Credit Card Fraud Detection

Final Project Report

INFO 6105

Farid Ghorbani
NUID: 002830498
College of Engineering
Northeastern University
Toronto, ON, Canada
ghorbani.f@northeastern.edu

1. Introduction

Credit card fraud is a significant concern for both consumers and financial institutions. The ability to accurately detect fraudulent transactions is crucial to protect customers from unauthorized charges and maintain trust in the financial system. In this project, we will explore the impact of an imbalanced dataset on the performance of various classification models in detecting credit card fraud. The goal of this project is twofold: first, to understand how an imbalanced dataset can impact the analysis and results of credit card fraud detection, and second, to evaluate the effectiveness of different classification models in handling this imbalance.

1.1 Project Improvement

In this project, we focused on enhancing our mid-term project in several key areas. Firstly, we tackled the challenge of unbalanced data by experimenting with two techniques: Undersampling and Oversampling.

By balancing the dataset, our aim was to enhance the performance of our models and achieve more accurate predictions. Furthermore, we compared the performance of these techniques, benchmarked all four models, and evaluated them against our previous model.

Additionally, we addressed the issue of outliers and worked on normalizing certain features to further enhance the quality of our analysis and predictions. Through these improvements, we aimed to refine our fraud detection system and enhance its effectiveness in identifying fraudulent transactions accurately.

2. Dataset

Our dataset consists of anonymized credit card transactions labeled as either fraudulent or genuine. The imbalance between fraudulent and genuine transactions is a common challenge in fraud detection. This dataset presents transactions that occurred in two days, with 492 frauds out of 284,807 transactions.

The dataset contains only numerical input variables, which are the result of a PCA transformation. PCA stands for Principal Component Analysis, a technique used to reduce the number of variables in a dataset while still retaining most of the information (Features V1, V2, … V28). The only features that have not been transformed with PCA are 'Time' and 'Amount'. The 'Time' feature shows the number of seconds that have passed since the first transaction in the dataset. The 'Amount' feature shows the amount of money involved in each transaction. The 'Class' feature is the response variable, and it takes the value 1 if the transaction is fraudulent and 0 if it is genuine.

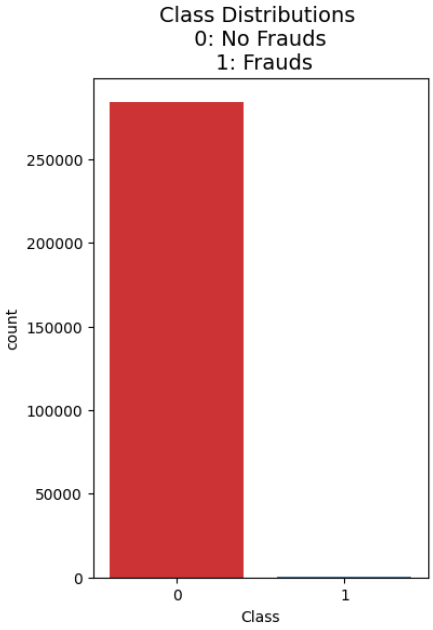
3. Data Preprocessing and Analysis

3.1 Missing Value

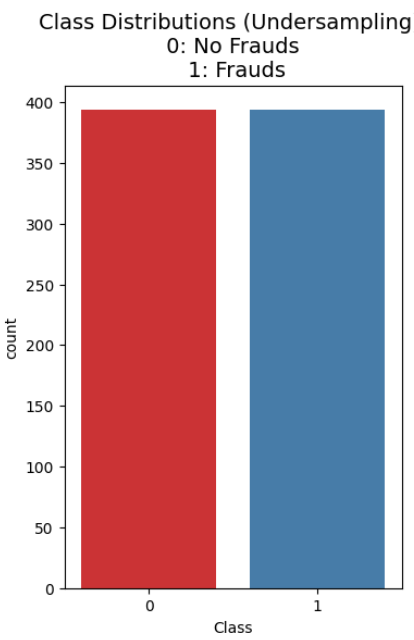
Since there are no "Null" values in the dataset, we do not need to worry about replacing missing values. This simplifies our data preprocessing step.

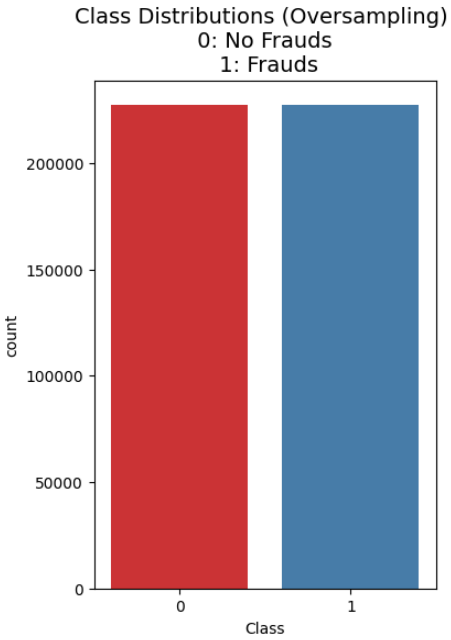
3.2 Unbalanced Data

The dataset is highly imbalanced, with Non-Fraud transactions accounting for 99.83% of the total, while Fraud transactions make up only 0.17%. This imbalance can pose a challenge when building predictive models, as the algorithms may become biased towards the majority class (Non-Fraud transactions) and may not perform well in detecting the minority class (Fraud transactions).



To address this issue, we use two resampling techniques and compare them: oversampling the minority class using SMOTE function and undersampling the majority class. These techniques aimed to rebalance the dataset and improve the performance of our predictive models in detecting fraudulent transactions.





3.3 Splitting the Data

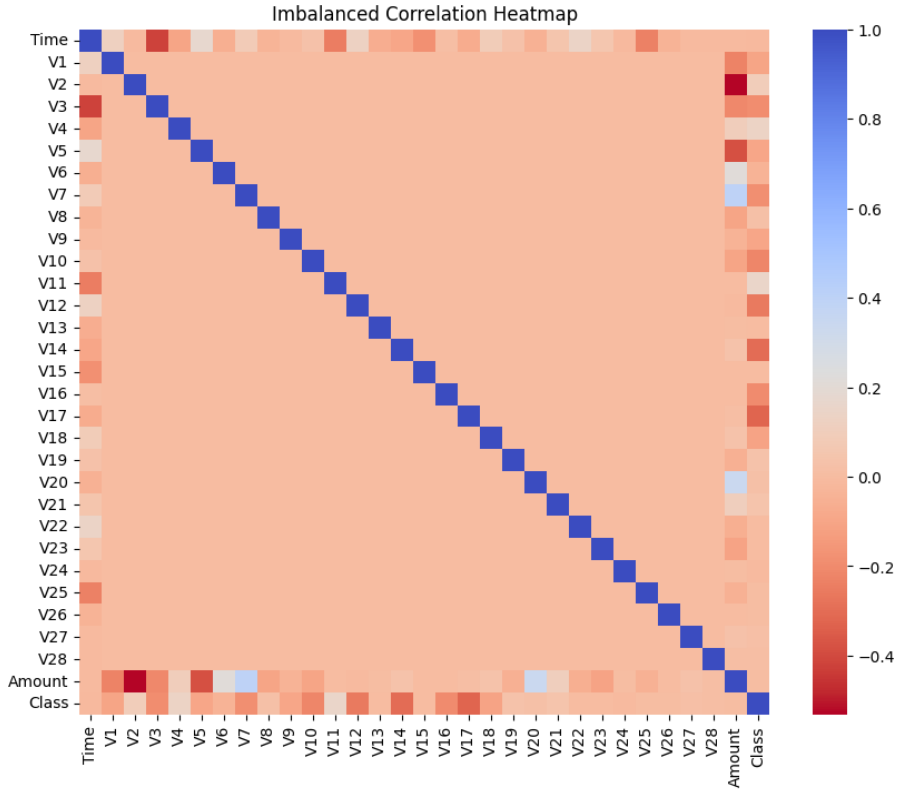
Before dealing with the Undersampling and Oversampling techniques, we need to separate the original dataframe. Our aim is to test our models on the original testing set, not on the testing set created by either of these techniques.

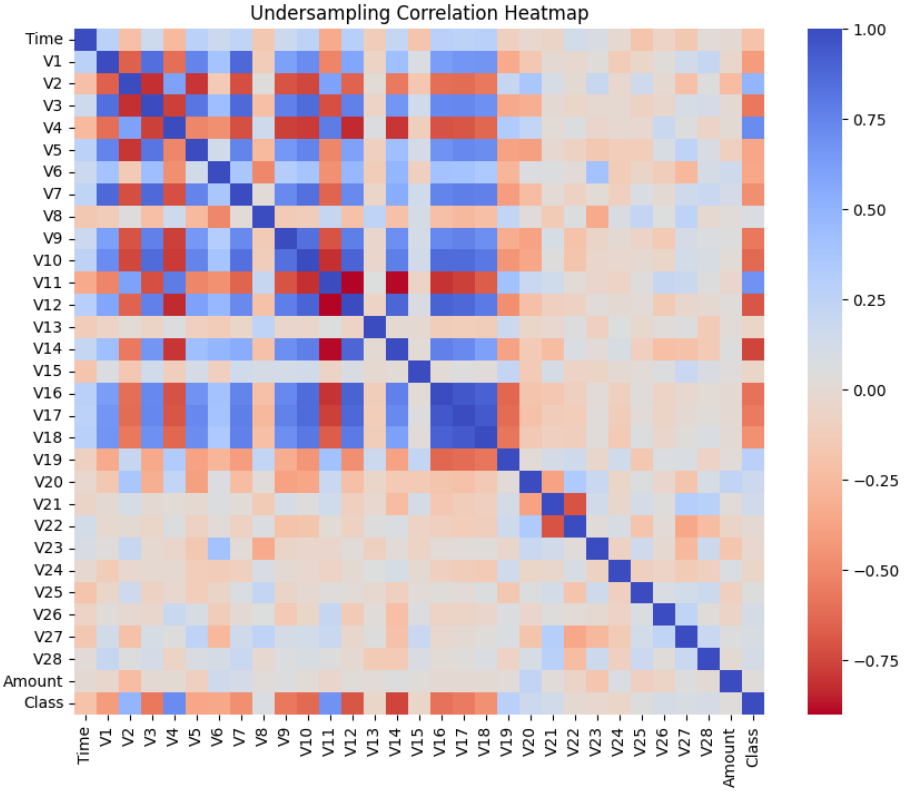
3.4 Correlation Amongst Attributes

Understanding the correlation among attributes help us identify important relationships in the data and inform our modeling decisions. Although we don't know what the "V" features stand for, we can still use them to understand how each of these features influences the result (fraud or no fraud) by examining their importance in the context of the model.

However, it's important to note that when dealing with unbalanced data, the correlation matrix may be influenced by the significant class imbalance. To address this effect, we used an undersampled dataset to explore which features have a high positive or negative correlation with regards to fraud transactions.

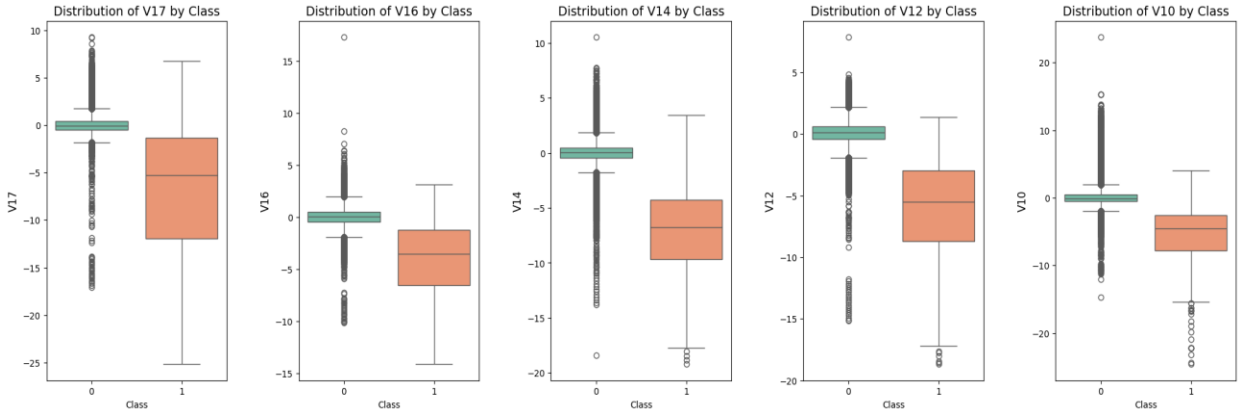
In an unbalanced dataset, identifying features with positive correlations can be challenging due to the disproportionate representation of classes. The imbalance can skew the correlation matrix, making it difficult to discern meaningful relationships between features and the target variable.


Imbalanced Correlation Heatmap
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Class
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


Undersampling Correlation Heatmap
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Class
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

3.5 Analyze Most Important Negative Correlations

By examining the correlation matrix, we identified the features with the most negative correlations with the target variable (fraud or no fraud). since these features are influence heavily in whether a specific transaction is a fraud, we used boxplots to visualize the distribution of these features and identify potential outliers.



Our analysis revealed that certain features, such as 'V14', 'V12' and 'V10', had some extreme outliers which can effect on the model, removing "extreme outliers" from features that have a high correlation with our classes make a positive impact on the accuracy of our models.

In both the undersampling and oversampling datasets, we implemented outlier removal for fraud cases using a threshold of 1.5 times the interquartile range (IQR).

Undersampling Dataset:

```
Feature V14:
iqr: 5.630894061347163
Threshold: 8.446341092020745
V14 Lower: -18.350020649139726
V14 Upper: 4.173555596248928
Feature V14 Outliers for Fraud Cases: 2
```

```
Feature V12:
iqr: 5.686459678795296
V12 Lower: -17.217866040005248
V12 Upper: 5.527972675175934
Feature V12 Outliers for Fraud Cases: 3
```

```
Feature V10:
iqr: 5.00649294545512
V10 Lower: -15.01185160911954
V10 Upper: 5.01412017270094
Feature V10 Outliers for Fraud Cases: 18
```

Oversampling Dataset:

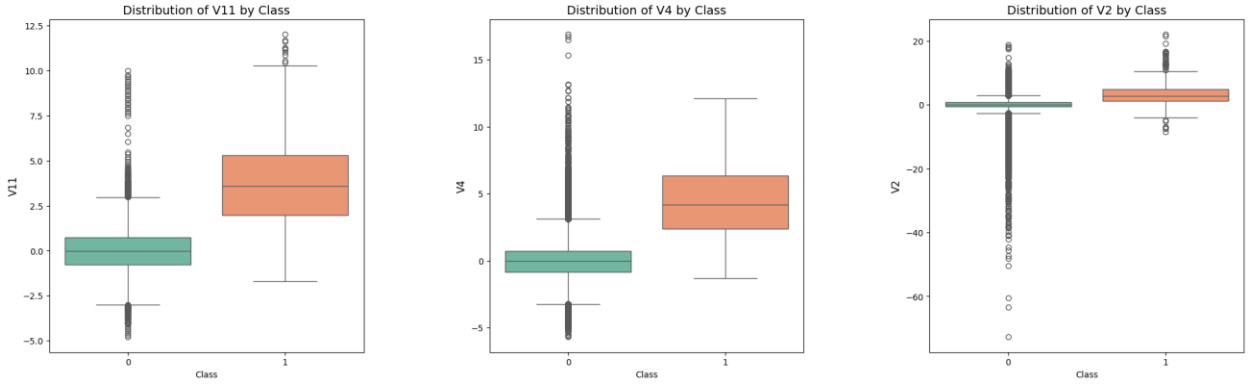
```
Feature V14:
iqr: 5.476096723658413
Threshold: 8.214145085487619
V14 Lower: -18.13313480548767
V14 Upper: 3.7712520891459 8
Feature V14 Outliers for Fraud Cases: 1036
```

```
Feature V12:
iqr: 5.577959365546505
V12 Lower: -17.017960094840504
V12 Upper: 5.293877367345516
Feature V12 Outliers for Fraud Cases: 3042
```

```
Feature V10:
iqr: 4.972410405064565
V10 Lower: -15.183020259582733
V10 Upper: 4.706621360675527
Feature V10 Outliers for Fraud Cases: 9630
```

3.6 Analyze Most Important Positive Correlations

Same as the most negative correlation features, we used boxplots to visualize the distribution of the most positive correlation features and identify potential outliers.



We identified outliers and removed them using the same thresholds for features (V11, V2) with the most positive correlations for fraud cases.

Undersampling Dataset:

```
Feature V11:
iqr: 3.3311869879084397
Threshold: 4.996780481862659
V11 Lower: -3.0874116628593042
V11 Upper: 10.237336288774454
Feature V11 Outliers for Fraud Cases: 7
```

```
Feature V2:
iqr: 2.997317148844532
Threshold: 4.495975723266798
V2 Lower: -3.3594857405188128
V2 Upper: 8.629782854859315
Feature V2 Outliers for Fraud Cases: 32
```

Oversampling Dataset:

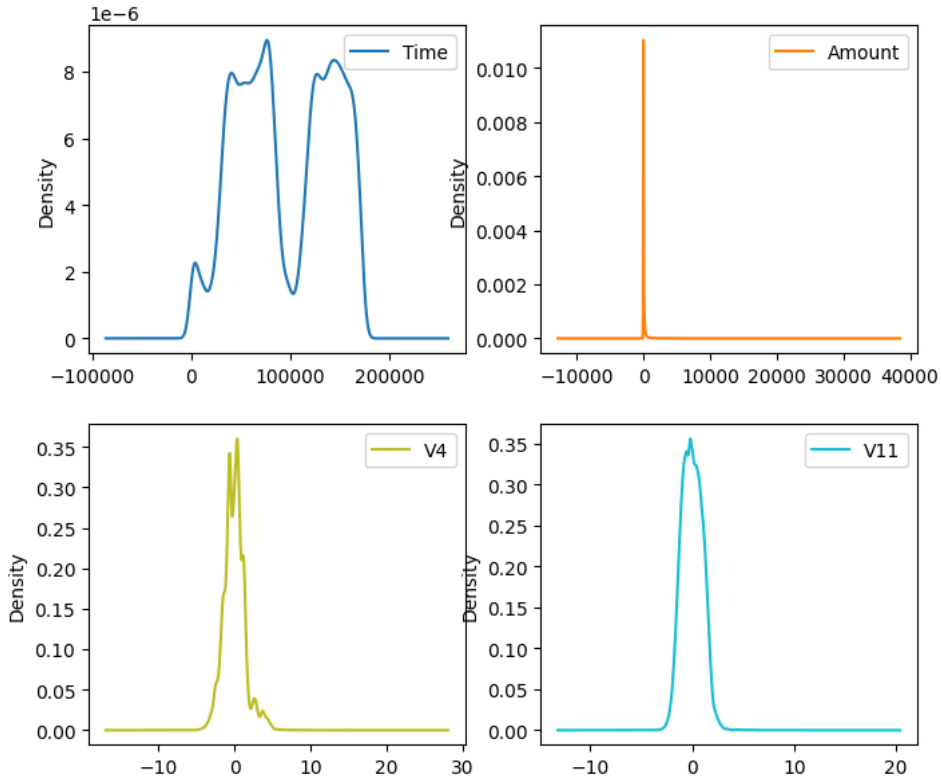
```
Feature V11:
iqr: 3.243293346582174
Threshold: 4.864940019873261
V11 Lower: -2.702995511699614
V11 Upper: 10.270177874629082
Feature V11 Outliers for Fraud Cases: 3935
```

```
Feature V2:
iqr: 2.815329521480927
Threshold: 4.222994282221391
V2 Lower: -3.0236301672324712
V2 Upper: 8.237687918691238
Feature V2 Outliers for Fraud Cases: 18062
```

3.7 More Data Visualization

By examining the distributions of the features, we can see how skewed are some features like. Skewed distributions can affect the performance of our models, as they may not accurately represent the underlying patterns in the data.

As most of our data has already been scaled, we proceeded to scale the remaining columns (Amount and Time) using RobustScaler, which is less prone to outliers.



4. Classifier Model

4.1 Preparing and Selection of Model:

So, I decided to choose 4 supervised learning models: Decision tree, KNN, XGBoost and RandomForest. As we mentioned earlier, our dataset is highly unbalanced, which makes our project more interesting, we don't have to focus so much on our accuracy score, in this scenario we are going to shift our focus on F1 score, Precision Score and Recall Score. If we talk about precision in this F1 score, "Of all the instances predicted as positive, how many are actually positive?" A high precision indicates that the model has a low false positive rate, which is crucial in fraud detection as it ensures that legitimate transactions are not mistakenly flagged as fraudulent. In case of recall, which also means sensitivity, it stated as "Of all the actual positive instances, how many were correctly predicted by the model?" A high recall indicates that the model effectively captures the majority of positive instances, minimizing false negatives.

One thing to note, we used cross validation techniques in all of my models to get training score.

Let's analyze all models step by step:

4.2 Decision Tree

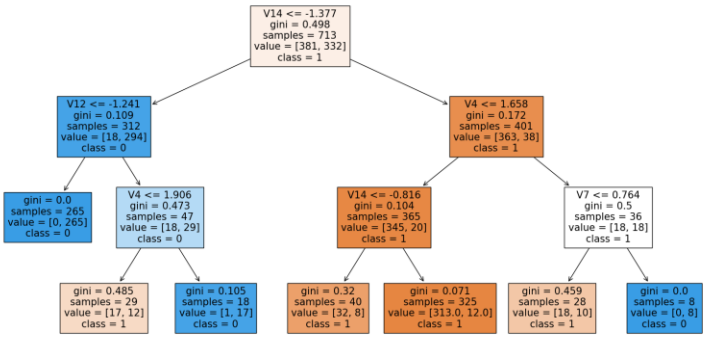
Decision trees offer transparency in decision-making, aiding in understanding why a transaction is labeled as fraudulent or not. Initially, I implemented a decision tree model without pruning. I assessed the model's performance using training and testing scores, as well as F1, precision, and recall scores suitable for unbalanced data. Then, I experimented with additional features using the GridSearchCV method, resulting in different outcomes compared to the unpruned model. This highlighted the risk of overfitting in the unpruned model, with improved results in both datasets. Moreover, a notable observation is that our decision tree is concise, comprehensible, and easy to interpret.

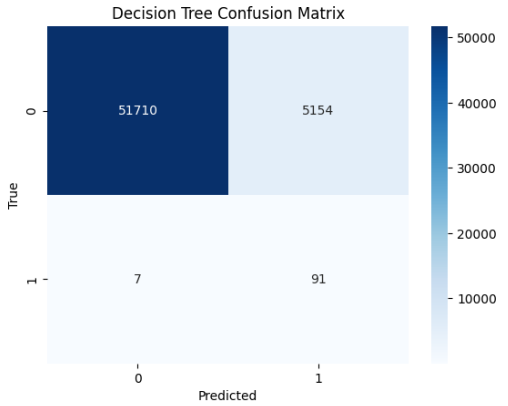
Next, we compare the performance of the two models on each dataset:

Undersampling Dataset:

For the decision tree without pruning in this dataset, we achieved a satisfactory training score of approximately 98.76% and a test score of around 90.05%. The recall score was 0.93, and the precision score was 0.03. In comparison, when utilizing the GridSearch technique for pruning, we retrained our model with parameters 'max_depth': 3 and 'max_features': 9. Consequently, there was a notable enhancement in the test score, which reached 98.02%. Additionally, the recall score improved to 0.91, and the precision score increased to 0.07, reflecting significant improvements over the previous model.

Below is an image of the pruned decision tree:



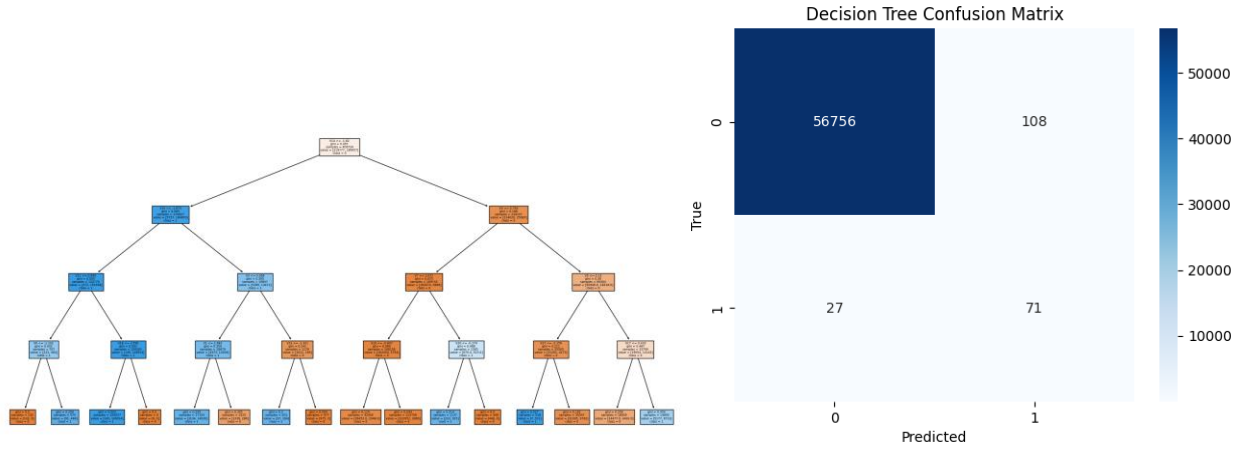


When examining the confusion matrix, it becomes evident that our model might misclassify some fraudulent transactions (5154 Records). However, despite this limitation, the simplicity of the decision tree enables the detection of non-fraudulent transactions with remarkably high accuracy with very low errors (7 Records).

Oversampling Dataset:

For this dataset, we initially trained the model without pruning, achieving a training score of approximately 99.87% and a test score of around 99.76%. The recall score was 0.72, and the precision score was 0.40. After applying GridSearch for pruning with parameters 'max_depth':4 and 'max_features':11, we obtained the following results:

Training Accuracy = 0.9966
Test Accuracy = 0.9563
Recall Score = 0.90
Precision Score = 0.03



When examining the confusion matrix, we observe that when using the Oversampling method, we have fewer incorrect fraud detections for non-fraudulent transactions (108 Records). However, there are more undetected fraudulent transactions (27 Records).

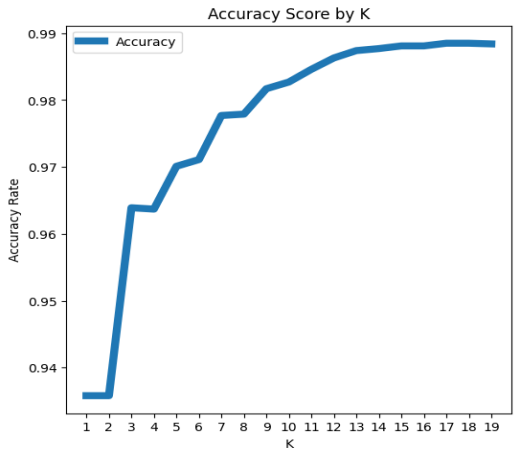
4.3 KNN

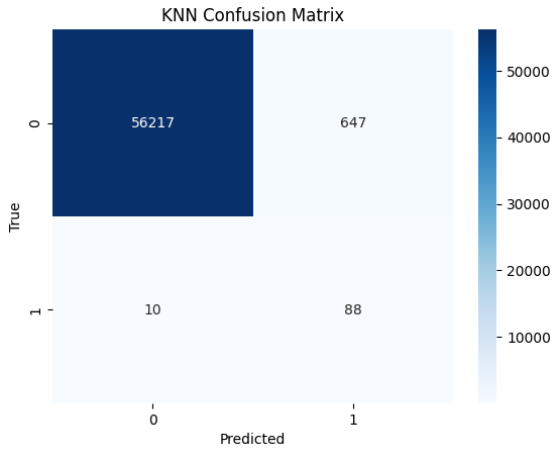
KNN is like a friendly neighbor. It looks at nearby data points to figure out where a new point fits in. This makes it great for finding weird stuff in credit card transactions. As before heading in to it, we have drop ped some of the coloumns of my data set which was irrelevant, Because KNN can be misled by irrelevant attributes. Unlike other models, KNN is very sensitive to the high dimensionality of data.

Undersampling Dataset:

First, we have to find the best value of k in the range from 1 to maximum, which is 20 and we found the optimal value of k that maximizes the accuracy of the KNN classifier for the given dataset. Then we implemented this model with value of **k=18** and these are the results that we encountered:

Training Accuracy = 0.9976
Test Accuracy= 0.9884
Recall Score = 0.90
Precision Score= 0.12

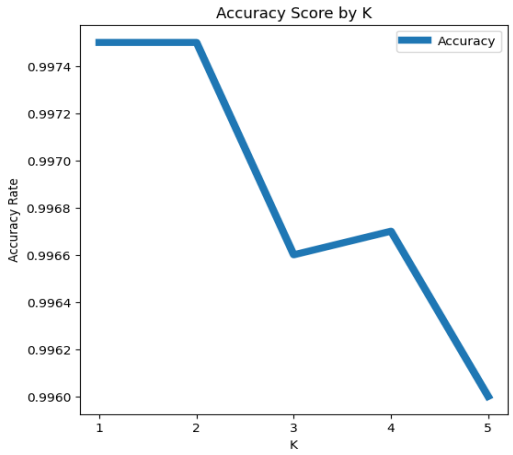


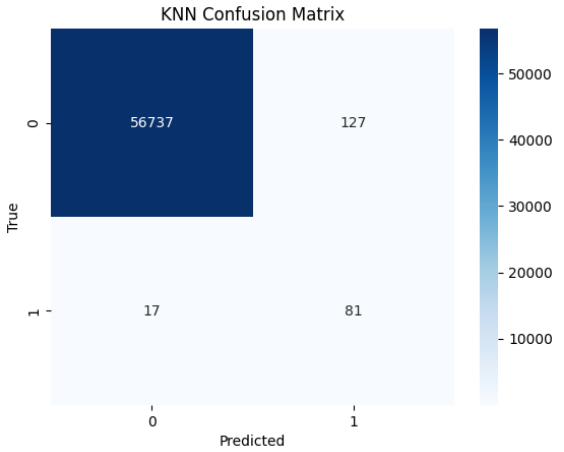


Looking at the confusion matrix for KNN, we notice that we have significantly fewer incorrect fraud detections compared to our previous models in Undersampling dataset.

Oversampling Dataset:

For this dataset, we found the optimal K value to be 2, and we trained the model with k=2. Here are the results we obtained:
Training Accuracy = 0.9991
Test Accuracy = 0.9974
Recall Score = 0.83
Precision Score = 0.39





When examining the confusion matrix, we notice a better trade-off between Recall and Precision compared to the Oversampling dataset. We have more undetected fraudulent transactions, but on the other hand, the number of non-fraudulent transactions that we incorrectly detected as fraudulent has decreased significantly.

4.4 XG-Boost:

Undersampling Dataset:

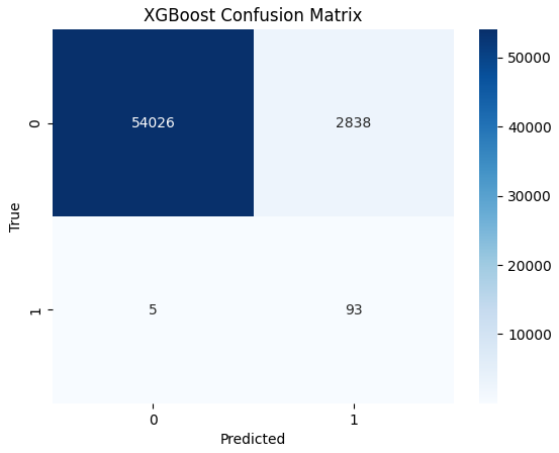
XG-Boost is like a superhero for data. It helps us see which parts of our data are super important, kind of like shining a spotlight on the most crucial bits. We implemented this model first with normal boost (without any feature) and these are the results that we encountered:

Training Accuracy = 0.9972
Test Accuracy= 0.9468
Recall Score = 0.95
Precision Score= 0.03

Let see, how our model performed after doing some hyper tunning of parameters. So, by using GridSearchCV I calculated the best values for n_estimator, learning_rate and max_depth. So, by adjusting this we got following results:

Training Accuracy = 0.99781
Test Accuracy= 0.95
Recall Score = 0.95
Precision Score= 0.03

The hyperparameter tuning process has not resulted in model improvements.



After evaluating the confusion matrix, it's evident that we have high errors in detecting fraudulent trans actions when they are actually non-fraudulent. However, we actually detect all fraudulent transactions.

Oversampling Dataset:

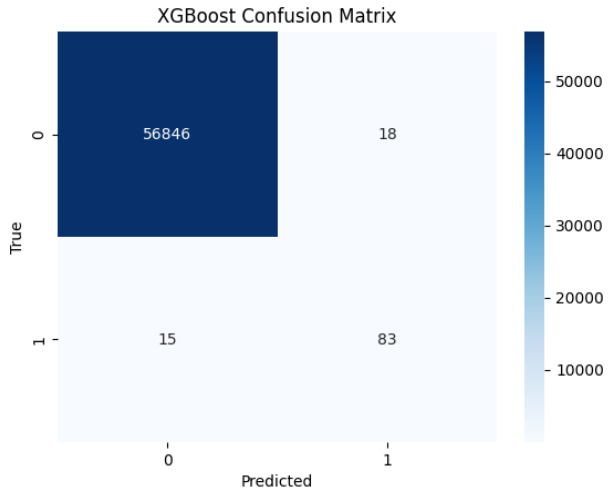
Same, we implemented this model first with normal boost (without any feature) and these are the results that we encountered:

Training Accuracy = 0.9998

Test Accuracy= 0.9994
Recall Score = 0.85
Precision Score= 0.82

Then, we performed some hyper tunning.So, by using GridSearchCV we calculated the best values for n_ estimator, learning_rate and max_depth. By adjusting this we got following results:
Training Accuracy = 0.9992
Test Accuracy= 0.9993
Recall Score = 0.84
Precision Score= 0.80

After hyperparameter tuning, our model's accuracy, recall score, and precision experienced a slight decrease.



Looking at the confusion matrix, we observe a very good balance between Recall and Precision scores. We have significantly fewer incorrect fraud detections compared to the Undersampling models.

4.5 RandomForest

Random Forest is an ensemble learning method that combines multiple decision trees, resulting in a more robust and accurate model. Its ability to handle unbalanced datasets and identify important features makes it well-suited for fraud detection tasks.

Undersampling Dataset:

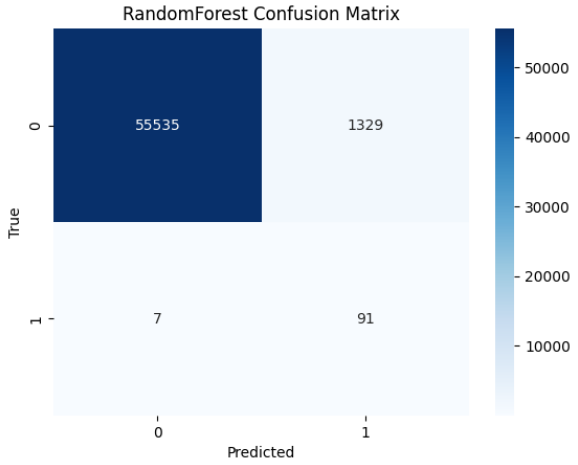
We implemented this model first with normal boost (without any feature) and these are the results that we encountered:

Training Accuracy = 0.9991
Test Accuracy= 0.9765
Recall Score = 0.93
Precision Score= 0.06

Then, we tried to hyper tunned the parameters and then will analyze how our model behave with that. So, by using GridSearchCV we calculated the best values for n_estimator, max_depth, min_samples_split. By adjusting this we got following results:

Training Accuracy = 0.9981
Test Accuracy= 0.9609
Recall Score = 0.93
Precision Score= 0.04

After hyperparameter tuning, our model's accuracy, recall score, and precision experienced a slight decrease.



By evaluating the random forest confusion matrix, we conclude that, similar to other models using undersampling methods, our model incorrectly identifies many non-fraudulent transactions as fraudulent.

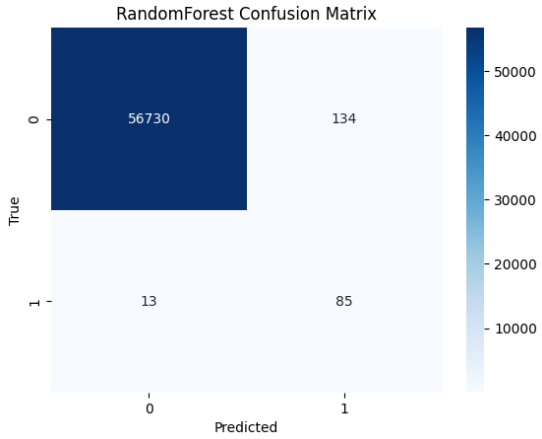
Oversampling Dataset:

For this dataset, we trained the model initially with normal boosting (without any additional features), and here are the results we obtained:

Training Accuracy = 0.9997
Test Accuracy= 0.9994
Recall Score = 0.79
Precision Score= 0.90
Then, by utilizing GridSearchCV, we calculated the best values for n_estimator, max_depth, and min_samples_split. Adjusting these parameters, we obtained the following results:

Training Accuracy = 0.9998

Test Accuracy= 0.9974
Recall Score = 0.87
Precision Score=0.39
After hyperparameter tuning, we observed an improvement in the Recall Score, but there was a significant decrease in the Precision Score. As a result, the newest model performs better at balancing recall and precision.



5. Comparing Models and Sampling Methods

__	UnderSampling Method				OverSampling Method			
Score	Train	Test	Recall	Precision	Train	Test	Recall	Precision
Decision Tree	98.76%	90.93%	93%	2%	99.89%	99.76%	72%	40%
Using GridSearch	99.91%	98.02%	91%	7%	98.66%	95.63%	90%	3%
KNN	99.86%	98.84%	90%	12%	99.91%	99.74%	83%	39%
XG-Boost	99.72%	94.68%	95%	3%	99.98%	99.94%	85%	82%
Using GridSearch	99.78%	95%	95%	3%	99.93%	99.93%	84%	80%
RandomForest	99.91%	97.65%	93%	6%	99.97%	99.94%	79%	90%
Using GridSearch	99.84%	96.09%	93%	4%	99.95%	99.74%	87%	39%

Precision, as the name suggests, indicates how precise or confident our model is in detecting fraudulent transactions. Recall, on the other hand, measures the proportion of actual fraudulent cases that our model is able to detect.

Choosing which method is better than the other depends on our goal and use case. In undersampling methods, we detect more fraudulent transactions correctly, whereas we achieve a higher recall score. However, there is a larger number of non-fraudulent transactions incorrectly identified as fraudulent (resulting in a very low precision score). In this scenario, it's maybe insufficient for customers if their transactions are flagged as fraudulent unnecessarily, the model's precision becomes a little crucial.

Undersampling methods achieve a more balanced approach between detecting fraud and non-fraud instances. SMOTE Technique provide greater accuracy compared to random under-sampling, but it typically requires more time for training due to volumes of data.

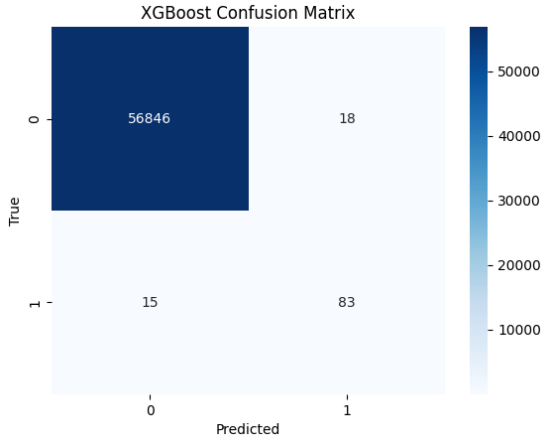
6. Conclusion

In conclusion, this project aimed to explore the impact of imbalanced datasets on the performance of various classification models in detecting credit card fraud. We evaluated four supervised learning models: Decision tree, KNN, XGBoost, and RandomForest on two datasets developed through resampling techniques.

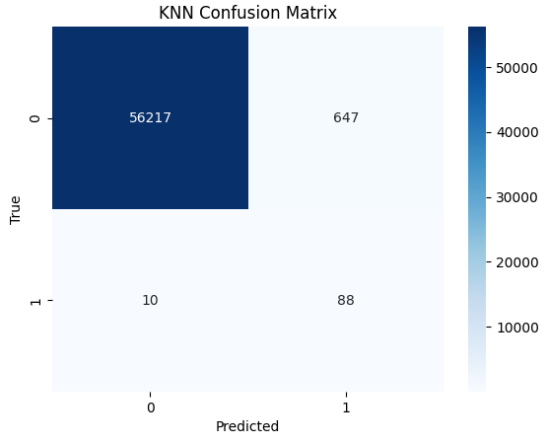
Our findings suggest that oversampling methods achieve a more balanced approach between detecting fraud and non-fraud instances. This technique provides greater accuracy but requires more training time due to data volumes.

For undersampling methods, KNN performed better, achieving a better balance between Recall and Precision.

For oversampling methods, XGBoost without tuning performed significantly better than the others and emerged as the best model in our analysis.



(1) Best Oversampling Result



(2) Best Undersampling Result

7. Acknowledgements

The dataset for this project has been obtained from the following websites:

https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud?select=creditcard.csv

8. Reference

https://www.datacamp.com/tutorial/decision-tree-classificaNon-python

https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets

https://www.scribbr.com/statistics/outliers/