Fraud Detection in credit cards transactions using python

Team No. 49; Team Members:

1) Rachna Tripathi (203350005)

2) Kaustubh Patil (203350013)

3) Rushikesh Dilip Patil (203350015)

4) Milindkumar Ukey (203350017)

Introduction:

This project is developed to help a credit card company in detecting potential fraud cases so that the customers are ensured that they won't be charged for the items they did not purchase. We have used a dataset containing the transactions between people, the information that they are fraud or not, and have done the differentiation between them. Our ultimate aim is to tackle this situation by creating a classification models to classify and distinguish fraud transactions. Classification is the process of predicting discrete variables (binary, Yes/no, etc.). and in the given scenario, it will be more optimistic to deploy a classification model rather than any others. Python helps in this regard compared to other languages as it has an extensive number of packages for machine learning, and can be learned easily.

Motivation:

To understand the basic machine learning models deeply and apply the python learning.

Methodology:

1. The packages are imported in python environment
2. The data is imported from open-source site
3. The data is processed and exploratory data analysis is carried out
4. The feature selection is carried out and data is split into training and testing data
5. In this case 6 types of classification models are applied
6. The evaluation of the classification model is done using metrics.

Implementation details:

For this project, our primary packages are going to be Pandas to work with data, NumPy to work with arrays, scikit-learn for data split, building and evaluating the classification models, and finally the xgboost package for the xgboost classifier model algorithm.

1. Data used for the project:

The data we have used is the dataset from Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection. It contains features V1 to V28 which are the principal components obtained by PCA. We have neglected the time feature which is of no use to build the models. The remaining features are the 'Amount' feature that contains the total amount of money being transacted and the 'Class' feature that contains whether the transaction is a fraud case or not.

2. Treatment of data and data visualization:

We have checked for no. of fraud cases and no-fraud cases in the data set used with computation of the percentage of fraud cases in the overall recorded transactions is also done. The data visualization is done using histogram, scatter matrix. Also, correlation matrix is shown using Spearman method and heat map is used for better visualization.

We got only 492 fraud cases out of 284,807 samples, which is only 0.17 percent of the total samples. Therefore, this highly imbalanced data is handled carefully during modelling and evaluating. After that, we got a statistical view of both fraud and non-fraud transaction amount data using the 'describe' method in python.

During observation of the the statistics, we saw that the values in the 'Amount' variable are varying enormously when compared to the rest of the variables. To reduce its wide range of values, we can normalize it using the 'StandardScaler' method in python.

3. Feature selection and data splitting:

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

4. Modelling:

In this step, we have built 6 different types of classification models which are Decision Tree, K-Nearest Neighbors (KNN), Logistic Regression, Support Vector Machine (SVM), Random Forest, and XGBoost. Even though there are many more models which we can use, these are the most popular models used for solving classification problems. All these models can be built feasibly using the algorithms provided by the scikit-learn package. Only for the XGBoost model, we are going to use the xgboost package. Though these algorithms sometimes take time to get implemented. All 6 different types of classification models used are described below:

Starting with the decision tree, we have used the 'DecisionTreeClassifier' algorithm to build the model. Inside the algorithm, we have mentioned the 'max_depth' to be '4' which means we are allowing the tree to split four times and the 'criterion' to be 'entropy' which is most similar to the 'max_depth' but determines when to stop splitting the tree. Finally, we have fitted and stored the predicted values into the 'tree_yhat' variable.

Next is the K-Nearest Neighbors (KNN). We used the model using the 'KNeighborsClassifier' algorithm and mentioned the 'n_neighbors' to be '5'. The value of the 'n_neighbors' is randomly selected but can be chosen optimistically through iterating a range of values, followed by fitting and storing the predicted values into the 'knn_yhat' variable.

There is nothing much to explain about the code for Logistic regression as we kept the model in a way more simplistic manner by using the 'LogisticRegression' algorithm and as usual, fitted and stored the predicted variables in the 'lr_yhat' variable.

We used the Support Vector Machine model using the 'SVC' algorithm and we didn't mention anything inside the algorithm as we managed to use the default kernel which is the 'rbf' kernel. After that, we stored the predicted values into the 'svm_yhat' after fitting the model.

The next model is the Random Forest model using the 'RandomForestClassifier' algorithm and we mentioned the 'max_depth' to be 4 just like how we did to build the decision tree model. Finally, fitting and storing the values into the 'rf_yhat'. Remember that the main difference between the decision tree and the random forest is that, decision tree uses the entire dataset to construct a single model whereas, the random forest uses randomly selected features to construct multiple models. That's the reason why the random forest model is used versus a decision tree.

Our final model is the XGBoost model. We built the model using the 'XGBClassifier' algorithm provided by the xgboost package. We mentioned the 'max_depth' to be 4 and finally, fitted and stored the predicted values into the 'xgb_yhat'.

With that, we have successfully built our six types of classification models and interpreted the code for easy understanding. Our next step is to evaluate each of the models and find which is the most suitable one for our case.

Evaluation:

As stated before, this model is evaluated by using the evaluation metrics provided by the scikit-learn package. To find the best model for the given case, we have used the accuracy score metric, F1 score metric, and confusion matrix.

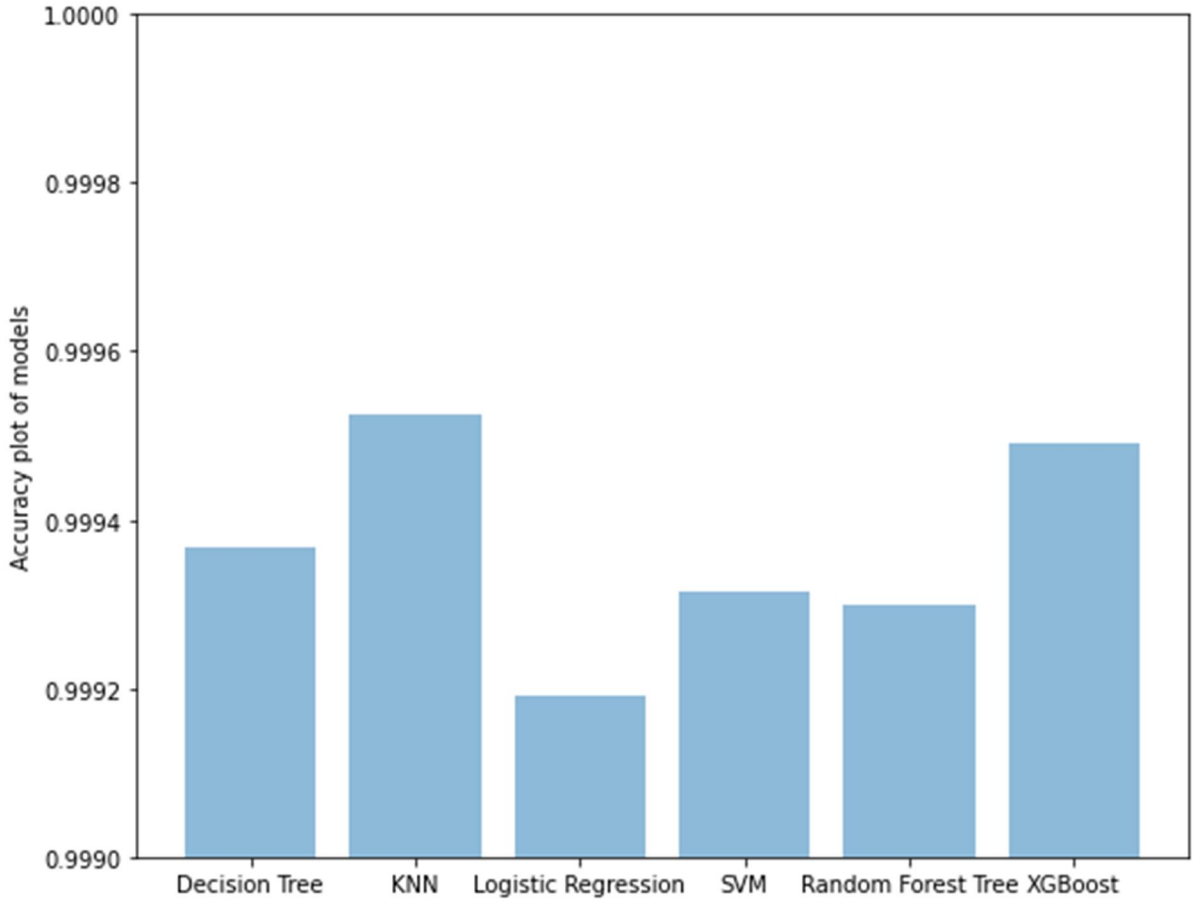
Output:

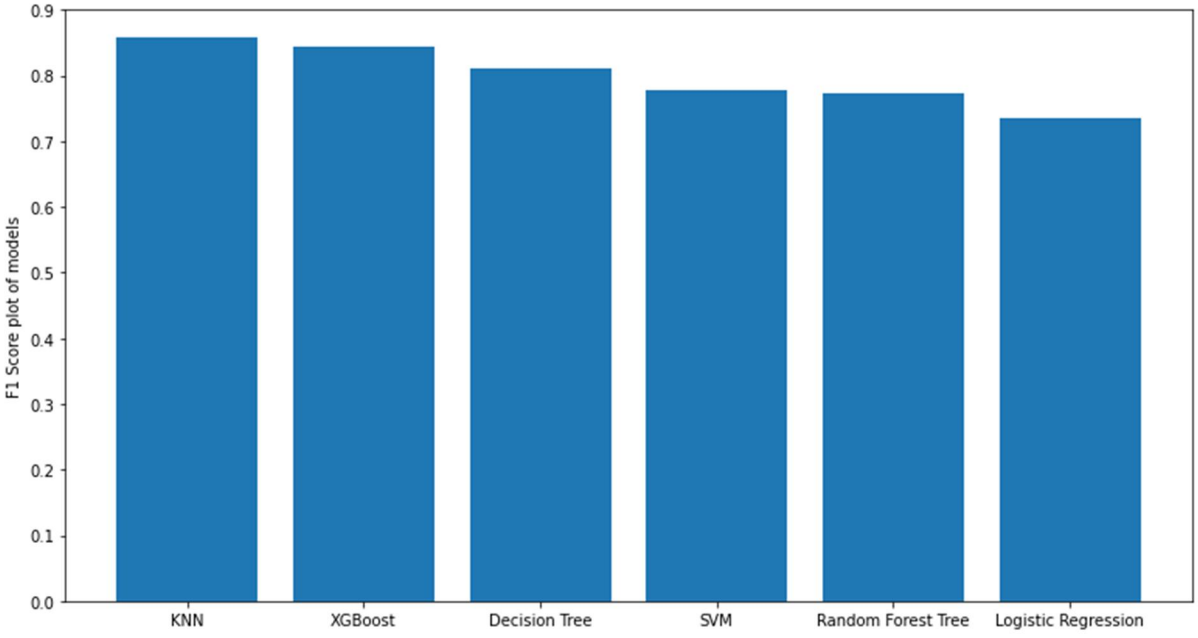
According to the accuracy score evaluation metric, the KNN model reveals to be the most accurate model and the Logistic regression model to be the least accurate model. But, after rounding up the results of each model, we got 0.99 (99% accurate) which is a very good score.

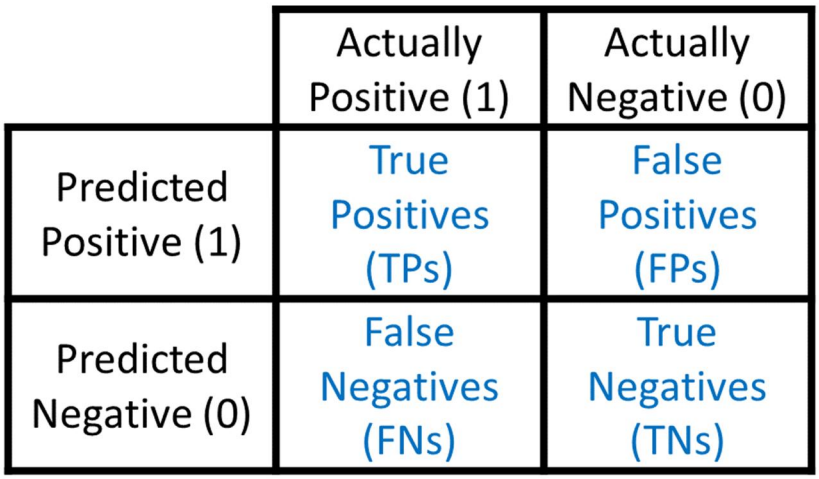
The F1 score metric evaluation shows the ranking of the models which is almost similar to the previous evaluation metric. On basis of the F1 score evaluation metric, the KNN model snatches the first place again and the Logistic regression model remains to be the least accurate model.

In comparison the confusion matrix of all the models, it is observed that the K-Nearest Neighbors model has performed a very good job of classifying the fraud transactions from the non-fraud transactions followed by the XGBoost model. Hence, we can conclude that the most appropriate model which can be used for our case is the K-Nearest Neighbors model and the model which can be neglected is the Logistic regression model.

Results:


1.0000
0.9998
0.9996
0.9994
0.9992
0.9990
Accuracy plot of models
Decision Tree
KNN
Logistic Regression
SVM
Random Forest Tree
XGBoost


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
F1 Score plot of models
KNN
XGBoost
Decision Tree
SVM
Random Forest Tree
Logistic Regression




```
Confusion matrix for Decision tree:
[[56849    12]
 [   24    77]]
Confusion matrix for Decision tree:
[[56854     7]
 [   20    81]]
Confusion matrix for Logistic Regression:
[[56852     9]
 [   37    64]]
Confusion matrix for Support Vector Machine:
[[56855     6]
 [   33    68]]
Confusion matrix for Random Forest Tree:
[[56854     7]
 [   33    68]]
Confusion matrix for XGBoost:
[[56854     7]
 [   22    79]]
```

Conclusion:

After a various type of procedure, we have successfully built six different types of classification models starting from the Decision tree model to the XGBoost model. After that, we have evaluated each of the models using the evaluation metrics and chose which model is most suitable for the given scenario. For this project, we have limited our model count to six but, there are many more models that one can explore.

References:

Kaggle Dataset. (2020). Credit Card Fraud Detection - Anonymized credit card transactions labeled as fraudulent or genuine. Retrieved from https://www.kaggle.com/mlg-ulb/creditcardfraud

Nikhil Adithyan. (2020). Credit Card Fraud Detection With Machine Learning in Python. Retrieved from https://medium.com/codex/credit-card-fraud-detection-with-machine-learning-in-python-ac7281991d87