**Credit Card Fraud Detection**

Fraud detection is a serious problem of the modern world. The news reports on countless attacks of credit card information being stolen annually. The transactions of fraudulent cards might seem minimal in occurrence but, on a larger scale these small occurrences can cost companies millions in losses. This is where machine learning comes into play learning from the patterns and adapting to new possible schemes.

Machine learning can give credit card companies the ability to recognize fraudulent credit card transactions so that customers are not charged for items that theydid not purchase. Fraud detection requires training an algorithm to identify uncommon observation from any normal observation.

Fraud detection in credit card is a data mining problem. It becomes challenging due to two major reasons – first, the profiles of normal and fraudulent behavior change frequently and second, the credit card fraud data sets are highly skewed. This paper investigates and checks the performance of Random Forest, Isolation Forest, Local Outlier Factor, XGBoost on highly skewed credit card fraud data. Dataset of credit card transactions is sourced from European cardholders containing 284,786 transactions. These techniques areapplied on the raw and pre-processed data.

**Outline**

* Problem
* Dataset
* Fraud Detection
* Machine Learning approaches in Fraud Detection
* Anomaly Detection
* Modeling
* Supervised Learning
* Unsupervised Learning
* Anomaly vs Supervised Learning
* Testing and Tuning
* Deployment and Pipeline
* Sum up

**Problem Statement**

Fraud detection is a set of activities that are taken to prevent money or property from being obtained through false pretenses. Fraud can be committed in different ways and in many industries. Most detection methods combine a variety of fraud detection datasets to form a connected overview of both valid and non-valid payment data to make a decision. The Credit Card Fraud Detection Problem includes modeling past credit card transactions with the knowledge of the ones that turned out to be a fraud. This model is then used to identify whether a new transaction is fraudulent or not. My aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud predictions.

**Data Collection**

The datasets contain transactions made by credit cards in September 2013 by European cardholders.

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.

. The dataset has been collected and analyzed during a research collaboration of Worldline and the Machine Learning Group of ULB (University Libre de Brucellas) on big data mining and fraud detection.

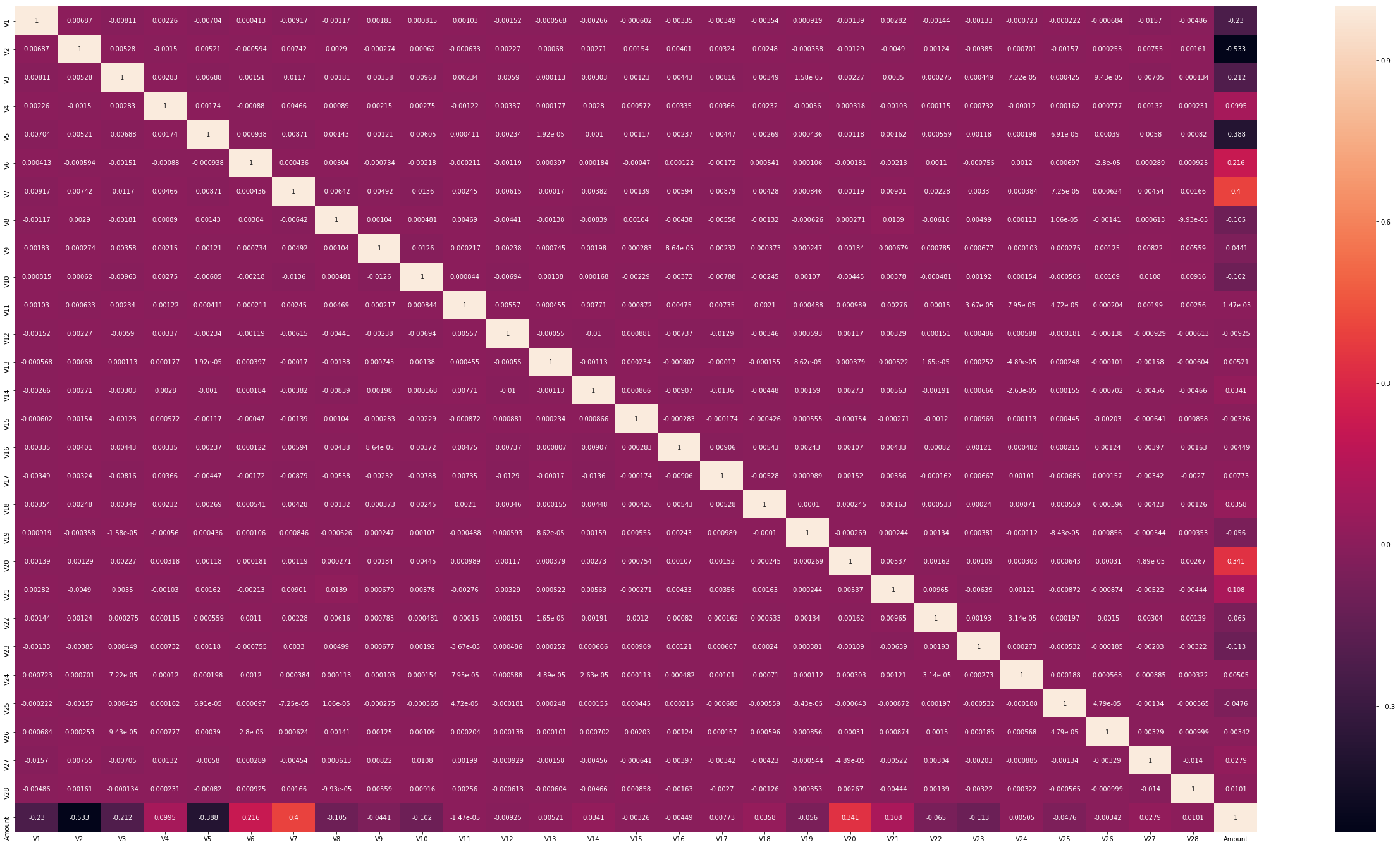
It contains only numeric input variables which are the result of a PCA transformation. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA is Time and Amount. Feature Time contains the seconds between each transaction and the first transaction in the dataset. The feature Amount is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning. Feature Class is the response variable and it takes value 1 in case of fraud and 0 otherwise.

**Exploratory Visualization**

**Fig 1.** A plot showing the ratio of fraudulent transactions to genuine transactions, from this visualization it is obvious that fraudulent cases are grossly misrepresented and may do some harm while training our model.



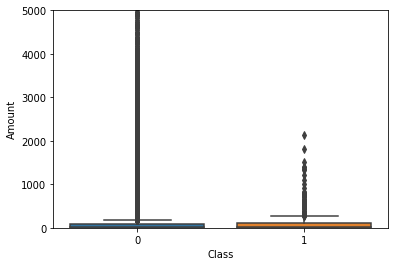
Correlation matrix graphically gives us an idea of how features correlate with each other and can help us predict what are the features that are most relevant for the prediction. Finally, it would be interesting to know if there are any significant correlations between our predictors, especially with regards to our class variable. One of the most visually appealing ways to determine that is by using a heatmap.



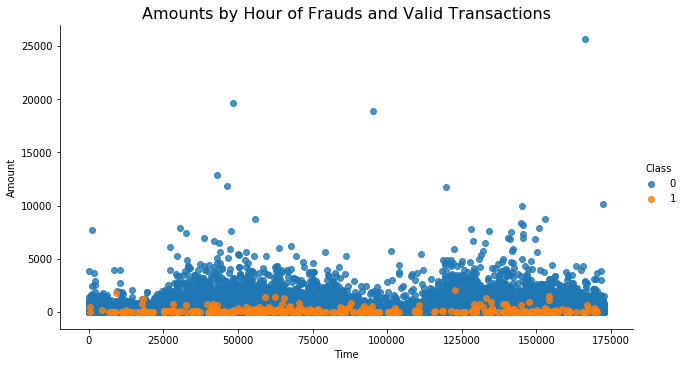
In the Heatmap we can clearly see that most of the features do not correlate to other features but there are some features that either has a positive or a negative correlation with each other. For example, V2 and V5 are highly negatively correlated with the feature called Amount. We also see some correlation with V20 and Amount. This gives us a deeper understanding of the Data available to us.

Let's plot the correlated and inverse correlated values on the same graph. I would like to start with V2, V5 (highly negatively correlated) with Amount, and also V7, V20 with Amount(highly positively correlated) with Amount.

# Feature engineering to a better visualization of the values

****

From above box plot we can easily infer that there are no fraud transactions occur above the transaction amount of 3000. All of the fraud transactions have transaction amount less than 3000. However, there are many transactions which have a transaction amount greater than 3000 and all of them are valid.



From the above plot it is clearly visible that there are frauds only on the transactions which have transaction amount approximately less than 2500. Transactions which have transaction amount approximately above 2500 have no fraud.

**Data Pre-processing**

The first step taken in pre-processing the dataset was **Normalization.** The Normalization procedure was applied only on the **Amount** Feature since it wasn’t on the same scale as the other features.

Next, the **SMOTE** algorithm was used to balance out the class ratio. It works by constructing new points from the minority class until it evens out the deficit in the class ratio.

**Implementation**

The implementation stage involved creating a training and predicting pipeline. This stage involved testing four algorithms to see which best suits the problem. The following algorithms were used;

**I. Random Forest**

**II. Isolation Forest**

**III. Local Outlier Factor**

**IV. XGBoost**

All these binary classification algorithms were initialized. Afterwards the training dataset was broken into four different portions, 1%, 10%, 50% and 100%. The purpose of breaking the dataset into different portions was to track the performance of each algorithm as the training set size increased.

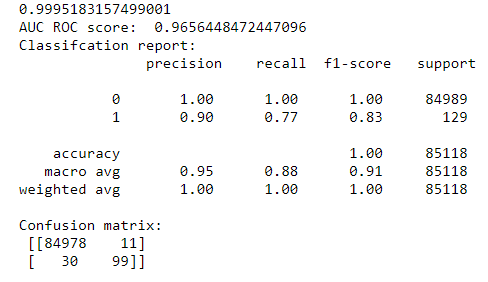
**Random Forest Classifier**

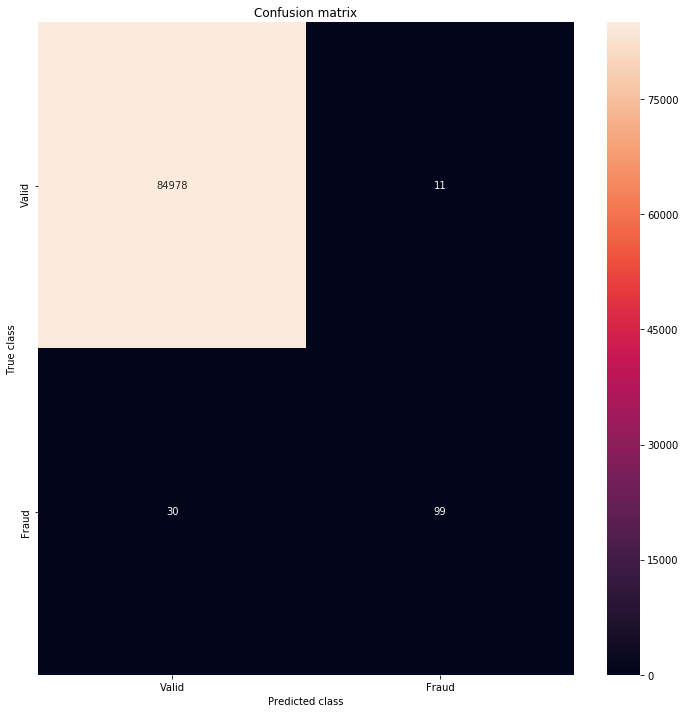
A random forest classifier fits a number of decision tree classifiers on various sub-samples of the dataset

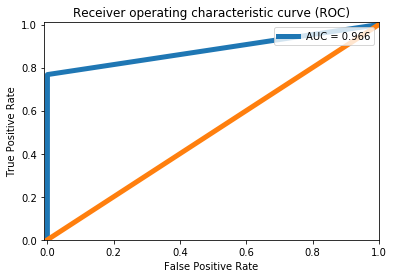
and use averaging to improve the predictive accuracy and control over-fitting. This classifier is very

popular to handle unbalanced classes. Following results are obtained by applying the algorithm on test

data set. All the features (maximum features = 30) were used for this method.





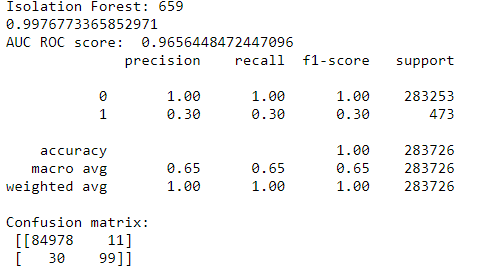
****

**Machine Learning techniques in Fraud Detection:**

One of the common techniques to detect fraud in credit card payment is Anomaly Detection that used to identify unusual patterns that do not conform to expected behavior, called outliers.

**Isolation Forest**

This algorithm is based on the fact that anomalies are data points that are few and different. As a result of these properties, anomalies are susceptible to a mechanism called isolation. Following results are obtained by applying the algorithm on test data set.



Observations:

* The IF Model is detecting fraud transactions 0.30 and LOF Model is detecting fraud transactions 0.04;
* Isolation Forest detected 659 errors versus Local Outlier Factor detecting 907 errors.
* Isolation Forest has a 99.76% more accurate than LOF of 99.66%.
* When comparing error precision & recall for 2 models, the Isolation Forest performed much better than the LOF as we can see  
  that the detection of fraud cases is 28 % versus LOF detection rate of just 0 %.
* So overall Isolation Forest Method performed much better in determining the fraud cases which is around 30%.
* We can also improve on this accuracy by increasing the sample size or use deep learning algorithms however at the cost of computational expense. We can also use complex anomaly detection models to get better accuracy in determining more fraudulent  
  cases.

**Local outlier factor**

Is an unsupervised outlier detection which compute the local density deviation of a given data with respect to its neighbors.

LOF scores can be used to detect outliers. Following points give a brief of how to do it:

* Find the k-nearest-neighbors
* For each instance, compute the local density
* For each instance compute the ratio of local densities
* Normal examples have scores close to 1.0
* Anomalies have scores > (1.2 ... 2.0)

With Isolation Forest, Local Outlier Factor Models, we have:

* 84976 transactions classified as valid and were actually valid (True Positive);
* 13 transactions classified as fraud but that were really valid (type 1 error);
* 29 transactions classified as valid but which were fraud (type 2 error);
* 100 transactions classified as fraud and were actually fraud.
* Look at the precision, recall, f1\_score. The accuracy looks good.

# XGBoost Classifier

# XGBoost has an inherent ability to handle missing values.

# Let’s see if we can improve their performance through hyperparameter optimization:

# I want to use GridSearchCV for hyperparameter tuning. In this approach machine learning is an evaluated for a range of hyperparameter values. but my result didn’t improve.

# 

# 

# EXPERIMENTAL RESULTS AND ANALYSIS

The machine learning models which were mentioned above were trained and subjected to evaluation metrics such as Accuracy Score, Area Under Curve, Confusion Matrix, Precision and Recall. In this project, the accuracy of all the methods implemented are compared using plots. The ROC score, Precision and Recall graphs, Confusion matrix of all the methods are plotted.

**Parameter Tuning:**

Parameter Tuning: Based on the preprocessing and model training, the accuracy of around 99% is obtained using the 4 models. To increase the accuracy of the models GridSearchCV sklearn tools were used to tune the parameters of all the models. Then the accuracy of all the algorithms increased by 1%.

I used Voting classifier to create multiple machine learning models to combine them to produce

a final result. More accurate prediction than a single model and can improve model performance.

**Lessons Learnt**

Data preprocessing is a very important step in machine learning workflow. Initially only 60% accuracy could be obtained due to lack of proper data preprocessing. After performing proper preprocessing of data, it was possible to achieve an accuracy of more than 90%.

**Dataset**

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

Dataset is transformed Principal Component Analysis (PCA) which is commonly used:

* Dimensionality reduction algorithm
* Speed-up Machine Learning algorithms

**Observation**

* some features (V1, V2, V3, ..., V28) transformed to PCA and Time, Amount features not transformed.
* target is Class (1-Fraud, 0-Valid)

**Metrics**

In a binary classification problem such as this, a model classifies examples as either positive (fraudulent) or negative (genuine). The decision made by the model, either positive or negative can be represented in a structure known as confusion matrix. This confusion matrix has four elements that define it, contextually they are:

* True Positive (TP) – An example where a transaction is fraudulent and is

classified correctly as fraudulent.

* False Positive (FP) – An example where a transaction is valid and is classified as incorrectly as fraudulent.
* True Negative (TN) – An example where a transaction is fraudulent but is classified incorrectly as valid.
* False Negative (FN) – An example that is valid and is classified correctly as fraudulent.

**Anomaly Detection algorithms I used:**

* Random Forest
* XGBoost
* Based on this initial EDA, this dataset does not have any null values and highly imbalance. Anomaly Detection is best for unbalanced dataset and supervised learning is better if dataset balanced.

**Benchmark**

For this project, considering the imbalance class ratio, accuracy would not be used to judge the model since it can be misleading. Instead, as a benchmark the model should have an Area Under the Precision-Recall Curve (AUPRC) score of % 97 or greater.

This score was gotten from a XGBoost classifier I built; this serves as benchmark towards building a better model.

**Improvement**

As stated earlier in the justification section, in the global context of credit card fraud detection this model might not be a silver bullet but it is definitely a great starting point to a larger project that seeks to tackle credit card fraud detection globally.

Improvement on solving the problem of credit card fraud detection would be industry/environment specific but some general rules apply, like a proper and detailed exploratory data analysis of variables that might explain the outcome of a transaction.

Also, more data could be collected, which would put the complexity and strength of advanced deep learning techniques to good use in building a more robust and accurate algorithm.

**Conclusion**

After the XGBoost algorithm was chosen, hyper parameter tuning was performed to optimize the model. Grid Search was used to find the optimal parameters, this implies that the Area Under Precision Recall Curve of models was 0.97 which is a very good score. A recall score of 0.78 implies that the final model predicted 78% of the fraudulent transactions in the test dataset correctly while the precision measures that fraction of cases predicted to be fraudulent that are truly fraudulent.

**Project Design**

The summarized workflow of this project is as follows:

* Download the dataset from Kaggle website (kaggle.com)
* Perform exploratory data analysis on the dataset to gain insights from the data structure (look for outliers, list the factors in order of relevance, etc.).
* Balance the classes using one or more strategies (under sampling, oversampling or SMOTE).
* Create and train different simple models commonly used on supervised training tasks (like random forest, XGBoost) and unsupervised learning (IF, LOF)
* Use the best result of the previous models as a benchmark utilizing the area under the ROC curve as a metric, precision and Recall.
* Use combinations of the simple models in an ensemble model to get a better result compared to the benchmark.