# Table of Contents

# Introduction

- The myriads of plastic cards in use worldwide are a gold mine for criminals, it is estimated that by 2027, there will be about $40 billion hit globally in credit card losses as compared to $27.85 billion in 2018. This significant increase is highly attributed to the rise of electronic transactions and card users in the world. Fraudulent methods are also getting more sophisticated and thus making it harder to spot by traditional fraud detection software and conventional methods.
- It is of very high importance that more sophisticated approaches are employed and used in mitigating this massive challenge, high hopes have thus been placed on machine learning models to help credit card companies to recognize credit card transactions which are suspicious and report them to an analyst while letting normal transactions be automatically processed.

# Problem Statement

The most common kind of identity theft is credit card fraud. With an estimated 1.5 billion credit cards in use in the United States alone, it's no surprise that millions of people fall victim to credit card fraud each year.

**How does fraud happen?**

- Unknown individuals receive a consumer's credit card number.
- Someone else uses a card that has been lost or stolen.
- Criminals steal mail from the intended receiver and utilize it for their own purposes.
- Employees at businesses copy the owner's cards or card numbers.

**Why does fraud happen?**

- Opportunity
- Pressure
- Rationalization

# Aim and Objectives

The aim of this project is to obtain an adoptable machine learning process that helps detect credit card fraud.

The following objectives are set to help achieve the aim:

- Applying a decision tree and logistic regression to detect fraudulent card transactions
- Selecting most significant variables
- Performing optimization to increase recall score of the model

# Review of Related Works

| s/n | Author | Approach | Achievement |
|---|---|---|---|
| 1. | (Ba 2019) | Generative Adversarial Networks | The results show that the Wasserstein-GAN is more stable in training and produce more realistic fraudulent transactions than the other variants of GANs. |
| 2. | (Dighe et al., 2018) | Naive Bayes, KNN, Decision Trees and Logistic Regression and Neural Network Algorithms | Results from the experiment showed that performance of KNN is better than other machine learning algorithms. |
| 3. | (Sahin & Duman, 2011 ) | Support Vector Machines and Decision Trees. | Decision Trees outperformed SVMs when the size of data set was small but with increase in size of dataset SVM reached accuracy of decision trees. |

# Methodology

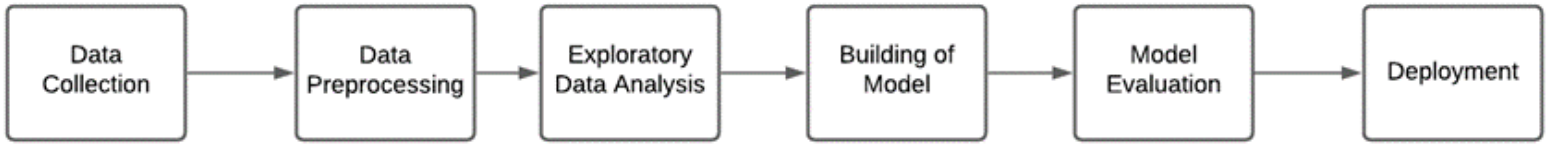


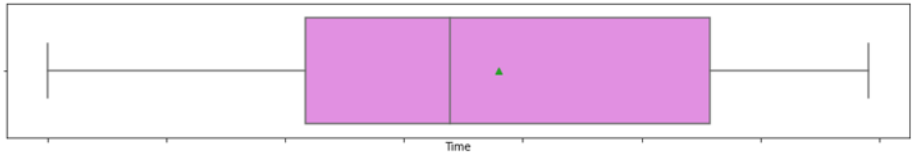

Figure 1: Machine learning System architecture workflow

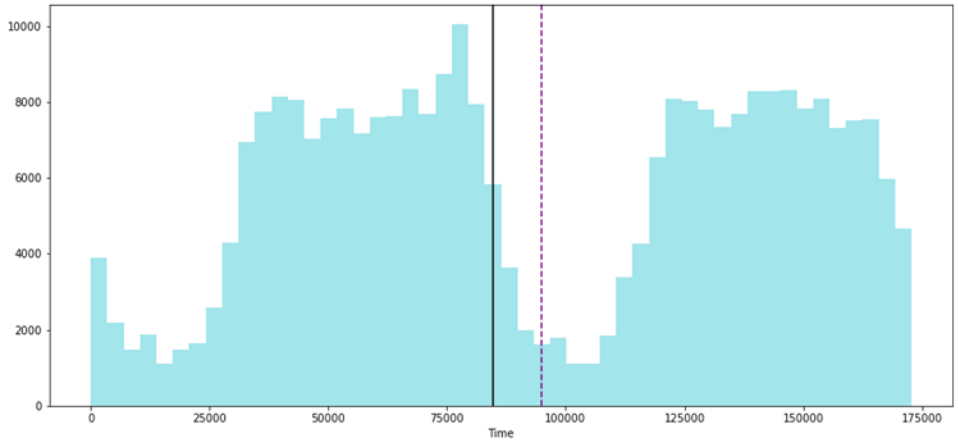
# Result and Discussion

## Exploratory Data Analysis

- Observation on Time

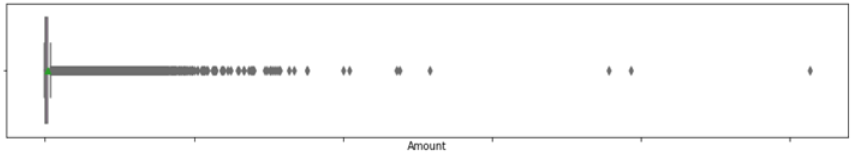
The box-plot and histogram charts below gives information on the univariate analysis of the time column, it shows that the time column is symmetrical at around 100,000 and at around 12,500.

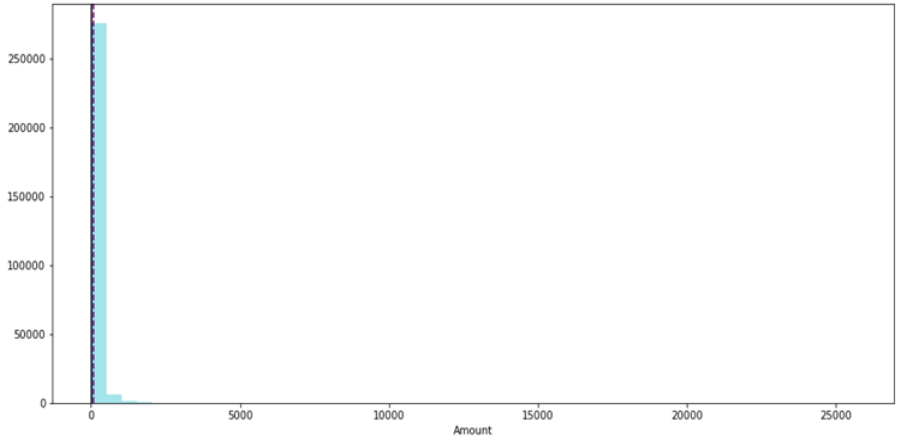



- Observation on Amount

The box-plot and histogram charts below gives information on the univariate analysis of the amount column, it shows that the amount column has lots of outliers which are potentially fraudulent activities.



# Logistic Regression Model

In [24]:
```
X_train1 = X_train.drop('Amount', axis = 1)
X_test1 = X_test.drop('Amount', axis = 1)

logit1 = sm.Logit(y_train, X_train1.astype(float))
lg1 = logit1.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.003891
         Iterations 13
```

In [25]:
```
# Let's check model performances for this model
scores_LR = get_metrics_score1(lg1,X_train1,X_test1,y_train,y_test,flag=True)
```

```
Accuracy on training set :  0.9992526233422283
Accuracy on test set :  0.9991807403766253
Recall on training set :  0.6582633053221288
Recall on test set :  0.5851851851851851
Precision on training set :  0.8969465648854962
Precision on test set :  0.8494623655913979
ROC-AUC Score on training set: 0.829063815851304
ROC-AUC Score on test set: 0.7925105369823332
```

Figure 2: Result of the logistic regression classifier after ‘Amount’ column was dropped.

# Logistic Regression Model With Optimal Threshold

- Recall increased from on the test set to 0.874, as compared to the previous model which was 0.585.
- As we will decrease the threshold value, Recall will keep on increasing but the Precision will decrease, but that's not right because it will lead to loss of resources, we need to choose an optimal balance between recall and precision.
- Area under the curve is has decreased as compared to the initial model.

In [71]:
```
# Model prediction with optimal threshold
pred_train_opt = (lg1.predict(X_train1.astype(float)) > optimal_threshold).astype(int)
pred_test_opt = (lg1.predict(X_test1.astype(float)) > optimal_threshold).astype(int)

print('Accuracy on train data:',accuracy_score(y_train, pred_train_opt) )
print('Accuracy on test data:',accuracy_score(y_test, pred_test_opt))

print('Recall on train data:',recall_score(y_train, pred_train_opt) )
print('Recall on test data:',recall_score(y_test, pred_test_opt))

print('Precision on train data:',precision_score(y_train, pred_train_opt) )
print('Precision on test data:',precision_score(y_test, pred_test_opt))

print('ROC-AUC Score on train data:',roc_auc_score(y_train, pred_train_opt) )
print('ROC-AUC Score on test data:',roc_auc_score(y_test, pred_test_opt))
```

```
Accuracy on train data: 0.9618637266507494
Accuracy on test data: 0.9616586496260665
Recall on train data: 0.9159663865546218
Recall on test data: 0.8740740740740741
Precision on train data: 0.041392405063291136
Precision on test data: 0.034942256440627775
ROC-AUC Score on train data: 0.9389562243767194
ROC-AUC Score on test data: 0.9179356631916767
```

Figure 3: Results of the model using the optimal threshold for ROC-AUC

# Logistic Regression Model With Precision Curve

At 0.085 threshold we get the highest recall with a good precision

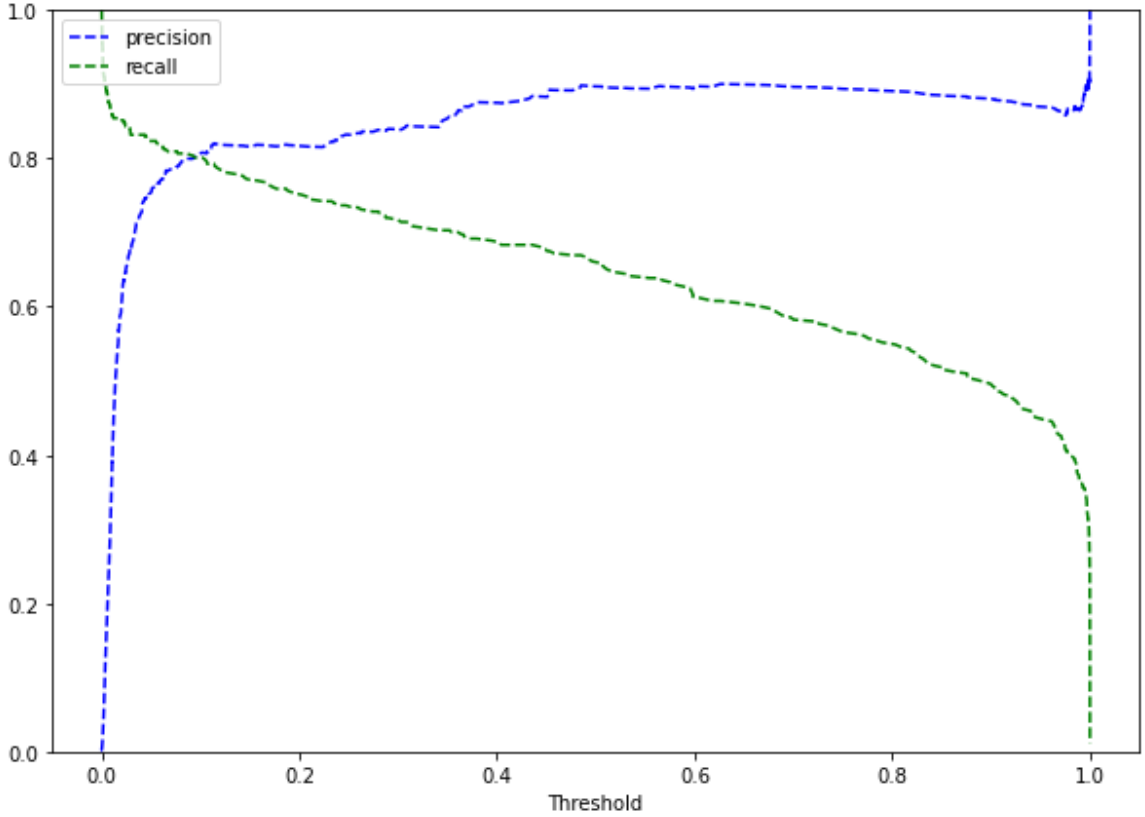
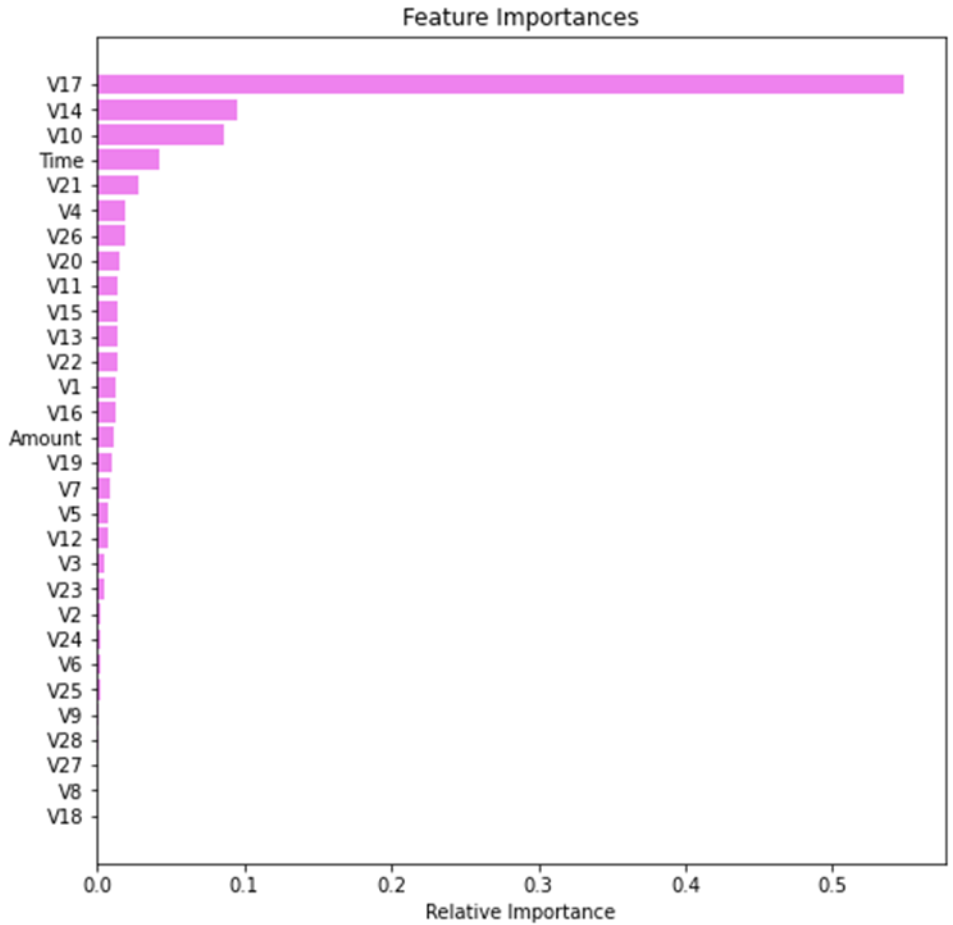



Figure 4: Precision-Recall curve

# Decision Tree




The decision Tree model gives the following Results:

Accuracy on training set: 0.999197447884272

Accuracy on test set: 0.9991807403766253

Recall on training set: 0.7703081232492998

Recall on test set: 0.7037037037037037

Precision on training set: 0.7790368271954674

Precision on test set: 0.76

ROC-AUC Score on training set: 0.8849580886186752

ROC-AUC Score on test set: 0.8516760184012963.

Figure 5: Feature Importance Chart after training the data

# Decision Tree Using Grid Search

```
In [79]: # Let's check model performances for this model
         scores_DT = get_metrics_score2(estimator,X_train,X_test,y_train,y_test,flag=True)

         Accuracy on training set :  0.999197447884272
         Accuracy on test set :  0.9991807403766253
         Recall on training set :  0.7703081232492998
         Recall on test set :  0.7037037037037037
         Precision on training set :  0.7790368271954674
         Precision on test set :  0.76
         ROC-AUC Score on training set: 0.8849580886186752
         ROC-AUC Score on test set: 0.8516760184012963
```

Figure 6: Result of the model after reducing overfitting.

The overfitting in terms of precision and recall has been reduced, while the ROC-AUC curve has reduced to 0.85167.

# Comparing Model Performance

| | Model | Train_Accuracy | Test_Accuracy | Train_Recall | Test_Recall | Train_Precision | Test_Precision |
|---|---|---|---|---|---|---|---|
| 0 | Logistic Regression | 0.999253 | 0.999181 | 0.658263 | 0.585185 | 0.896947 | 0.849462 |
| 1 | Decision Tree | 1.000000 | 0.999146 | 1.000000 | 0.733333 | 1.000000 | 0.727941 |
| 2 | Decision Tree(pre-pruned) | 0.999197 | 0.999181 | 0.770308 | 0.703704 | 0.779037 | 0.760000 |
| 3 | Logistic Regression with precision-recall curv... | 0.999293 | 0.999251 | 0.806723 | 0.740741 | 0.800000 | 0.775194 |
| 4 | Logistic Regression with optimal threshold | 0.961864 | 0.961659 | 0.915966 | 0.874074 | 0.041392 | 0.034942 |

Fig 6: Model performance table for both logistic and decision tree.

From the models above, the best model for giving us the highest recall is the logistic regression optimal threshold, followed by the logistic regression with precision-recall curve. But if the bank wants a harmonic of both the precision and the recall, then the pruned decision tree can be used.

# Conclusion and Recommendation

Machine learning models have been built to detect and classify fraudulent card transactions, with their performances compared. We can conclude that the decision tree is more prone to overfitting than the logistic regression. Pruning the decision tree helps to avoid overfitting of the model and thereby resulting into a model with higher recall value. Also, in terms of recall on the test set, logistic regression variants outperformed the decision tree, making logistic regression more suitable for this type task.

I therefore recommend that the various machine learning models utilized in this study can be extended to deep learning models in the future. Other methods for feature selection and dealing with the problem of data set imbalance can be utilized in conjunction for improved outcomes.

# References


- Ba, H. (2019). Improving detection of credit card fraudulent transactions using generative adversarial networks. *arXiv preprint arXiv:1907.03355*.

- Dighe, D., Patil, S., & Kokate, S. (2018, August). Detection of credit card fraud transactions using machine learning algorithms and neural networks: A comparative study. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)* (pp. 1-6). IEEE.

- Şahin, Y.G. and Duman, E., 2011. Detecting credit card fraud by decision trees and support vector machines.