, making it one of the fastest algorithms for classification tasks.
  - Handling Imbalanced Data: XGBoost has built-in support for handling imbalanced datasets through its use of custom loss functions and class weighting.

  Implementation for Fraud Detection: For this project, we used XGBoost to predict fraud cases in transaction data. The model was trained on features such as transaction amount, city population, merchant fraud probability, and time of day, using a standard binary classification approach where transactions are classified as either fraudulent or legitimate.

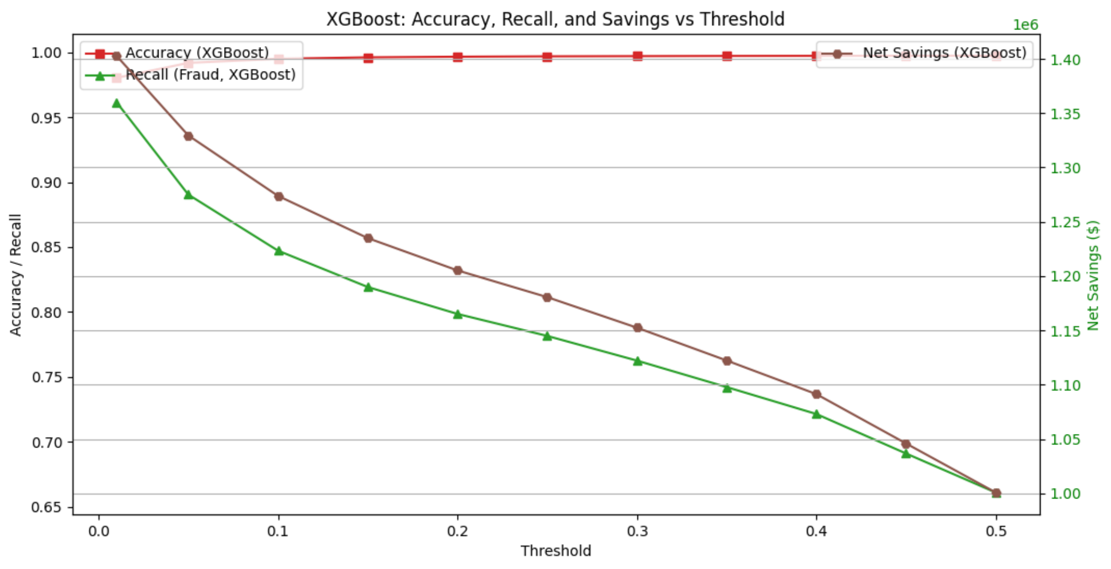


Figure 3 Model XGBoost

Results:

- Threshold 0.01: High accuracy (98.03%) and substantial fraud savings (1,402,904.61 USD), but with a high number of false positives.
- Threshold 0.05: Improved accuracy (99.18%) and slightly decreased fraud savings (1,329,536.47 USD). Balance between fraud detection and reducing false positives.
- Threshold 0.5: High accuracy (99.74%) but lower savings (1,000,728.40 USD), indicating the model is more conservative, focusing on precision but at the cost of missing some fraud cases.

Summary:

The XGBoost model performs well across various thresholds, with accuracy steadily improving as the threshold increases. At very low thresholds (0.01), fraud savings are maximized, but the high number of false positives increases operational costs. As the threshold increases, accuracy improves, and fraud savings decrease due to fewer false positives, which reduces the investigation cost. The model becomes more conservative as the threshold approaches 0.5, leading to higher precision but a trade-off in fraud detection (lower recall). The model strikes a balance between detecting fraud and minimizing the costs of false positives.

Conclusion/Recommendation:

For maximizing fraud detection and savings, thresholds between 0.01 to 0.1 are optimal, offering high accuracy and good fraud savings. However, if minimizing false positive costs is the primary goal, thresholds between 0.2 to 0.25 provide a better balance, as the model becomes more precise, though at the expense of slightly lower recall and fraud detection. The threshold choice depends on the trade-off between detecting fraud and managing operational costs.

- *Support Vector Machines (SVM)*

SVM is a powerful classifier that works well in high-dimensional spaces. It aims to find the hyperplane that best separates the two classes by maximizing the margin between them. SVM can work effectively with both linear and non-linear decision boundaries by using kernel tricks.

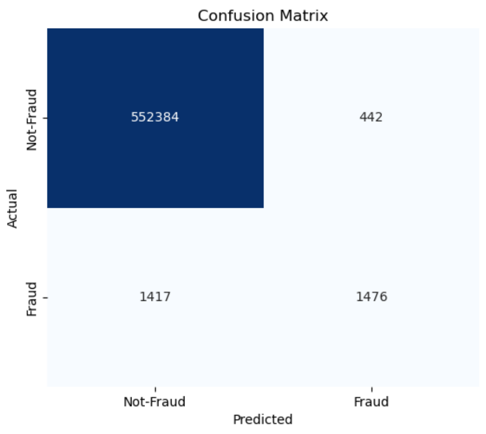
In fraud detection, SVM can help detect fraudulent transactions by finding the optimal boundary between legitimate and fraudulent transactions. Due to its flexibility, SVM can work well even with complex datasets, though it can be computationally expensive for large datasets.

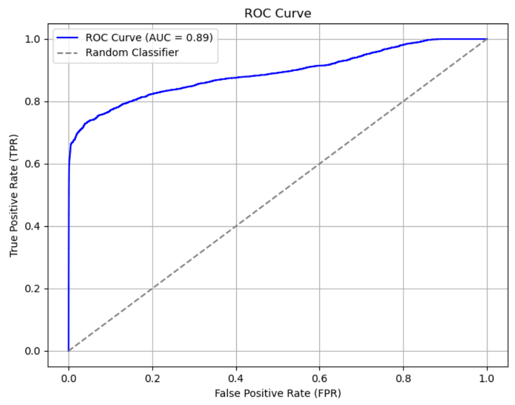
An optimum SVM classifier with RBF kernel and tuned parameters with 'C': 10, 'gamma': 'scale' is created after various iterations.

Performance:

- Classification metrics: Much better performance than Naïve bayes is seen. High accuracy and precision is seen but recall is still a challenge.
- AUC Curve: AUC score of 89% shows good ability of the model to separate out Fraud from Non-Fraud cases.
- While SVM showed a significant gain in prediction, it does pose some concerns over recall ability.

Model	Accuracy	Precision	Recall	ROC-AUC
SVM (RBF)	99.7%	77.0%	51.0%	89.0%





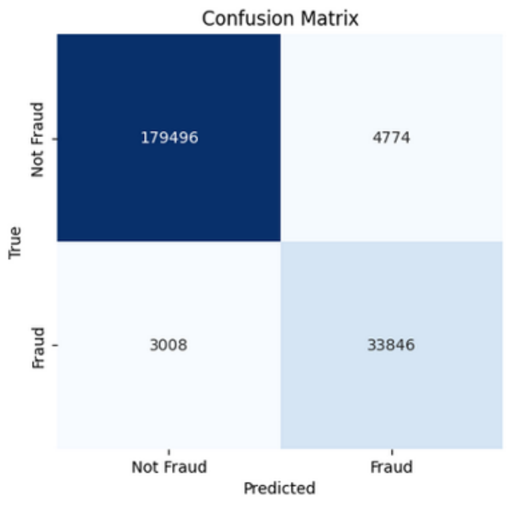
- *Neural Networks*

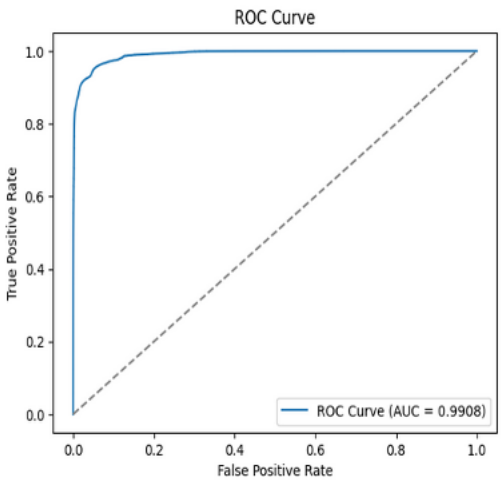
Neural networks are a supervised learning model which can be used to assign input test data into predefined categories. It consists of interconnected layers that maps input to output labels through training on labeled data. Neural network models are great at handling complex non-linear patterns in the data and are widely used in tasks like fraud transaction detection, image recognition etc.

Here we trained a customizable and flexible neural network model using Keras(via tensorflow) with two hidden layers and a sigmoid output layer for predicting classification probabilities. Input layer was a fully connected layer with 32 neurons and ReLU activation. The hidden layer was a fully connected layer with 16 neurons. The output layer was a single neuron with sigmoid activation function using cross entropy loss.

Performance: We can see here that although the accuracy of the model is not as high as the other models, but it achieved the highest recall of 91.84% implying that it was able to capture the greatest number of fraud transactions while reducing the false positives as evident from 87.6% precision. Overall, this is a more general model which was able to capture the most nonlinear relationship between the features. ROC-AUC of 99.1% suggests excellent classification ability.

Model	Accuracy	Precision	Recall	ROC-AUC
Neural Network	96.5%	87.6%	91.8%	99.1%





- *Anomaly detection methods*

One-Class SVM is an unsupervised learning model that is used for anomaly detection. It assumes that most of the data points belong to a single class (non-fraudulent transactions) and detects the outliers (fraudulent transactions). It is particularly useful when the dataset is heavily imbalanced, and fraud cases are rare.

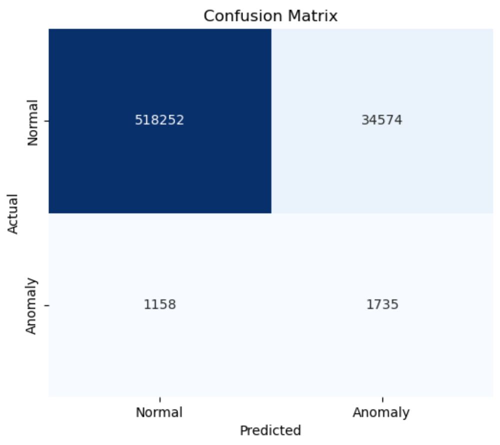
In this project, One-Class SVM was trained only on the non-fraudulent transactions, and it was expected to detect fraud by identifying deviations from the learned patterns of legitimate transactions. This method is useful in fraud detection where there are far fewer fraud cases compared to legitimate ones.

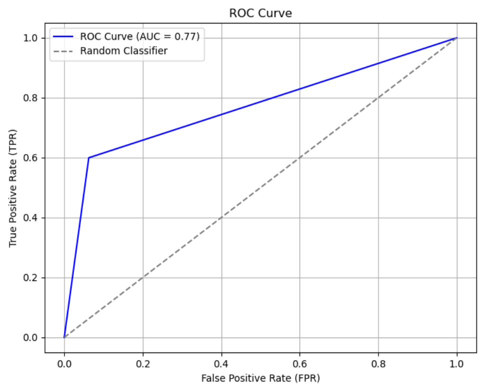
An optimum one class SVM classifier with RBF kernel and tuned parameters with 'nu': 0.06, 'gamma': 'scale' is created after various iterations.

Performance:

- Classification metrics: Shows poor performance compared to Naïve bayes and SVM. While Recall is looking better, Precision is unacceptable.
- AUC Curve: AUC score of 77% shows poor ability of the model to separate out Fraud from Non-Fraud cases. Also, the curve is not monotonous due to the fact that it creates 1/0 a and no continuous prediction probability unlike some of the other techniques.
- The poor performance indicates how a non-supervised algorithm is ineffective to solve such a complex problem and is also reflective of how the technique is incapable of handling an imbalanced data like this.

Model	Accuracy	Precision	Recall	ROC-AUC
Anomaly Detection (One Class SVM)	93.6%	4.8%	60.0%	76.9%





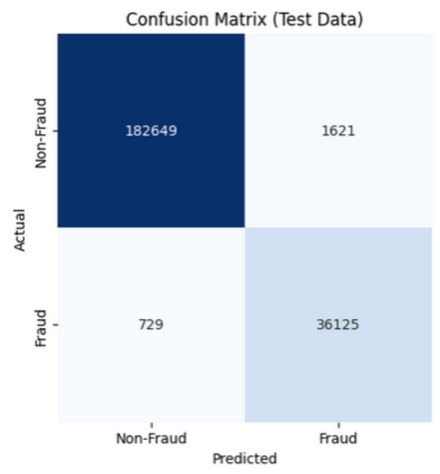
- *K-Nearest Neighbor Classifier*

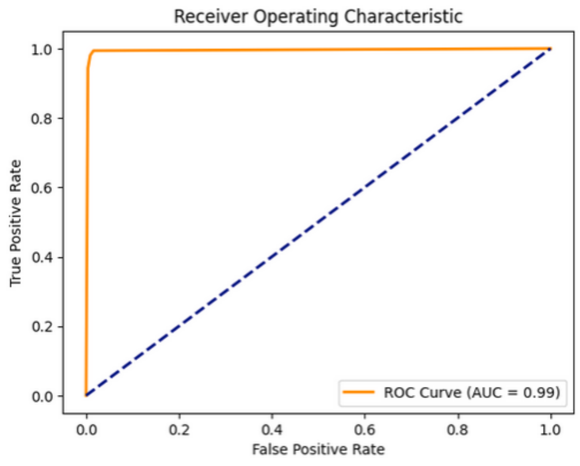
K-Nearest Neighbor (KNN) is a supervised learning technique that can be used for classification tasks. It identifies the k nearest data points (neighbors) to a given data point and assigns a majority class label among these neighbors. KNN is simple to understand and effective for detecting fraudulent transactions by grouping similar data points based on proximity in the dataset.

Since the choice of k (or number neighbors) affects the model performance, we performed parameter tuning and found that for k= {8-12} we were getting good performance. This model also faced issues with under-sampled dataset, so the data set was oversampled using SMOTE. This improved the overall performance of the model.

Performance: The performance of the KNN model was surprisingly exceptional which goes on to show that simpler models often perform better. The accuracy, precision and recall are both above 98% indicating that it is an excellent overall model which avoids overfitting and produces good results.

Model	Accuracy	Precision	Recall	ROC-AUC
K-Nearest Classifier	99.0%	98.0%	98.0%	98.0%





5.5 Model Evaluation

S.No	Model	Accuracy	Precision	Recall	ROC-AUC
1.1	Logistic Regression(0.05 threshold)	99.%	39%	38%	68.8%
1.2	Logistic Regression with Balanced data	88.8%	3%	76%	82.4%
2.1	Naïve Bayes with SMOTE	97.7%	14.7%	70.0%	81.7%
2.2	Naïve Bayes w/o SMOTE	99.1%	27.2%	46.8%	82.3%
3	Random Forest	97.0%	13.4%	87.3%	98.0%
5	XGboost(0.3 threshold)	99.7%	70%	76%	88%
5	SVM (RBF)	99.7%	77.0%	51.0%	89.0%
6	Anomaly Detection (One Class SVM)	93.6%	4.8%	60.0%	76.9%
7	Neural Network	96.5%	87.6%	91.8%	99.1%
8	K-Nearest Classifier	99.0%	98.0%	98.0%	98.0%

6. Results

The results of the fraud detection models are summarized in the table above, showcasing the performance of various algorithms across key metrics such as accuracy, precision, recall, and ROC-AUC.

Key Insights:

- **K-Nearest Neighbors (KNN)** and **Neural Networks** emerged as the best-performing models with the highest recall (98% and 91.8%) and ROC-AUC (99%), making them highly effective at detecting fraudulent transactions while maintaining strong precision (87.6%).

- **XGBoost (0.3 threshold)** offered a robust balance of precision (70%) and recall (76%), making it a strong candidate for practical deployment where both fraud detection and false positive minimization are priorities.
- **SVM (RBF)** achieved the highest precision (77%) but had a lower recall (51%), indicating its potential for scenarios where minimizing false positives is critical.
- **Logistic Regression** and **Naïve Bayes** models struggled with either low recall or precision, particularly when data imbalance was not addressed adequately.

Comparison Visualizations:

- **Confusion Matrices:** Highlighting the trade-offs in false positives and false negatives across models.
- **ROC Curves:** Demonstrating the classification performance for the top models (Neural Networks, XGBoost, and SVM).
- **Precision-Recall Curves:** Emphasizing the models' effectiveness at different thresholds.

7. Conclusion and scope for Future Discussion

Conclusion:

- The study demonstrated the effectiveness of various machine learning algorithms in detecting fraudulent credit card transactions. Key conclusions include:
- KNN and Neural Networks achieved superior performance, making them the ideal choice for applications where maximizing fraud detection (recall) is critical.
- XGBoost, with its balance of precision and recall, is well-suited for operational environments where minimizing false positives is equally important.
- The choice of the optimal model depends on the specific business objectives, such as prioritizing fraud detection versus minimizing operational costs.

Challenges:

- **Imbalanced Dataset:** The dataset's inherent imbalance posed significant challenges for traditional models, requiring techniques like SMOTE to boost recall.
- **Computational Complexity:** Training Neural Networks and ensemble methods (like Random Forest and XGBoost) required substantial computational resources.

Future Scope:

- **Explainability:** Develop interpretability frameworks for complex models like Neural Networks to increase user trust.
- **Feature Engineering:** Explore additional domain-specific features, such as cardholder transaction history or social network analysis.

- **Hybrid Models:** Combine the strengths of Neural Networks and XGBoost to further enhance performance.
- **Scalability:** Optimize models to handle larger datasets for deployment in high-transaction environments.
- **Real-Time Fraud Detection:** Incorporate streaming data techniques to enable real-time fraud detection.
- The findings of this project provide a strong foundation for designing robust, scalable fraud detection systems, contributing significantly to financial security and operational efficiency.

8. References

Data - https://www.kaggle.com/datasets/kartik2112/fraud-detection/data

9. Appendices

- Link to Git hub for codes: https://github.gatech.edu/aasthana41/ISYE-6740