# Project 3 – Credit Card Fraud Detection

## Milestone 3 : White Paper

**********

College of Science and Technology, Bellevue University

DSC680-T301 Applied Data Science (2245-1)

Sashidhar Bezawada

May 24 , 2024

- **Introduction**

  Credit Card Frauds are the cases of using someone else's credit cards for financial transactions without the information of the card owner. Credit Cards were made available in order for the people to increase their buying power, it is an agreement with your bank that lets the user use the money lended by the bank in exchange for the repayment of this lended money on the due date or incur interest charges. As all the things in the nature are binary, cases of credit card frauds have also achieved high numbers. Global economy pays the price of more than $ 24 billion per year due to these frauds. It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase. Thus, it becomes essential to solve this problem and building automated models for such a rising problem statement is necessary and AI - ML is the key for it!

- **Business Problem**

  Develop a machine learning model to address the below:
  - Classify whether a credit card transaction is fraudulent or genuine and handle unbalanced dataset.

- **Datasets**

  I am using the dataset, which is from Kaggle. The dataset contains transactions made by credit cards in September 2013 by European cardholders.

  https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data

  **File descriptions:**

  **creditcard.csv** - This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

  It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Here's a brief version of what you'll find in the data description file:

**Dataset Attributes**

V1 - V28 : Numerical features that are a result of PCA transformation.
Time : Seconds elapsed between each transaction and the 1st transaction.
Amount : Transaction amount.
Class : Fraud or otherwise (1 or 0)

- **Exploratory Data Analysis**

The dataset have any 1081 duplicates, which are no good for the model training. So removed them. Has no missing values and no Categorical value. Data types are all float values excluding the target (integer).Generally, when making a predictive model, it would be preferable to train a model with features that are not too correlated with one another so that we do not need to deal with redundant features. . In assessing the 'Class' variable distribution, it became apparent that the classes were highly imbalanced in their representation of the fraudulent transactions , as visualized in **Appendix**. The value '1' representing the frauds has only 0.2% of the overall records, this imbalance represented a concern for the modeling portion of the project. Therefore, to handle this before modeling, I used a concept called SMOTE to generate new and synthetic data we used for training our model. The strategy helped to ensure that bias was not introduced towards the majority class during the predictive modeling steps. There are too many features in the dataset and it is difficult to understand anything. Hence, we will plot the correlation map only with the target variable.

**Feature reduction** has been performed in two ways:

- **Based on Correlation Plot : ( 10 columns considered )**
  For feature selection, we will exclude the features having correlation values between [-0.1,0.1].
  V4, V11 are positively correlated and V7, V3, V16, V10, V12, V14, V17 are negatively correlated with the Class feature. These Charts are also shown in **Appendix.**

- **Based on ANOVA Score : ( 21 columns considered )**

  Higher the value of the ANOVA score, higher the importance of that feature with the target variable. And have rejected features with values less than 50.

**Data Balancing :**
For best performance, I have used oversampling technique known as SMOTE to treat the imbalance and created the training and test datasets.

**Model Selection :**
Five models which are appropriate for a classification model were used to evaluate performance accuracy. The models evaluated are listed below :

- KNN - Straight forward pattern recognition model which allows the testing of several k values and leaf sizes to determine the best performance.
- Decision tree - A decision tree is a supervised learning algorithm that performs strong in classification problems.
- Random Forest - Expands beyond a decision tree by constructing multiple decision trees to remediate forcing a binary decision.
- XGBoost - an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning model.
- GradientBoost Classifier - Gradient boosting works by building simpler (weak) prediction models sequentially where each model tries to predict the error left over by the previous model.

Post-data exploration, the models were performed on all of the 31 features in the dataset. The model performance was then evaluated based on the models' accuracies and confusion matrices. Precision and recall can be derived from a confusion matrix; recall helps to determine the ability to find all relevant instances of a class in a data set, whereas precision is used to determine the proportion of data points that the model says exists within the relevant class were indeed relevant. The F1 score is calculated as the harmonic mean of precision and recall. The AUC measures the probability that the model will assign a randomly chosen positive instance a higher predicted probability compared to a randomly chosen negative instance. All accuracy, precision, recall, F1-Score, AUC results can be viewed in **Appendix.**

- **Results interpretation:**

SMOTE uses a nearest neighbors algorithm to generate new and synthetic data we used for training our model.

**Decision Tree** - Model accuracy and Precision is the lowest among the 5 Models.
**K-Nearest Neighbor** - Has good Accuracy Score, and 2nd highest F1 and Recall Scores. The False positive & False negative cases are higher than all models.
**Random Forest** - Accuracy score is good, however the model is not predicting the Fraud correctly (precision is low).
**XGBoost** - Accuracy is highest as well as it is identifying the Fraud better with the Highest Precision Score). Also, it has high Recall & F1 Scores.
**Gradient Boosted Classifier** - Accuracy is high. While precision is low. It also has the highest Recall, F1 & AUC Scores.

Also, The Scores are higher with feature reduction done with ANOVA test than Correlation plot.

- **Ethical Considerations**

  **Bias :** Models can be programmed to favor certain groups of people over others. This can lead to unfair treatment and discrimination, kind of like a robot referee who always calls fouls against one team but not the other.

  **Privacy :** It's important to make sure that this data is used in a way that respects our privacy and autonomy, so we don't feel like we're constantly being watched by a robot spy.

  **Responsibility :** If an model makes a mistake or causes harm, who is to blame? Is it the developer who created the system, the company that deployed it, or the robot itself? It's like a game of hot potato, but with ethical and legal implications.

- **Assumption**

  As the data looks normalized, I have made the assumption that it can be as a result of the desensitization process or through decomposition processes (i.e. PCA).

- **Challenges/Issues**

  This is a simulated dataset and has only 31 attributes. This is an imbalanced dataset. To improve the effectiveness, SMOTE has been used.

- **Conclusion**

  This is a great dataset to learn about binary classification problem with unbalanced data. As the features are disguised, feature selection cannot be assisted based on the domain knowledge of the topic. Statistical tests hold the complete importance to select features for modeling.
  It appears that all models achieved very high accuracy on the test data. However, it's essential to keep in mind that accuracy might not be the best evaluation metric for imbalanced datasets like the one in this case. Since fraudulent transactions are rare, a high accuracy score can be misleading, as a model could simply predict all transactions as non-fraudulent and still achieve a high accuracy due to the class imbalance. In such cases, metrics like precision, recall, F1-score, or the area under the precision-recall curve (AUPRC) are more informative for evaluating model performance. These metrics provide insights into how well the model performs specifically on identifying fraudulent transactions.

- **Future Uses**

  Financial companies can use these fraud detecting models to be implemented in Real-time for minimizing the losses they incur due to these fraudulent transactions.

- **Recommendations**

  The current dataset is a simulated one with less number of attributes. Another dataset with more attributes can be explored to improve the Model

- **Implementation Plan**

  As we already discussed in the methodology section about some of the implementation details. So, the language used in this project is Python programming. We're running python code in anaconda navigator's Jupyter notebook. Jupyter notebook is much faster than Python IDE tools likePyCharm or Visual studio for implementing ML algorithms. The advantage of Jupyter notebook is that while writing code, it's really helpful for Data visualization and plotting some graphs like Piechart , heatmaps.

- **References**

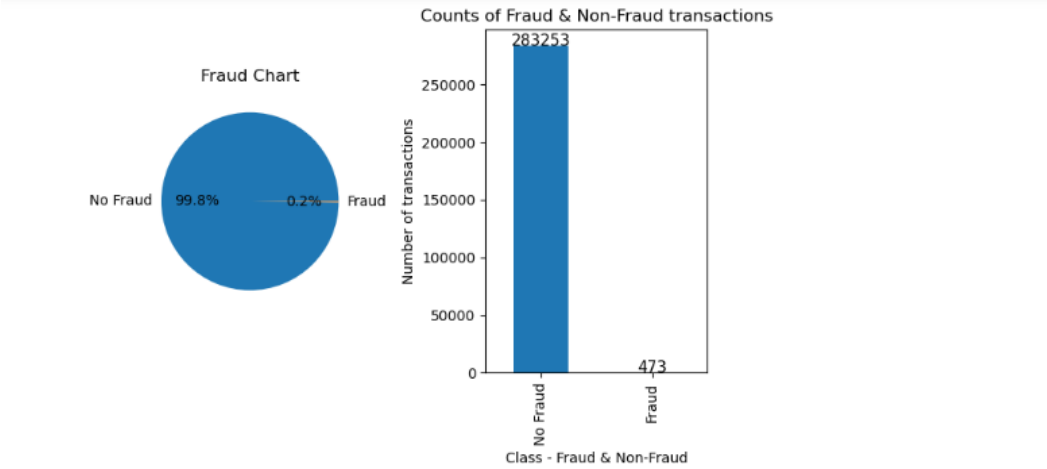

  https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

  https://www.chargebackgurus.com/blog/credit-card-fraud-detection

  https://www.cnbc.com/select/what-is-a-credit-card/

  https://www.fortunebusinessinsights.com/industry-reports/fraud-detection-and-prevention-market-100231

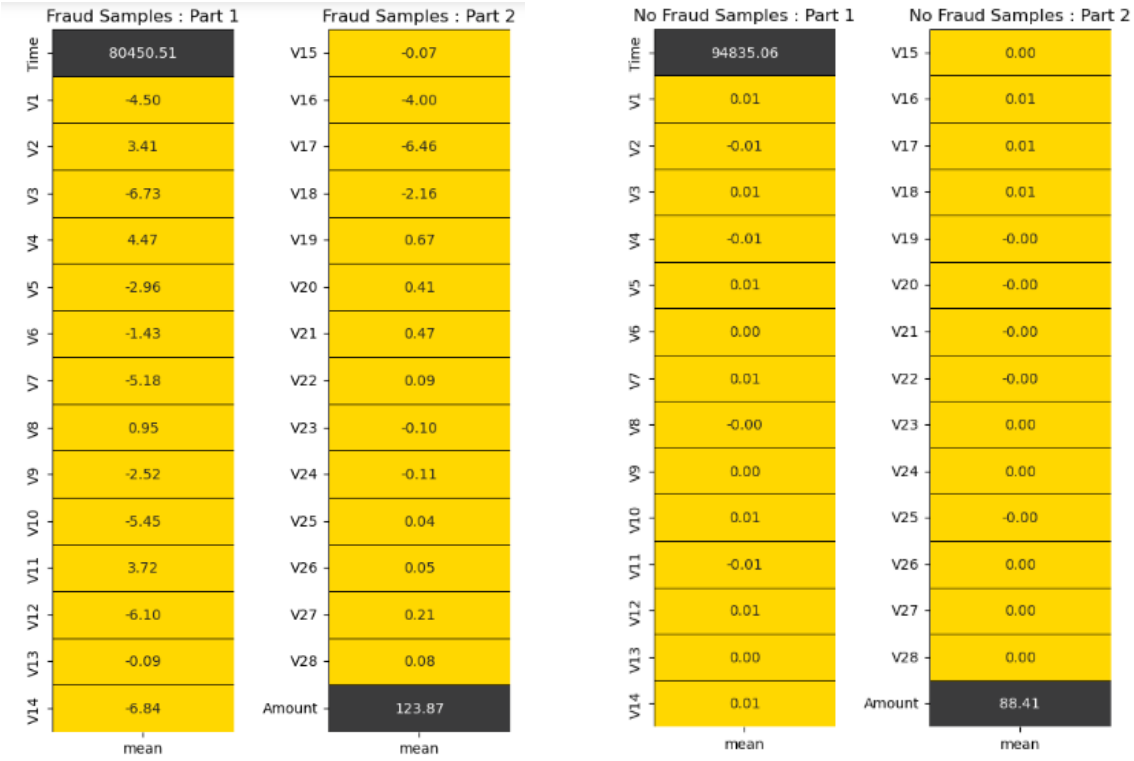
- **Appendix A - Target Class Distribution**



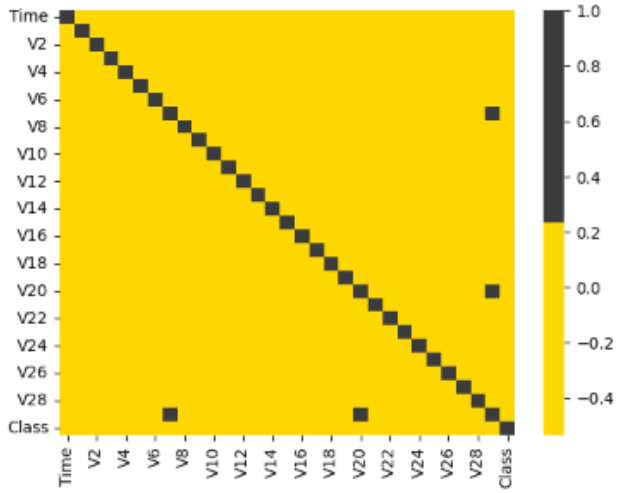

The data is clearly highly unbalanced with majority of the transactions being No Fraud. And the classification model will bias its prediction towards the majority class, No Fraud.

Hence, data balancing is required for building a robust model.

- **Appendix B - Feature Importance Visualizations**
  Mean values of features for Fraud & No Fraud cases!



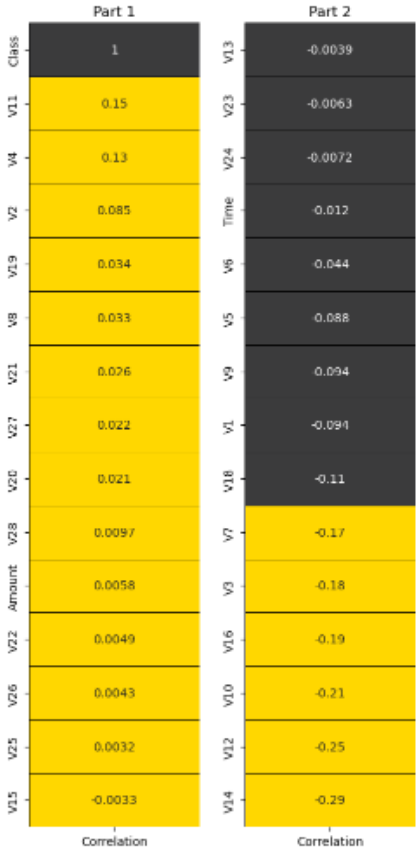
- **Appendix C – Correlation Matrix**




There are too many features in the dataset and it is difficult to understand anything.

Hence, we will plot the correlation map only with the target variable.

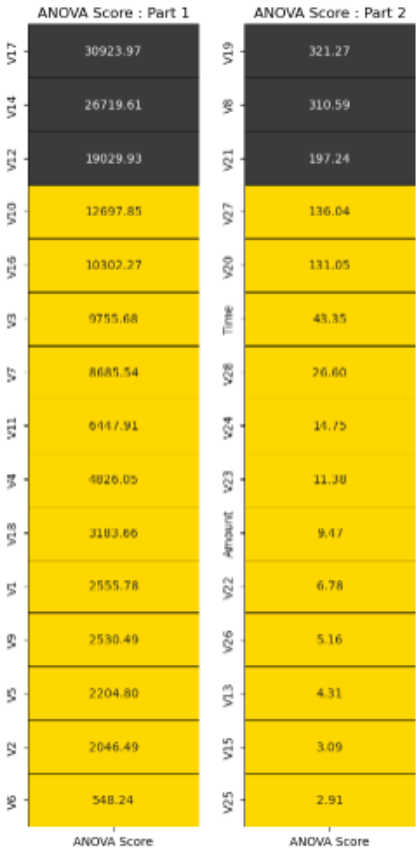
- **Heatmap - Correlation Plot**




For feature selection, we will exclude the features having correlation values between [-0.1,0.1].

V4, V11 are positively correlated and V7, V3, V16, V10, V12, V14, V17 are negatively correlated with the Class feature.

**Heatmap - ANOVA Score**




Note : Higher the value of the ANOVA score, higher the importance of that feature with the target variable.

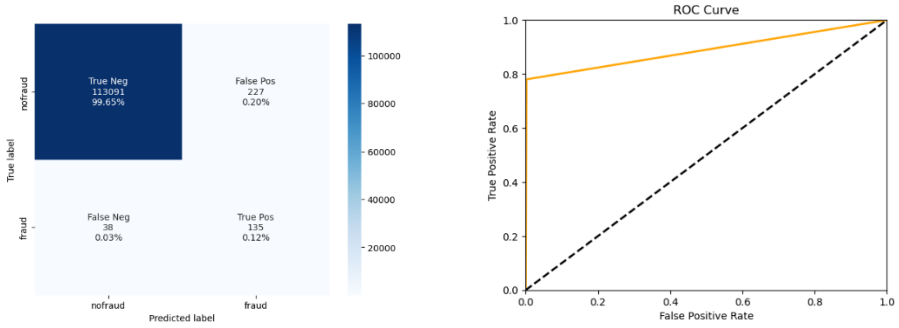
From the above plot, we will reject features with values less than 50.

Also, 2 Datasets are created based on Correlation plot and other based on ANOVA Scores.

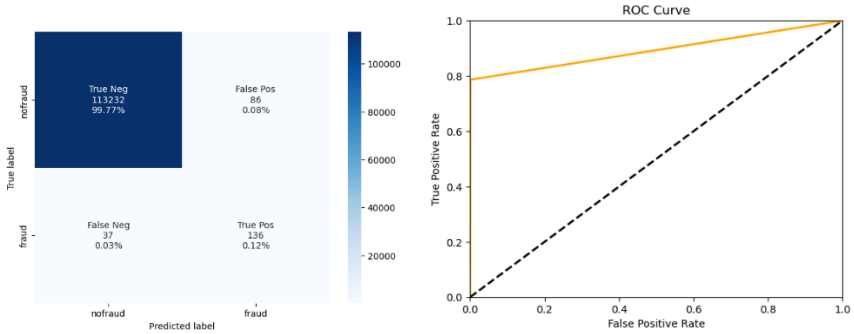
- **Appendix D - Model Evaluation Confusion Matrices & ROC Cuves**

**XGBoost Confusion matrix & ROC Curve results ( Best performance )**

**1.** Results based on Correlation Plot




**2.** Results based on ANOVA Score




- **Appendix E - Model Evaluation Results**

Model Results based on Correlation Plot :

| Model | K-Nearest Neighbor(SMOTE) | Random Forest (SMOTE) | Decision Tree (SMOTE) | XGBoost (SMOTE) | Gradient Boosted Classifier (SMOTE) |
|---|---|---|---|---|---|
| Test_accuracy : | 0.99742 | 0.99508 | 0.97321 | 0.99766 | 0.98713 |
| Test_Precision_Score : | 0.34936 | 0.21396 | 0.04390 | 0.37292 | 0.09451 |
| Test_F1_Score : | 0.48591 | 0.34042 | 0.08323 | 0.50467 | 0.17045 |
| Test_Recall_Score : | 0.79768 | 0.83236 | 0.79768 | 0.78034 | 0.86705 |
| Test_AUC_Score : | 0.89770 | 0.91385 | 0.88558 | 0.88917 | 0.92718 |
| Test_Balanced_Accuracy_Score: | 0.89770 | 0.91385 | 0.88558 | 0.88917 | 0.92718 |

Model Results based on ANOVA Score :

| Model | K-Nearest Neighbor(SMOTE) | Random Forest (SMOTE) | Decision Tree (SMOTE) | XGBoost (SMOTE) | Gradient Boosted Classifier (SMOTE) |
|---|---|---|---|---|---|
| Test_accuracy : | 0.99859 | 0.99666 | 0.97593 | 0.99891 | 0.99006 |
| Test_Precision_Score : | 0.52452 | 0.29376 | 0.04901 | 0.61261 | 0.12073 |
| Test_F1_Score : | 0.63470 | 0.43582 | 0.09238 | 0.68860 | 0.21229 |
| Test_Recall_Score : | 0.80346 | 0.84393 | 0.80346 | 0.78612 | 0.87861 |
| Test_AUC_Score : | 0.90117 | 0.92041 | 0.88983 | 0.89268 | 0.93442 |
| Test_Balanced_Accuracy_Score: | 0.90117 | 0.92041 | 0.88983 | 0.89268 | 0.93442 |