MACHINE LEARNING IN FINANCE

CREDIT CARD FRAUD DETECTION DEALING WITH UNBALANCED DATA

1. Business Understanding

Credit card fraud can be defined as a case where someone uses someone else´s credit card for personal reasons while the owner and the card issuing authorities are unaware of the fact that the card is being used.

In an era of digitalization, the need to identify credit card frauds is necessary. Fraud detection involves monitoring and analysing the behaviour of various users to perceive, detect or avoid undesirable actions. To perform credit card fraud detection effectively, there is a need to understand what is behind it.

This problem is particularly challenging from the perspective of learning, credit card fraud is usually associated with class imbalance. Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally. In credit card fraud it is very common that the number of legitimate transactions far outnumber the ones that are considered fraudulent.

Machine learning models are employed to analyse all the authorized transactions and report any suspicious activities. These reports are then investigated by professional who contact cardholders to confirm if the transaction performed was genuine or not. So, in hindsight, there is a need to train and update these models regularly to improve fraud-detection performance overtime.

Using machine learning to detect fraud is part of the wider effort of anomaly detection, which aims to identify rare events which differ significantly from most of the data. There are two main categories within anomaly detection:

- Unsupervised anomaly detection
- Supervised anomaly detection

Unsupervised techniques aim to detect anomalies in unlabelled datasets by grouping events according to their features. Clustering is a commonly used technique, and, more recently, autoencoders have also proven useful for feature extraction. The assumption here is that in each dataset where most events are non-fraudulent, the model will learn to reconstruct them better than anomalous ones and the reconstruction error will be higher for the outlier events.

Supervised techniques require the dataset to contain a label indicating whether an event is fraudulent or not. However, labelled datasets are limited, with only a few high-quality public datasets available.

The methodology of this report is to use the latest machine learning algorithms to detect any anomalous activities, called outliers.

Firstly, we obtained the dataset from Kaggle[1], a data analysis website that provides public datasets. The dataset contains transactions made by credit cards in September 2013 by European cardholders. We know present a brief description of each variable:

Table I – Variable Description

Variable	Description
Time	Represents the seconds elapsed between that particular transaction and the first transaction in our data.
Vn	It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, additional information regarding the original features cannot be provided.
Amount	Represents the transaction amount
Class	Represents our class variable where it takes the value of 1 if we are dealing with a fraudulent transaction and 0 for genuine transactions

[1] Dataset available at: https://www.kaggle.com/mlg-ulb/creditcardfraud

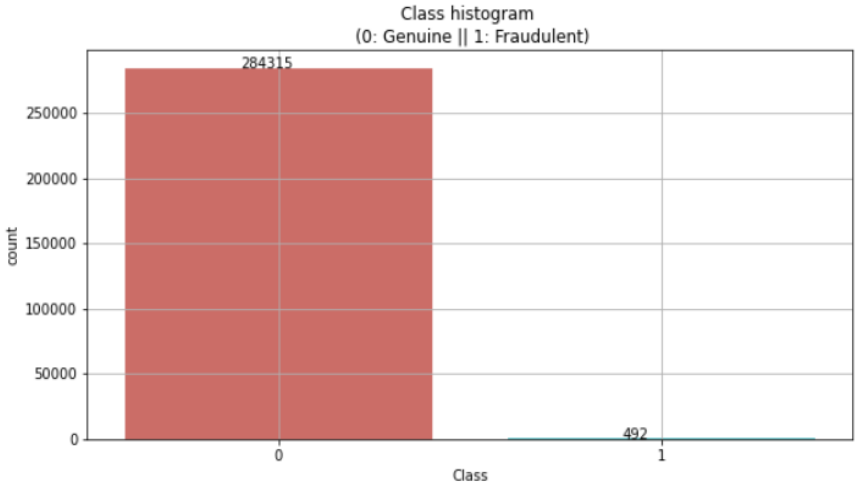
2. Data Understanding

The first basic step is to understand our data. The unique thing about our data set is that other than the columns “transaction” and “amount”, rest of the columns are anonymised for privacy concerns.

It is mentioned in the data description available at Kaggle that the data went through PCA transformation. It also mentions that it can be assumed that prior to the application of PCA, data was scaled as per normal convention.

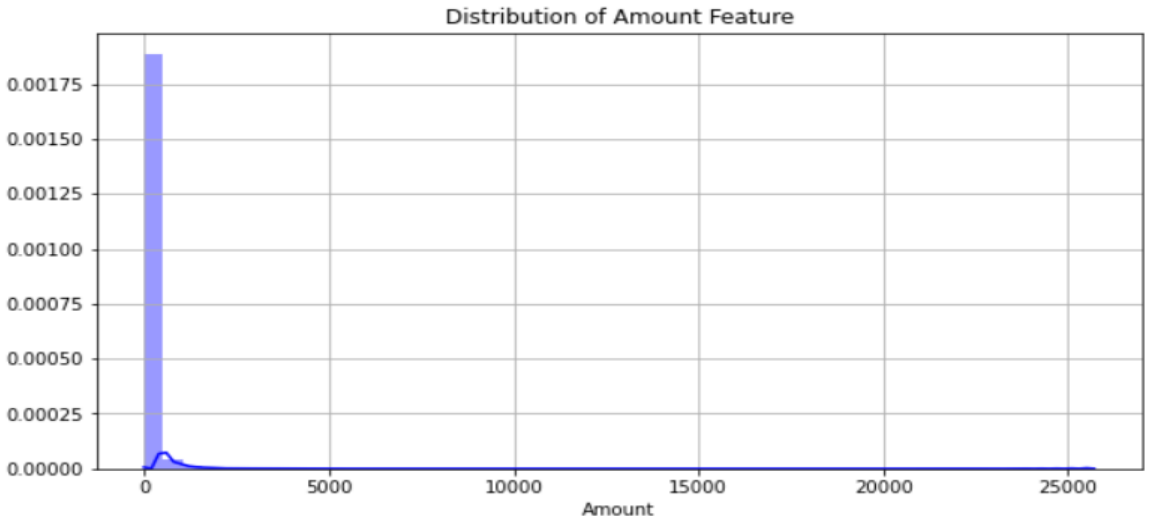
Inside this dataset there are 31 columns and 284.807 rows. We start this data understanding by plotting different graphs to check for any inconsistencies in the dataset:

```
0    284315
1       492
Name: Class, dtype: int64
```



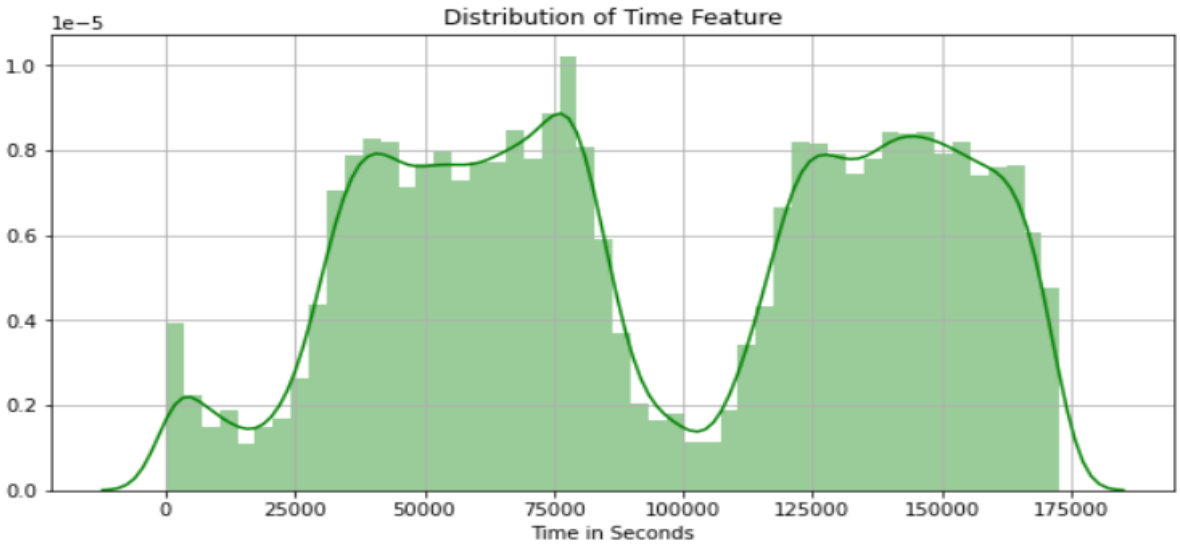
This simple countplot shows that the number of fraudulent transactions are much less than the legitimate ones. In fact, 284.315 correspond to non fraudulent transations and 492 are related to fraudulent ones. If we apply a classification algorithm, it will learn well about the genuine transactions and not learn much from the fraudulent transactions since there are very few (class imbalance problem).

Since nearly all predictors have been anonymised, we decided to focus on the non-anonymised predictors time and amount of the transaction. The mean value of all transactions is $88.35 while the largest transaction recorded in this data set amounts to $25.691,16. Plotting the distribution, we have:



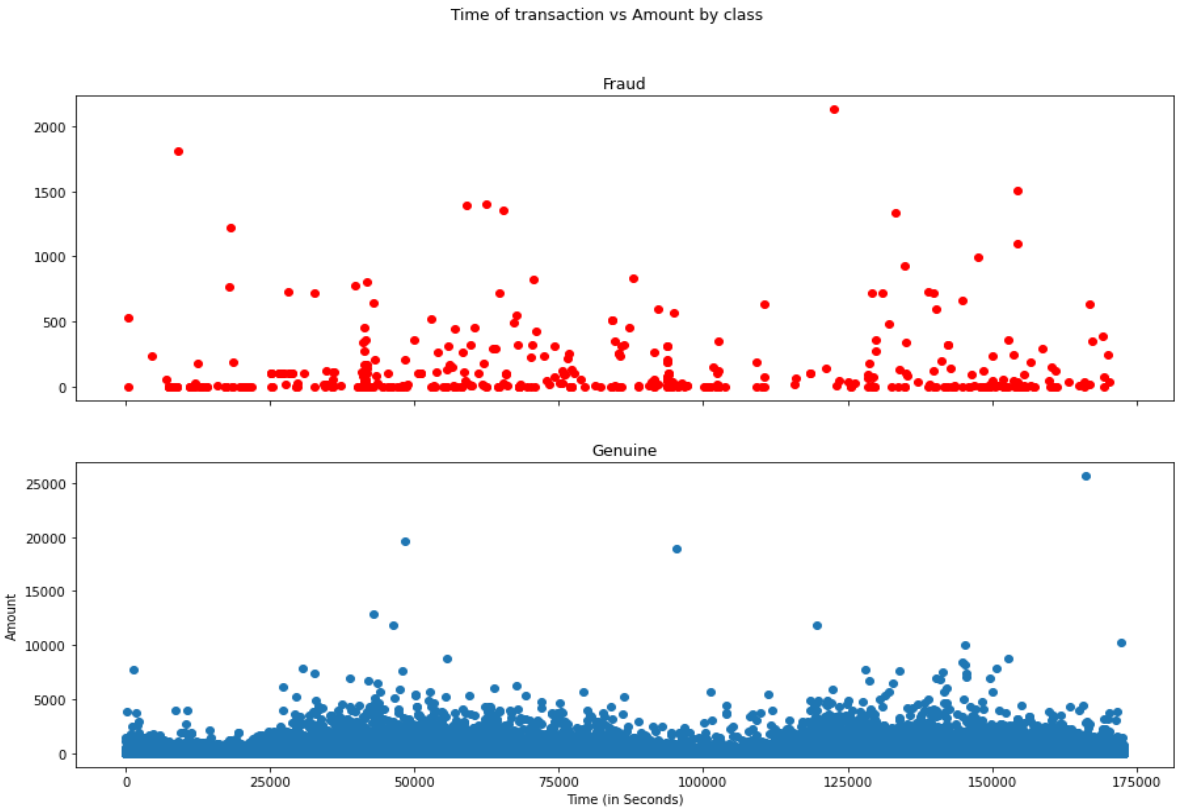
Based on the mean and maximum value, the distribution of the monetary value of all transactions is heavily right-skewed. Most transactions are relatively small and only a small fraction of transactions comes even close to the maximum.

We know present the distribution of the time feature:



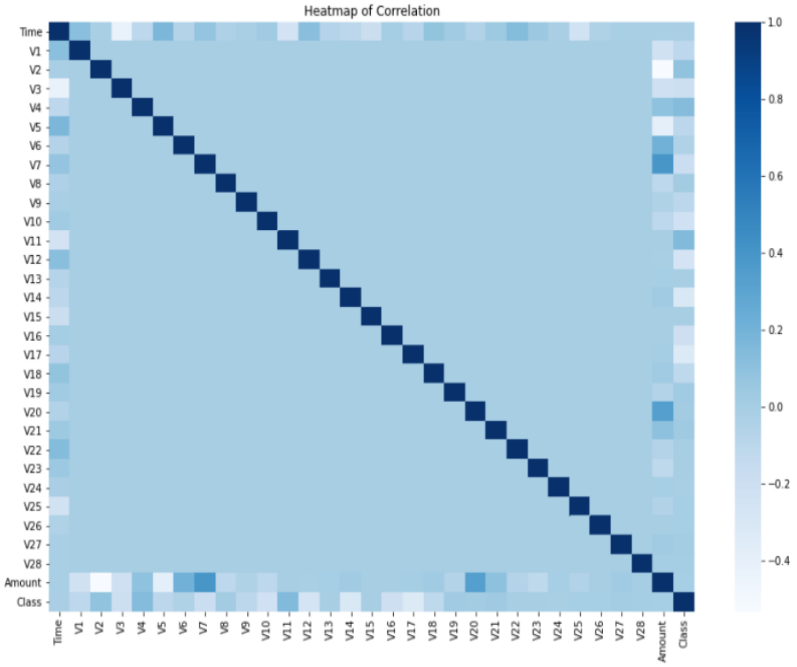
According to the dataset description these transactions occurred in two days. If we take a closer look at the graph, it is evident that fewer transactions occurred at night-time.

We then decided to split the dataset into genuine and fraudulent data to see if we can pinpoint if fraudulent transactions occur more frequently at daytime or at night-time.



Looking at both scatterplots, the data looks to be equally distributed so we can not make any conclusions regarding the timings at which the frequency of fraudulent activities is higher.

Finally, it would be interesting to know if there are any significant correlations between our predictors, especially with regards to our class variable. One of the most visually appealing ways to determine that is by using a heatmap.



Looking at the heatmap, some of the predictors seem to be correlated with the class variable. Nonetheless, there seem to be relatively little significant correlations for such a big number of variables. The huge class imbalance might distort the importance of certain correlations with regards to our class variable.

Looking at the correlation map the highest correlations found were:

- Time and V3 with -0.42.
- Amount and V2 with -0.53.
- Amount and V4 with 0.4.

While these correlations are relatively high in comparison to the others, we considered that these were not high enough to consider the risk of multicollinearity.

3. Data Preparation

In this process, we are going to define the independent (X) and the dependent variables (Y). Using the defined variables, we will split the data into a training set and testing set which is further used for modelling and evaluating. We can split the data easily using the 'train_test_split' algorithm in python.

While also dealing with the data preparation we opted to scale the amount feature since its heavily right-skewed and to avoid any kind of distortion. We then excluded the time and old amount feature for better and more accurate results.

4. Modelling

To predict fraudulent transactions, we opted to build five different classification models more specifically:

- **Decision Tree:** A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.
- **K-Nearest Neighbours (KNN):** The k-nearest neighbours (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

- **Logistic Regression**: Logistic regression is another technique borrowed by machine learning from the field of statistics. It is a method used for binary classification problems.
- **Support Vector Machine (SVM**): supervised learning models with associated learning algorithms that analyse data for classification and regression analysis.
- **Random Forest**[2]: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

All these models can be easily built using the algorithms provided by the scikit-learn package. We now present a summary of each model used.

Table II – Classification Models

Classification models	Algorithm	Stored Predictions (train set)	Stored Predictions (test set)
Decision Tree	DecisionTreeClassifier	tree_y_train	tree_yhat
K-Nearest Neighbours	KNeighboursClassifier	knn_y_train	knn_yhat
Logistic Regression	LogisticRegression	lr_y_train	lr_yhat
Support Vector Machine	SVC	svm_y_train	svm_yhat
Random Forest	RandomForestClassifier	rf_y_train	rf_yhat

Our next step is to evaluate each of the models and find which is the most suitable one.

5. Evaluation and Performance Metrics

In this final phase we are going to evaluate the results of our built models using the evaluation metrics available at the scikit/learn package. We considered the following metrics:

Confusion Matrix: A confusion matrix is a table layout which allows the visualization of the performance of an algorithm. A confusion matrix gives us four important measures:

[2] We opted for both the decision tree and the random forest because the first uses the entire dataset to construct a single model whereas, the random forest uses randomly selected features to construct multiple models.

Table III – Confusion matrix

		Actual Values	
		Negative (Genuine)	Positive (Fraud)
Predicted Values	Negative (Genuine)	True Negative (TN)	False Positive (FP)
	Positive (Fraud)	False Negative (FN)	True Positive (TP)

Explaining each of them we have:

- TP – Fraudulent cases that are correctly classified as fraudulent transactions;
- TN – Genuine cases that are correctly classified as genuine transactions;
- FP - Transactions that are genuine, but the model predicts as fraud;
- FN – Transactions that are fraud, but the model predicts as genuine.

Accuracy Score: Accuracy score is one of the most basic evaluation metrics which is widely used to evaluate classification models. The accuracy score is calculated simply by dividing the number of correct predictions made by the model by the total number of predictions made by the model. The formula is presented below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: Precision measures how accurately the model can capture fraud i.e out of the total predicted fraud cases, how many turned out to be fraud.

$$Precision = \frac{TP}{TP + FP}$$

Recall: Recall measures out of all the actual fraud cases; how many the model could predict correctly as fraud. This is an important metric to be considered as we want find out how the model behaves in predicting fraudulent transactions.

$$Recall = \frac{TP}{TP + FN}$$

F1-score: this is a balance between precision and recall.

$$F1 - Score = 2 * \frac{(precision * recall)}{precision + recall}$$

We know present the result for both the training and test set for each of the classification models (each value was rounded up to 3 decimal points):

Table IV – Results of each classification model

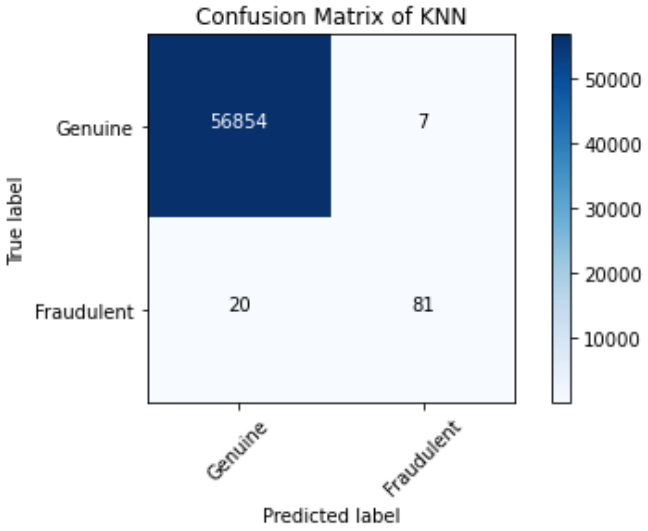
	DT		KNN		LR		SVM		RF	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Accuracy	0.999	**0.999**	1.000	**1.000**	0.999	**0.999**	1.000	**0.999**	0.999	**0.999**
Recall	0.793	**0.762**	0.798	**0.802**	0.611	**0.634**	0.806	**0.673**	0.729	**0.673**
Precision	0.891	**0.865**	0.957	**0.920**	0.879	**0.877**	0.981	**0.919**	0.944	**0.909**
F1 Score	0.839	**0.811**	0.870	**0.857**	0.721	**0.736**	0.885	**0.777**	0.823	**0.787**

Before looking at the values that were predicted using each classification metric, we first are going to elaborate on why we are not going to use accuracy as performance metric. Accuracy simply refers to the proportion of correctly classified instances. It is usually the first metric you look at when evaluating a model. Since we are dealing with imbalanced data accuracy can be misleading, a model can predict the value of the dominant class for all predictions and achieve a high accuracy score. This means that additional performance metrics are going to be required meaning we are mainly going to focus on the recall score because it focuses on correctly predicting fraud transactions.

At first glance the evaluations metrics suggest that the KNN model is the most accurate model to predict fraudulent transactions while the logistic regression model looks to be the least accurate model. An additional remark, is the difference between both the training and test set, is is possible that we may be dealing with some overfitting.

Let us know take the confusion matrix of the KNN model as an example[3]. The first row is for transactions whose actual fraud value in the test set is 0 meaning the genuine ones, while the second row is for the transactions that are fraudulent where the test set is equal to 1.

[3] Please consider the python file attached to this report which contains the plots of the remaining confusion matrices.



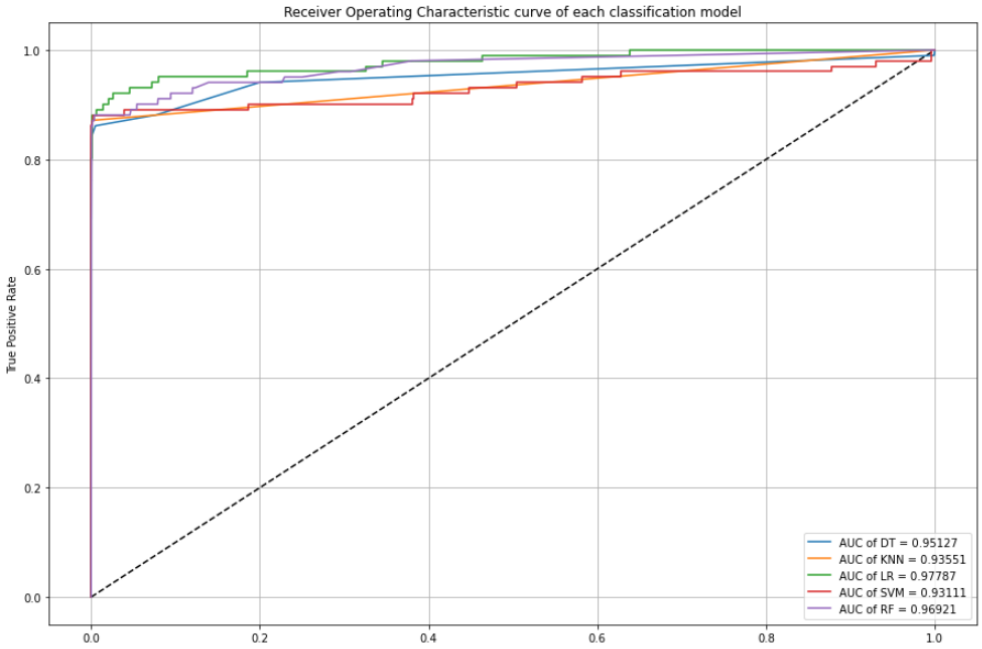
Looking at the first row the KNN model classified 56.861 as genuine transactions. Out of these 56.861 transactions the model correctly predicted 56.854 (TN) as genuine transactions and only 7 (FP) were mistakenly classified as fraud. Considering the second row of a total of 101 transactions the model correctly predicted 81 (TP) as fraudulent ones while classifying incorrectly 20 (FN) fraudulent transactions as genuine.

Comparing the confusion matrix for all the models, the KNN model has performed a very good job of classifying the fraud transactions from the non-fraud transactions.

So, at first glance, the KNN model looks to be the most appropriate model.

As an additional performance measure, we also considered looking at the Receiver Operating Characteristics (ROC) curve and the area under the curve (AUC).

AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represent the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting genuine cases as genuine and fraudulent cases as fraudulent. By analogy, the Higher the AUC, the better the model is at distinguishing both classes.



Looking at the AUC score of each model it is evident that every model is, in some regard, very good at predicting both genuine and fraudulent transactions.

From this point on, we decided to tackle this classification problem by resampling our data to see if we were able to obtain better results.

6. Dealing with imbalanced data: Resampling Techniques

Since we exposed the problem of imbalanced data and how it provides misleading classification we would like to know if there is a way of improving these results.

Sampling is a good approach to deal with imbalanced data. We change the dataset in such a way that the model gets balanced data as input. There are two types of sampling — undersampling and oversampling.

The idea of undersampling is to delete some instances of the oversampled class to obtain class equivalence. If you have a very large dataset, it is a good approach to try undersampling. We must make sure that we do not lose any relevant information by undersampling.

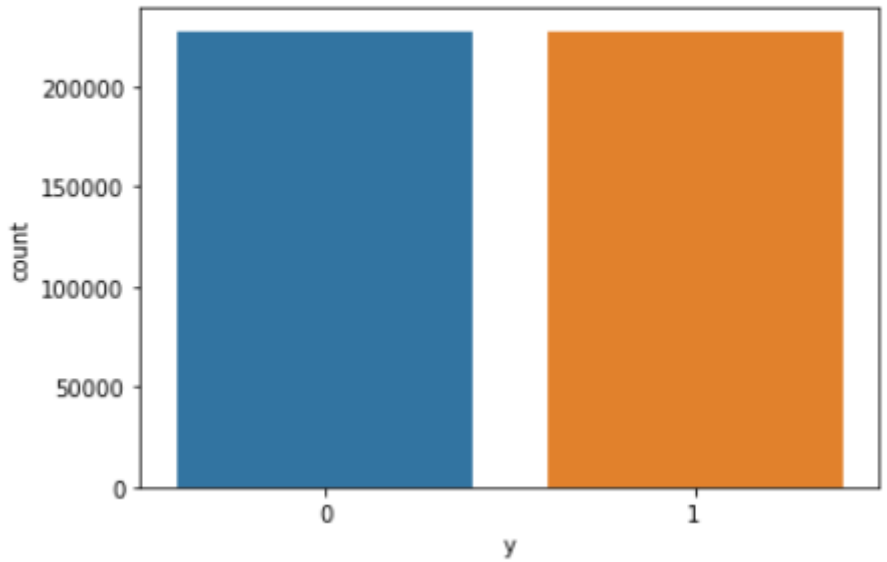
Oversampling uses copies of the underrepresented class to get a balanced data. In our case, we add copies of fraudulent class so that we get equal number of genuine and fraudulent cases. This new

dataset is given as input to the classification algorithm. If you are working with a small dataset, it is a good step to apply oversampling.

Synthetic Minority Oversampling Technique (SMOTE) is a very popular oversampling technique. SMOTE technique generates new synthetic minority class data which is used to train the model. It selects minority class data points that are close in the feature space and draws a line between the selected data points. A new sample is generated at a point along this line. This procedure is used to create the required number of synthetic samples of the minority class. This approach is effective because new synthetic samples from the minority class are relatively close in feature space to existing minority class data.

By using the SMOTE we can generate new instances of the minority class as presented in this simple plot:

```
Counter({0: 227454, 1: 227454})
<matplotlib.axes._subplots.AxesSubplot at 0x16c82d271f0>
```



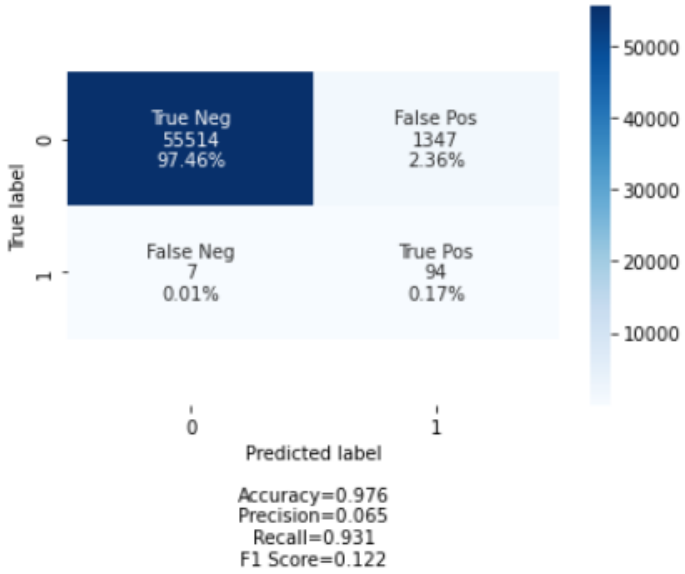
For this approach we are going to apply SMOTE technique to every model to see if our scoring metrics benefit from it.

We present model each a table containing the results of each metric after applying the SMOTE technique:

Table IV – Results of each classification model

	DT		KNN		LR		SVM		RF	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Accuracy	0.999	**0.998**	0.999	**0.998**	0.942	**0.976**	0.965	**0.986**	1.000	**1.000**
Recall	1.000	**0.861**	1.000	**0.861**	0.908	**0.931**	0.944	**0.911**	1.000	**0.842**
Precision	0.999	**0.518**	0.999	**0.518**	0.974	**0.065**	0.986	**0.107**	1.000	**0.895**
F1 Score	0.999	**0.647**	0.999	**0.647**	0.940	**0.122**	0.965	**0.192**	1.000	**0.867**

It is evident that all the classification models improved their recall score, more specifically the Logistic Regression and Support Vector Machine. So, in hindsight, considering only the recall as a performance measure after applying the SMOTE the logistic regression is the model that was able to correctly predict more fraudulent transactions. Presenting the confusion matrix of the logistic regression:



The model correctly predicted 56.854 (TN) as genuine transactions and only 1.347 (FP) were mistakenly classified as fraud. Considering the second row of a total of 101 transactions the model correctly predicted 94 (TP) as fraudulent ones while classifying incorrectly 7 (FN) fraudulent transactions as genuine.

If we consider, the Logistic Regression confusion matrix without the SMOTE we were only able to correctly predict as fraud 64 out of 101 fraudulent transactions.

However, if we consider the other metrics, we noticed that the Logistic Regression precision score was reduced significantly to 0.065 so even though the model shows a clear improvement in the recall score

the decrease in the precision score ends up hurting the performance of this specific classification problem.

The best model using the SMOTE approach is clearly the Random Forest. The model did end up sacrificing a bit of precision but in return the recall did increase tremendously.

The ROC AUC in general in all the models did not suffer many changes as expected these classification models work well in imbalance problems.

To conclude our analysis the model that should be used will depend on the kind of analysis that we are trying to make. If we do not consider resampling techniques then the KNN model seems to be the best model out of the five presents, considering resampling techniques then the Random Forest is the most appropriate even though the Logistic Regression ended up having a higher recall value.

7. Conclusions

Fraud detection is a key area where machine learning can lead to billions of savings for businesses while providing customers with a safer environment. Through sophisticated feature engineering and modeling techniques, such as, classification models, autoencoders and clustering, machine learning can help detect fraudulent events as anomalies in a customer purchasing journey. Implementing the right machine learning solution means going beyond simple solutions.

In this report, we have successfully built five different types of classification models starting from the Decision tree model to the Support Vector Machine model. After that, we have evaluated each of the models using the evaluation metrics and chose which model is most suitable for the given case. Later we decided to approach this problem by using resampling techniques to see if our results improved. We limited our model count to five but, there are many more models to explore that may end up providing with better predictions.

Since the entire data set consist of transaction records of only two days, the machine learning algorithm will only increase its efficiency over time as more data is put into it.

While we could not reach out 100% fraud detection, we did end up creating a system that can, with enough time and data, get very close to that goal. As with any project, there is some room for improvement here. This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of versatility to the project. More room for improvement can be found in the dataset.

Hence, more data will surely make the model more accurate in detecting frauds and reduce the number of incorrect predictions. However, this will require access to relevant information that can only be provided from the banks themselves.