**CREDIT CARD FRAUD DETECTION – FINAL PROJECT REPORT**

**Problem Statement:** With the world moving towards digitization almost all the services can be paid online, and the introduction of e-commerce websites has led to significant increase in number of transactions. Credit card transactions hold 23% percent share in the total transactions made and detecting fraud transactions is of great importance for any credit card company. The main problem in the credit card fraud detection is that less than 1% of the total transaction are actual frauds. This exceedingly high imbalance between fraud and legitimate transactions make it a difficult task. To solve this problem, we are going to create leverage Machin learning models to accurately classify fraud and non fraud transactions. we try implementing a deep neural network (DNN) models and two traditional machine learning models (Random Forest and Decision Tree) and calculate various metrics like accuracy, precision, recall and f1 score to check for the best performing model.

**Exploring Data and Data Insights:** For the project we are using the dataset from Kaggle credit cart fraud (<https://www.kaggle.com/mlg-ulb/creditcardfraud>). The dataset contains transactions made by credit cards of European cardholders in two days. The dataset has 492 fraud transactions out of 284,807 transactions. The dataset is **highly imbalanced 99.83%** of non fraud transaction and **0.17%** of fraud transactions that is if we use this dataset as the base for prediction models we end with lot of errors as the models will assume that most of the transactions are not fraud. We want our model to detect the hidden patterns to classify fraud and non-fraud transactions.

Once we open the zip file, we have creditcard.csv which hold the dataset. The creditcard.csv has 31 columns out of which Time, Amount and Class are only the known columns whereas the other columns (V1 - V28) have been scaled and PCA transformations (Dimensionality Reduction Technique). The main purpose of scaling and name of the column not being shown due to privacy reasons. Time and Amount are features were the columns which did not undergo PCA transformations. The Time column is in seconds and Amount column is in USD with a mean of approximately **88 USD** and mean of fraud transactions is **122 USD** approximately. The highest non-fraud transaction was **25691.16 USD** and highest fraud transaction was **2125.87 USD**. The below graph is transaction at different minutes of an hour with their values.

Chart, scatter chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

The graph on the top right is histogram of transactions of both fraud and non-fraud per hour. Other key insight we found from the data when we created a plot to check for all transaction/ hour we saw that the transactions were **increasing** as the day progressing. The case remains the same for non-Fraud Transactions but non fraud transactions we can see that maximum fraud transaction occurring between **2AM – 3AM**.

**Data Preprocessing:** As most of the columns in the dataset were already scaled there was not much preprocessing left to do. First, we checked if there are any null values in the dataset. As there were no null value, we proceed with scaling the two other features (Time and Amount) as the other columns have already been scaled. We have used robust scaler as it is prone to less prone to outliners. Below is the Distribution for Amount and Time before scaling.

Chart

Description automatically generated Chart, histogram

Description automatically generated

Then we split the dataset into two training and target. Training contains all the columns expect class whereas the target contains only class which has zero and one values for fraud and non-fraud transactions. Then we split the data into 80-20 for training and testing the dataset using train\_test\_split method. We use the output from this to train and test the model.

**Discussing Model Performance:** As stated in the problem statement we have successfully evaluated the different DNN models and traditional Machine Learning models. We have calculated different metrics like accuracy, precision score, recall score and f1 score for each of the implemented models. As we can see in the table below the accuracy of all the models is greater than 96%. In the case of imbalanced dataset accuracy cannot be considered as metrics to evaluate the performance of models. By definition accuracy is the sum of true positives and true negatives divided by the size of the dataset. As more than 95 percent of the data is non-fraud (true negatives) our models will smartly forecast them as negatives. Our aim to detect to detect true positives (fraud transactions) hence we need to use other metrics like precision and recall.

First, we implemented decision tree by importing DecisionTreeClassifier from sklearn.tree we split the data into 80-20 percent which will used for training and testing, respectively. We pass X\_train, y\_train to fit (train) the model. We predict the model performance by passing the X\_test to predict function of decision tree classifier. Decision Tree classifier produced a precision score of 75%, recall score of 75.6% and f1 score of 75.7%. Decision tree was the worst performing model in all the implemented models. Second, we implemented RandomForestClassifier from sklearn.ensemble. we created the RF model with 100 estimators and passed X\_train and y\_train to fit method to train the model. After the model was trained, we passed X\_test to predict function of RF and the model achieved a precision score of 93.97%, recall score of 78% and f1 score of 85.2%. Random forest model performed better than decision tree, but it performed very poorly when compared to under sampled and oversampled DNN model

Chart

Description automatically generated

Chart

Description automatically generated

As DNN model was our best performing model we are going discuss the model in detail in the next section as this section only focusses on the model performance. The model used was all the DNN cases was the same the input data which was passed was different in each case. In case of Regular DNN model the precision, score is 82.9%, recall score is 79% and f1 score is 81%. A lot of fraud transactions were misclassified as fraud. As there was a lot of scope for improvement, we decided to try out DNN model with UnderSampling. The data for fraud and non fraud was selected randomly but equally (492 fraud and 492 non-fraud transactions). With the under sampled data we trained the same DNN model again. The results showed a precision score of 96.8%, recall score of 95.8% and f1 score of 96.3% which were an improvement of the regular DNN model. In the last DNN model we have implemented SMOTE (Synthetic Minority Over-Sampling). It works by duplicating the fraud transaction data such that we have equal fraud and non-fraud transactions (284315 of each class). After retraining the model with the over sampled data DNN model with SMOTE achieved precision score of 99.7%, recall score of 99.9% and f1 score of 99.8%. This model has provided highest score compared to all other models implemented as part of this project.

Table

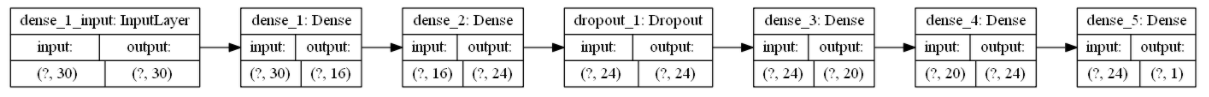
Description automatically generated

We have also implemented ROC Curve (Receiver Operating characteristic Curve) which shows the performance of different classification models at different thresholds. We do this by passing true positive rate (TPR), false positive rate (FPR) and auc score of different classification models respectively by passing them to a custom-built function to plot the graph. From the roc curve we can see that the DNN with SMOTE has the best area under the curve score of 99.8% followed by DNN with under sampling with a score of 99.6%. The least performing model was Decision tree with a auc score of 88.1%. We can clearly observe that the DNN models performed much better than the traditional machine learning algorithms.

Diagram

Description automatically generated

**Best Model Implementation and Graphs:** For this Project we have implemented 3 DNN models (Regular DNN model, DNN model with under sampling, DNN model with SMOTE). A keras Sequential model was created with five dense layers and starting input dimensions of 30. The activation function used is relu for all layers except the last one where sigmoid was used. A dropout layer with dropout rate of 50% was used after the second layer. To compile the model Adam optimizer was used, a binary cross entropy as loss function and metrics used was accuracy. The above-mentioned model was used in all the cases and only the data being passed has been altered.



For the **regular DNN model** we have passed the whole imbalanced dataset and then we have split the data into 80-20 for training and testing using the test\_train\_split method. Below are the results obtained from the confusion matrix. This model had a precision score of 82.9% and recall score of 79.5%, hence we decided to optimise the model performance by under sampling.

Chart, bar chart

Description automatically generatedChart

Description automatically generated

For the **DNN model with under sampling** we have passed the have randomly selected equal number of fraud and non-fraud transactions (492 of both classes) then we have split the data into 80-20 for training and testing using the test\_train\_split method. Below are the results obtained from the confusion matrix. This model has precision score of 96.8% and recall score of 95.8%, hence we decided to further optimise the model performance by SMOTE (oversampling).

Chart, bar chart

Description automatically generatedChart

Description automatically generated

For the **DNN model with SMOTE** we have over sampled the dataset by creating the duplicates of non-fraud cases such that equal number of fraud and non-fraud transactions are present in the dataset (284315 of both classes). This oversampling will not be a problem as the duplicate fraud transactions created having similar features as the regular fraud transaction. Then we have split the data into 80-20 for training and testing using the test\_train\_split method. Below are the results obtained from the confusion matrix. This model had a precision score of 99.7% and recall score of 99.9%, hence this was the most optimal model out of the 3 DNN models.

Chart, bar chart

Description automatically generatedChart, treemap chart

Description automatically generated

We have created a validation dataset to for plotting graphs like **training and validation accuracy** and **training and validation loss** which can help us understand why this model is the best model.

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

From the graph training accuracy vs validation accuracy, we can observe that validation accuracy is always more at different epochs. Validation accuracy started at 96.61% and ended at 99.84% whereas Training accuracy started at 99.02% and ended at 99.7%. Which is a sign of good model performance as the model performs better unseen data.

From the graph training loss vs validation loss, we can observe that the model has less validation loss compared to training loss. Validation loss started at 1.33% and ended at 0.77% whereas training loss started 2.93% and ended at 0.99%. Which is another good sign that the model had less loss and performed better in case of unseen data.

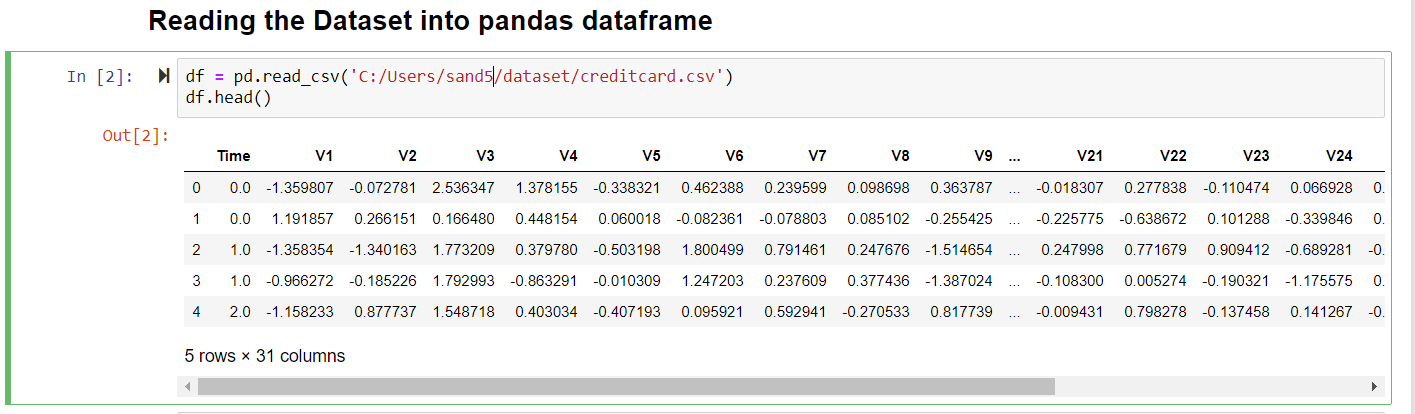
**LINK TO THE DATASET:**

<https://www.kaggle.com/mlg-ulb/creditcardfraud>

**CODE:**

**Graphical user interface, text, application

Description automatically generated**

****

**Graphical user interface, text, application, Excel

Description automatically generated**

**A picture containing chart

Description automatically generated**

**Text, application

Description automatically generated**

**Chart, histogram

Description automatically generated**

**Chart, histogram

Description automatically generated**

**Chart, histogram

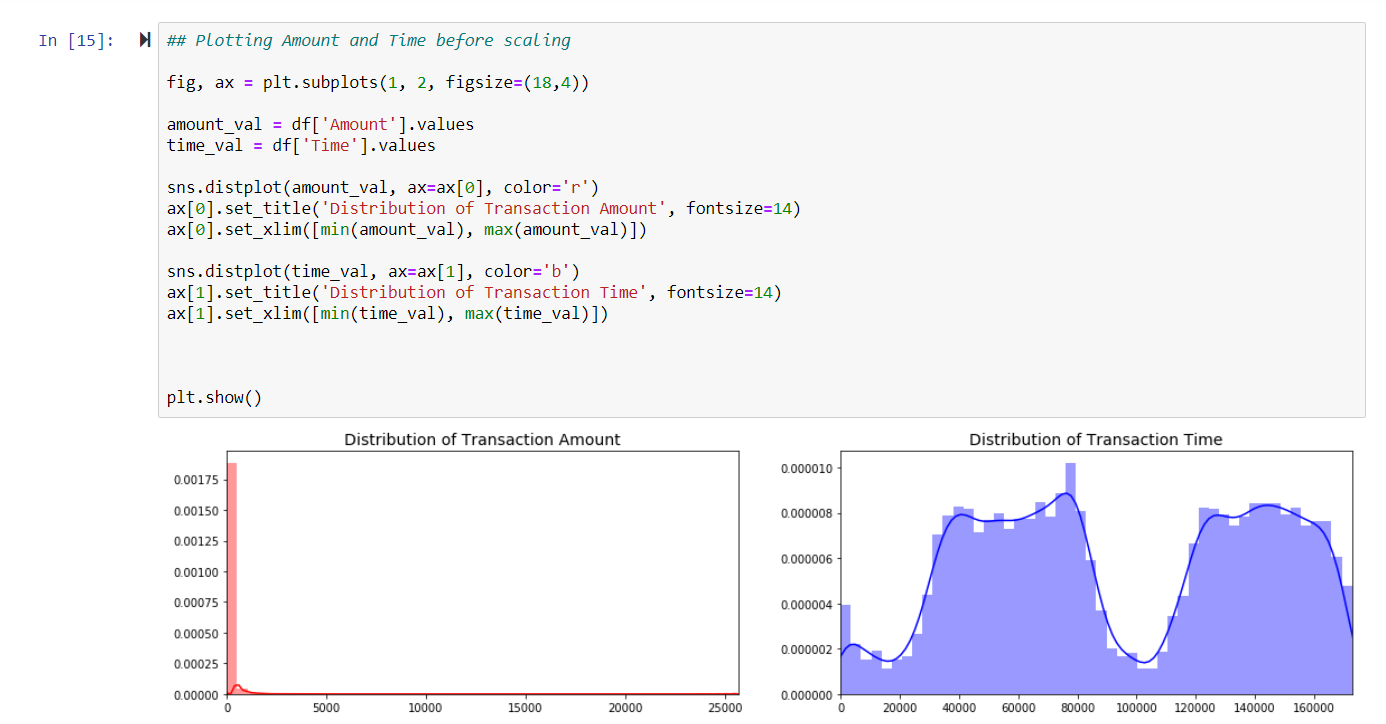
Description automatically generated**

**Chart, histogram

Description automatically generated**

**Chart, scatter chart

Description automatically generated**

****

**A picture containing text

Description automatically generated**

**Graphical user interface, text

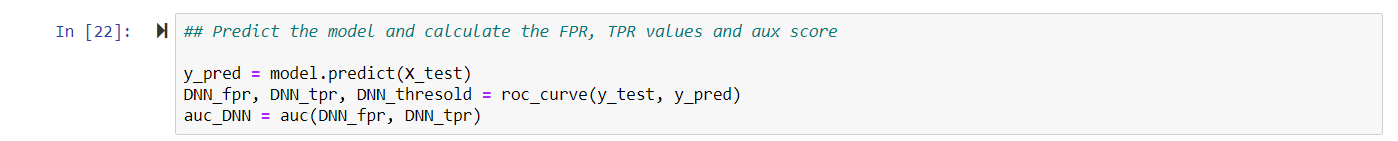
Description automatically generated**

**A picture containing graphical user interface

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

****

**Text

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**A picture containing text

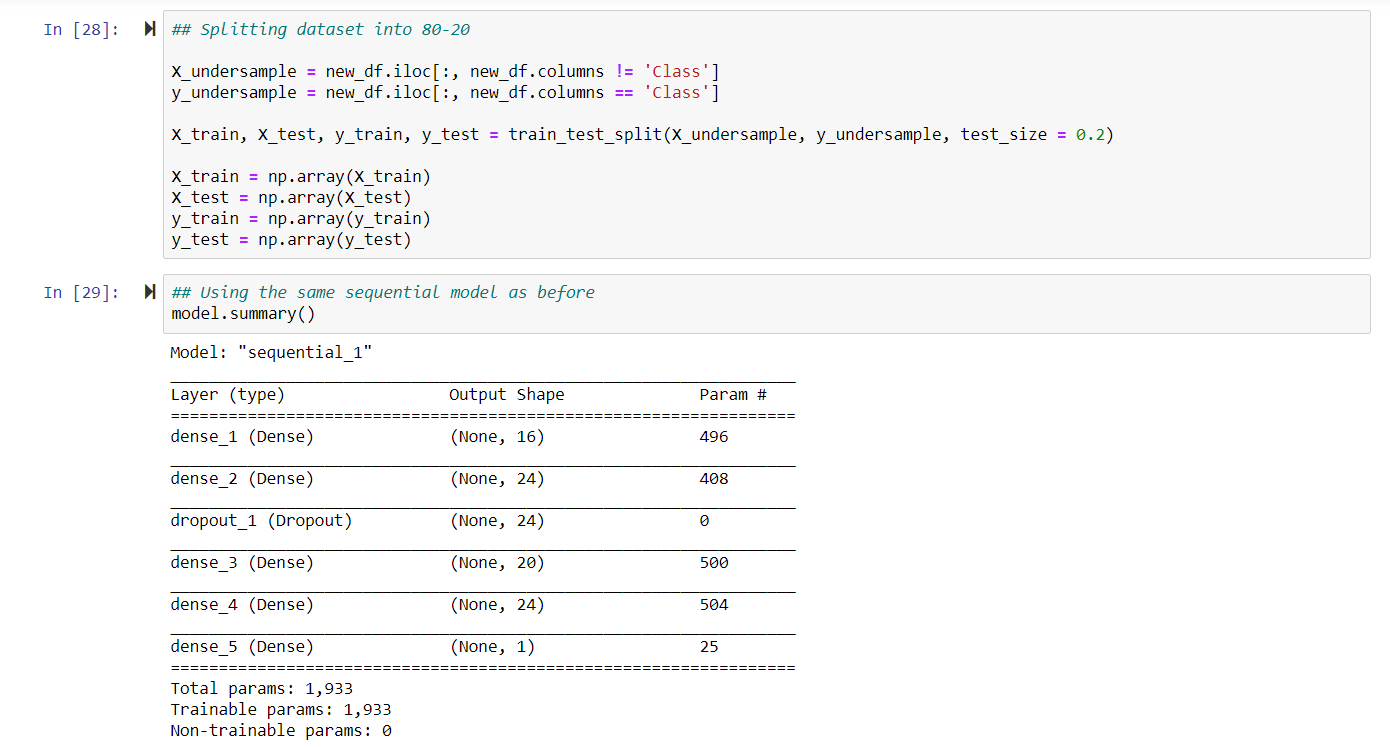
Description automatically generated**

**A picture containing calendar

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

****

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**A picture containing chart

Description automatically generated**

**Table

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Text

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Text

Description automatically generated**

**Graphical user interface

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Text

Description automatically generated**

**A picture containing scatter chart

Description automatically generated**

**Diagram

Description automatically generated**

**A picture containing text

Description automatically generated**

**GITHUB LINK:**