Data Analytics

MSc Artificial Intelligence 6120 Assignment-2

Title:

Credit Card Fraud Detection Classification Model.

Submitted by:

Abdul Nayim IMRIT (2022783)

Kiran RAMAH (2022523)

Date: 13 March 2022

Table of Contents

Abstract 3
Introduction 4
Literature Review 5
Proposed Methodology 6
    Details on Dataset 6
    Data Analysis 7
    Techniques used to analyze data 14
    Cat Boost 14
    LightGBM 15
    ANN 16
Discussions 18
Conclusion 20
References 21

Abstract

Successful businesses today make extensive use of data analysis. As such, a buying pattern or even a specific behavior of an individual can be exploited for revenue. Data can be structured, unstructured and semi-structured. Making use of data outside business control is making use of Bigdata. Major characteristics of big data are the 5Vs namely, Volume, Velocity, Veracity, Variety and Value. Bigdata benefits include improved customer service, increased operational efficiency, and better decision making, anticipation or predictions among others. The aim of this paper is to find a new way for the financial sector to flag a transaction as fraudulent or not. A credit card company will want to detect potential fraud cases so that customers are not required to pay for products they did not order. Gradient Boosting algorithms, Artificial Neural Network (ANN) and Random Forest were used to build and train different models and the more favorable one was further investigated.

Keywords: Gradient Boosting Algorithm, Big data, ANN, Random Forest.

Introduction

With technologies like ‘touch and pay’, ‘juice’, ‘aliexpress’ or even ‘ebay’, credit cards are becoming vulnerable to theft and can be used by anyone. Nowadays, especially because of the pandemic, we observe an amplification of the usage of online payment system. Credit card are often used for these transactions. To encourage people to use cards for payment or purchases, banks and other associated institution must be able to cater for anomalies or even detect suspicious activities for all of its holder. This will help institutions retain and also gain more customer base. Many frauds detection approaches, such as predictive analytics and data mining, aid in the prevention of fraud in the financial industry, particularly modeling algorithms that integrate clustering techniques and anomaly detection [1][2].

We cannot overlook the significant number of cases of fraud in the financial industry, thus, makes it a must to adopt fraud detection and prevention. There are many ways thief get details of a credit card. We have hoax phone calls, stolen cards, eavesdropping and maybe hackers were able to hack in some bank account details. The main challengers here are data collection (big data), data confidentiality and different dimensionality of data.

In this paper, we will make use of mostly boosting algorithms models such as XGboost, Catboost, LigthGBM to try to determine which techniques can better detect cases of legit and fraudulent activities. Other technique will also be explored such as ANN which will be explained later in the paper.

Literature Review

In this section, we review several papers from researchers regarding credit card fraud detection using artificial intelligence. The goal is to identify the research gap to review and test new methodologies.

In the study of (Dornadula & Geetha, 2019), there are multiple supervised and semi-supervised machine learning techniques for fraud detection. They mention algorithms such as Decision Trees, Naïve Bayes, Least Square Regression, Logistic Regression, SVM, Random Forest, K-Nearest Neighbor, Adaboost and Majority voting methods. Their methodology was to use a clustering method as high, medium and low groups of cardholders in order to extract the behavioral patterns. Synthetic Minority Over-Sampling technique (SMOTE) was used to balance the dataset. The methods which were used and tested are Local Outlier factor, Isolation Forest, Logistic Regression, Decision Tree and Random Forest whereby the last three gave better results.

(Bagga et al., 2020) proposed a pipeline and ensemble learning model in comparison with Logistic Regression, Random Forest, K-Nearest Neighbors and Naïve Bayes. Moreover, they mention the use of ANN, SVM and decision tree as well. They tackled this problem as a binary classification problem based on the spending behavior of the cardholder and five types of variables namely "all transactions, regional statistics, categories of merchants, time-based statistics and card profile. Zero mean and unit variance were applied on the dataset for preprocessing and ADASYN for data sampling to make data balanced. To integrate their pipeline, they used selectKBest (Sklearn) along with Random Forest Classifier for the classification and prediction. To add the ensemble learning method, they made use of Bagging Classification because of randomization in the development process. To conclude, their model appears to be working better than all the models and K-nearest neighbors being the worst. Yet, their accuracy appears to be unrealistic with almost perfect scores of 99.99%.

In the studies mentioned above, the main challenge regarding credit card fraud detection lies in the dataset which contains confidential data hence are transformed and the fact that they are usually highly imbalanced. Both source of dataset contains features from V1 to V28 which are the results of the principal component analysis. (Chen et al., 2020) retrieved their dataset via Kaggle and they observe that hybrid model is more accurate than single process models.

Proposed Methodology

After reviewing several research papers, the most popular methods we came across were logistic regression, K-nearest neighbors, random forest (whether alone or with ensemble learning of bagging classification), Adaboost, SVM, decision tree and ANN. As for the data sampling, we observe that SMOTE and ADASYN were used.

Since hybrid models appears to be performing better than single models, boosting methods could potentially create more powerful models. As such, models using Gradient Boosting Decision Trees could potentially be used: XGboost, Catboost, LigthGBM. Moreover, deep learning was briefly mentioned in (Chen et al., 2020) but no tested. Therefore, we may also test a sequential model using Keras and MLP Classification using SKlearn.

The goal is to determine if the proposed models would be more efficient. Therefore, as for the data sampling: we shall maintain the use of SMOTE for data sampling as it proven to be effective.

Details on Dataset

Our dataset contains credit card transactions with labels to classify if they are fraudulent or not. However, this dataset is highly imbalanced

The "Time" column shows how long has passed since the first transaction.

The "Class" column indicates whether the transaction is authentic (0) or fraudulent (1), while the "Amount" column displays the transaction amount.

The transaction details of the customer are highly confidential hence the features of the dataset (V1 to V28) were transformed using principal component analysis (PCA).

There are no missing values in the dataset's 284,807 rows.

Attribute Name	Types of Attributes	Description	Values Present
Time	Numerical	Interval transactions were made	Unique values for each row
Class	Boolean	Whether fraudulent/ not	0- Not fraud 1- fraud
Amount	Numerical	Currency of transaction amount done	0-10,000
V1 – V28	Numerical	Confidential Data	-27.67 - 2.454

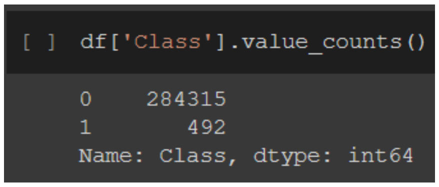
Data Analysis

The data was cleaned before use so that it could be used with different algorithms and optimizer. However, we also seen some places where we could not fit the data for analysis like in SVM.

Below are our steps for pre-processing of data:

1) Determining the number of classes:

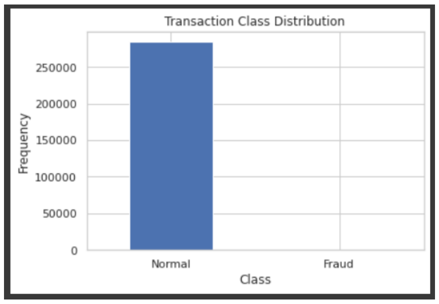
   We have approximately 284,807 rows in our excel. Classes generated were 0/1. Rows with class 0 is legit and the 1s are fraudulent.



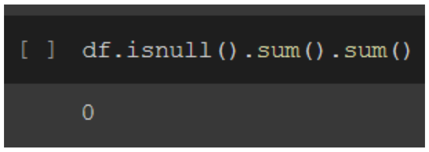
```
[ ]  df['Class'].value_counts()

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

   Our data is imbalanced. So, in the next step, we cater for imbalances via various techniques.



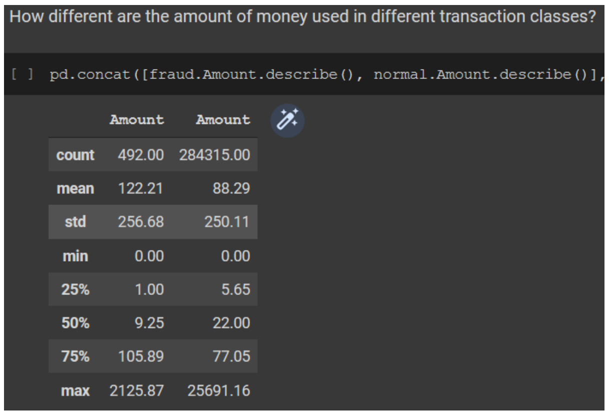
   Missing values:



```
[ ]  df.isnull().sum().sum()

     0
```

   Money used in fraudulent and normal transactions:

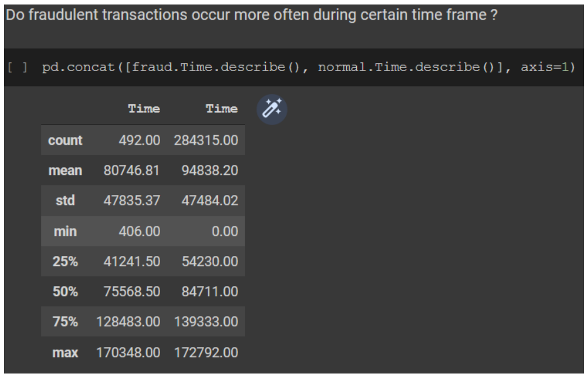


How different are the amount of money used in different transaction classes?

```
[ ]  pd.concat([fraud.Amount.describe(), normal.Amount.describe()],
```

	Amount	Amount
count	492.00	284315.00
mean	122.21	88.29
std	256.68	250.11
min	0.00	0.00
25%	1.00	5.65
50%	9.25	22.00
75%	105.89	77.05
max	2125.87	25691.16

Time frame fraudulent activities occurs:

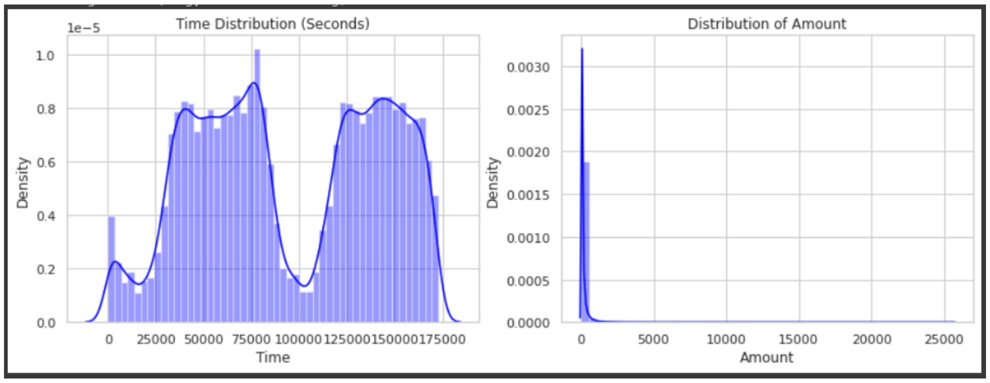


Do fraudulent transactions occur more often during certain time frame ?

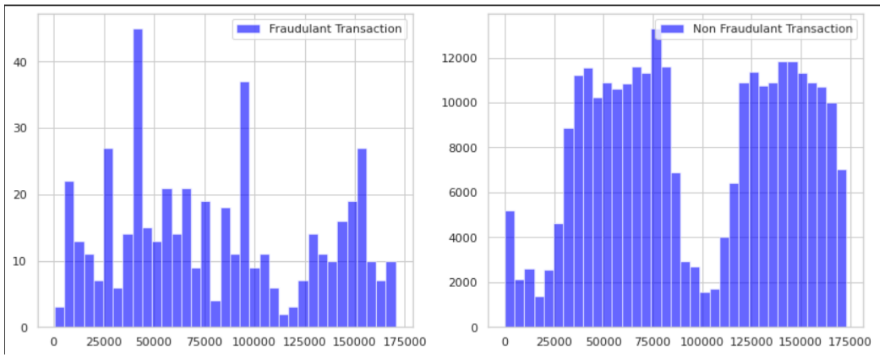
```
[ ] pd.concat([fraud.Time.describe(), normal.Time.describe()], axis=1)
```

	Time	Time
count	492.00	284315.00
mean	80746.81	94838.20
std	47835.37	47484.02
min	406.00	0.00
25%	41241.50	54230.00
50%	75568.50	84711.00
75%	128483.00	139333.00
max	170348.00	172792.00

Side-by-side comparisons of Amount and Time



Do time of transactions matters? We will see it in the skewness below:



According to the foregoing observation, the time of transaction does not appear to be a significant factor.

Key findings:

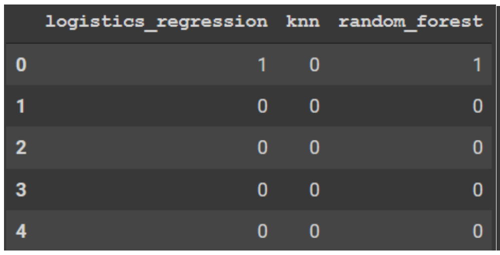
- Amount of transaction is small. We did not find any value above 10,000.
- When we verified for null values, we had none.
- Most transactions fall in the 0 class which is a non-fraud category.

2) Dealing with imbalanced data:

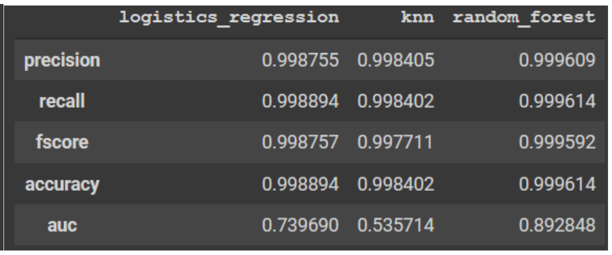
Our options were:

- Keep the excel sheet as is;

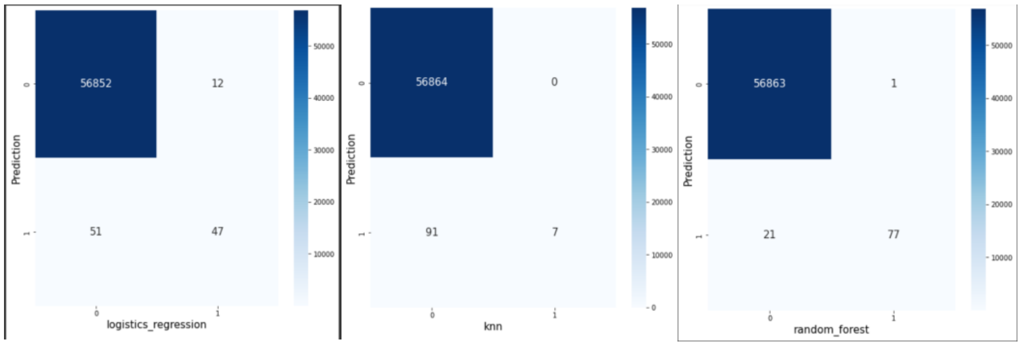
Result:



	logistics_regression	knn	random_forest
0	1	0	1
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0



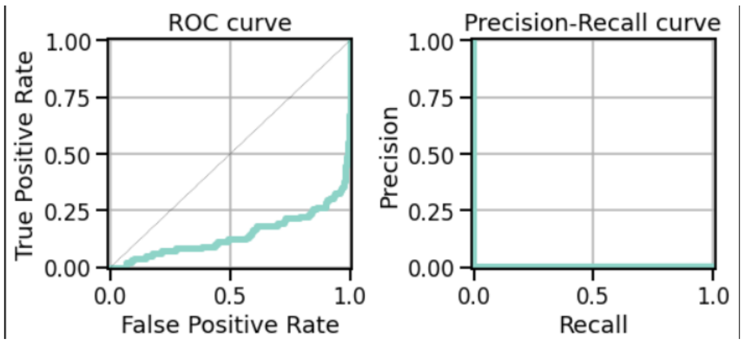
	logistics_regression	knn	random_forest
precision	0.998755	0.998405	0.999609
recall	0.998894	0.998402	0.999614
fscore	0.998757	0.997711	0.999592
accuracy	0.998894	0.998402	0.999614
auc	0.739690	0.535714	0.892848



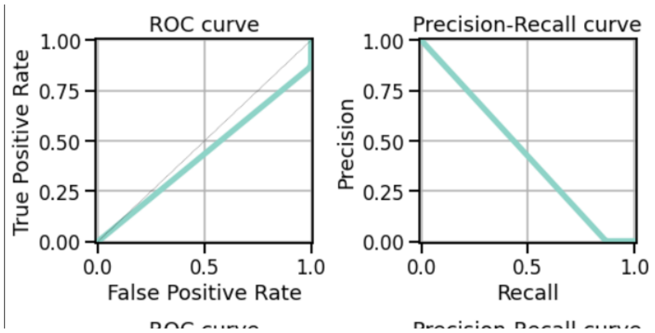
Out of 98 known frauds, Logistic regression accurately detected 47, KNN:7 and Random Forest:77, despite the high precision and recall values.

Below is the precision-recall curve for the above 3 matrix.

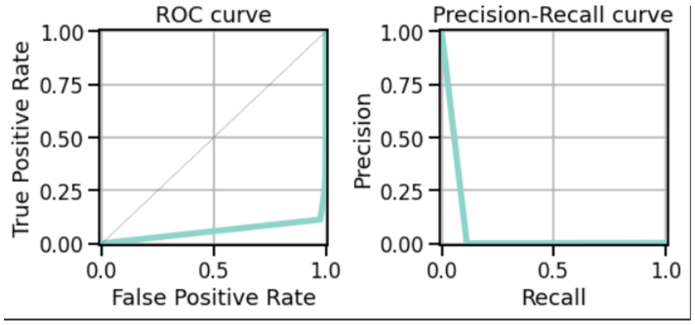
Logistic Regression



KNN



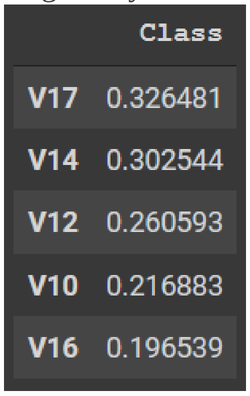
Random Forest



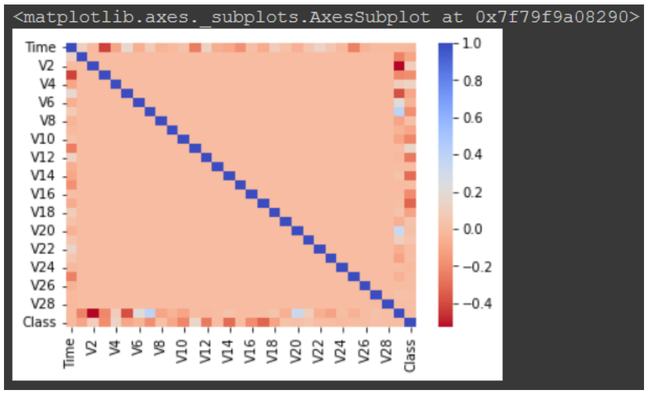
- Using oversampling to balance the data;
- Using undersampling to balance the data;
- Using Synthetic Minority Oversampling Technique.

We did all tests above to see where better results were obtained. The below observation was made:

- There are skew aspects in data that must be adjusted for improved prediction using non-tree models.
- Negatively corelated variables were:



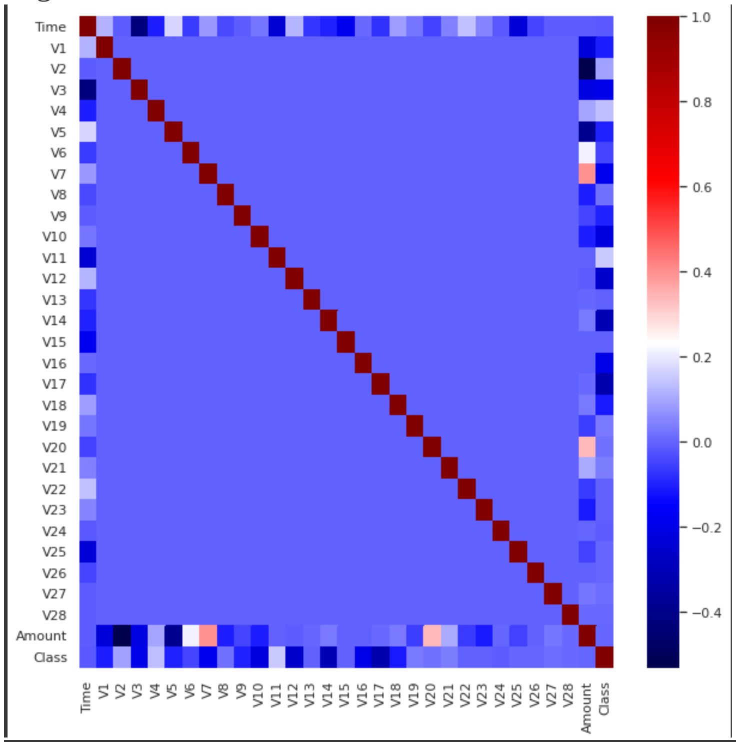
	Class
V17	0.326481
V14	0.302544
V12	0.260593
V10	0.216883
V16	0.196539

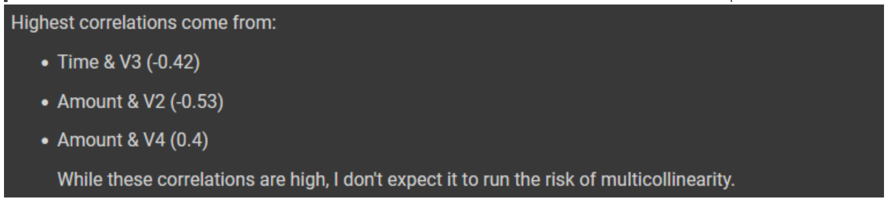


```
<matplotlib.axes._subplots.AxesSubplot at 0x7f79f9a08290>
```

- It's worth noting that the lower these numbers are, the more likely it is that the final result will be a fraudulent transaction.

- Highest correlation variables are:



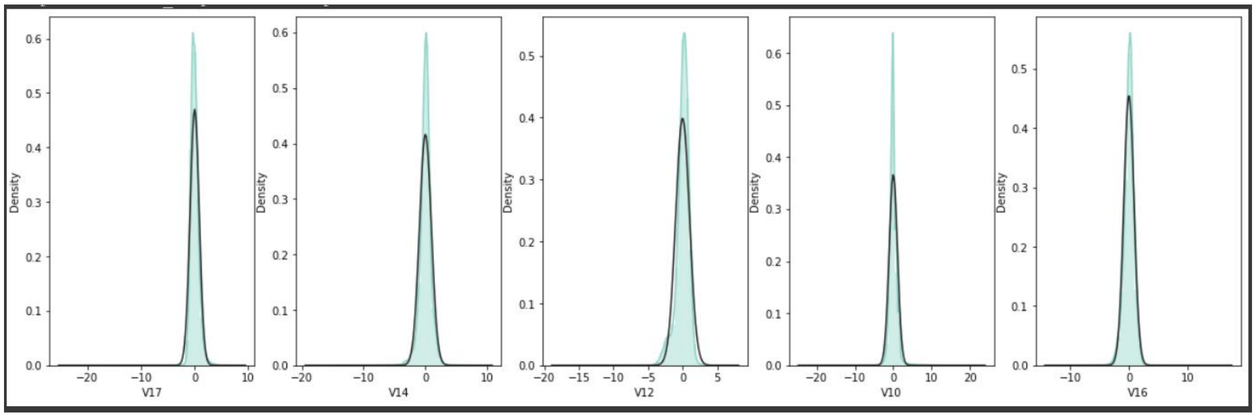


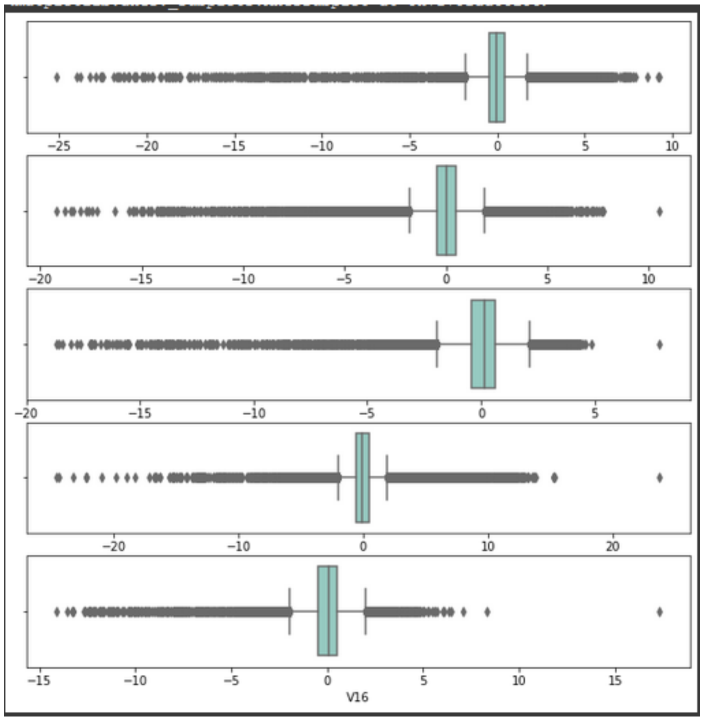
Highest correlations come from:

- Time & V3 (-0.42)
- Amount & V2 (-0.53)
- Amount & V4 (0.4)

While these correlations are high, I don't expect it to run the risk of multicollinearity.

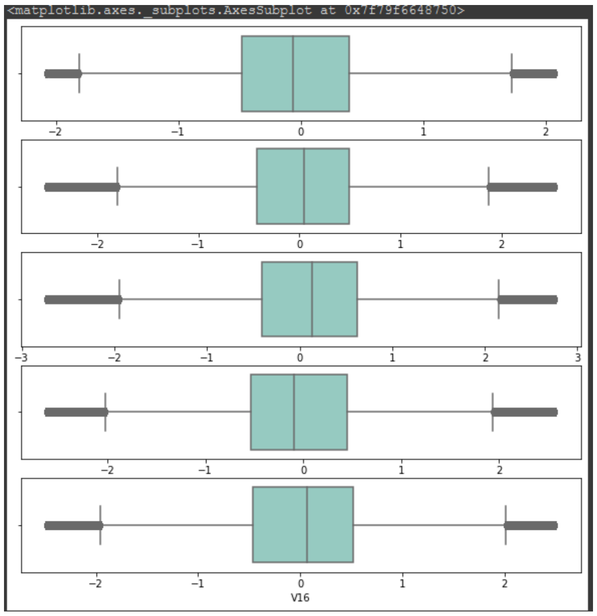
- Skewness & outlier analysis of negatively corelated variables were as follows:





3) Dealing with outliers:

   To improve our model's accuracy, extreme outliers in our boxplot above shall be removed. To cater for this, we used outlier capping based on boundary values. The results were as follows:



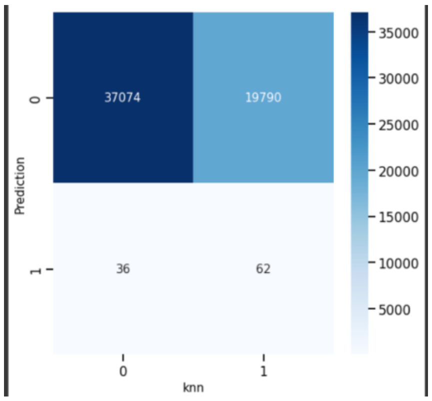
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f79f6648750>
```

Points to be noted:

- We conclude that V14 (second boxplot from the top) has a Gaussian distribution.
- Data imbalance is a problem if we are dealing with accuracy but not a problem if we want precision/ recall/ F1 scores.

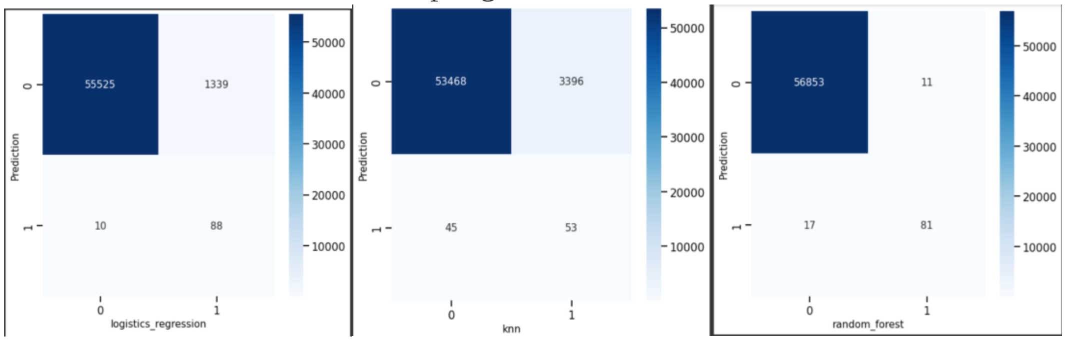
4) Best method in dealing with data imbalances:

From all the tests: leaving data unchanged, oversampling, undersampling and SMOTE, we were able to get better results in undersampling using the method: RandomUnderSampler with the algorithm of KNN (KNeighborsClassifier). Other algorithms used are: logistics_regression, random_forest.



Observations:

- SMOTE misidentifies a large number of non-frauds instance as a single fraud case. Since examples are synthesized, they can give a false impression of the model. Even though we tried using SMOTE with an undersampling approach, results were not better than that of the above. SMOTE alone gave better results; however, it might have worked better with oversampling. Results of smote alone:



Techniques used to analyze data

Cat Boost

It is a boosting algorithm, released in 2017 by Yandex in Russia, which works faster than XGBoost and performs well where we lack training data; in our case, fraudulent activities, where time is crucial (detecting frauds) and where users lack skills to tune parameters. All our data present in the excel file is all numerical so there was no need to pass cat_feature in our model.fit dry run. The process involved in CatBoost is as follows:

1) The set of input observations is permuted in a random order.
2) A large number of random permutations are created.
3) Converting a floating point or category label value to an integer.
4) The following formula is used to convert all category feature values to numeric values:

$$avg\_target = \frac{countInClass + prior}{totalCount + 1}$$

The **countInClass** parameter is the classes we currently have: 0/1.

The **prior** parameter is the numerator's preliminary value. The beginning settings decide this.

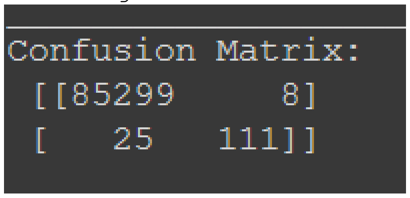
The entire number of objects with a categorical feature value that matches the current one is called **totalCount**.

The accuracy of our test result was: 99.96%. Training time was approximately 7 minutes in all.

CATBOOST	0	1	accuracy	Macro avg	Weight avg
precision	1	0.93	1	0.97	1
recall	1	0.82	1	0.91	1
F1 - score	1	0.87	1	0.94	1
support	85307	136	1	85443	85443

Our confusion matrix for CatBoost was as follows:

```
True positive: 85299
False positive: 8
False negative: 25
True negative: 111
```



```
Confusion Matrix:
 [[85299     8]
 [   25   111]]
```

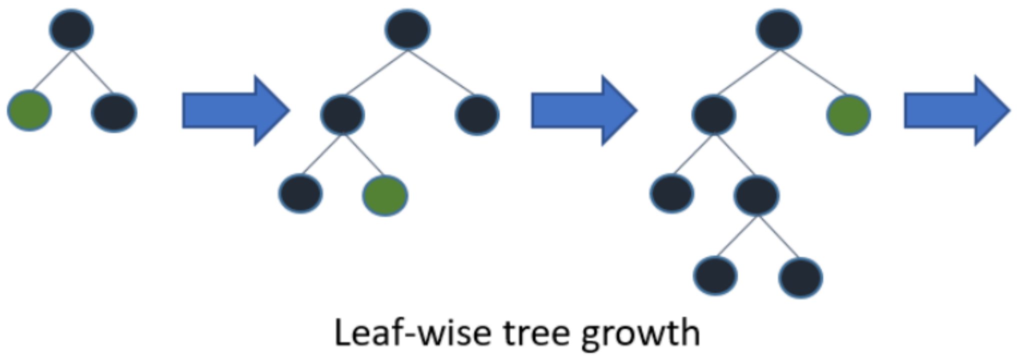
Analyzing cases of fraudulent activities, we are unable to get '1', so, there is still few data we are missing. In this case, we have a classifier with a high precision but a lower recall. Precision has increased since the classifier is now more certain that the transaction is fraudulent. Now that the classifier threshold has been set so high, fewer transaction are labeled as being a fraud so giving a lower recall.

The F1-score is the weighted average of precision and recall. Our F1-score is not perfect (1) but better than average; a value of 0.87% detected as valid fraud.

LightGBM

LightGBM, like CatBoost, can handle category features by taking feature names as input. It works faster than CatBoost but there is no guarantee that the results will be better. LGBM employs a unique approach to determine the split value of category features which is known as Gradient-based One-Side Sampling (GOSS). The process involved in lightGBM are:

1) Enumerate all features for each node.
2) Sorting is done by instances of feature values in step 1
3) To determine the appropriate split along that feature foundation information gain, use a linear scan.
4) Choose the optimal split solution based on all of the features.



Leaf with higher gradient/error is used for growing further in LGBM

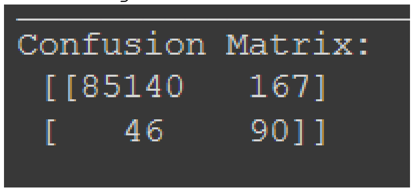
The underlying assumption here is that samples with small gradient training instances have lower training error and are already well-trained. To maintain the same data distribution, GOSS applies a constant multiplier while computing the information gain for data instances with minor gradients. As a result, GOSS strikes a reasonable compromise between lowering the amount of data instances and maintaining the accuracy of learnt decision trees.

The accuracy of our test result was: 99.75% and our training score was 99.86%. Training time was merely less than 5 minutes.

LIGHTGBM	0	1	accuracy	Macro avg	Weight avg
precision	1	0.35	1	0.67	1
recall	1	0.66	1	0.83	1
F1 - score	1	0.46	1	0.73	1
support	85307	136	1	85443	85443

Our confusion matrix for LightGBM was as follows:

```
True positive: 85140
False positive: 167
False negative: 46
True negative: 90
```



```
Confusion Matrix:
 [[85140   167]
 [   46    90]]
```

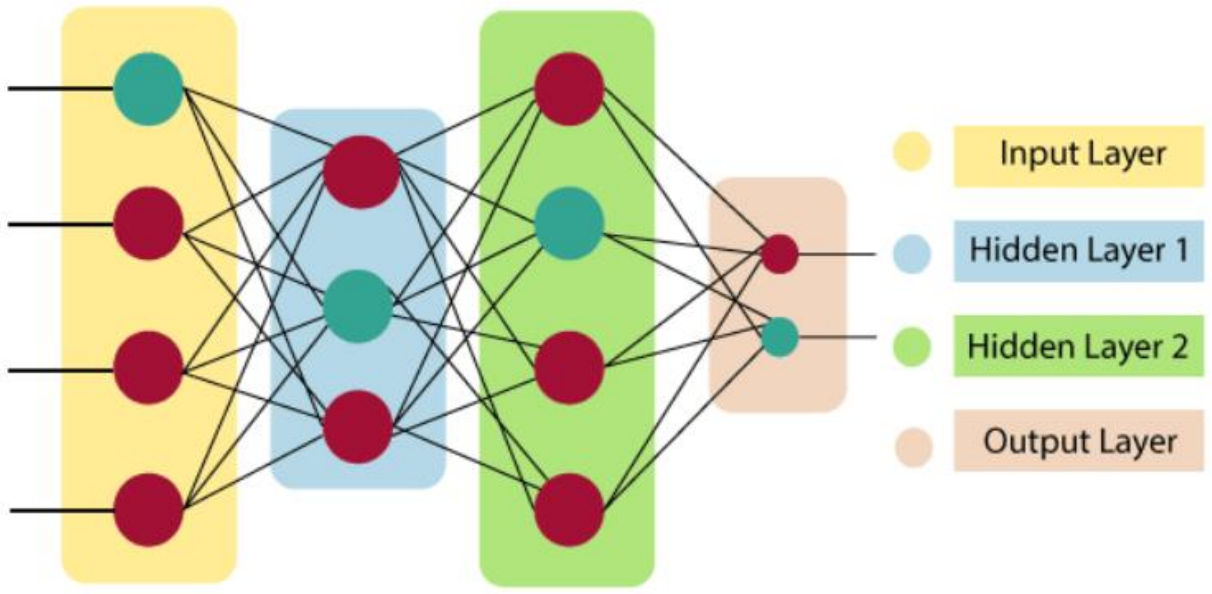
This technique is very fast but not as accurate as CatBoost model above.

The F1-score is the weighted average of precision and recall. Our F1-score is not perfect (1) but not better than average; a value of 0.46% detected as valid fraud. Using this method would be catastrophic for companies dealing with credit/online transactions.

ANN

Artificial neural networks are said to mimic the brain of human beings.

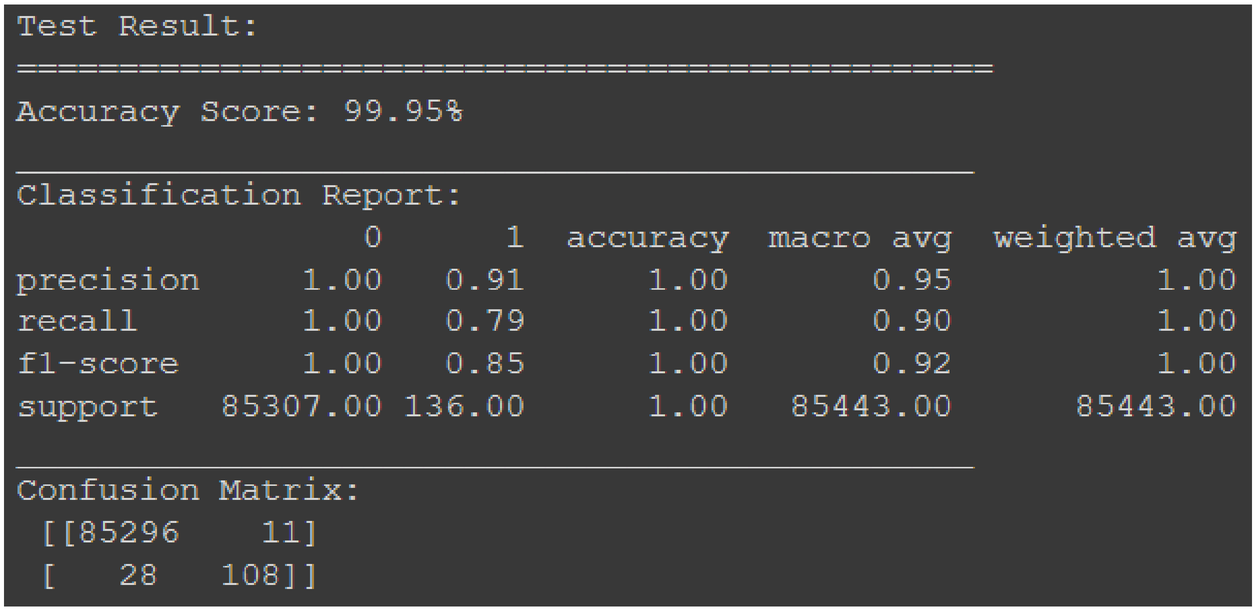
Artificial Neural Network primarily consists of three layers:



All the layers are responsible for a specific task. As input, it crawls all available resources fed into its system. The hidden layer does all of the math to uncover hidden characteristics and patterns. Finally, after lots of combinations or transformation of data it weight a total to produce an output.

Our result for ANN is as follows:

```
2671/2671 [==============================] - 13s 5ms/step - loss: 0.0038 - fn: 28.0000 - fp: 11.0000 - tn: 85296.0000 - tp: 108.0000 - precision: 0.9076 - recall: 0.7941
[0.0038311469834297895, 28.0, 11.0, 85296.0, 108.0, 0.9075630307197571, 0.7941176295280457]
```



```
Test Result:
================================================
Accuracy Score: 99.95%
_______________________________________________
Classification Report:
                 0      1  accuracy  macro avg  weighted avg
precision     1.00   0.91      1.00       0.95          1.00
recall        1.00   0.79      1.00       0.90          1.00
f1-score      1.00   0.85      1.00       0.92          1.00
support   85307.00 136.00      1.00   85443.00      85443.00
_______________________________________________
Confusion Matrix:
 [[85296    11]
 [   28   108]]
```

Discussions

Our model should be analyzed on precision, recall and F1-score rather than accuracy.

Precision: Precision refers to the percentage of your results which are relevant and is calculated as follows:

```
True Positives/(True Positives + Flase Positives)
```

Recall: Recall refers to the percentage of total relevant results correctly classified by your algorithm and is calculated as follows:

```
True Positives/(True Positives + False Negatives)
```

F1 Score: It is the harmonic mean of precision and recall and is calculated as follows:

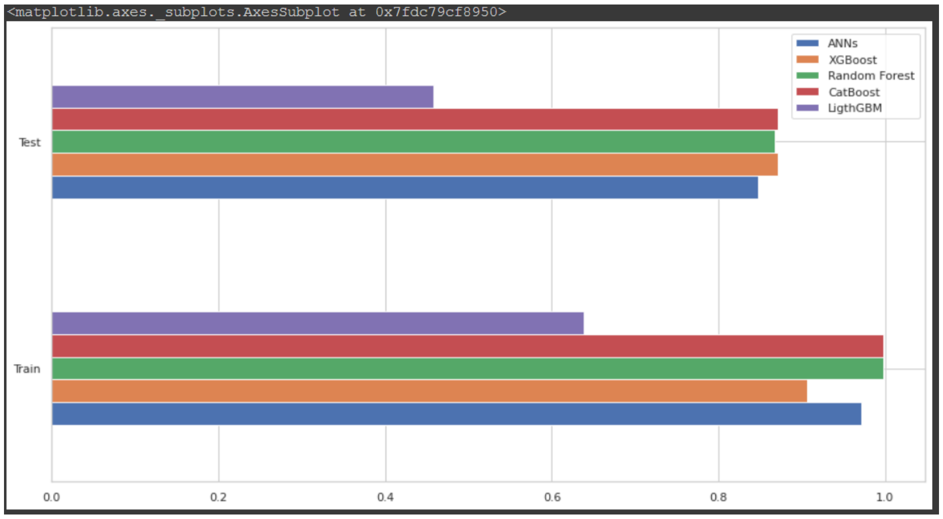
```
(2 x Precision x Recall)/(Precision + Recall)
```

The bigger the F1-score, the better the model is. Because we have accuracy of over 90% in almost all of our model tested, our confusion matrix was taken into consideration: True positives, False Negatives, False Positives and True negatives.

The Cat Boost model was faster and more accurate than other techniques. ANN showed very good results but took almost 3 hours to train and predict results.

Models	Performance Measure Metrics	Precision	Recall	F1-score	Confusion Matrix
Proposed model	Catboost	0.93	0.82	0.87	True positive: 85299 False positive: 8 False negative: 25 True negative: 111
Another model	ANN	0.91	0.79	0.85	True positive: 85296 False positive: 11 False negative: 28 True negative: 108
	XGBoost	0.93	0.82	0.87	True positive: 85299 False positive: 8 False negative: 25 True negative: 111
	Random Forest	0.93	0.82	0.87	True positive: 85298 False positive: 9 False negative: 25 True negative: 111
	LightGBM	0.35	0.66	0.46	True positive: 85140 False positive: 167 False negative: 46 True negative: 90

Below is a graph of our F1-scores combined.



```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdc79cf8950>
```

Conclusion

To conclude, CatBoostClassifier from catboost should be used. If we are to choose from the above techniques, catboost would be easily selected amongst others because of its already tweaked parameters and its speed. For companies in the financial sector, as real time data are being dealt with, faster detection is better.

References

[1] McCue, C. Advanced Topics. Data Mining and Predictive Analysis; Butterworth-Heinemann: Oxford, UK, 2015; pp. 349–365

[2] Mdpi-res.com. 2022. [online] Available at: <https://mdpi-res.com/d_attachment/electronics/electronics-11-00662/article_deploy/electronics-11-00662.pdf> [Accessed 13 March 2022].