UMBC

Independent Study on Big data and Apache Spark

Credit card fraud prediction using python and spark

Under the guidance of
Dr. Jianwu Wang

Report Submitted by:

AISHWARYA BHONDE

SIDDHESH SAWANT

NEELESH SURYAWANSHI

ROHIT KOUL

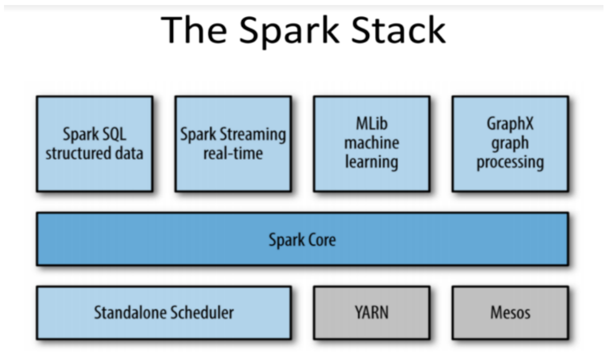
Contents:

1. Introduction:

Before the real presentation of apache-spark, how about we attempt to examine why this was presented what was the requirement for this innovation in this day and age and how is it helping ventures and everyday citizens to change the manner in which information is taken a gander at or dealt with. Well 'Information' is only the assortment of realities like estimations, numbers, words, or any kind of perceptions which is interpreted in a manner the PC can comprehend and process it with some important bits of knowledge. Spark is a universally useful circulated information handling engine that is reasonable for use in a wide scope of conditions. In any use of decision, we can utilize libraries like AI, SQL, stream handling, and diagram calculation on spark center information preparing engine.

Presently days with the expansion in the utilization of internet-based life and innovation there is a humungous measure of information that is produced and should be taken care of such that it gives valuable bits of knowledge to the business and individuals. Along these lines, with the expansion in the information ideas like Big information came into the picture and to deal with a similar apache-spark was presented. The manner in which spark performs is, any utilization of flash runs as an individual procedure and is facilitated by the Spark meeting object in the driver program and afterward the group administrator appoints undertakings according to the segment, in turn, the assignment applies its unit of work to the dataset in its parcel and yields another segment and by applying iterative calculations errands run their part and reserving is done across cycles. At last, results are sent back to the driver application.

As we are going to implement different statistical models using pyspark and python. Everyone has started with python basics along with the spark libraries. In this project, we will use various predictive models to see how accurate they are in detecting whether a transaction is a normal payment or a fraud. As described in the dataset, the features are scaled, and the names of the features are not shown due to privacy reasons. Nevertheless, we can still analyze some important aspects of the dataset.



2. Problem Statement:

Credit card fraud is one of the major problems in today's world where all the transaction is based on online services. The project goal is to predict fraudulent credit card transaction base on features such as 'Time' and 'Amount' using different statistical models like Logistic Regression, GBT and KNN. We are also demonstrating the behavior of potentially fraudulent providers; the exploratory analysis will discover important variables to understand the fraudulent patterns in the provider's claims.

3. Dataset Description:

The dataset contains just numerical information factors which are the consequence of a PCA change. Shockingly, because of classification issues unique highlights are not given and more foundation data about the information is additionally not present.

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.

Dataset description:

Time - Number of seconds elapsed between the current transaction and the first transaction in the dataset

V1-V28 - PCA Dimensionality reduction to protect user identities and sensitive features(v1-v28)

Amount - Transaction amount

Class - 1 for fraudulent transactions, 0 otherwise

Features V1, V2, ... V28 are the primary parts acquired with PCA, the main highlights which have not been changed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds slipped by between every exchange and the main exchange in the dataset. The component 'Amount' is the exchange Amount, this element can be utilized for instance dependent cost-delicate learning. Feature 'Class' is the response variable and it takes value 1 if there should arise an occurrence of fraud and 0 in any case.

Spark Techniques:

- Spark Session: This is a session created when we are running spark applications. It is a unified entry point for any spark application. It provides a way to interact with various spark functionalities.
- Pyspark: This helps to interface with resilient distributed datasets in apache spark and python. Py4J is a popular library integrated within pyspark that lets python interface dynamically with java virtual machine objects (RDD’s).
- Pyspark.sql.functions: This is one of the apache spark’s module that helps to work on structured data.
- Pyspark.ml.feature: This is a new package added to spark after version 1.2 that aims to provide a uniform set of high-level APIs which in turn helps users to tune or create their dataset by using practical machine learning pipelines.

4. Knowledge gained during the project:

This course has brought us into an alternate measurement and a significant piece of the Data world which is 'Big Data' and applications used for running the equivalent, instruments, and distinctive programming's utilized. The principal goal was to find out about what is the spark. For what reason is that utilized? How could that be utilized? What kind of platform we can use to perform the programming? In our project we have used the Databricks and created cluster to perform the various models.

Why we used seaborn and matplotlib?

Seaborn is a Python library made for upgraded information representation and visualization. It's a convenient and pertinent tool for information experts working today exactly in light of the fact

that successful information perception – and correspondence when all is said in done – is an especially basic ability. Having the option to overcome any issues among information and understanding is colossally significant, and Seaborn is an apparatus that fits easily in the toolchain of anybody keen on doing only that.
matplotlib has a broad codebase that can be overwhelming to numerous new clients. Be that as it may, the majority of matplotlib can be comprehended with a genuinely straightforward theoretical structure and information on a couple of significant focuses. Plotting requires activity on a scope of levels, from the broadest (e.g., 'form this 2-D exhibit') to the most explicit (e.g., 'shading this screen pixel red'). The reason for a plotting bundle is to help you in imagining your information as effectively as could be expected under the circumstances, with all the important control – that is, by utilizing generally elevated level orders more often than not, and still can utilize the low-level orders when required.

5. Solution and Execution:

As per the requirement of the data set, we have taken the following steps:

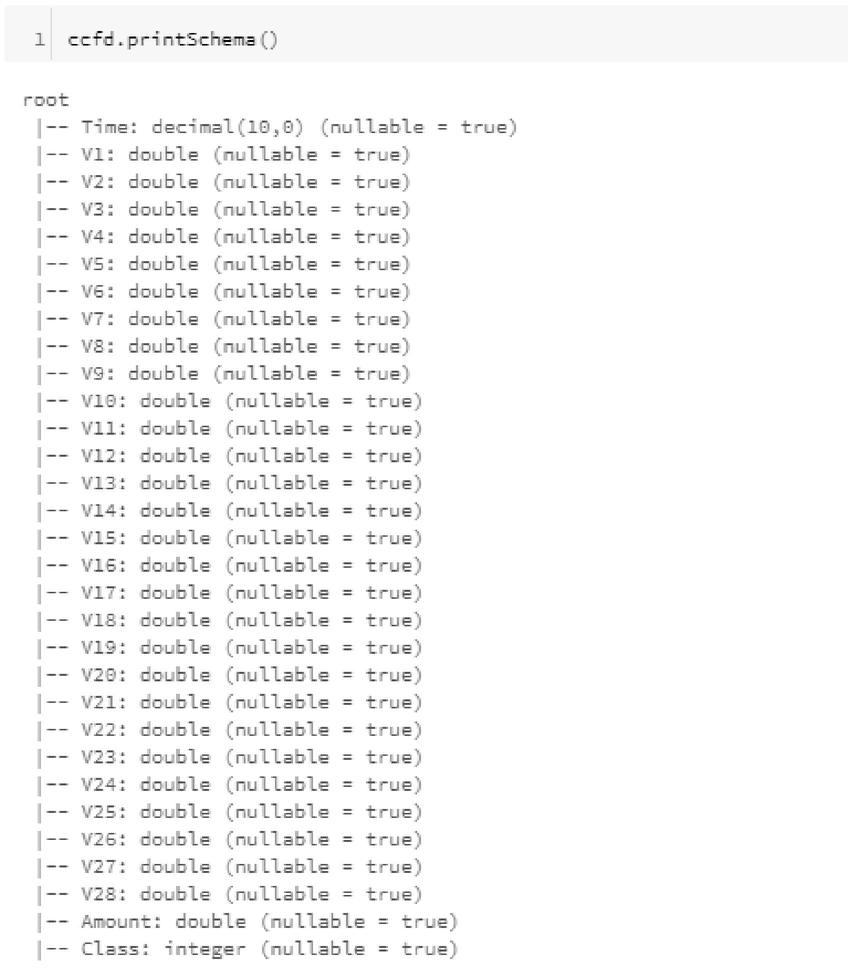
1. Creating cluster in data bricks
2. Uploading data set
3. Loading the data into the work environment
4. Importing the libraries
5. Printing the schema
6. Exploratory data analysis -to identify outliers
7. Data preprocessing
    a. Removing outliers
    b. Standardizing data
8. Data splitting into training and testing data
9. Logistic Regression
10. GBT Classifier
11. KNN Classifier
12. TSNE

13. Model Accuracy and other metrics

6. Model description:

Gradient-Boosted Trees (GBTs) is a learning algorithm for classification. It is a member of the group of ensemble models. The GBT build several weak models that, when combined, form a strong classifier. It supports binary labels, as well as both continuous and categorical features. The general idea for using GBT algorithm is to compute a sequence of simple trees, where each successive tree is built for prediction residuals of the preceding tree. The predictions made by final ensemble model is therefore the weighted sum of the predictions made by the previous tree models.

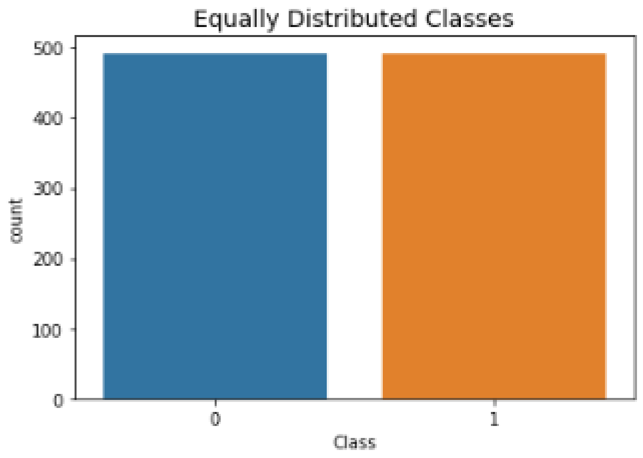
7. Screenshots of the implementation:



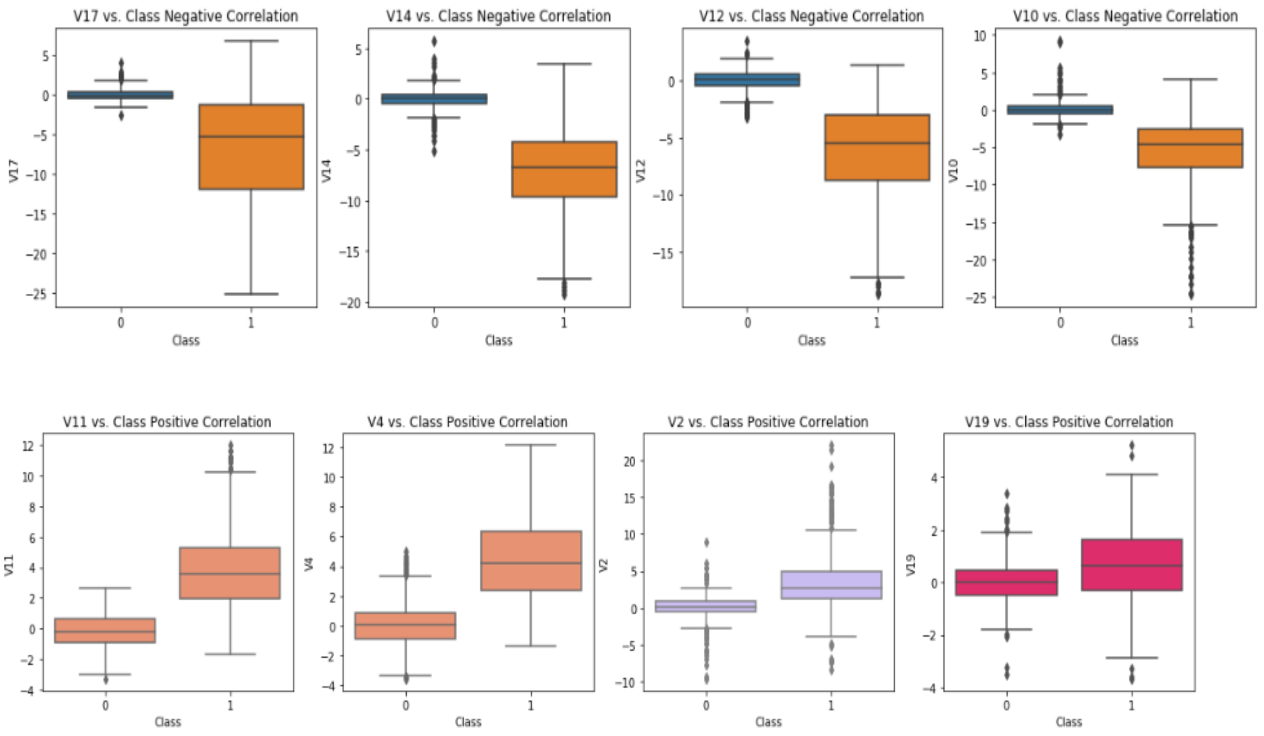
```
ccfd.printSchema()
```

```
root
 |-- Time: decimal(10,0) (nullable = true)
 |-- V1: double (nullable = true)
 |-- V2: double (nullable = true)
 |-- V3: double (nullable = true)
 |-- V4: double (nullable = true)
 |-- V5: double (nullable = true)
 |-- V6: double (nullable = true)
 |-- V7: double (nullable = true)
 |-- V8: double (nullable = true)
 |-- V9: double (nullable = true)
 |-- V10: double (nullable = true)
 |-- V11: double (nullable = true)
 |-- V12: double (nullable = true)
 |-- V13: double (nullable = true)
 |-- V14: double (nullable = true)
 |-- V15: double (nullable = true)
 |-- V16: double (nullable = true)
 |-- V17: double (nullable = true)
 |-- V18: double (nullable = true)
 |-- V19: double (nullable = true)
 |-- V20: double (nullable = true)
 |-- V21: double (nullable = true)
 |-- V22: double (nullable = true)
 |-- V23: double (nullable = true)
 |-- V24: double (nullable = true)
 |-- V25: double (nullable = true)
 |-- V26: double (nullable = true)
 |-- V27: double (nullable = true)
 |-- V28: double (nullable = true)
 |-- Amount: double (nullable = true)
 |-- Class: integer (nullable = true)
```

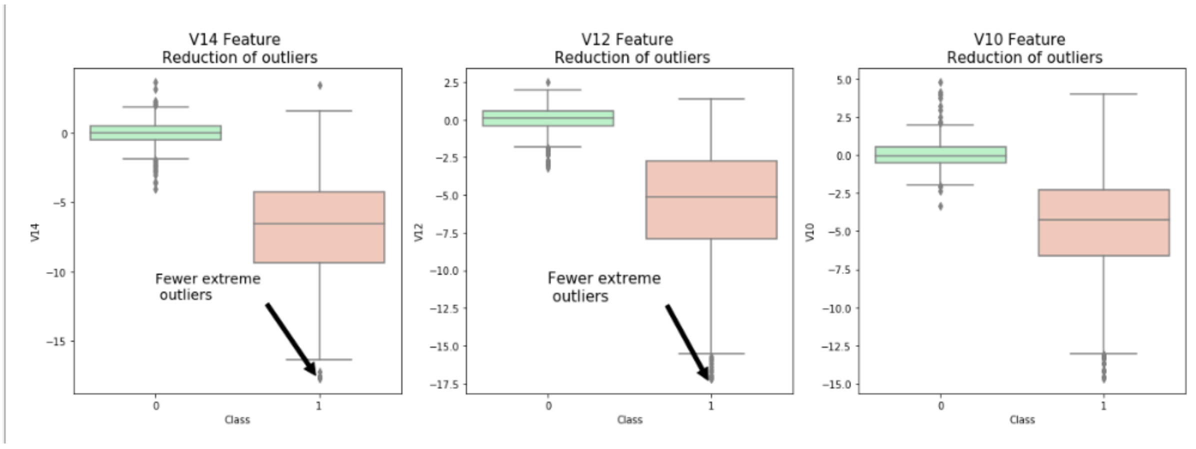
Distribution of the Classes in the subsample dataset



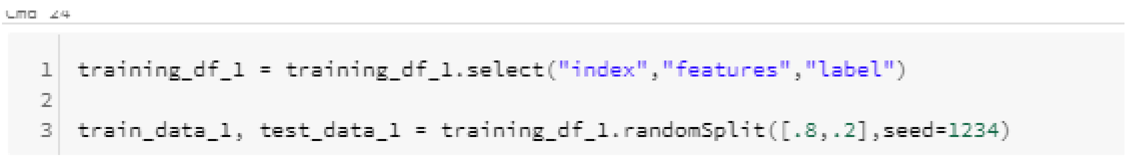
Identification of Outliers using boxplots



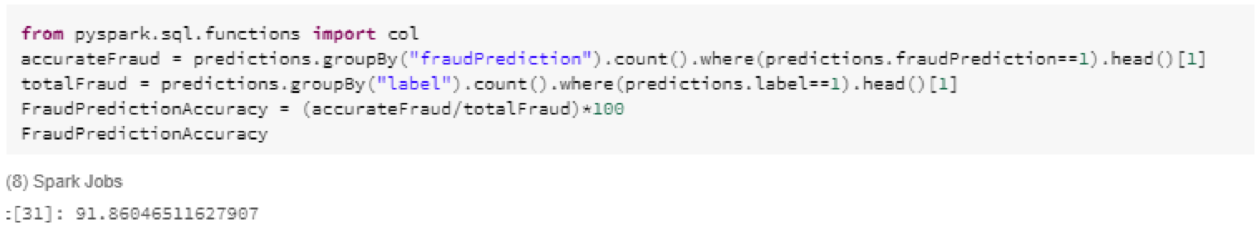
Removing the outliers



Splitting of the dataset

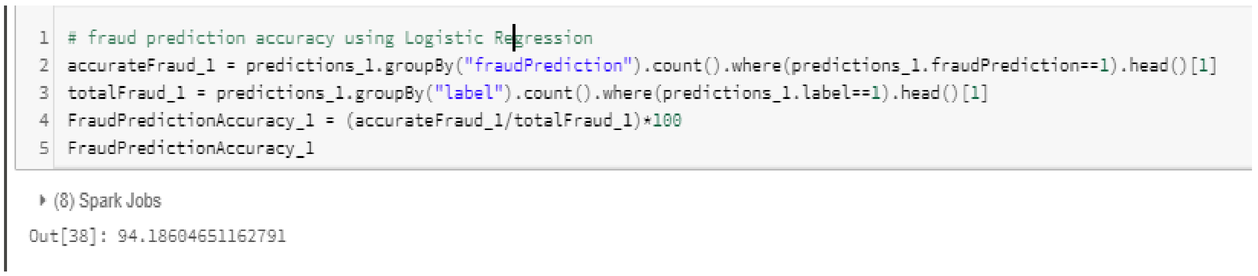

```
Cmd 24
training_df_1 = training_df_1.select("index","features","label")

train_data_1, test_data_1 = training_df_1.randomSplit([.8,.2],seed=1234)
```

Fraud prediction accuracy using GBT


```
from pyspark.sql.functions import col
accurateFraud = predictions.groupBy("fraudPrediction").count().where(predictions.fraudPrediction==1).head()[1]
totalFraud = predictions.groupBy("label").count().where(predictions.label==1).head()[1]
FraudPredictionAccuracy = (accurateFraud/totalFraud)*100
FraudPredictionAccuracy
```

```
(8) Spark Jobs
:[31]: 91.86046511627907
```

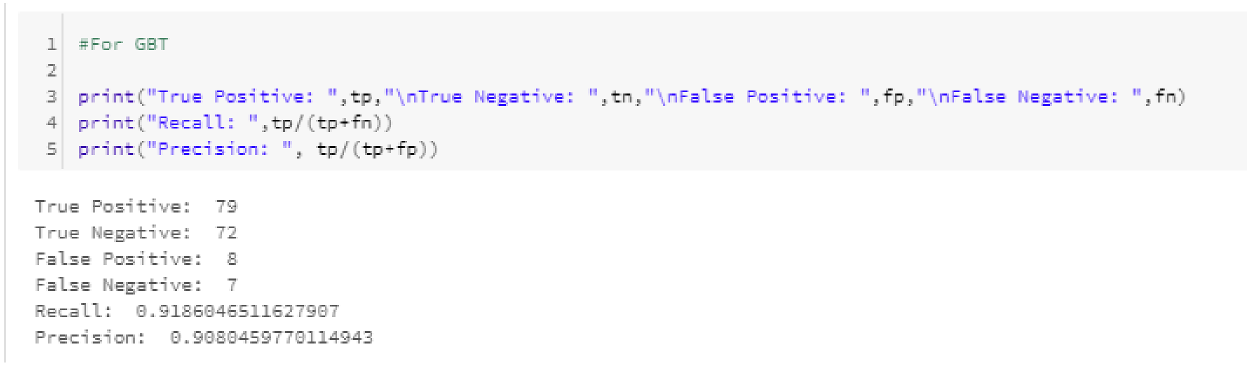
Fraud prediction accuracy using Logistic Regression


```
# fraud prediction accuracy using Logistic Regression
accurateFraud_1 = predictions_1.groupBy("fraudPrediction").count().where(predictions_1.fraudPrediction==1).head()[1]
totalFraud_1 = predictions_1.groupBy("label").count().where(predictions_1.label==1).head()[1]
FraudPredictionAccuracy_1 = (accurateFraud_1/totalFraud_1)*100
FraudPredictionAccuracy_1
```

```
(8) Spark Jobs
Out[38]: 94.18604651162791
```

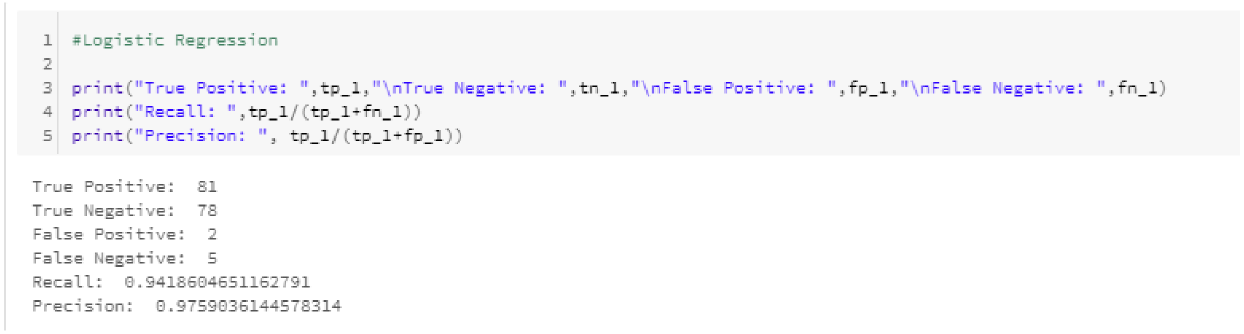
Calculating Recall and Precision for GBT Classifier, Logistic Regression and KNN classifier

Precision and recall are two extremely important evaluation metrics for the model. While precision refers to the percentage of your results that are relevant, recall refers to the percentage of total relevant results that your algorithm correctly classifies. Unfortunately, all these metrics cannot be maximized at the same time since one comes at the expense of another.



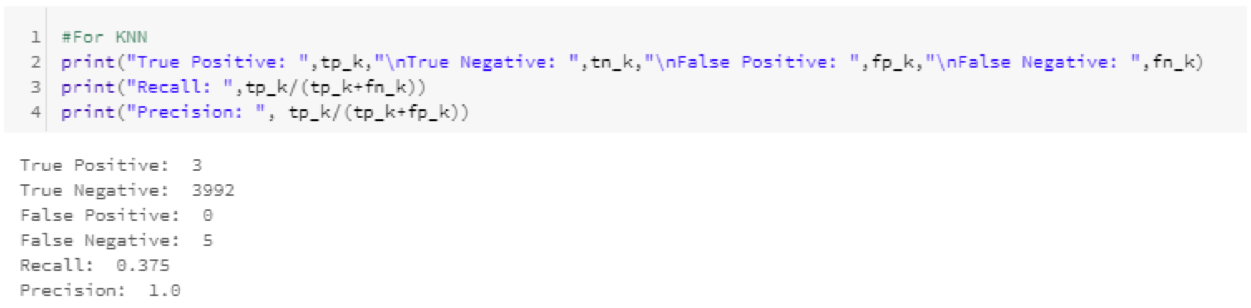
```python
#For GBT

print("True Positive: ",tp,"\nTrue Negative: ",tn,"\nFalse Positive: ",fp,"\nFalse Negative: ",fn)
print("Recall: ",tp/(tp+fn))
print("Precision: ", tp/(tp+fp))
```

```
True Positive:  79
True Negative:  72
False Positive:  8
False Negative:  7
Recall:  0.9186046511627907
Precision:  0.9080459770114943
```



```python
#Logistic Regression

print("True Positive: ",tp_l,"\nTrue Negative: ",tn_l,"\nFalse Positive: ",fp_l,"\nFalse Negative: ",fn_l)
print("Recall: ",tp_l/(tp_l+fn_l))
print("Precision: ", tp_l/(tp_l+fp_l))
```

```
True Positive:  81
True Negative:  78
False Positive:  2
False Negative:  5
Recall:  0.9418604651162791
Precision:  0.9759036144578314
```



```python
#For KNN
print("True Positive: ",tp_k,"\nTrue Negative: ",tn_k,"\nFalse Positive: ",fp_k,"\nFalse Negative: ",fn_k)
print("Recall: ",tp_k/(tp_k+fn_k))
print("Precision: ", tp_k/(tp_k+fp_k))
```

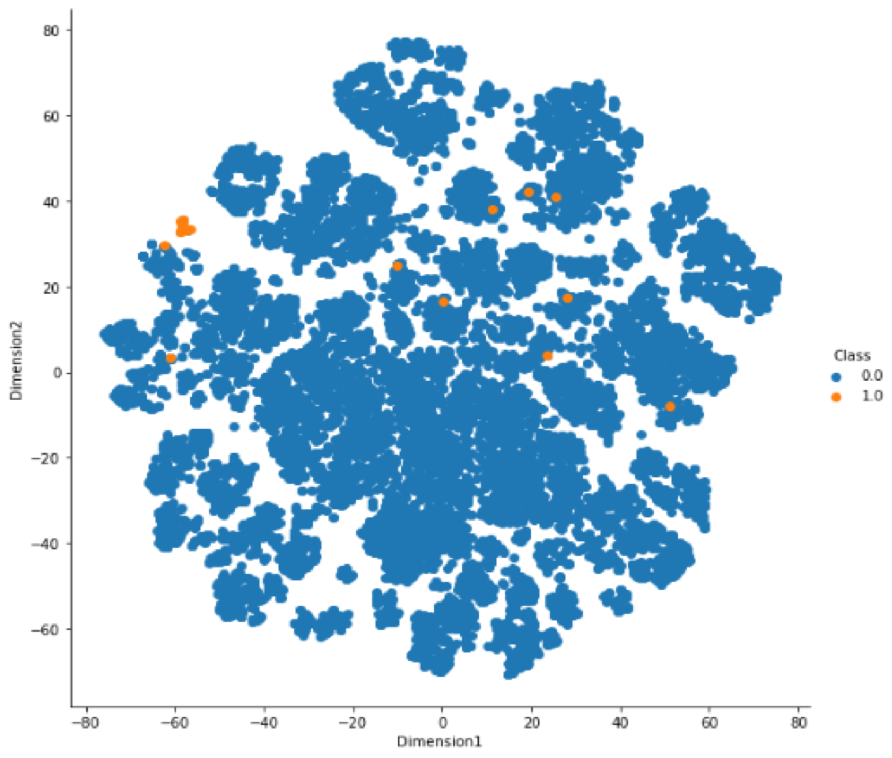
```
True Positive:  3
True Negative:  3992
False Positive:  0
False Negative:  5
Recall:  0.375
Precision:  1.0
```

TSNE:

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear, unsupervised technique commonly used for data discovery and high-dimensional data visualization. In simpler terms, t-SNE gives you a feel or intuition of how the data is arranged in a high-dimensional space.

(t-SNE) t-Distributed Stochastic Neighbor Embedding is a non-linear dimensionality reduction algorithm used for exploring high-dimensional data. It maps multi-dimensional data to two or more dimensions suitable for human observation. With help of the t-SNE algorithms, you may have to plot fewer exploratory data analysis plots next time you work with high dimensional data. dimensionality reduction is the technique of representing multi-dimensional data (data with multiple features having a correlation with each other) in 2 or 3 dimensions.

```python
from sklearn.manifold import TSNE
data_25k = standardized_data_1[0:25000]
labels_25k = cc[0:25000]

model = TSNE(n_components=2, random_state=0, perplexity=50, n_iter=1000)

tsne_data = model.fit_transform(data_25k)

# creating a new data frame which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, labels_25k)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dimension1", "Dimension2", "Class"))

# Ploting the result of tsne
sns.FacetGrid(tsne_df, hue="Class", size=8).map(plt.scatter, 'Dimension1', 'Dimension2').add_legend()
plt.show()
display()
```



8. Conclusion:

The prevention of credit card is very much important for the smooth transaction among the credit card holders. It was an extraordinary encounter implementing this project, likewise we confronted a great deal of challenges taking care of the errors and executing the models. We utilized pyspark, pyspark.sql.functions and pyspark.ml.features so as to structure the information and actualize the calculated relapse, GBT, TSNE and KNN Classifier to tune our information and get most extreme precision. The accuracy achieved for this undertaking is 91.86% for GBT and for Logistic Regression is 94.19%.

Here is the difference we found between spark libraries and python libraries:

PySpark is an API composed for utilizing Python alongside Spark structure. As we as a whole know, Spark is a computational motor, that works with Big Data and Python is a programming language. During the use of PySpark and python we get to know that PySpark is a tool to support python on Spark and the time consumption in using the PySpark library is much lesser then the python. It concludes that if we are working on Big Data and Data Mining which may take much time to process, python is not enough.
PySpark is unmistakably a requirement for information researchers, who are not truly open to working in Scala in light of the fact that Spark is essentially written in Scala. On the off chance that you have a python software engineer who needs to work with RDDs without learning another programming language, at that point PySpark is the main way.
Instead of it, python has a standard library that underpins a wide assortment of functionalities like databases, computerization, content preparation and logical processing. On the other hand, PySpark uses the library know as Py4j, an API in Python.

If you want to learn more about the project code and methods used, here is the link of GitHub:

9. References:

- https://mapr.com/blog/spark-101-what-it-what-it-does-and-why-it-matters/

- https://spark.apache.org/docs/2.3.0/ml-classification-regression.html#gradient-boosted-trees-gbts
- https://spark.apache.org/docs/2.3.0/ml-classification-regression.html#gradient-boosted-tree-classifier
- https://www.analyticsvidhya.com/blog/2016/10/spark-dataframe-and-operations/
- https://towardsdatascience.com/apache-spark-mllib-tutorial-ec6f1cb336a9
- https://distill.pub/2016/misread-tsne/
- https://seahorse.deepsense.ai/operations/gbt_classifier.html

Github Link:

https://github.com/rkoul1992/Learning-Studio-Website