# Credit Card Fraud Detection

Andromeda - Pranathi Alluri, Sanjana Kandunoori, Anagha Kadoo, Sinuo Song, Chen Li
Northeastern University

# Abstract

With the increase of online transactions and ease of access to personal information, credit card fraud continues to be a challenge that costs billions of dollars annually. This not only threatens individuals financial security but also erodes trust in e-commerce systems. Millions of real-time transactions are made on a daily basis, making it difficult to distinguish between legitimate and fraudulent transactions quickly. Furthermore, the methods used by fraudsters are always evolving, making it hard to pinpoint breaches. The vast amount of information, as well as the need to find patterns and make predictions, makes this not only a relevant, but an important data mining problem. Detecting fraudulent transactions in real-time is essential to guarding financial systems and protecting consumers.

In this project, well-known machine learning models for classification, including logistic regression, support vector machines (SVMs), decision trees, random forest, and k-means clustering, are applied to generated transaction data. The project also simulates a dataset highlighting most of the issues that are possible in fraud detection for real-world data. However, the dataset generated from the simulation is an imbalanced dataset. To mitigate any potential biases arising from imbalanced data, we have employed the Borderline SMOTE technique. To ensure robust model performance, a comprehensive k-fold cross-validation approach is adopted, enabling the fine-tuning of hyperparameters for each model. Subsequently, the selection of the optimal fraud detection model is based on an evaluation of key performance metrics, including accuracy, precision, and recall. This evaluation uncovers how each model performs in the intricate landscape of fraud detection.

By training the dataset with the specified models and assessing their performance using the accuracy metric, it becomes evident that the Random Forest and Decision Tree models showcase the highest accuracy rates, demonstrating their robust ability to make precise predictions regarding credit card fraud transactions. On the other hand, K-means Clustering, while useful for grouping data points, yields a considerably lower accuracy, highlighting its limitations in supervised prediction tasks.

# Introduction

The financial industry faces an ongoing and dynamic challenge with credit card fraud due to the growing popularity of digital transactions and online commerce. Fraudsters are constantly developing sophisticated methods to exploit payment system vulnerabilities. Real-time detection of fraudulent credit card transactions is essential to prevent financial losses, safeguard consumers' assets, and uphold confidence in the financial ecosystem. Therefore, this project seeks to propose an efficient solution leveraging advanced machine learning algorithms to accurately and promptly identify fraudulent activities, addressing the urgent demand for an effective credit card fraud detection system [7].

The importance of solving the credit card fraud problem cannot be overstated due to its wide-ranging impact on various stakeholders in the financial ecosystem. Credit card fraud may result in significant financial losses for both financial institutions and consumers. By effectively detecting and preventing fraudulent activities, these losses can be minimized, leading to more stable financial operations for banks and protecting consumers' hard-earned money.

In addition, credit card fraud incidents might erode consumers' trust in financial institutions and online transactions. A successful fraud detection system instills confidence in consumers, encouraging them to use credit cards for their purchases and online transactions, thereby fostering a healthy and thriving digital economy.

Furthermore, a successful credit card fraud detection system reduces the likelihood of false positives and false negatives, resulting in fewer legitimate transactions being flagged as fraudulent. This ensures a smoother and seamless experience for consumers during their financial transactions.

As fraudsters continuously develop new and sophisticated fraud techniques, it is crucial for the financial industry to keep up with the evolving landscape. An advanced fraud detection system using cutting-edge machine learning algorithms allows financial institutions to stay ahead of fraudsters and detect emerging patterns effectively. Therefore, resolving the credit card fraud problem is of utmost importance in safeguarding financial institutions, protecting consumers' assets, maintaining trust in digital transactions, and upholding the stability and integrity of the entire financial ecosystem. By addressing this pressing issue with advanced machine learning algorithms, the project aims to provide a practical and effective solution that will have a positive impact on the financial industry and the global economy as a whole.[8]

# Methodology

To develop the best model for binary classification on credit card transactions, as fraudulent or not, various common machine learning models will be run on a generated dataset of credit card transactions.

## 2.1 Data Description

The dataset utilized in this study has been generated using a transaction data simulator [1]. This simulated dataset serves as an approximation of reality, albeit with a simpler design compared to the intricate dynamics that underlie real-world payment card transaction data. By imposing straightforward preliminary rules, the simulator generates transactions and fraudulent behaviors, aiding in the interpretation of patterns discerned by diverse fraud detection techniques.

To be specific, generating the data included creating customer and terminal datasets, which included geographical locations, unique_ids, and spending habits. Then, the simulator iterates over the customer profiles and generates transactions, based on proximity to terminals (within a 4500 km radius). In the final step, transactions are categorized as legitimate or fraudulent. For our dataset we used two types of fraud scenarios:

1. The first scenario represents phishing where two terminals are randomly selected each day and all transactions on these terminals are labeled as fraudulent over the next 28 days.
2. The second scenario represents stolen card numbers where three customers are randomly selected each day. Over the next 14 days, 1/3 of their transactions have their amounts multiplied by 5 and are marked as fraudulent.

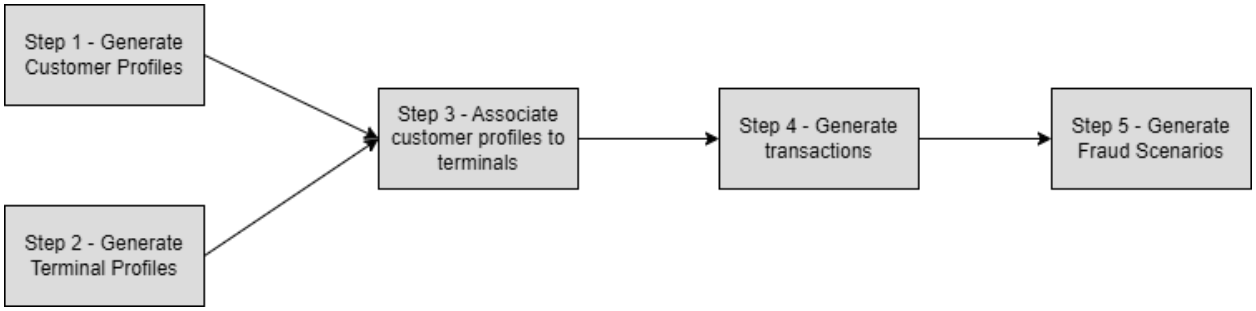



Fig 1. Transaction Data Generation Process

In addition to its interpretative value, the simulated dataset, with 1,000,000 transactions, sheds light on the challenges encountered by fraud detection practitioners when working with genuine

data. The dataset encompasses several critical aspects, including class imbalance, where fraudulent transactions represent less than 1% of the total, a combination of numerical and categorical features, intricate relationships between features, and scenarios of fraud evolving over time.

Importantly, the decision to use a generated dataset stems from the sensitivity of real credit card transactions. Handling authentic transaction data raises concerns about the confidentiality of sensitive information. To ensure the privacy and security of customers' personal details, a generated dataset that emulates authentic transaction patterns is adopted. This approach not only mitigates the risk of exposing sensitive information but also allows for comprehensive analysis and experimentation without compromising the integrity of individuals' personal data.

## 2.2 Data Preprocessing

In order to deal with the imbalance of credit card transaction data, Borderline SMOTE was run to generate instances of the minority class, and balance the data. Feature engineering, including finding fraud rates and total transactions, were then run on terminal and customer ids with a window of 30 days, to better detect fraud scenarios without overfitting to IDs. After dropping categorical variables, the data was normalized using min-max scalar and split into 80% training and 20% testing.

## 2.3 Creating Machine Learning Models

### 2.3.1 Logistic Regression

Logistic Regression is a type of statistical model used for binary classification tasks. It serves as an effective tool for predicting the probability that a given input instance belongs to one of two possible outcomes. based on prior observations of the data set. The model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. In this context, the model predicts the legitimacy of a credit card transaction by analyzing various attributes, including the transaction amount, the location or terminal where the transaction occurs, and the timestamp of the transaction, among others. During training, the model learns the optimal weights for the input features, aiming to minimize the difference between predicted probabilities and the actual class labels. Once trained, the model can predict the probability of an instance belonging to the positive or negative class. This model is relatively faster to train than the rest of the models discussed below.

### 2.3.2 SVM

Unlike logistic regression, which models the probability that a given data point belongs to a specific class, Support Vector Machines (SVM) aim to find the hyperplane that optimally separates the two classes. In our context, we utilize soft margin SVM to account for some misclassification errors, thereby enhancing the model's generalization to unseen data.

Three primary hyperparameters require careful tuning for SVM – the regularization parameter (C), gamma (controlling the kernel width), and the choice of kernel type. The overarching objective is twofold: maximize the margin between classes while minimizing classification errors. However, given the extensive dataset, performing 20-fold fits for each of the 80 parameter combinations (totaling 1600 fits) would pose significant computational demands, potentially requiring several days to complete.

To overcome this computational challenge, we make use of the linear kernel (linearSVC) due to its lower computational complexity compared to other kernels. Additionally, linear kernels only have a single hyperparameter, C, which controls the trade-off between maximizing the margin and minimizing classification errors.

To find the optimal C value, Bayesian search cross-validation is employed. This approach efficiently explores the hyperparameter space while leveraging the results of previous iterations. 10 iterations with 5 cross-folds were used, striking a balance between thorough parameter exploration and computational efficiency.

### 2.3.3 Decision trees

Using decision trees for credit card fraud detection is a practical and effective approach to identify potentially fraudulent transactions within large datasets. Decision trees can automatically learn patterns and features in the data that distinguish between legitimate and fraudulent transactions, aiding in building a predictive model for fraud detection. However, decision trees are prone to overfitting, especially when they're deep and complex[4]. At the beginning, I trained a model using the DecisionTreesClassifier without setting any parameters to get a picture of the general performance. It looks good on the surface. However, the train score is 1.0 which indicates potential issues like overfitting.

In order to prevent overfitting and improve generalization, I tried to do a little hyperparameter tuning[5]. I selected ‘max_depth’ to define the maximum depth of the tree, since deeper trees can model more complex relationships but are more prone to overfitting. I selected ‘min_samples_split’ to define the minimum number of samples required to split an internal node, since larger values can prevent overfitting by ensuring a

minimum number of samples in each split. I selected ‘max_features’ to define the number of features to consider when looking for the best split, since it can take different values, each influencing how the features are selected. Then using Grid Search Cross Validation specified a range of possible values to explore. Instead of evaluating accuracy, precision, recall, F1-score for each combination of hyperparameters. I used Scikit-learn's GridSearchCV to make the process easier, let the Grid Search systematically find the best combination of hyperparameters that maximizes accuracy on the validation sets.

### 2.3.4 Random Forest

To experiment with different algorithms, Random Forest [2] can be a choice which is a suitable model for binary classification tasks like credit card fraud detection. Unlike Decision Trees, Random Forest is an ensemble model that combines multiple decision trees using different random subsets of the data and features to make predictions. It’s a learning algorithm that can handle both categorical and numerical data, making it effective for detecting fraudulent transactions based on various features. It combines multiple decision trees to make predictions and is known for its robustness, accuracy, and resistance to overfitting.

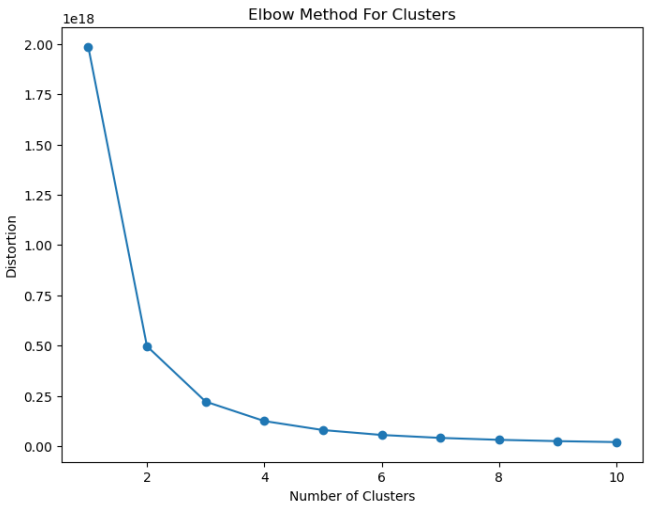
Before training the Random Forest model, tuning hyperparameters [3] for a Random Forest is essential to optimize the model's performance. Two crucial hyperparameters to fine-tune are "n_estimators" and "max_depth." The "n_estimators" controls tree count, affecting complexity and overfitting. "max_depth" limits tree depth, impacting complexity and pattern capture. "min_samples_split" sets node split threshold, reducing noise-based nodes. In addition to tuning "n_estimators" and "max_depth," two other important hyperparameters for Random Forest are "min_samples_split" and "min_samples_leaf." The "min_samples_split" sets node split threshold, reducing noise-based nodes. "min_samples_leaf" enforces leaf sample minimum, preventing noise-driven predictions. These hyperparameters collectively curb complexity and overfitting, resulting in a robust, accurate model for new data. By carefully selecting appropriate values for these parameters during the hyperparameter tuning process, we can build a more robust and accurate model that generalizes well to new, unseen data.

### 2.3.5 K-Means Clustering

Using K-Means for credit card fraud detection involves treating the transaction data as points in a high-dimensional feature space and partitioning them into K clusters based on similarity. The algorithm aims to minimize the distance between each point and the

centroid of its assigned cluster. To determine the number of clusters K we can perform a clustering analysis using the distortion for each value of K.

By calculating the distortion (inertia) - a measure of how spread out the data points are within their assigned clusters - for different cluster counts we can determine the “elbow point” to determine the optimal number of clusters K. This can be done by generating a plot showing the number of clusters on the x-axis and the corresponding distortions on the y-axis. This plot is used to visually identify the "elbow point," where the distortion begins to decrease at a slower rate, helping to determine an optimal number of clusters for the data. For our dataset the optimal number of clusters is found to be 2 which is used for clustering.




Furthermore the dataset was divided with a 80-20 split where 20 percent was used as testing data. The init parameter was set to k-means++ which is used for deciding the initial cluster centroids before the algorithm starts iterating to optimize the clusters. The algorithm parameter was set to the value elkan and the random_state was set to 0.

# Results & Analysis

## 3.1 Model Evaluation and Visualization

The dataset used is divided into training and testing subsets with an 80-20 split, where 80% of the data is used for training and 20% for testing. The models are trained on the training data, learning patterns from the input features to predict corresponding binary labels. Once trained, the model is employed to predict labels for the test dataset, and these predictions are stored in a prediction array. In addition, to assess the model's performance, evaluation metrics like Classification Report and ROC-AUC curve is used. Classification report summarizes the model's performance, offering metrics like precision, recall, F1-score, and support for different classes. It evaluates how well the model performs in a classification problem. On the other hand, the ROC-AUC curve is particularly useful for understanding the model's ability to discriminate between the positive and negative classes across different thresholds.

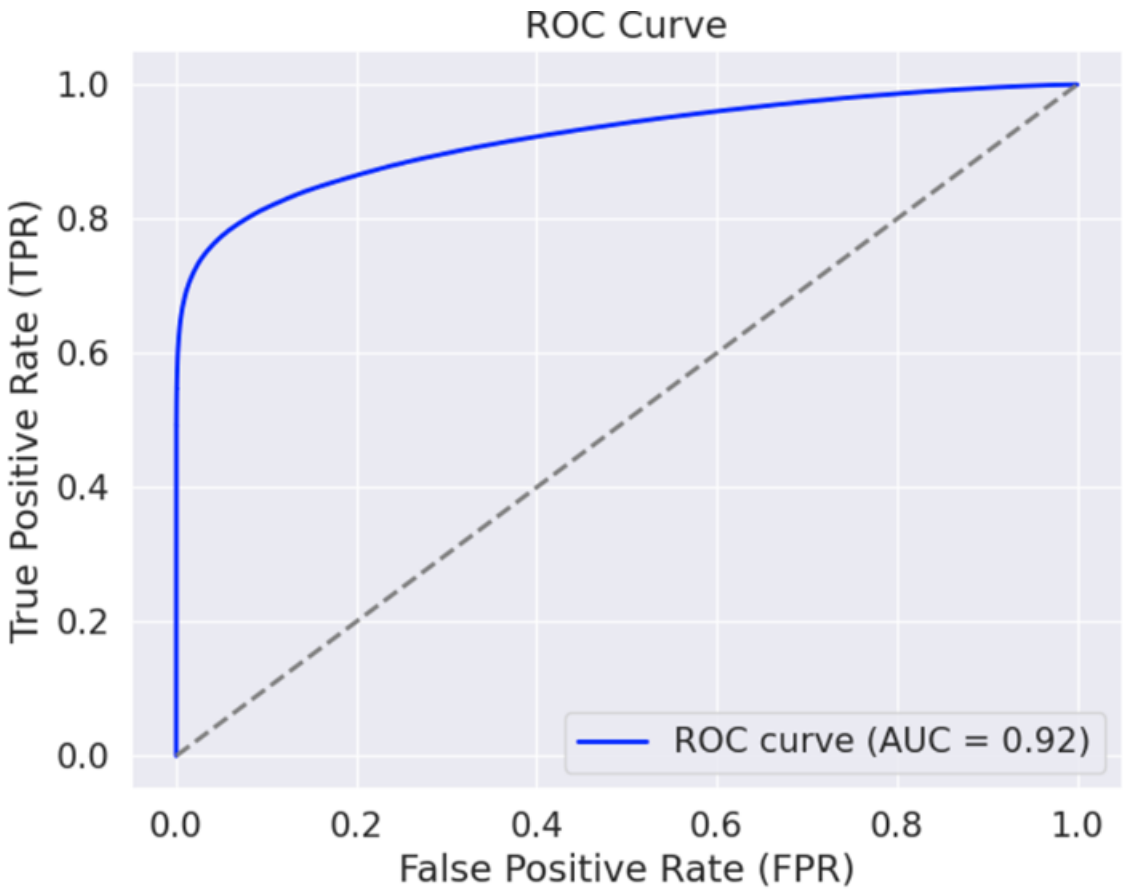
### 3.1.1 Logistic Regression - Results

The logistic regression model is initialized with a maximum of 1000 iterations for convergence. Based on the classification report provided, among all instances predicted as fraud, 90% of these predictions are indeed correct. Additionally, the model can identify 81% of the actual instances that belong to the fraud class.

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.91 | 0.87 | 383551 |
| 1 | 0.90 | 0.81 | 0.85 | 384292 |
| accuracy | | | 0.86 | 767843 |
| macro avg | 0.86 | 0.86 | 0.86 | 767843 |
| weighted avg | 0.86 | 0.86 | 0.86 | 767843 |

The accuracy of the model, representing the proportion of correct predictions out of the total predictions, is calculated, and reported as approximately 86%. The ROC AUC score, indicating the model's ability to distinguish between the classes, is 0.92. Looking at the graph below, it's evident that the curve is positioned closer to 1, indicating the model's significant performance.




**An accuracy of 86% and a high ROC AUC score of 0.92, signifies the model's proficiency in addressing the challenge of identifying fraudulent transactions.**

### 3.1.2 SVM - Results

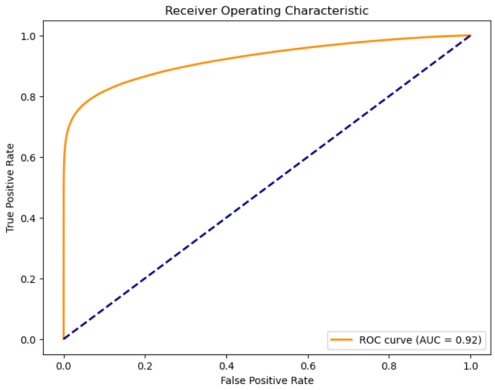
After running Bayesian cross validation it was found that the best parameter value for C, taking into consideration accuracy and computational power, was 0.15. The choice of a small value for C is influenced by the presence of a large dataset with concept drift. This requires a larger margin and more tolerance for classification errors to create a model that generalizes well over time.

Fitting the entire model on the data, resulted in an accuracy of 86% on both the training and testing data. Looking at the precision scores it seems that when a model is predicted

as fraudulent it's correct 91% of the time, which is quite high. However, the model struggles at actually identifying fraudulent transactions in the first place, as seen with a recall value of 80%.

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.82 | 0.92 | 0.87 | 383551 |
| 1 | 0.91 | 0.80 | 0.85 | 384292 |
| accuracy | | | 0.86 | 767843 |
| macro avg | 0.87 | 0.86 | 0.86 | 767843 |
| weighted avg | 0.87 | 0.86 | 0.86 | 767843 |




The similar results to logistic regression are not surprising, due to the use of a linear kernel. After all, with such a large dataset SVM training becomes slow, and the optimization process might not converge properly, making RBF and polynomial kernels extremely inaccurate and inefficient for credit card fraud classification. Also, since SVM aims to find the optimal hyperplane to best separate the classes, it struggles when the data is not well-separated or has overlapping regions. Since BorderLine SMOTE was used to generate synthetic samples for borderline instances of the minority class, near the decision boundary, this might be causing SVM to not perform as well as expected. [6]

With the provided results and weakness of SVM with extremely large datasets, it's evident that it's not the best model to use for credit card fraud classification.

### 3.1.3 Decision trees - Results

With the decision trees classifier, the optimal accuracy the model can achieve is 0.9682, ROC-AUC Score is 0.97, which shows a pretty good performance in credit card fraud detection scenarios.

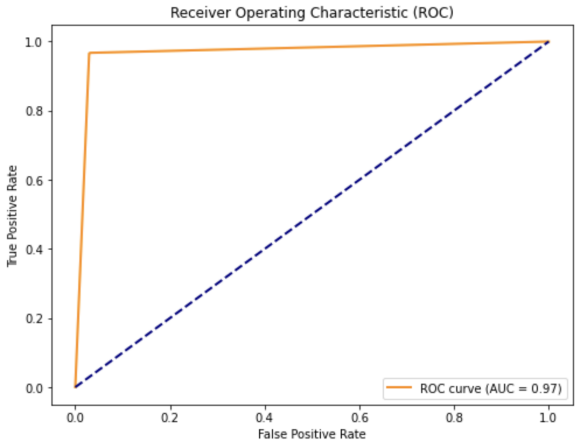
The best_params give the optimal train score is 0.9999 and the test score is 0.96821, which has ignorable differences from the DecisionTreesClassifier. Also, comparing the tuned best model with the classifier on classification reports, they only have minor differences.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97    383551
           1       0.97      0.97      0.97    384292

    accuracy                           0.97    767843
   macro avg       0.97      0.97      0.97    767843
weighted avg       0.97      0.97      0.97    767843
```

```
Precision: 0.9705350138591078
Recall: 0.9658072507364192
F1-score: 0.9681653606569349
ROC-AUC: 0.96896281828251
```




The Precision value is 0.970 indicates that the model's positive predictions are highly accurate. The Recall is 0.9666 suggests that the model is good at minimizing the number of positive instances that are being missed or incorrectly classified as negative. The F1-score is 0.968 which indicates a high balance between precision and recall especially working on our uneven class distribution. The ROC curve toward the top-left corner, indicating a higher True Positive Rate for a given False Positive Rate. The AUC is 0.97 which performs perfect classification.

In conclusion, the Decision Trees model performs very well on our dataset. The main reason for that is because it's a non-linear dataset. The advantage of decision trees classifier is partition the feature space into regions and fit different decision boundaries for each region. Also, decision trees are less sensitive to outliers compared to some other algorithms like linear models.

### 3.1.4 Random Forest - Results

By conducting a systematic search, such as grid search or random search, across a range of values for these parameters, we can strike a balance between model complexity and generalization to find the optimal combination that maximizes the Random Forest's predictive power.

To train the Random Forest model, one set of parameters was manually tried for the RandomForestClassifier, named rf_model. The model achieved an accuracy of **88.63%**, showing a reasonable performance. The precision and recall scores were high at **96.16%**

and **80.49%**, respectively, indicating that the model is good at correctly identifying positive cases and has a high true positive rate. The F1 score, which combines precision and recall, was **87.63%**, indicating a balanced performance between precision and recall. The ROC-AUC score was **96.03%**, suggesting that the model has a good ability to distinguish between positive and negative cases.

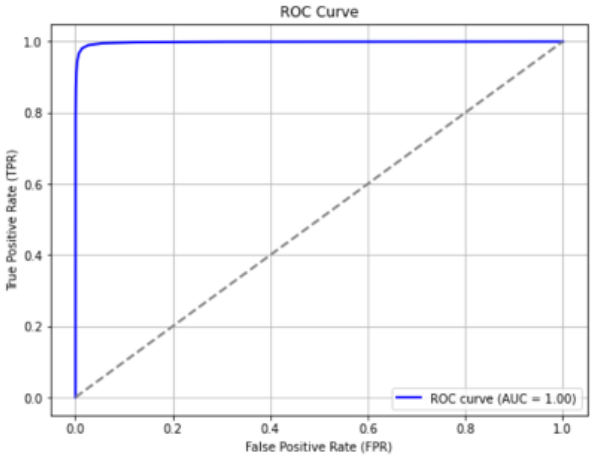
Furthermore, RandomizedSearchCV was used to find the best parameters for the model, which resulted in the similar feature importances as before. The model trained using the best parameters achieved a better performance to rf_model, with much higher accuracy of **97.99%**, precision of **99.28%**, recall of **99.69%**, and F1 score of **97.96%**, and a higher ROC-AUC score of **99.81%**.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98    383551
           1       0.99      0.97      0.98    384292

    accuracy                           0.98    767843
   macro avg       0.98      0.98      0.98    767843
weighted avg       0.98      0.98      0.98    767843

Precision: 0.9927671762863846
Recall: 0.9668663412196975
F1 Score: 0.9796455907129052
ROC-AUC Score: 0.9980771196410041
```




Overall, the Random Forest models showed promising performance in the given machine learning application. The feature importances remained consistent across the models, indicating the importance of selected features in the classification task. The high precision and recall scores suggest the model's effectiveness in correctly identifying positive cases, while the high ROC-AUC scores demonstrate its ability to distinguish between positive and negative cases. These results make the Random Forest model a suitable choice for the given dataset and classification problem.

### 3.1.5 K-Means Clustering - Results

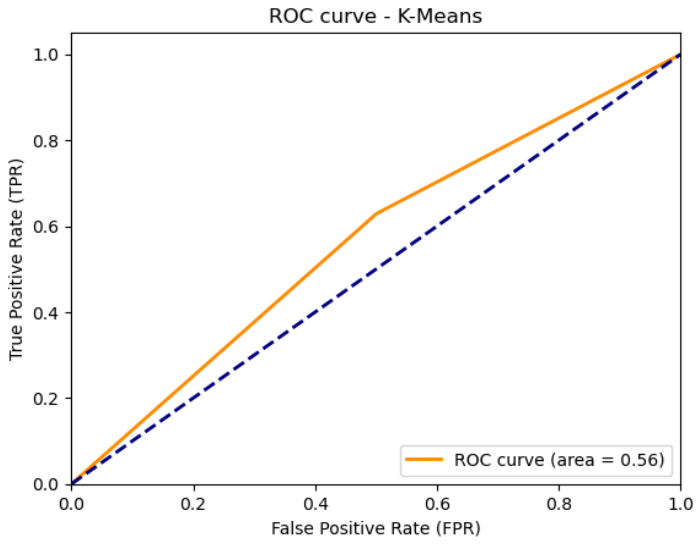
Using the value 2 as K for clustering the K-Means algorithm produced the following results. The model had a Recall score of 0.6289 i.e. 62%. The Accuracy of the model was found to be low 0.5002 i.e. 50%. The ROC-AUC score was found to 0.5641 i.e. 56%. The F-1 score was found to be 0.0167 which is less than 2%. However since in clustering there are no true positives or negatives as in the case of classification models, precision, recall, and consequently the F1 score are not good metrics for performance evaluation. On the other hand the value of Davies Bouldin score was found to be 0.500 indicating that the clusters generated are reasonably well-separated and distinct from each other.

```
Classification Report :
              precision    recall  f1-score   support

           0       0.99      0.50      0.67    383551
           1       0.01      0.63      0.02    384292

    accuracy                           0.50    767843
   macro avg       0.50      0.56      0.34    767843
weighted avg       0.99      0.50      0.66    767843

Accuracy :  0.5002667259390976
F-1 score :  0.01672395847000271
Recall score :  0.6289473684210526
Davies Bouldin Score :  0.5000290444611917
```




K-Means, originally used for clustering purposes, could encounter challenges in accurately pinpointing instances of fraud. Furthermore the optimal number of clusters (K) needs to be predefined in K-Means algorithm, which might not align with the natural distribution of fraud cases for this dataset. This could be the reason for the less than optimal performance as compared to the other algorithms.

## 3.2 Comparison of different methods

The table below shows the performance comparison table across all models. Based on the accuracy comparison of various models, it is evident that Decision Tree and Random Forest models exhibit the highest accuracy rates, achieving 97% and 98% respectively. These models demonstrate a strong ability to make correct predictions on credit card fraud detection.

| Model | Accuracy (in percent) |
|---|---|
| Logistic Regression | 86 |
| SVM | 86 |
| Decision Tree | 97 |
| Random Forest | 98 |
| K-Means Clustering | 50 |

On the other hand, Logistic Regression and SVM models both attain an accuracy of 86%, indicating moderate predictive performance. K-means Clustering, while useful for grouping data points, yields a considerably lower accuracy of 50%, highlighting its limitations in supervised prediction tasks. Therefore, among the models evaluated, the ensemble methods of Decision Tree and Random Forest prove to be the most accurate in capturing patterns within the dataset and generating accurate predictions.

# Conclusions

After an in-depth analysis of performance metrics across various machine learning models including Logistic Regression, and Decision Tree, SVM, Random Forest, and K-Means, while all models except K-Means get to an accuracy above 85%, it becomes evident that Random Forest stands out as the optimal choice for tackling the credit card fraud detection problem. The precision, recall, and F1-score are high, indicating that it performs well in correctly identifying both positive and negative instances of fraud. The ROC-AUC score is also close to 1, suggesting excellent discrimination between the classes.

The Random Forest model showcases consistently high accuracy and ROC scores. This might be attributed to several inherent reasons. The ensemble nature of Random Forest combines multiple decision trees, allowing it to capture complex relationships within the data and reduce overfitting. Moreover, its ability to handle imbalanced datasets, a common characteristic in fraud detection, contributes to its effectiveness. The model's feature selection process and robustness to outliers further enhance its performance. Taken together, these traits make Random Forest a powerful solution for identifying fraudulent transactions and maintaining the integrity of credit card transactions.

However, K-Means model seems not a suitable predictive model for our problem. There can be multiple reasons for this. Firstly our problem is a binary classification problem wherein each transaction can be classified as either fraudulent or not. However since K-Means being a clustering algorithm assigns data points to clusters without considering class labels. And thus it doesn't produce good results on a binary classification task.

Furthermore, our dataset has labeled data – transactions are either marked as fraudulent or legitimate. K-Means on the other hand doesn't utilize these labels during the clustering process. It tries to minimize the variance within clusters, which does not align with the goal of separating fraudulent and non-fraudulent transactions. Also K-Means relies on a distance metric (usually Euclidean distance) to determine cluster assignments. However, for our problem, the concept of distance between transactions might not be relevant in the context of fraud patterns. The other models used for our problem can incorporate labeled data and capture complex patterns associated with fraud and thus have a better accuracy as compared to K-Means.

# Future work

While random forest worked well for detecting fraudulent transactions, as the landscape of fraud detection continues to evolve, it is imperative to proactively refine existing models and augment their capabilities to address the ever-changing nature of fraudulent activities.

For one, the way we generate transitions and fraud scenarios can be improved. Currently, customers are limited to making transactions at terminals within 4500 km of their geographic location. While this is a large area that worked well for the purpose of this project, with the increase of online transactions, this restriction is limiting. Extending the scope beyond the existing radius, would prevent legitimate transactions from being inadvertently hindered by geographic constraints, ensuring the model's continued relevance in a globalized marketplace.

Furthermore, the efficacy of any fraud detection system hinges upon its ability to recognize a diverse range of fraudulent scenarios. Beyond the foundational phishing and account theft cases, there is a need to revamp the approach to generating fraudulent instances. By incorporating more sophisticated techniques and encompassing a broader spectrum of fraud types, the model can capture novel and intricate fraudulent schemes that are increasingly prevalent in today's world. Additionally, improving the model to also be able to detect the type of fraud would be beneficial to know whether the customer or terminal is compromised, allowing fraud to be stopped earlier. Similarly, more techniques that account for the concept drip that is occurring would help improve the performance of current models. As of now, fixed timeframes, such as 28 or 14 days, are being used, which is not the case in the real world.

Finally, one of the main challenges with stopping fraudulent activity in real time detection, Future work can include looking into incorporating this capability, to stop phishing and suspicious terminal instantly, Afterall, this is a basic feature that is available on all major search engines and stores, One way to do this is to look into unsupervised machine learning models. The current Supervised machine learning techniques require the data to be trained, although this worked well on test data indicating beach model's ability to generalize well, this might not always be the case as there are always new types of fraud as there are always new types of fraud occurring. Although k-means didn't perform well, it still might be beneficial to look into anomaly detection approaches for improved defraud detection and other unsupervised machine learning models.

# References


1. *https://fraud-detection-handbook.github.io/fraud-detection-handbook/Chapter_3_GettingStarted/SimulatedDataset.html - Generating dataset code and handbook*
2. *https://www.ibm.com/topics/random-forest - Random Forest*
3. *https://www.geeksforgeeks.org/hyperparameters-of-random-forest-classifier/ - Random Forest Hyperparameters*
4. *https://scikit-learn.org/stable/modules/tree.html - Decision Trees*
5. *https://www.educba.com/decision-tree-hyperparameters/ -Decision Tree Hyperparameters*
6. *https://towardsdatascience.com/everything-about-svm-classification-above-and-beyond-cc665bfd993e#:~:text=SVM%20algorithm%20is%20not%20suitable,samples%2C%20the%20SVM%20will%20underperform. - SVM*
7. https://www.equifax.com/personal/education/credit-cards/credit-card-fraud/ - What Is Credit Card Fraud?
8. https://www.fraud.com/post/credit-card-fraud - Credit Card Fraud – Ways to Detect and Prevent It

# Annex-A

- List of details regarding the individual contribution of each member of the project group
    - Pranathi Alluri: Preprocessing and cleaning data, SVM, report writing and the presentation
    - Sanjana Kandunoori: Contributed to the dataset generation, Logistic Regression Code, report writing and the presentation
    - Anagha Kadoo: Contributed to the dataset generation process, K-Means clustering code and result analysis, report writing and creating the video presentation
    - Sinuo Song: Contribute to data generation code, Random Forest model code and result analysis, report writing and the presentation
    - Chen Li:

- Explanation for skipping Code section

  We decided to exclude the code section from the main report. The complexity and length of the code exceeded the scope of the report, making it impractical to include within the document without sacrificing readability and brevity. Additionally, including detailed code snippets could potentially overwhelm the main narrative of the report and detract from the primary focus on methodology, results, and analysis. In order to ensure clarity and maintain a streamlined flow, a separate set of comprehensive Python notebooks has been prepared to accompany the report. These notebooks contain the full code implementation of dataset generation and model training.