A STUDY ON CREDIT CARD FRAUD DETECTION

A Report for the Evaluation 3 of Project

NIKHIL MISHRA

Admission No.:16SCSE101394

Under the Supervision of

Prof. U Samson Ebenezar

School of Computing Science and Engineering

Greater Noida, Uttar Pradesh

Winter 2019-2020

GALGOTIAS
UNIVERSITY

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report “ **A study on credit card fraud detection** ” is the bonafide work of “ **Nikhil Mishra (1613101440)** ” who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
	Mr U. Samson Ebenezar
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor & Dean	**Professor**
School of Computing Science & Engineering	**School of Computing Science & Engineering**

TABLE OF CONTENTS

ABSTRACT

Fraud is one of the major ethical issues in the credit card industry. The main aims are, firstly, to identify the different types of credit card fraud, and, secondly, to review alternative techniques that have been used in fraud detection. The sub-aim is to present, compare and analyze recently published findings in credit card fraud detection.

This project defines common terms in credit card fraud and highlights key statistics and figures in this field. Depending on the type of fraud faced by banks or credit card companies, various measures can be adopted and implemented.

The proposals made in this project are likely to have beneficial attributes in terms of cost savings and time efficiency. The significance of the application of the techniques reviewed here is in the minimization of credit card fraud. Yet there are still ethical issues when genuine credit card customers are misclassified as fraudulent.

INTRODUCTION

Financial fraud is an ever-growing menace with far consequences in the financial industry. Data mining had played an imperative role in the detection of credit card fraud in online transactions. Credit card fraud detection, which is a data mining problem, becomes challenging due to two major reasons – first, the profiles of normal and fraudulent behaviors change constantly and secondly, credit card fraud data sets are highly skewed. The performance of fraud detection in credit card transactions is greatly affected by the sampling approach on dataset, selection of variables and detection technique(s) used. This project implements the naïve bays classification algorithm on highly skewed credit card fraud data. Fraud means obtaining services/goods and/or money by unethical means, and is a growing problem all over the world nowadays. Fraud deals with cases involving criminal purposes that, mostly, are difficult to identify.

Furthermore, the face of fraud has changed dramatically during the last few decades as technologies have changed and developed. Dataset of credit card transactions is sourced from European cardholders containing 284,807 transactions. A hybrid technique of under-sampling and oversampling is carried out on the skewed data.

Credit cards are one of the most famous targets of fraud but not the only one; fraud can occur with any type of credit products, such as personal loans, home loans, and retail. Fraud is one of the major ethical issues in the credit card industry. The main aims are, firstly, to identify the different types of credit card fraud, and, secondly, to review alternative techniques that have been used in fraud detection. The sub-aim is to present, compare and analyze recently published findings in credit card fraud detection. The proposals made in this project are likely to have beneficial attributes in terms of cost savings and time efficiency.

The significance of the application of the techniques reviewed here is in the minimization of credit card fraud. Yet there are still ethical issues when genuine credit card customers are misclassified as fraudulent .A critical task to help businesses, and financial institutions including banks is to take steps to prevent fraud and to deal with it efficiently and effectively. Credit card fraud may happen in various ways, which depend on the type of fraud concerned; it encapsulates bankruptcy fraud, theft fraud / counterfeit fraud, application fraud and behavioral fraud. Each of these sub-fraud categories has its own definition and specificity.

The complexity of credit card fraud is that it may be committed in various ways, including theft fraud, application fraud, counterfeit fraud, bankruptcy fraud.

By not paying enough attention to fraud prevention or detection, the risk for the bank is that “credit card fraud remains usually undetected until long after the criminal has completed the crime”. Therefore, it will generate irrecoverable costs for the bank. Credit is a method of selling goods or services without the buyer having cash in hand. A credit card is only an automatic way of offering credit to a consumer.

Today, every credit card carries an identifying number that speeds shopping transactions. In the credit card business, it can be an internal party but most likely an external party. As an external party, fraud is committed being a prospective/existing customer or a prospective/existing supplier. Three different profiles can be identified for external fraudsters: the average offender, criminal offender, and organized crime offender. For many companies sometimes dealing with millions of external parties, it is cost-prohibitive to manually check the majority of the external parties’ identity and activities. Indeed, to investigate each suspicious transaction, they incur a direct overhead cost for each of them. In order to avoid these overheads and depending on the type of fraud committed, diverse solutions can be implemented.

Proposed Model

The model proposed in this project is to implement machine learning algorithms in to analyze and compare different fraud detection techniques. Our aim is to classify the highest possible degree of accuracy, precision and specificity of the detection techniques which will lead to increase in the probability of detecting the frauds in credit card . Our dataset gathered is highly unbalanced and huge . So firstly we need to do the dimensionality reduction and then follow the methods. After initial data exploration, we knew would implement a logistic regression model, a k-means clustering model, and a neural network. Some challenges we observed from the start were the huge imbalance in the dataset: frauds only account for 0.172% of fraud transactions. In this case, it is much worse to have false negatives than false positives in our predictions because false negatives mean that someone gets away with credit card fraud. False positives, on the other hand, merely cause a complication and possible hassle when a cardholder must verify it .

Principle Component Analysis

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

Exploratory Data Analysis

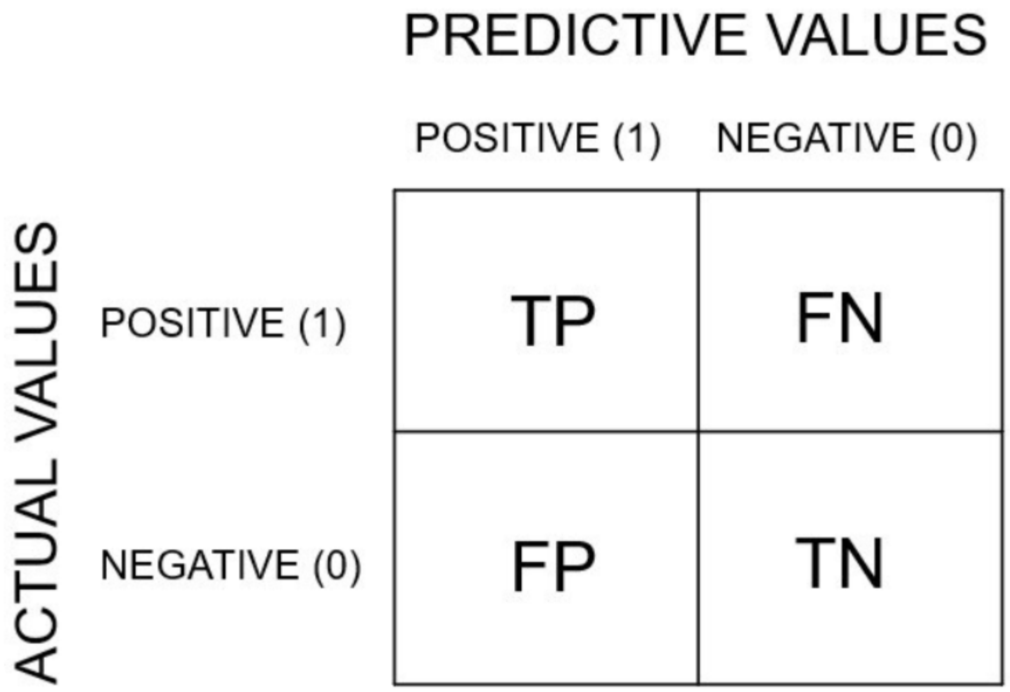
Exploratory Data Analysis (EDA) is anapproach/philosophy for data analysis that employs a variety of techniques (mostly graphical) tomaximize insight into a data set, uncover underlying structure, extract important variables, detect outliers and anomalies, test underlying assumptions, develop parsimonious models anddetermine optimal factor settings.The EDA approach is precisely that--an approach--not a set oftechniques, but an attitude/philosophy about how a data analysis should be carried out.

Correlation Matrix

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. 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.

Confusion Matrix

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made. A confusion matrix for binary classification shows the four different outcomes: true positive, false positive, true negative, and false negative. The actual values form the columns, and the predicted values (labels) form the rows. The intersection of the rows and columns show one of the four outcomes. For example, if we predict a data point is positive, but it actually is negative, this is a false positive.



Confusion Matrix for the Binary Classification

Accuracy

Accuracy is calculated as the total no. of corrected prediction divided by the total number of dataset. Accuracy works well on the balanced dataset. In case of imbalanced dataset accuracy mislead the performance.

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

Precision

Precision is defined as the ratio of the total number of correctly classified positive classes divided by the total number of predicted positive classes. Or, out of all the predictive positive classes, how much we predicted correctly. Precision should be high.

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

Recall

Recall is defined as the ratio of the total number of correctly classified positive classes divide by the total number of positive classes. Or, out of all the positive classes, how much we have predicted correctly. Recall should be high.

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

F-1 Score

It is difficult to compare two models with different Precision and Recall. So to make them comparable, we use F-Score. It is the Harmonic Mean of Precision and Recall. As compared to Arithmetic Mean, Harmonic Mean punishes the extreme values more. F-score should be high.

$$\textbf{\textit{F - score}} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$

Now it is time to start building the model .The types of algorithms we are going to use to try to do anomaly detection on this dataset are as follows

Isolation Forest Algorithm :

One of the newest techniques to detect anomalies is called Isolation Forests. The 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.

This method is highly useful and is fundamentally different from all existing methods. It introduces the use of isolation as a more effective and efficient means to detect anomalies than the commonly used basic distance and density measures. Moreover, this method is an algorithm with a low linear time complexity and a small memory requirement. It builds a good performing model with a small number of trees using small sub-samples of fixed size, regardless of the size of a data set.

Typical machine learning methods tend to work better when the patterns they try to learn are balanced, meaning the same amount of good and bad behaviors are present in the dataset.

How Isolation Forests Work The Isolation Forest algorithm isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. The logic argument goes: isolating anomaly observations is easier because only a few conditions are needed to separate those cases from the normal observations. On the other hand, isolating normal observations require more conditions. Therefore, an anomaly score can be calculated as the number of conditions required to separate a given observation.

The way that the algorithm constructs the separation is by first creating isolation trees, or random decision trees. Then, the score is calculated as the path length to isolate the observation.

Outlier detection formula of an anomaly score is required for decision prediction. For Isolation Forest it is defined as

S(x, n) =2 ^ -E(h(x))/c(n)

Where,

h(x) = is the path length of observation x,

c(n)= is the Avg path length of failed search in a BST (Binary Search Tree)

(n)= is the number of other nodes.

Each n observation is given an anomaly score and therefore the following call are often created on its basis:

- Score near to 1 precise the outlier
- Score less than 0.5 show legal transactions
- In condition of scores which they are near to 0.5 than the rest of sample does not seem clearly detect anomalies.

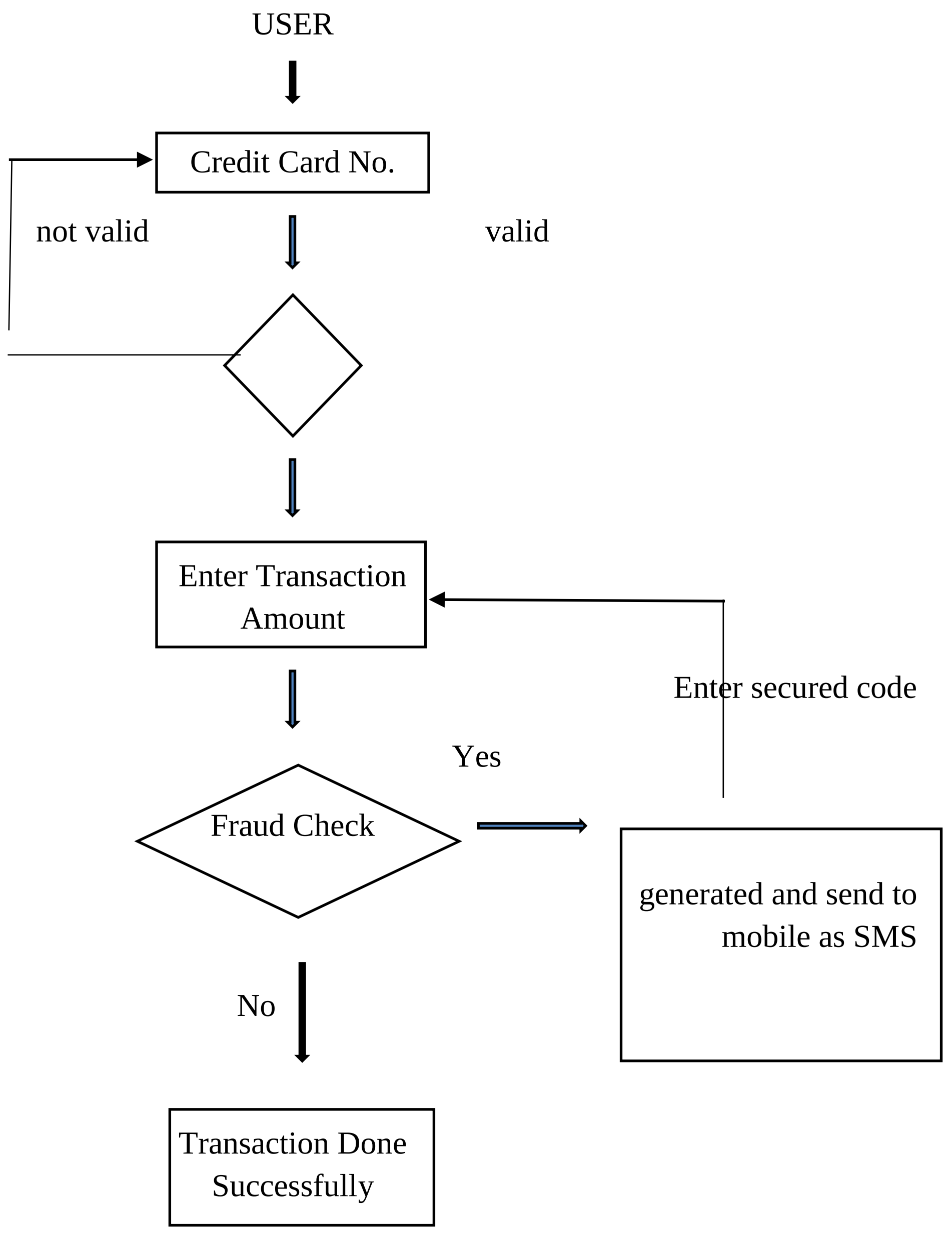
Local Outlier Factor(LOF) Algorithm

The LOF algorithm is an unsupervised outlier detection method which computes the local density deviation of a given data point with respect to its neighbors. It considers as outlier samples that have a substantially lower density than their neighbors.

The number of neighbors considered, (parameter n_neighbours) is typically chosen 1) greater than the minimum number of objects a cluster has to contain, so that other objects can be local outliers relative to this cluster, and 2) smaller than the maximum number of close by objects that can potentially be local outliers. In practice, such information are generally not available, and taking n_neighbors=20 appears to work well in general.

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble . Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction. The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is: A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don’t constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction.


USER
Credit Card No.
not valid
valid
Enter Transaction Amount
Enter secured code
Yes
Fraud Check
generated and send to mobile as SMS
No
Transaction Done Successfully

Existing Model

1.) Decision tree

A decision tree is flowchart like structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label. The topmost node in the tree is the root node. The construction of decision tree classifiers does not require any domain knowledge discovery. Decision trees can handle multi-dimensional data. Their representation of acquired knowledge in tree form is intuitive and generally easy to assimilate by humans. The learning and classification steps of decision tree induction are simple and fast. Decision tree induction algorithms have been used for classification in many application areas such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology. The idea of a similarity tree using decision tree logic has been developed. A similarity tree is defined recursively: nodes are labelled with attribute names, edges are labelled with values of attributes that satisfy some condition and 'leaves' that contain an intensity factor which is defined as the ratio of the number of transactions that satisfy these condition(s) over the total number of legitimate transaction in the behavior. The advantage of the method that is suggested is that it is easy to implement, to understand and to display. However, a disadvantage of this system is the requirements to check each transaction one by one. Nevertheless, similarity trees have given proven results also worked on decision trees and especially on an inductive decision tree in order to establish an intrusion detection system, for another type of fraud.

2.) Classification

It is the organization of data in given classes. Also known as supervised classification, classification uses given class labels to order the objects in the data collection. Classification approaches normally use a training set where all objects are already associated with known class labels. The classification algorithm learns from the training set and builds a model. The model is used to classify new objects. Common techniques for classification are decision tree, neural networks, SVM etc. Algorithms are often recommended as predictive methods as a means of detecting fraud. One algorithm that has been suggested by Bentley et al. (2000) is based on genetic programming in order to establish logic rules capable of classifying credit card transactions into suspicious and non-suspicious classes. Basically, this method follows the scoring process. They conclude from their investigation that neighborhood-based and probabilistic algorithms have been shown to be appropriate techniques for classification, and may be further enhanced using additional diagnostic algorithms for decision-making in borderlines cases, and for calculating confidence and relative risk measures.

3.) Clustering techniques

It is a division of data into groups of similar objects. Each group, called cluster, consists of objects that are similar amongst them and dissimilar compared to objects of other groups. Representing data by fewer clusters necessarily loses certain fine details, but achieve simplification. It represents many data objects by few clusters, and hence, it models data by its clusters . Some algorithms are Model Based algorithms , Density Based algorithms etc. Bolton & Hand (2002) suggest two clustering techniques for behavioral fraud. The peer group analysis is a system that allows identifying accounts that are behaving differently from others at one moment in time whereas they were behaving the same previously. Those accounts are then flagged as suspicious. Fraud analysts have then to investigate those cases. The hypothesis of the peer group analysis is that if accounts behave the same for a certain period of time and then one account is behaving significantly differently, this account has to be notified. Breakpoint analysis uses a different approach. The hypothesis is that if a change of card usage is notified on an individual basis, the account has to be investigated. In other words, based on the transactions of a single card, the break-point analysis can identify suspicious behavior. Signals of suspicious behavior are a sudden transaction for a high amount, and a high frequency of usage.

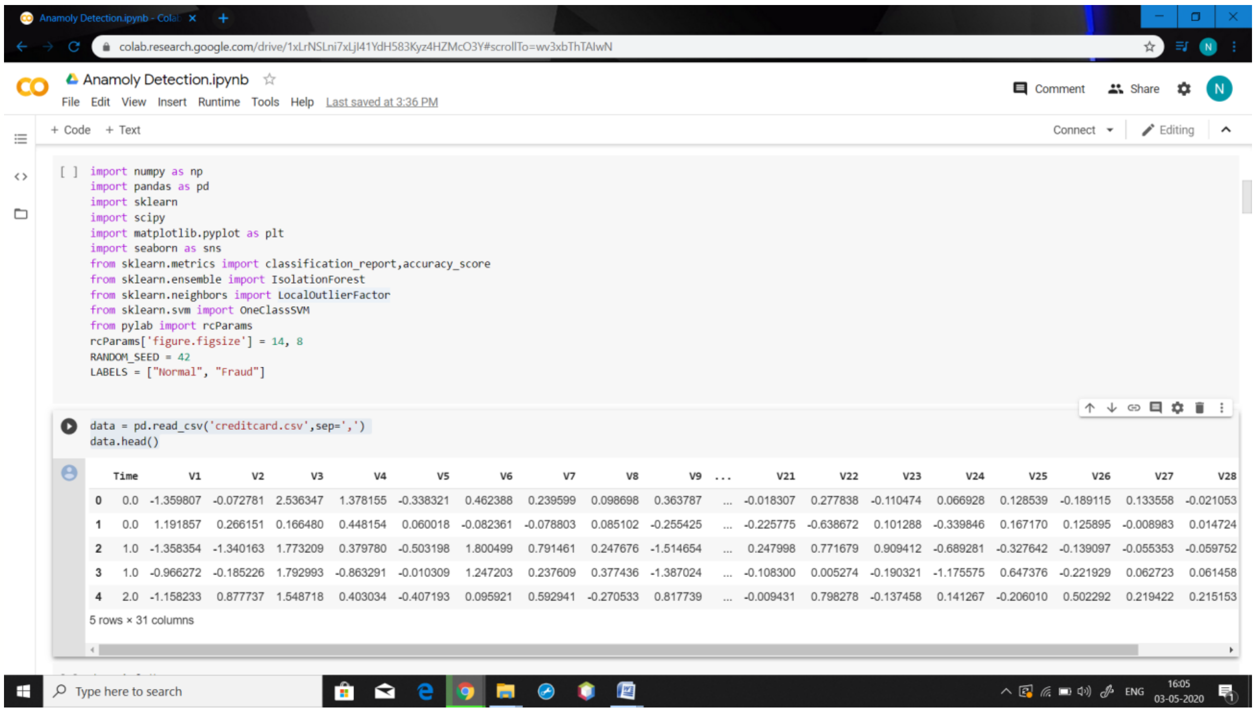
4.) Neural networks

Fraud detection methods based on neural network are the most popular ones. An artificial neural network consists of an interconnected group of artificial neurons .The principle of neural network is motivated by the functions of the brain especially pattern recognition and associative memory. The neural network recognizes similar patterns, predicts future values or events based upon the associative memory of the patterns it was learned. It is widely applied in classification and clustering. The advantages of neural networks over other techniques are that these models are able to learn from the past and thus, improve results as time passes. They can also extract rules and predict future activity based on the current situation. By employing neural networks, effectively, banks can detect fraudulent use of a card, faster and more efficiently. Among the reported credit card fraud studies most have focused on using neural networks. In more practical terms neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. There are two phases in neural network training and recognition. Learning in a neural network is called training. There are two types of NN training methods supervised and unsupervised. In supervised training, samples of both fraudulent and non fraudulent records are used to create models. In contrast, unsupervised training simply seeks those transactions, which are most dissimilar from the norm. On other hand, the unsupervised techniques do not need the previous knowledge of fraudulent and non fraudulent transactions in database. NNs can produce best result for only large transaction dataset. And they need a long training dataset.

Implementation

The datasets contains 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.

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

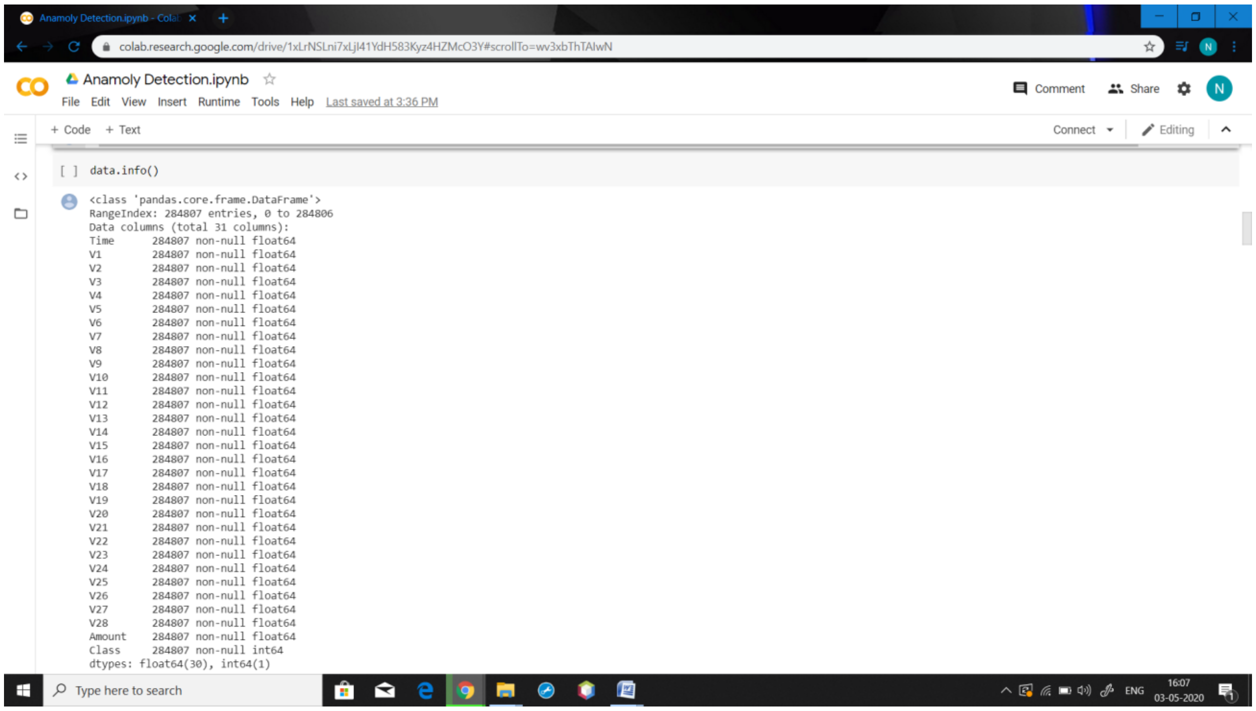

Anamoly Detection.ipynb - Colab
colab.research.google.com/drive/1xLrNSLni7xLjI41YdH583Kyz4HZMcO3Y#scrollTo=wv3xbThTAIwN
Anamoly Detection.ipynb
File Edit View Insert Runtime Tools Help Last saved at 3:36 PM
Comment Share
+ Code + Text
Connect Editing
Type here to search
ENG 16:05 03-05-2020

```
import numpy as np
import pandas as pd
import sklearn
import scipy
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report,accuracy_score
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn.svm import OneClassSVM
from pylab import rcParams
rcParams['figure.figsize'] = 14, 8
RANDOM_SEED = 42
LABELS = ["Normal", "Fraud"]
```

```
data = pd.read_csv('creditcard.csv',sep=',')
data.head()
```

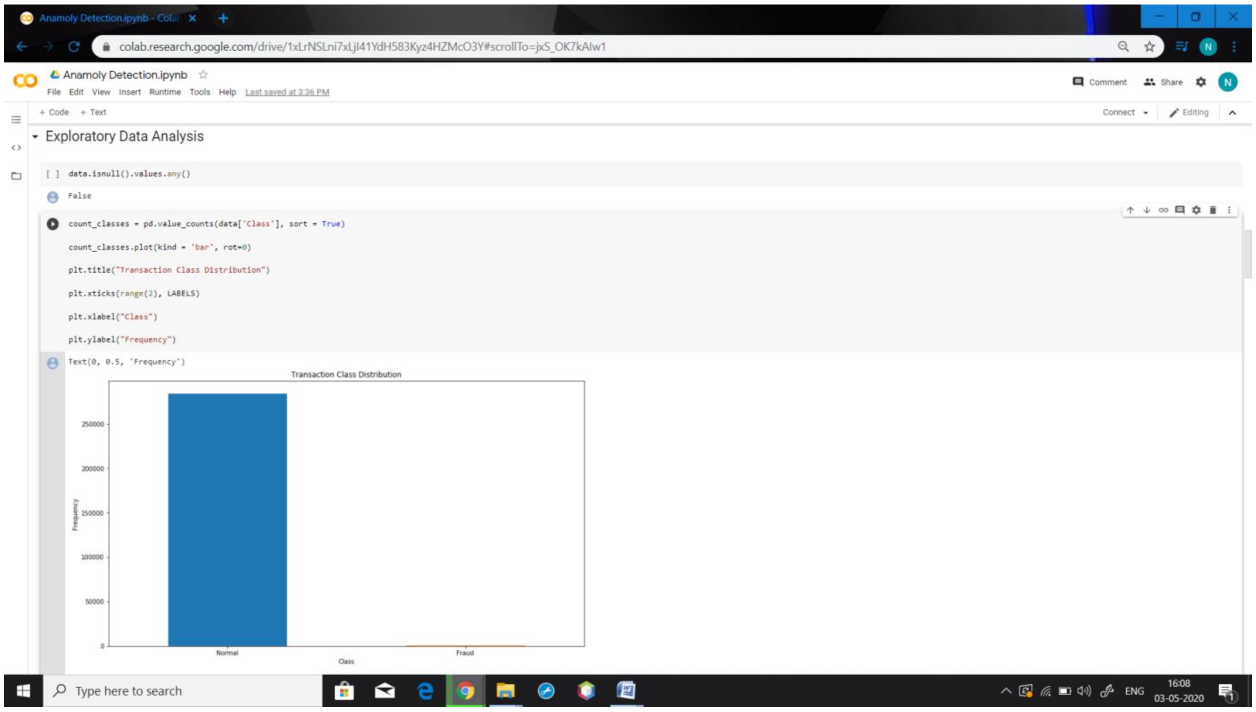
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153

5 rows × 31 columns


Anamoly Detection.ipynb - Colab
colab.research.google.com/drive/1xLrNSLni7xLjl41YdH583Kyz4HZMcO3Y#scrollTo=wv3xbThTAIwN
Anamoly Detection.ipynb
File Edit View Insert Runtime Tools Help Last saved at 3:36 PM
Comment Share
+ Code + Text
Connect Editing
Type here to search
ENG 16:07 03-05-2020

```
[ ] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
Time      284807 non-null float64
V1        284807 non-null float64
V2        284807 non-null float64
V3        284807 non-null float64
V4        284807 non-null float64
V5        284807 non-null float64
V6        284807 non-null float64
V7        284807 non-null float64
V8        284807 non-null float64
V9        284807 non-null float64
V10       284807 non-null float64
V11       284807 non-null float64
V12       284807 non-null float64
V13       284807 non-null float64
V14       284807 non-null float64
V15       284807 non-null float64
V16       284807 non-null float64
V17       284807 non-null float64
V18       284807 non-null float64
V19       284807 non-null float64
V20       284807 non-null float64
V21       284807 non-null float64
V22       284807 non-null float64
V23       284807 non-null float64
V24       284807 non-null float64
V25       284807 non-null float64
V26       284807 non-null float64
V27       284807 non-null float64
V28       284807 non-null float64
Amount    284807 non-null float64
Class     284807 non-null int64
dtypes: float64(30), int64(1)
```


Anamoly Detection.ipynb - Colab
colab.research.google.com/drive/1xLrNSLni7xLjl41YdH583Kyz4HZMcO3Y#scrollTo=jxS_OK7kAlw1
Anamoly Detection.ipynb
File Edit View Insert Runtime Tools Help Last saved at 3:36 PM
Comment Share
+ Code + Text
Connect Editing
Transaction Class Distribution
250000
200000
150000
100000
50000
0
Frequency
Normal
Fraud
Class
Type here to search
ENG 16:08 03-05-2020

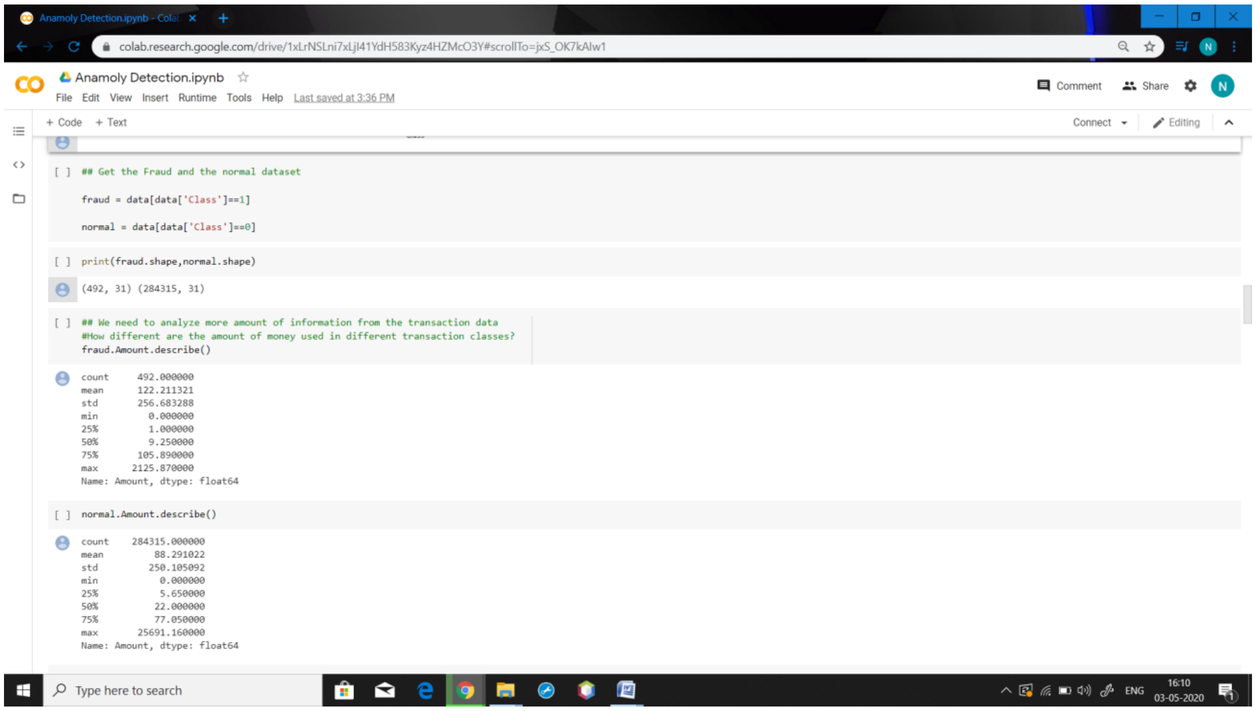
Exploratory Data Analysis

```
[ ] data.isnull().values.any()
```

```
False
```

```
count_classes = pd.value_counts(data['Class'], sort = True)

count_classes.plot(kind = 'bar', rot=0)

plt.title("Transaction Class Distribution")

plt.xticks(range(2), LABELS)

plt.xlabel("Class")

plt.ylabel("Frequency")
```

```
Text(0, 0.5, 'Frequency')
```



```
## Get the Fraud and the normal dataset

fraud = data[data['Class']==1]

normal = data[data['Class']==0]
```

```
print(fraud.shape,normal.shape)
```

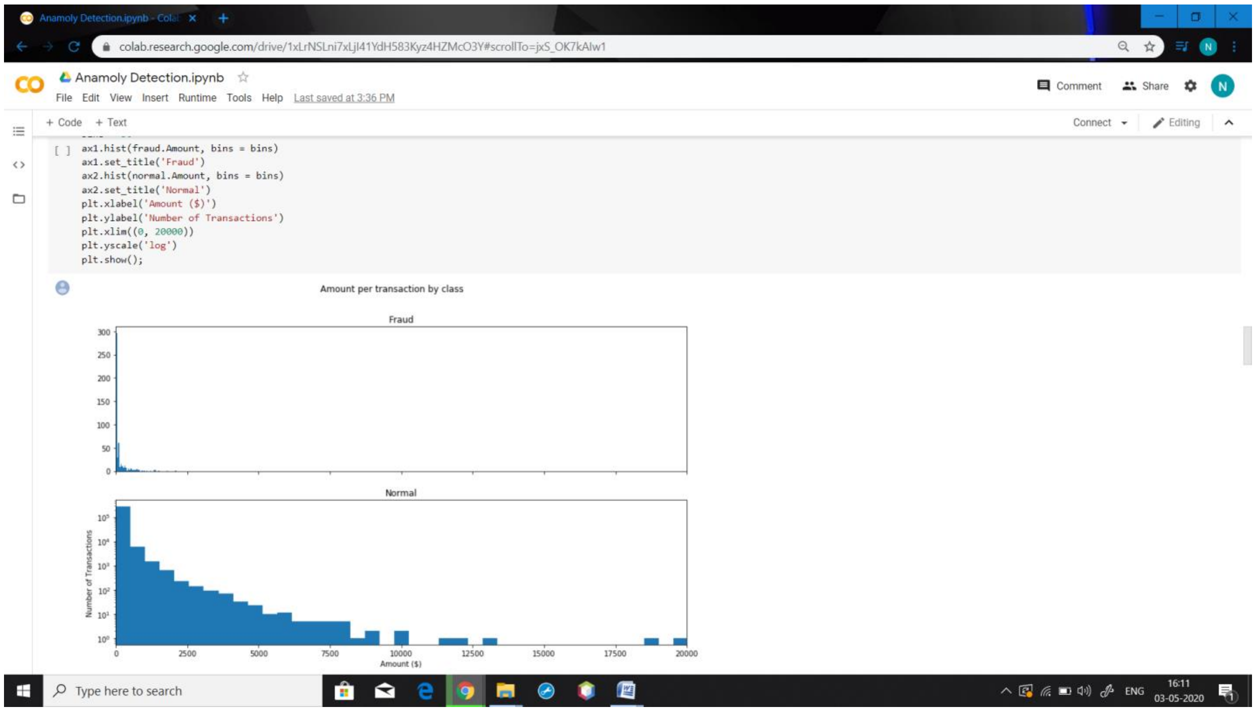
```
(492, 31) (284315, 31)
```

```
## We need to analyze more amount of information from the transaction data
#How different are the amount of money used in different transaction classes?
fraud.Amount.describe()
```

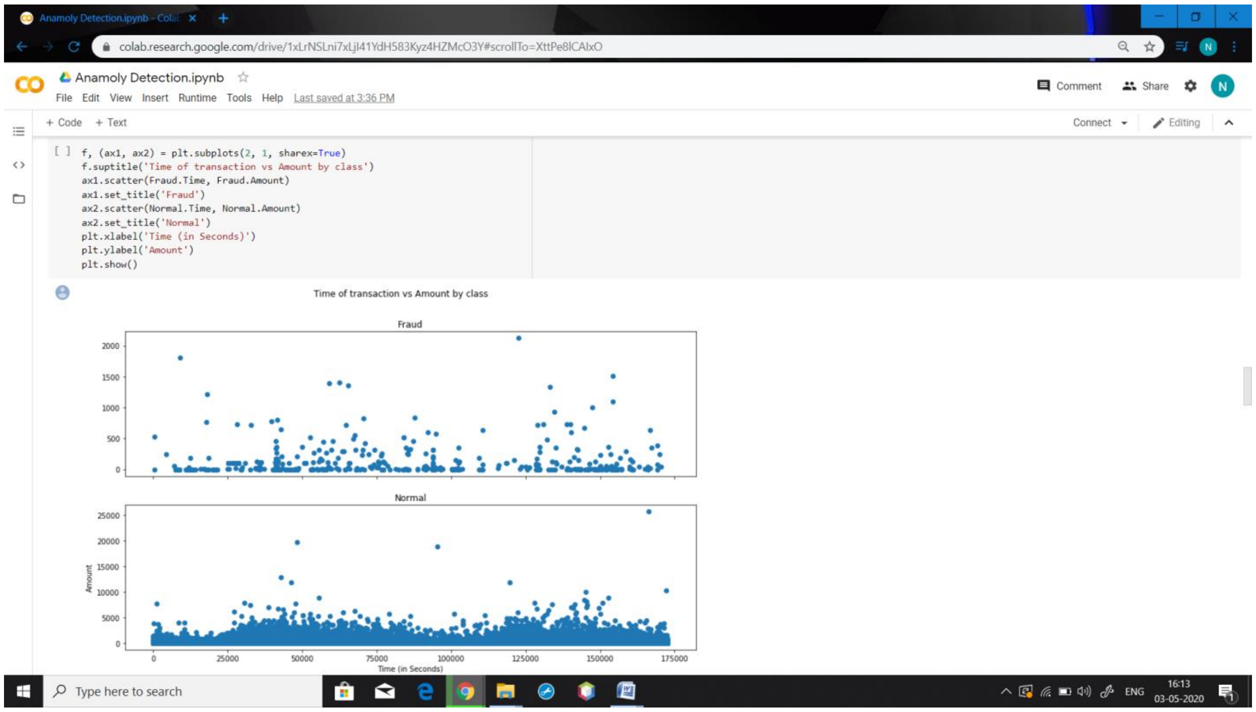
```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

```
normal.Amount.describe()
```

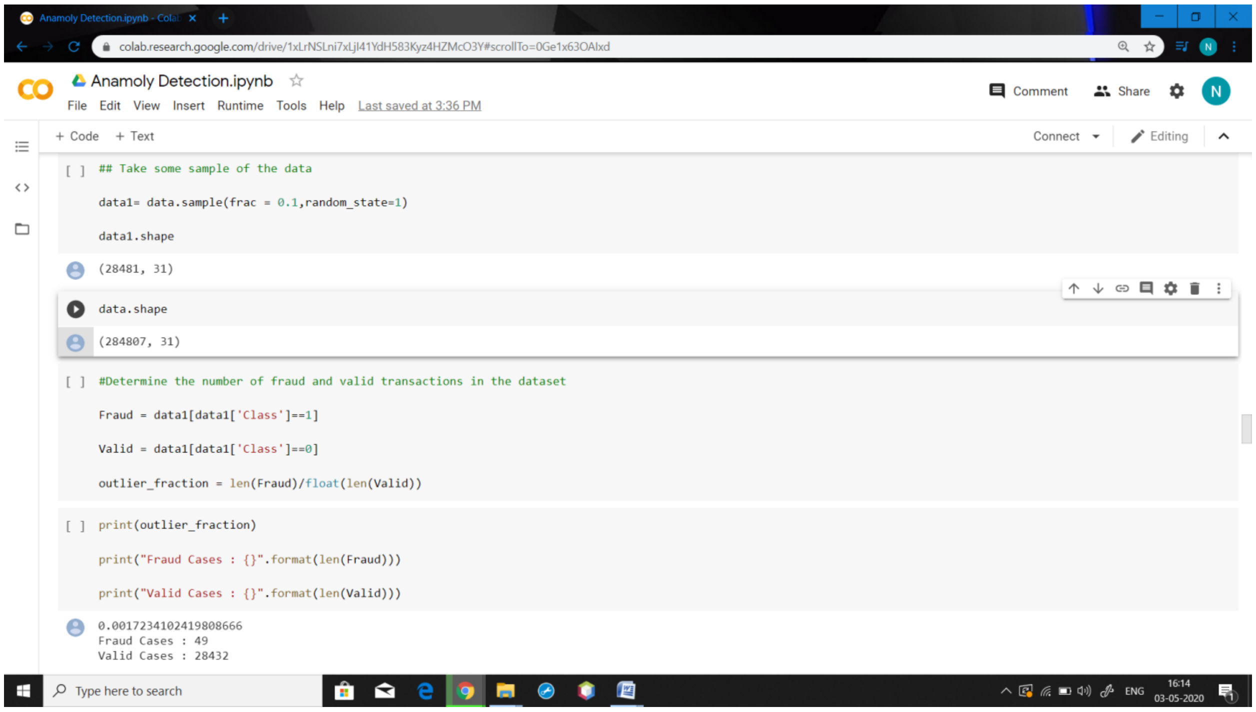
```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```



```
ax1.hist(fraud.Amount, bins = bins)
ax1.set_title('Fraud')
ax2.hist(normal.Amount, bins = bins)
ax2.set_title('Normal')
plt.xlabel('Amount ($)')
plt.ylabel('Number of Transactions')
plt.xlim((0, 20000))
plt.yscale('log')
plt.show();
```


Anamoly Detection.ipynb - Colab
colab.research.google.com/drive/1xLrNSLni7xLjl41YdH583Kyz4HZMcO3Y#scrollTo=XttPe8ICAlxO
Anamoly Detection.ipynb
File Edit View Insert Runtime Tools Help Last saved at 3:36 PM
Comment Share
+ Code + Text
Connect Editing
Time of transaction vs Amount by class
Fraud
Normal
Amount
Time (in Seconds)
Type here to search
ENG 16:13 03-05-2020

```
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
f.suptitle('Time of transaction vs Amount by class')
ax1.scatter(Fraud.Time, Fraud.Amount)
ax1.set_title('Fraud')
ax2.scatter(Normal.Time, Normal.Amount)
ax2.set_title('Normal')
plt.xlabel('Time (in Seconds)')
plt.ylabel('Amount')
plt.show()
```


Anamoly Detection.ipynb - Colab
colab.research.google.com/drive/1xLrNSLni7xLjl41YdH583Kyz4HZMcO3Y#scrollTo=0Ge1x63OAlxd
Anamoly Detection.ipynb
File Edit View Insert Runtime Tools Help Last saved at 3:36 PM
Comment Share
+ Code + Text
Connect Editing
Type here to search
ENG 16:14 03-05-2020

```
## Take some sample of the data

data1= data.sample(frac = 0.1,random_state=1)

data1.shape
```

```
(28481, 31)
```

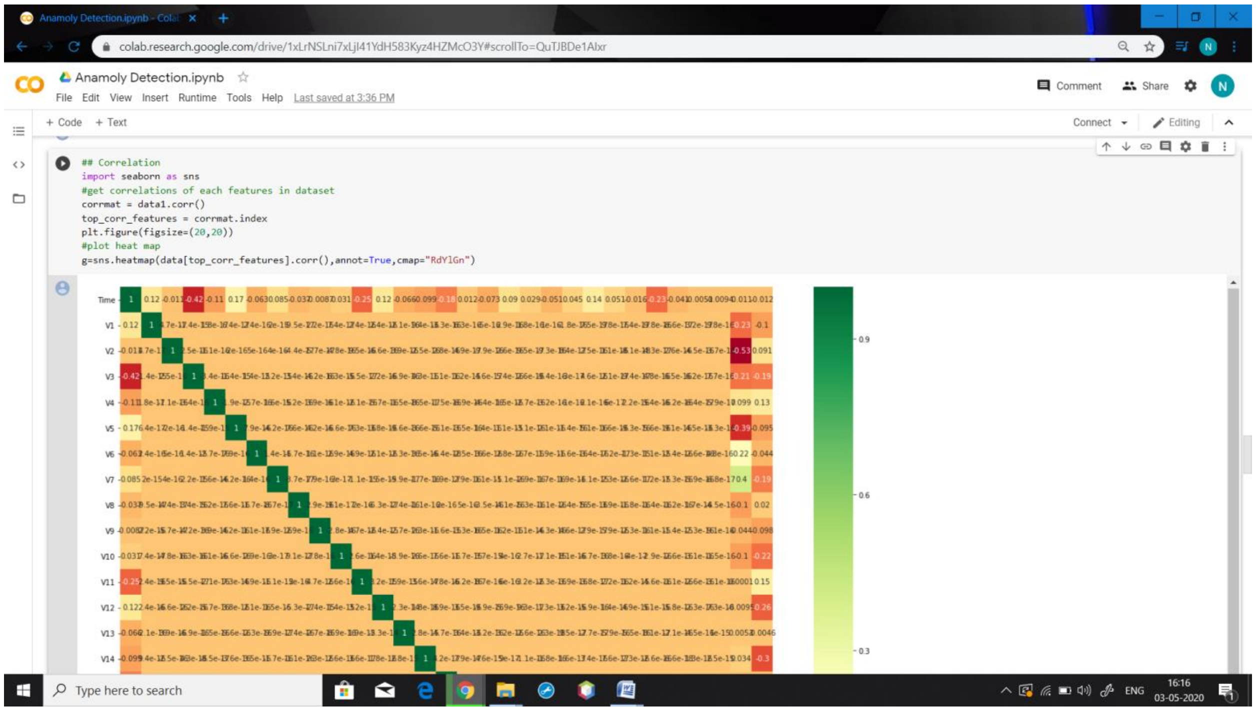
```
data.shape
```

```
(284807, 31)
```

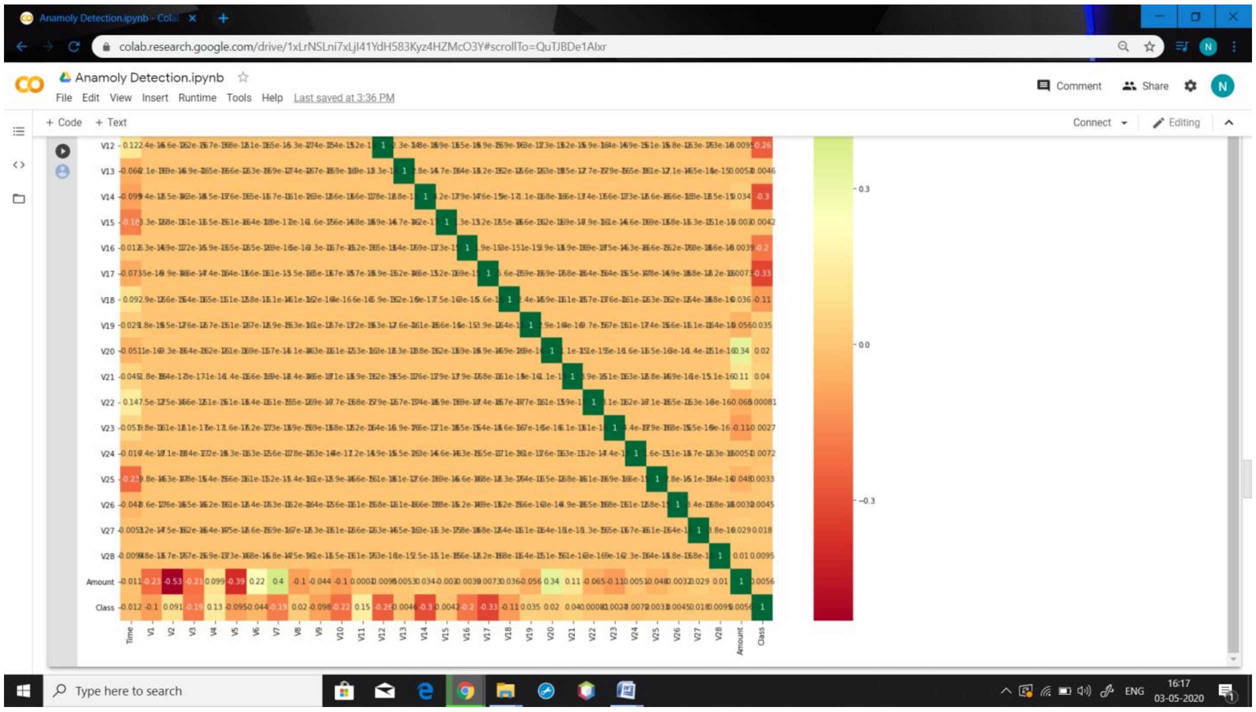
```
#Determine the number of fraud and valid transactions in the dataset

Fraud = data1[data1['Class']==1]

Valid = data1[data1['Class']==0]

outlier_fraction = len(Fraud)/float(len(Valid))
```

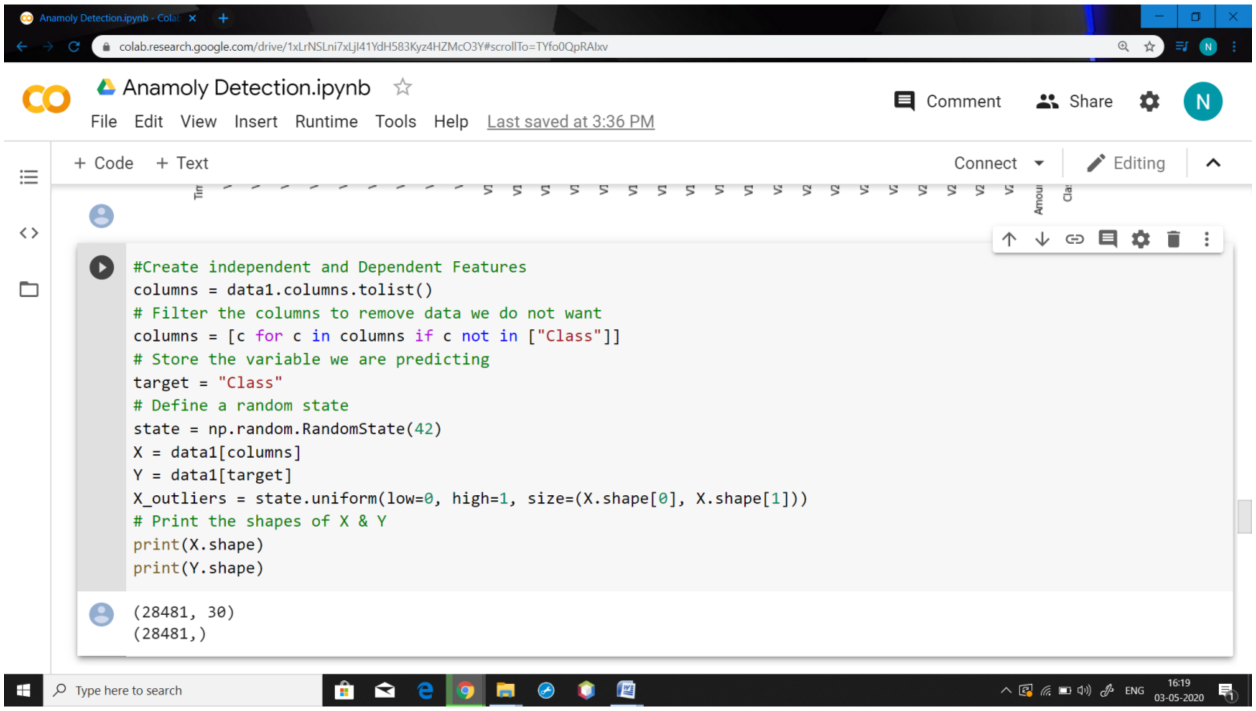
```
print(outlier_fraction)

print("Fraud Cases : {}".format(len(Fraud)))

print("Valid Cases : {}".format(len(Valid)))
```

```
0.0017234102419808666
Fraud Cases : 49
Valid Cases : 28432
```


Anamoly Detection.ipynb - Colab
colab.research.google.com/drive/1xLrNSLni7xLjl41YdH583Kyz4HZMcO3Y#scrollTo=QuTJBDe1Alxr
Anamoly Detection.ipynb
File Edit View Insert Runtime Tools Help Last saved at 3:36 PM
Comment Share
+ Code + Text
Connect Editing
Type here to search
ENG 16:16 03-05-2020

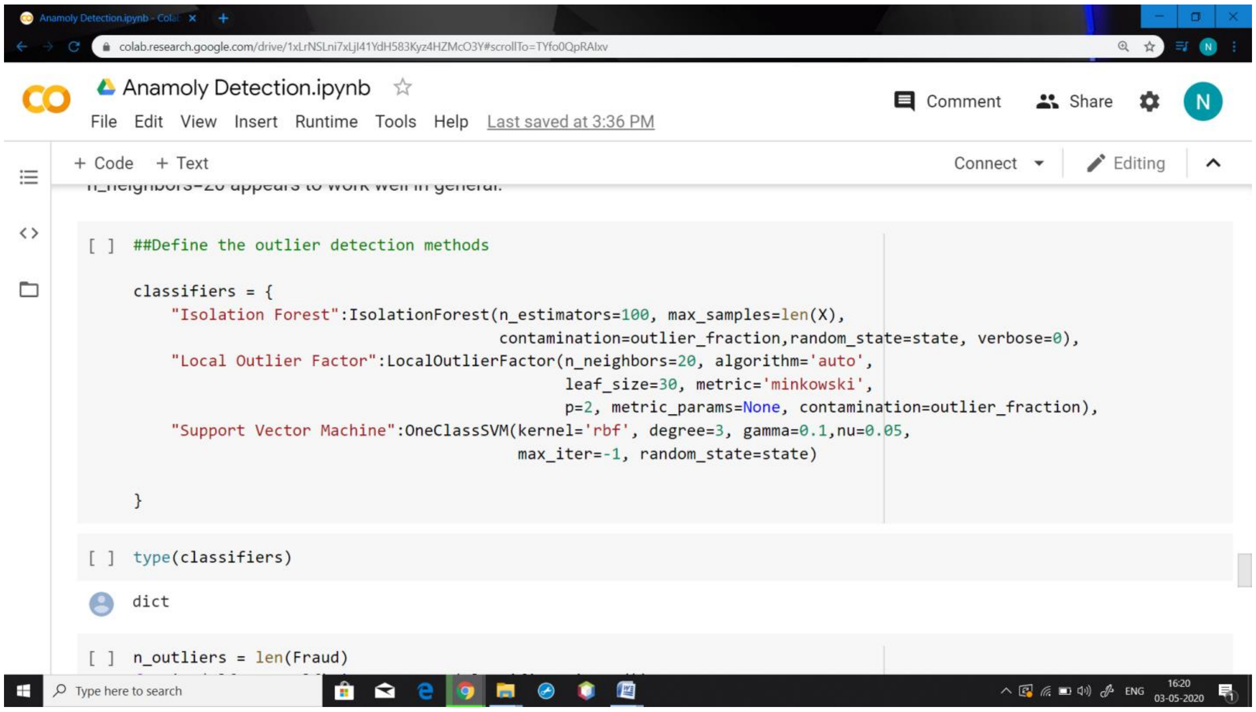
```
## Correlation
import seaborn as sns
#get correlations of each features in dataset
corrmat = data1.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```




Anamoly Detection.ipynb - Colab
colab.research.google.com/drive/1xLrNSLni7xLjl41YdH583Kyz4HZMcO3Y#scrollTo=TYfo0QpRAlxv
Anamoly Detection.ipynb
File Edit View Insert Runtime Tools Help Last saved at 3:36 PM
Comment Share
+ Code + Text
Connect Editing
Type here to search
16:19 03-05-2020

```python
#Create independent and Dependent Features
columns = data1.columns.tolist()
# Filter the columns to remove data we do not want
columns = [c for c in columns if c not in ["Class"]]
# Store the variable we are predicting
target = "Class"
# Define a random state
state = np.random.RandomState(42)
X = data1[columns]
Y = data1[target]
X_outliers = state.uniform(low=0, high=1, size=(X.shape[0], X.shape[1]))
# Print the shapes of X & Y
print(X.shape)
print(Y.shape)
```

```
(28481, 30)
(28481,)
```


Anamoly Detection.ipynb - Colab
colab.research.google.com/drive/1xLrNSLni7xLjl41YdH583Kyz4HZMcO3Y#scrollTo=TYfo0QpRAlxv
Anamoly Detection.ipynb
File Edit View Insert Runtime Tools Help Last saved at 3:36 PM
Comment Share
+ Code + Text
Connect Editing
Type here to search
16:20 03-05-2020

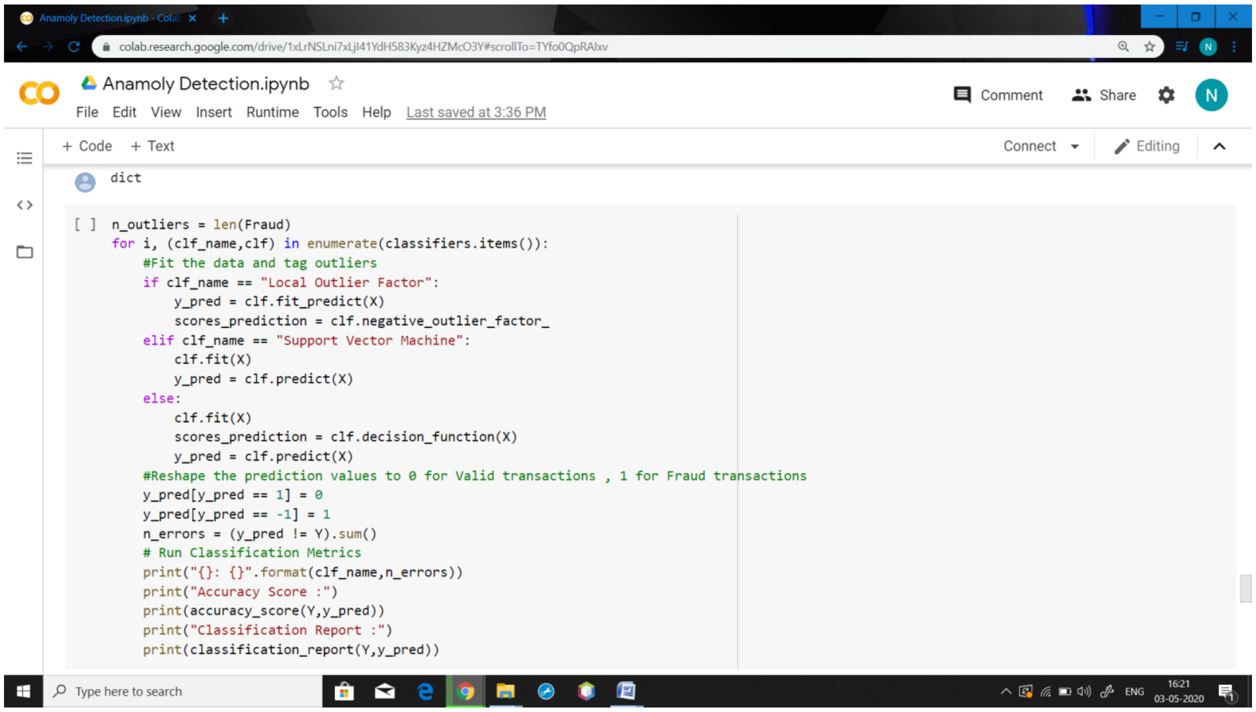
n_neighbors=20 appears to work well in general.

```python
##Define the outlier detection methods

classifiers = {
    "Isolation Forest":IsolationForest(n_estimators=100, max_samples=len(X),
                                       contamination=outlier_fraction,random_state=state, verbose=0),
    "Local Outlier Factor":LocalOutlierFactor(n_neighbors=20, algorithm='auto',
                                              leaf_size=30, metric='minkowski',
                                              p=2, metric_params=None, contamination=outlier_fraction),
    "Support Vector Machine":OneClassSVM(kernel='rbf', degree=3, gamma=0.1,nu=0.05,
                                         max_iter=-1, random_state=state)

}
```

```python
type(classifiers)
```

```
dict
```

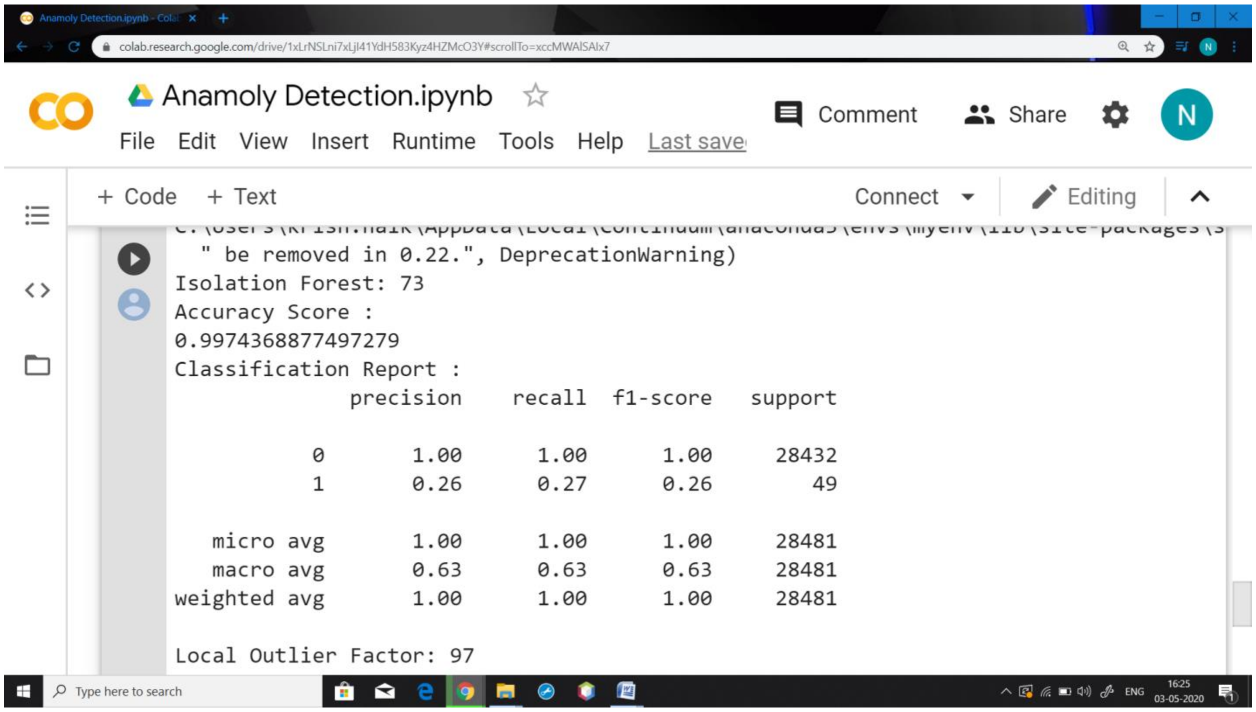
```python
n_outliers = len(Fraud)
```


Anamoly Detection.ipynb - Colab
colab.research.google.com/drive/1xLrNSLni7xLjI41YdH583Kyz4HZMcO3Y#scrollTo=TYfo0QpRAlxv
Anamoly Detection.ipynb
File Edit View Insert Runtime Tools Help Last saved at 3:36 PM
Comment
Share
+ Code + Text
Connect
Editing
Type here to search
ENG
16:21
03-05-2020

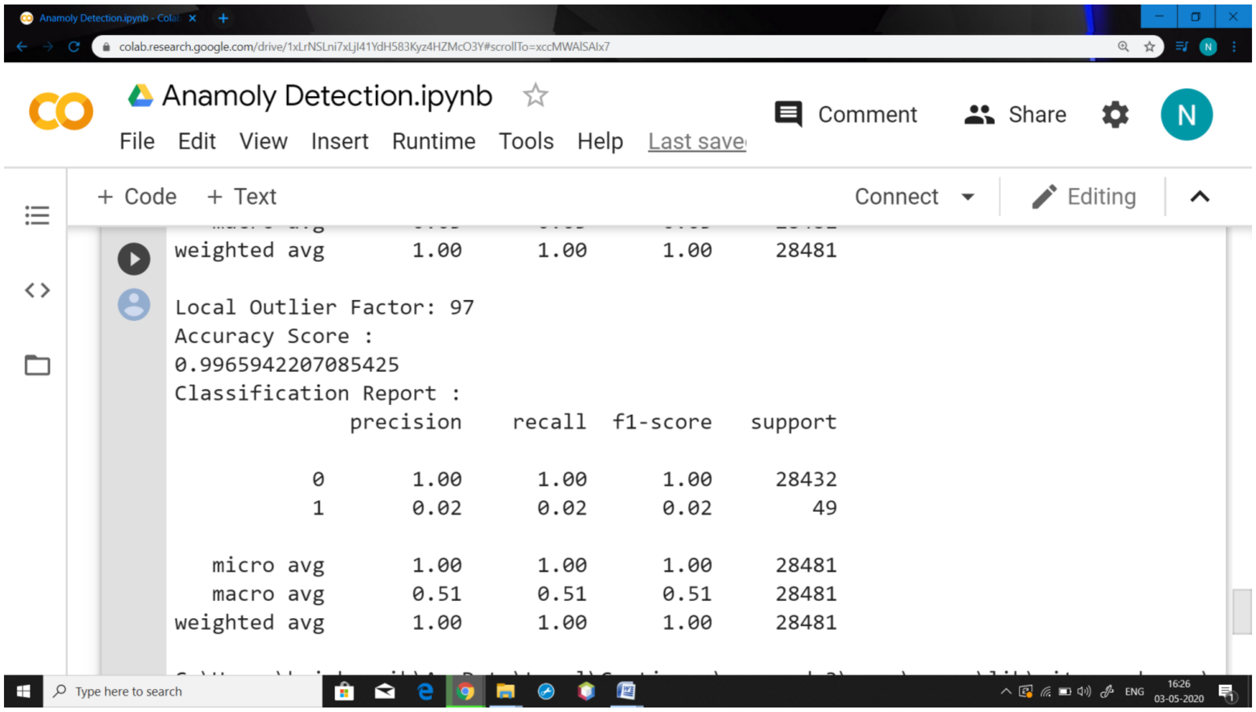
```
dict
```

```
[ ] n_outliers = len(Fraud)
    for i, (clf_name,clf) in enumerate(classifiers.items()):
        #Fit the data and tag outliers
        if clf_name == "Local Outlier Factor":
            y_pred = clf.fit_predict(X)
            scores_prediction = clf.negative_outlier_factor_
        elif clf_name == "Support Vector Machine":
            clf.fit(X)
            y_pred = clf.predict(X)
        else:
            clf.fit(X)
            scores_prediction = clf.decision_function(X)
            y_pred = clf.predict(X)
        #Reshape the prediction values to 0 for Valid transactions , 1 for Fraud transactions
        y_pred[y_pred == 1] = 0
        y_pred[y_pred == -1] = 1
        n_errors = (y_pred != Y).sum()
        # Run Classification Metrics
        print("{}: {}".format(clf_name,n_errors))
        print("Accuracy Score :")
        print(accuracy_score(Y,y_pred))
        print("Classification Report :")
        print(classification_report(Y,y_pred))
```

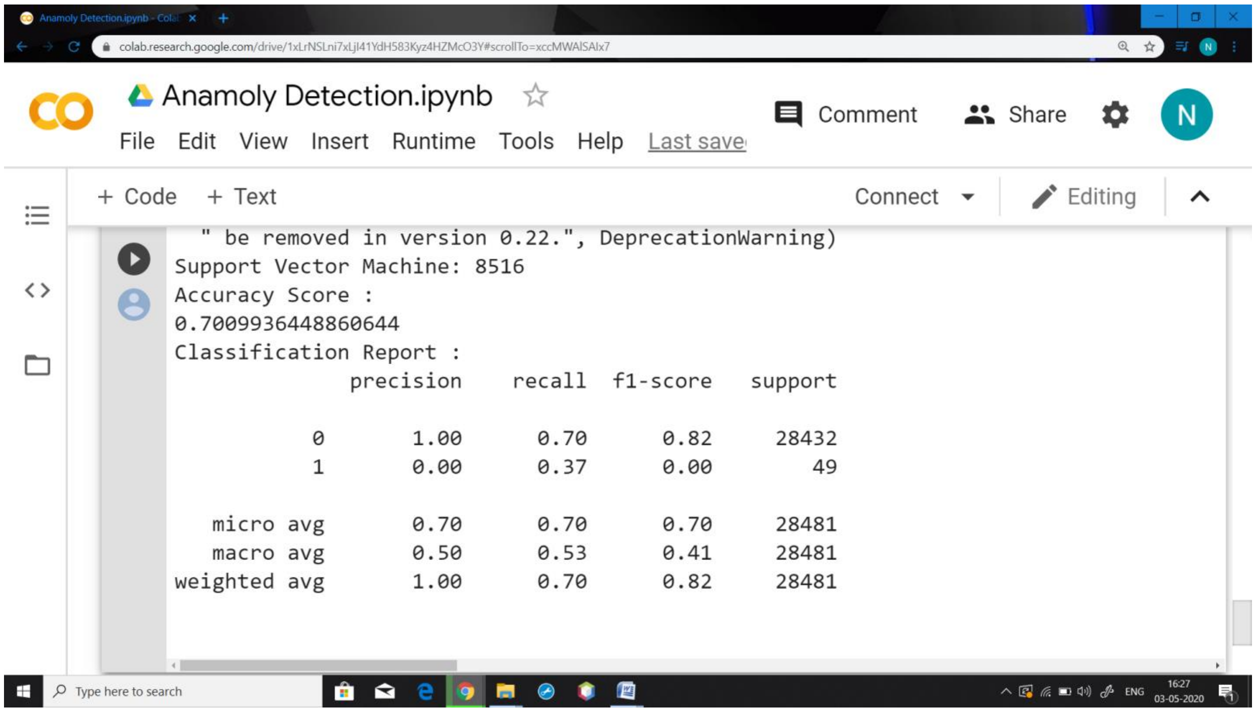
OUTPUT


Anamoly Detection.ipynb
File Edit View Insert Runtime Tools Help Last save
Comment Share
+ Code + Text Connect Editing

```
 " be removed in 0.22.", DeprecationWarning)
Isolation Forest: 73
Accuracy Score :
0.9974368877497279
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28432
           1       0.26      0.27      0.26        49

   micro avg       1.00      1.00      1.00     28481
   macro avg       0.63      0.63      0.63     28481
weighted avg       1.00      1.00      1.00     28481

Local Outlier Factor: 97
```


Anamoly Detection.ipynb
File Edit View Insert Runtime Tools Help Last save
Comment Share
+ Code + Text Connect Editing

```
weighted avg       1.00      1.00      1.00     28481

Local Outlier Factor: 97
Accuracy Score :
0.9965942207085425
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28432
           1       0.02      0.02      0.02        49

   micro avg       1.00      1.00      1.00     28481
   macro avg       0.51      0.51      0.51     28481
weighted avg       1.00      1.00      1.00     28481
```


Anamoly Detection.ipynb - Colab
colab.research.google.com/drive/1xLrNSLni7xLjl41YdH583Kyz4HZMcO3Y#scrollTo=xccMWAlSAlx7
Anamoly Detection.ipynb
File Edit View Insert Runtime Tools Help Last save
Comment
Share
+ Code + Text
Connect
Editing
Type here to search
16:27
03-05-2020

```
  " be removed in version 0.22.", DeprecationWarning)
Support Vector Machine: 8516
Accuracy Score :
0.7009936448860644
Classification Report :
              precision    recall  f1-score   support

           0       1.00      0.70      0.82     28432
           1       0.00      0.37      0.00        49

   micro avg       0.70      0.70      0.70     28481
   macro avg       0.50      0.53      0.41     28481
weighted avg       1.00      0.70      0.82     28481
```

Conclusion

- Isolation Forest detected 73 errors versus Local Outlier Factor detecting 97 errors vs. SVM detecting 8516 errors

- Isolation Forest has a 99.74% more accurate than LOF of 99.65% and SVM of 70.09

- When comparing error precision & recall for 3 models , the Isolation Forest performed much better than the LOF as we can see that the detection of fraud cases is around 27 % versus LOF detection rate of just 2 % and SVM of 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

References

1. www.kaggle.com
2. www.wikipedia.com
3. www.google.com
4. www.researchgate.net
5. www.youtube.com