Configuration Manual

MSc Research Project
Msc in Data Analytics

Ketan R.Chalwadi
Student ID: 19222840

School of Computing
National College of Ireland

Supervisor: Prof.Hicham Rifai

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Ketan R.Chalwadi
Student ID:	19222840
Programme:	Msc in Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Prof.Hicham Rifai
Submission Due Date:	16/08/2021
Project Title:	Configuration Manual
Word Count:	318
Page Count:	14

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Ketan R.Chalwadi
Date:	23rd September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Ketan R.Chalwadi
19222840

1 Overview

The main objective of the present research work is to check to what Classification of CRedit Card Fraudulent Transactions using Neural Network and Oversampling Technique can accurately classify credit card frauds on the basis of transaction data that has taken from European Card Holder.Agrawal et al. (2015)Neural Networks with multi layer structure can classify target variables more accurately.

In this research, XGBoost, ADABoost,Random Forest and Decision Tree Model are also used for comparison Purpose.Entire Project has been implemented using python libraries.This configuration manual is divided into four individual sections: Overview, System Specification and requirements, Installation Process, Implementation and Evaluation of results.

2 System Requirements

Processor : Intel® Core™ i5-10210U CPU @ 1.60GHz × 8
Memory(RAM) Installed : 8 GB DDR4 2667 MHz
System Type : Ubuntu 18.04, 64 Bit Operating System with x64-based processor
Storage: 500 GB SSD
GPU : 4 GB, Intel(R) UHD Graphics

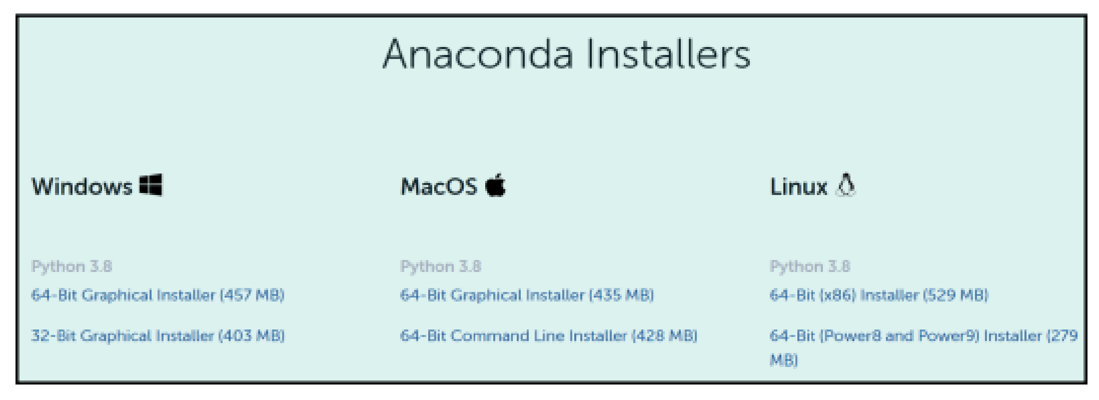
2.1 Software Requirements

This research work requires applications such as Anaconda Navigator(Anaconda3), jupyter notebook,Microsoft Installed.

3 System Requirements

3.1 Installing softwares

Anaconda Python package for the Ubuntu OS platform has to be downloaded and installed.Fig 1 shows the version of anaconda package to be downloaded. Fig 2 shows the Matplotlib version to be installed. Fig 3 installing matplotlib in ubuntu terminal



Anaconda Installers

Windows

Python 3.8

64-Bit Graphical Installer (457 MB)

32-Bit Graphical Installer (403 MB)

MacOS

Python 3.8

64-Bit Graphical Installer (435 MB)

64-Bit Command Line Installer (428 MB)

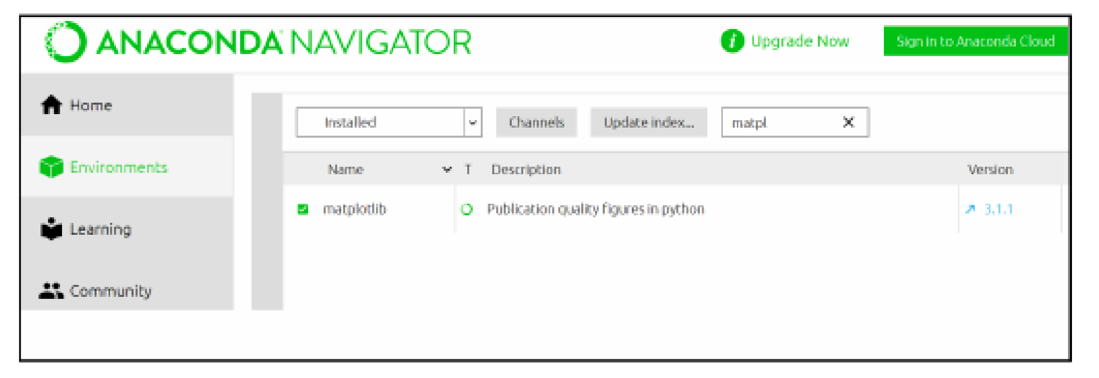
Linux

Python 3.8

64-Bit (x86) Installer (529 MB)

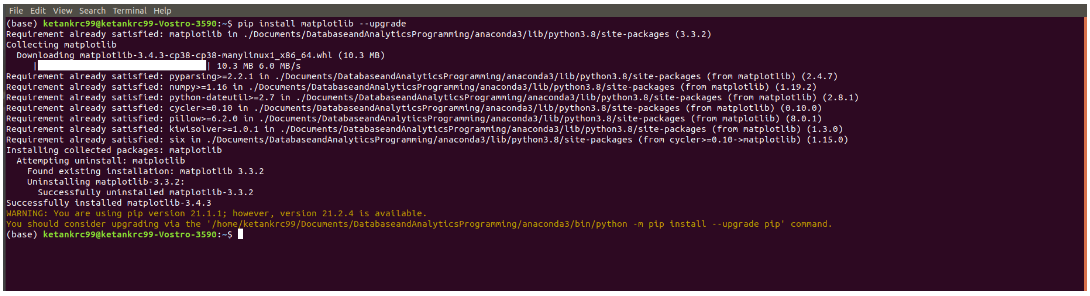
64-Bit (Power8 and Power9) Installer (279 MB)

Figure 1: Anaconda3 Download



	Name	T	Description	Version
☑	matplotlib	○	Publication quality figures in python	↗ 3.1.1

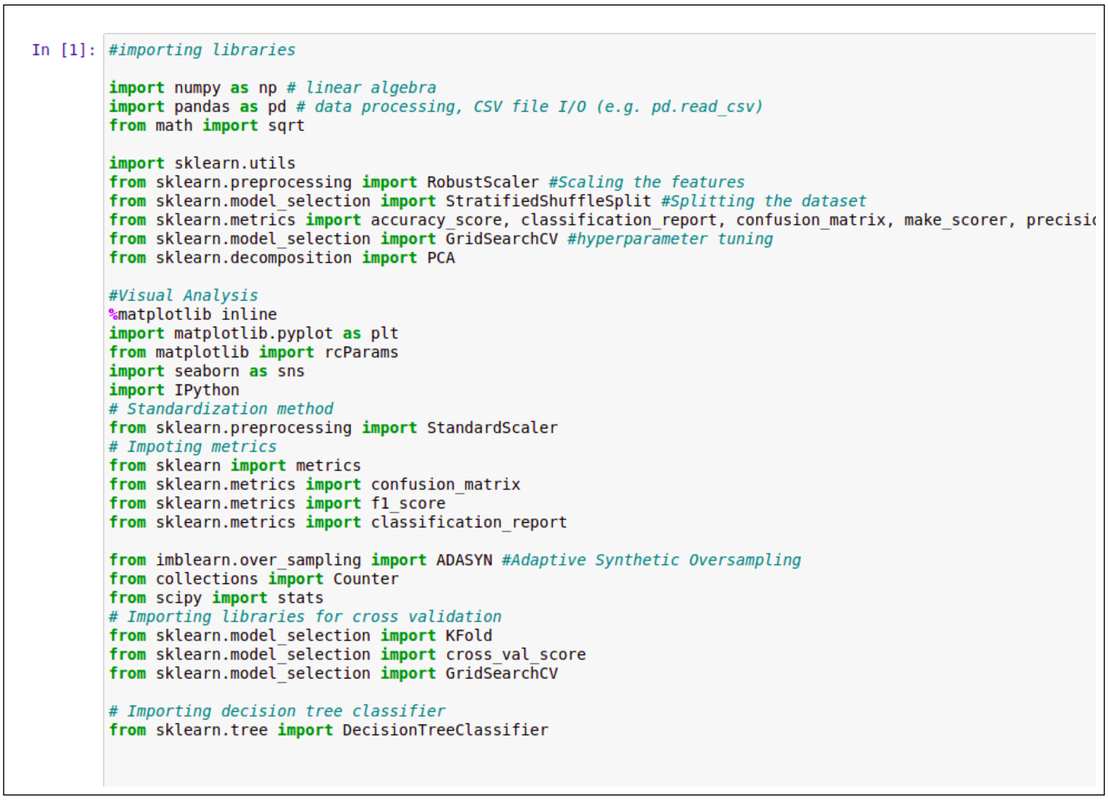
Figure 2: Matplotlib Installation process



```
(base) ketankrc99@ketankrc99-Vostro-3590:~$ pip install matplotlib --upgrade
Requirement already satisfied: matplotlib in ./Documents/DatabaseandAnalyticsProgramming/anaconda3/lib/python3.8/site-packages (3.3.2)
Collecting matplotlib
  Downloading matplotlib-3.4.3-cp38-cp38-manylinux1_x86_64.whl (10.3 MB)
     |████████████████████████████████| 10.3 MB 6.0 MB/s
Requirement already satisfied: pyparsing>=2.2.1 in ./Documents/DatabaseandAnalyticsProgramming/anaconda3/lib/python3.8/site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: numpy>=1.16 in ./Documents/DatabaseandAnalyticsProgramming/anaconda3/lib/python3.8/site-packages (from matplotlib) (1.19.2)
Requirement already satisfied: python-dateutil>=2.7 in ./Documents/DatabaseandAnalyticsProgramming/anaconda3/lib/python3.8/site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: cycler>=0.10 in ./Documents/DatabaseandAnalyticsProgramming/anaconda3/lib/python3.8/site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in ./Documents/DatabaseandAnalyticsProgramming/anaconda3/lib/python3.8/site-packages (from matplotlib) (8.0.1)
Requirement already satisfied: kiwisolver>=1.0.1 in ./Documents/DatabaseandAnalyticsProgramming/anaconda3/lib/python3.8/site-packages (from matplotlib) (1.3.0)
Requirement already satisfied: six in ./Documents/DatabaseandAnalyticsProgramming/anaconda3/lib/python3.8/site-packages (from cycler>=0.10->matplotlib) (1.15.0)
Installing collected packages: matplotlib
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.3.2
    Uninstalling matplotlib-3.3.2:
      Successfully uninstalled matplotlib-3.3.2
Successfully installed matplotlib-3.4.3
WARNING: You are using pip version 21.1.1; however, version 21.2.4 is available.
You should consider upgrading via the '/home/ketankrc99/Documents/DatabaseandAnalyticsProgramming/anaconda3/bin/python -m pip install --upgrade pip' command.
(base) ketankrc99@ketankrc99-Vostro-3590:~$ 
```

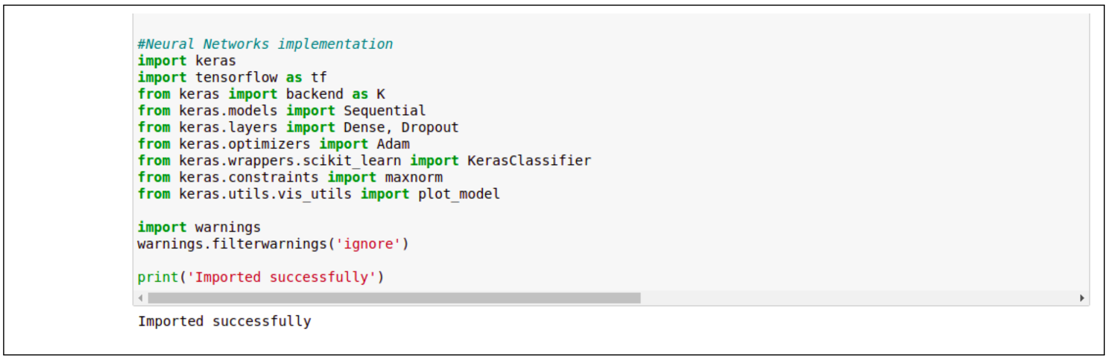
Figure 3: Matplotlib Installed

3.2 Installing python packages/libraries



```
In [1]: #importing libraries

        import numpy as np # linear algebra
        import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
        from math import sqrt

        import sklearn.utils
        from sklearn.preprocessing import RobustScaler #Scaling the features
        from sklearn.model_selection import StratifiedShuffleSplit #Splitting the dataset
        from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, make_scorer, precisi
        from sklearn.model_selection import GridSearchCV #hyperparameter tuning
        from sklearn.decomposition import PCA

        #Visual Analysis
        %matplotlib inline
        import matplotlib.pyplot as plt
        from matplotlib import rcParams
        import seaborn as sns
        import IPython
        # Standardization method
        from sklearn.preprocessing import StandardScaler
        # Impoting metrics
        from sklearn import metrics
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import f1_score
        from sklearn.metrics import classification_report

        from imblearn.over_sampling import ADASYN #Adaptive Synthetic Oversampling
        from collections import Counter
        from scipy import stats
        # Importing libraries for cross validation
        from sklearn.model_selection import KFold
        from sklearn.model_selection import cross_val_score
        from sklearn.model_selection import GridSearchCV

        # Importing decision tree classifier
        from sklearn.tree import DecisionTreeClassifier
```

Figure 4: Python Library and Packages



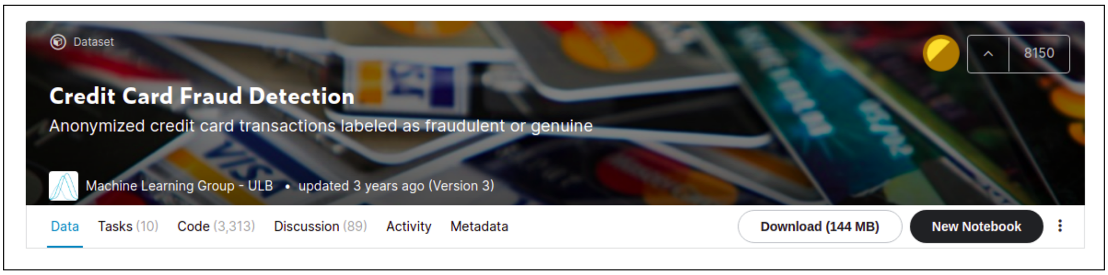
```
#Neural Networks implementation
import keras
import tensorflow as tf
from keras import backend as K
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import Adam
from keras.wrappers.scikit_learn import KerasClassifier
from keras.constraints import maxnorm
from keras.utils.vis_utils import plot_model

import warnings
warnings.filterwarnings('ignore')

print('Imported successfully')
```

```
Imported successfully
```

Figure 5: Additional Python Library and Packages

4 Implementation Flow and Performance Evaluation of Model

4.1 Dataset Selection



Dataset

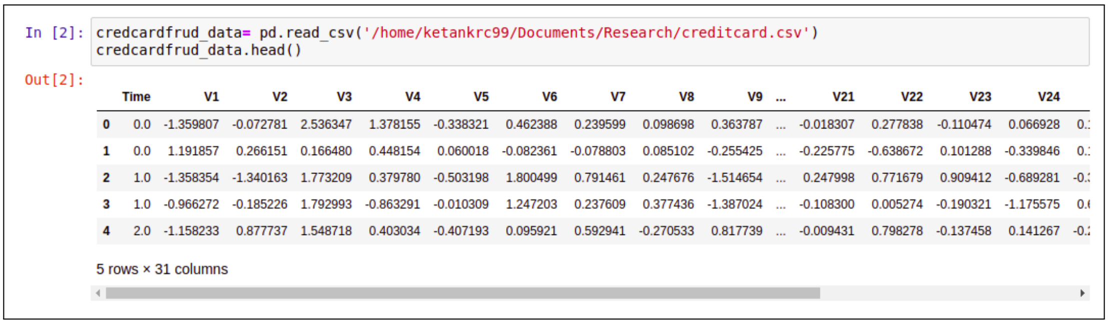
Credit Card Fraud Detection

Anonymized credit card transactions labeled as fraudulent or genuine

Machine Learning Group - ULB • updated 3 years ago (Version 3)

Figure 6: Dataset

4.2 Loading of Dataset



```
In [2]: credcardfrud_data= pd.read_csv('/home/ketankrc99/Documents/Research/creditcard.csv')
        credcardfrud_data.head()
```

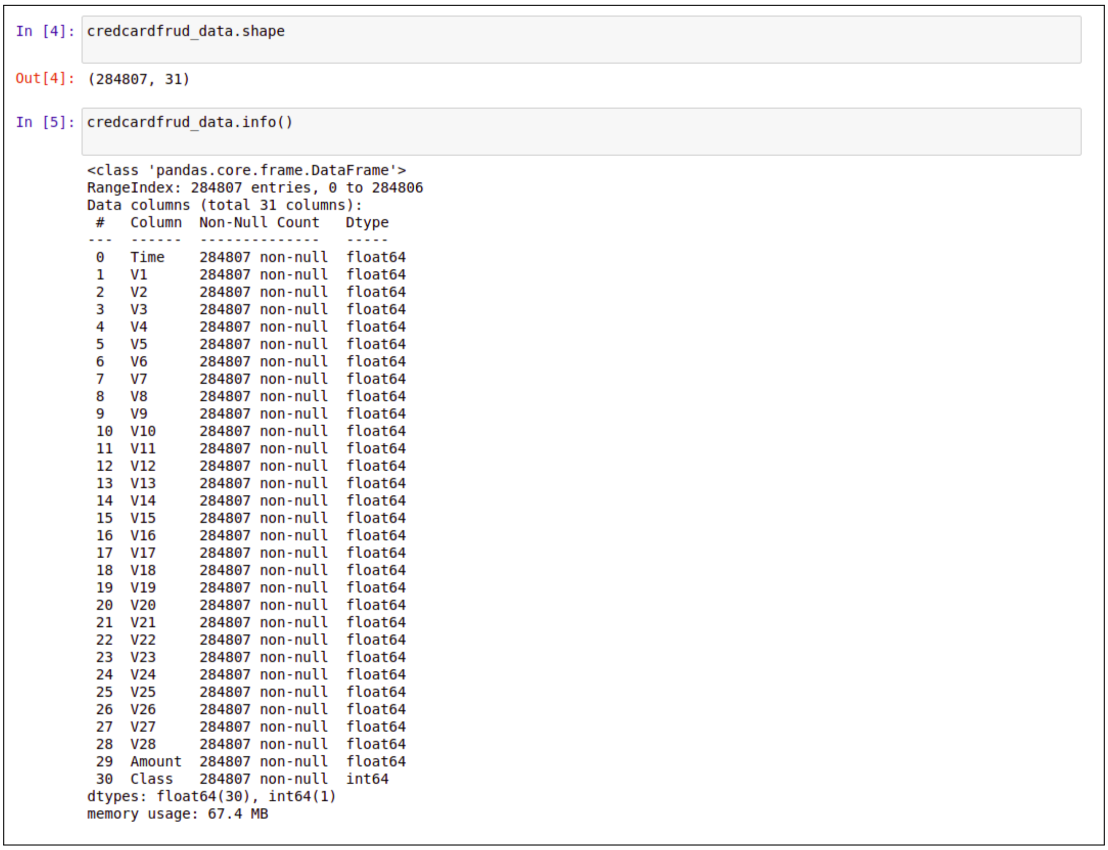
Out[2]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	
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.
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.
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.
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.
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.

5 rows × 31 columns

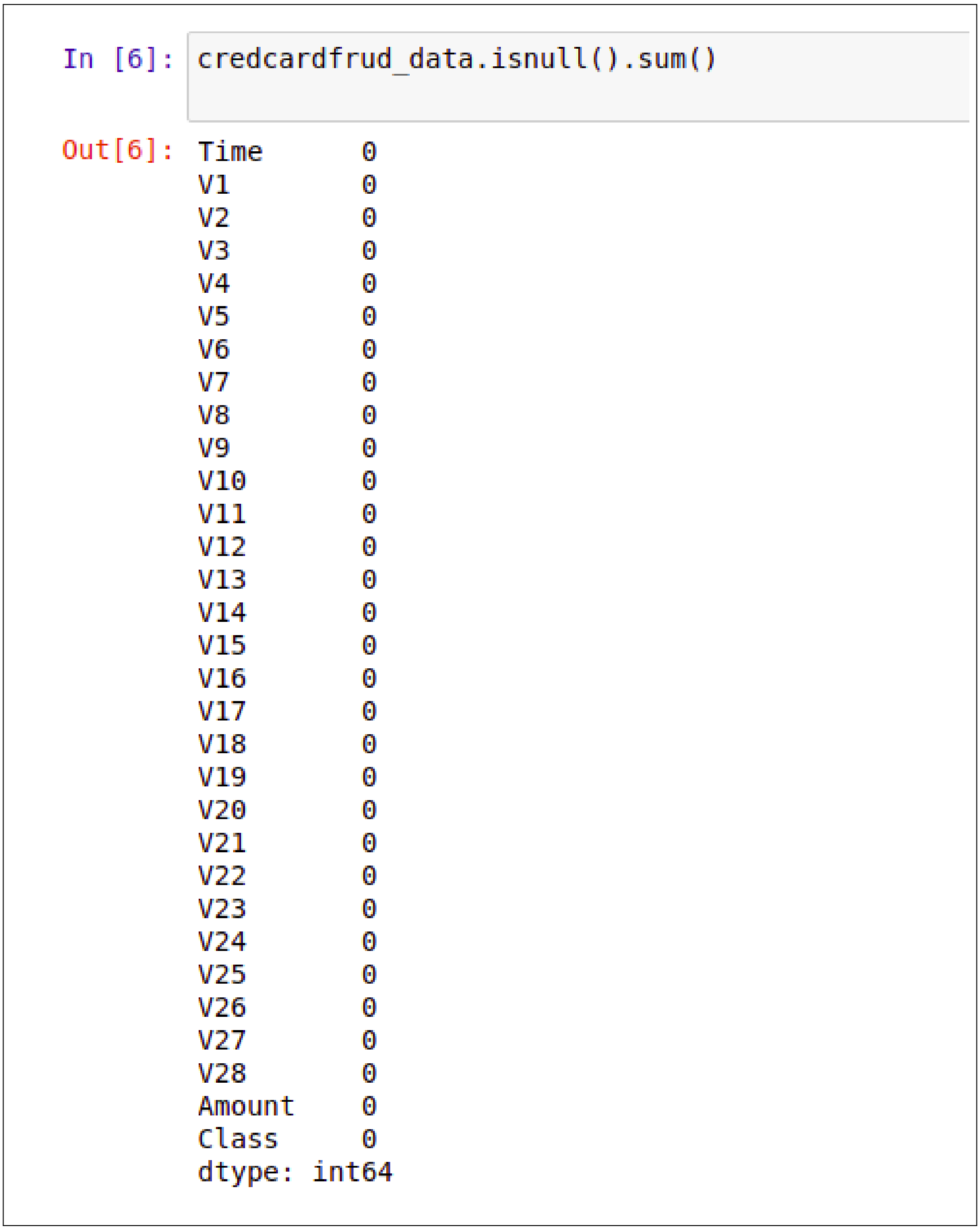
Figure 7: Dataset Loading

4.3 Data Preprocessing



```
In [4]: credcardfrud_data.shape

Out[4]: (284807, 31)

In [5]: credcardfrud_data.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 284807 entries, 0 to 284806
        Data columns (total 31 columns):
         #   Column  Non-Null Count   Dtype  
        ---  ------  --------------   -----  
         0   Time    284807 non-null  float64
         1   V1      284807 non-null  float64
         2   V2      284807 non-null  float64
         3   V3      284807 non-null  float64
         4   V4      284807 non-null  float64
         5   V5      284807 non-null  float64
         6   V6      284807 non-null  float64
         7   V7      284807 non-null  float64
         8   V8      284807 non-null  float64
         9   V9      284807 non-null  float64
         10  V10     284807 non-null  float64
         11  V11     284807 non-null  float64
         12  V12     284807 non-null  float64
         13  V13     284807 non-null  float64
         14  V14     284807 non-null  float64
         15  V15     284807 non-null  float64
         16  V16     284807 non-null  float64
         17  V17     284807 non-null  float64
         18  V18     284807 non-null  float64
         19  V19     284807 non-null  float64
         20  V20     284807 non-null  float64
         21  V21     284807 non-null  float64
         22  V22     284807 non-null  float64
         23  V23     284807 non-null  float64
         24  V24     284807 non-null  float64
         25  V25     284807 non-null  float64
         26  V26     284807 non-null  float64
         27  V27     284807 non-null  float64
         28  V28     284807 non-null  float64
         29  Amount  284807 non-null  float64
         30  Class   284807 non-null  int64  
        dtypes: float64(30), int64(1)
        memory usage: 67.4 MB
```

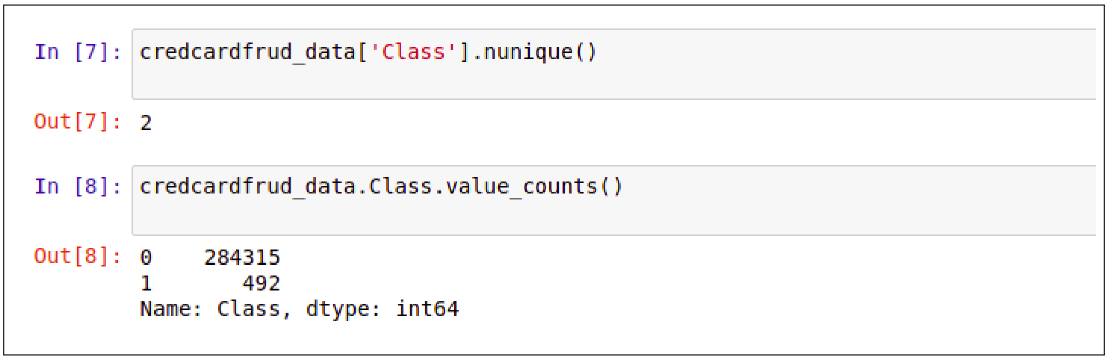
Figure 8: Dataset Info



```
In [6]: credcardfrud_data.isnull().sum()
```

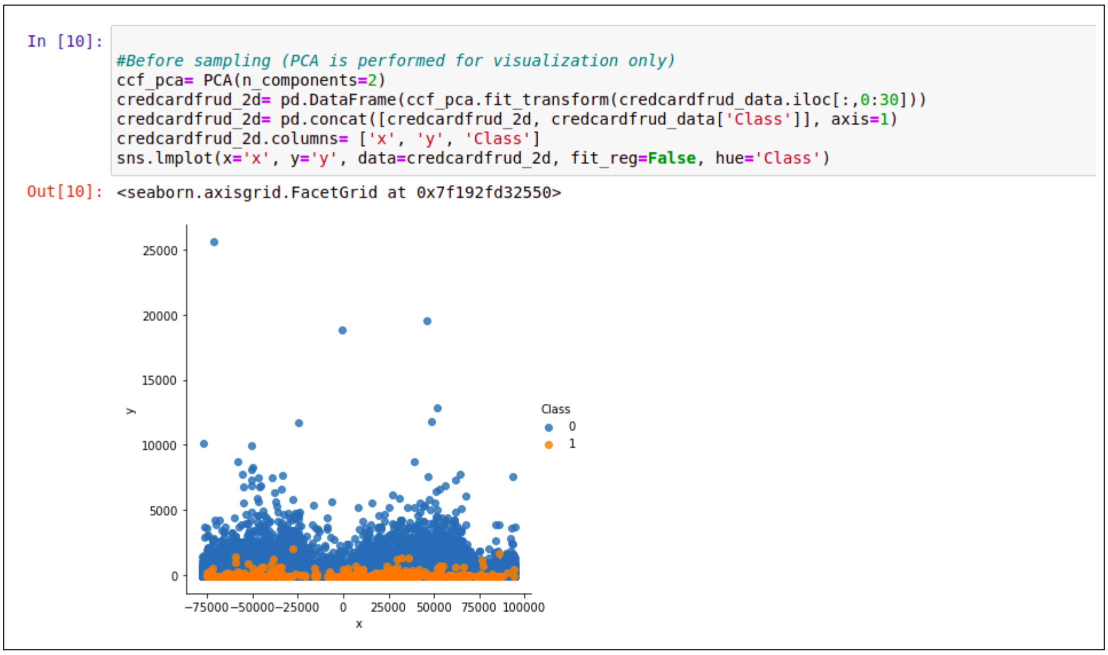
```
Out[6]: Time      0
        V1        0
        V2        0
        V3        0
        V4        0
        V5        0
        V6        0
        V7        0
        V8        0
        V9        0
        V10       0
        V11       0
        V12       0
        V13       0
        V14       0
        V15       0
        V16       0
        V17       0
        V18       0
        V19       0
        V20       0
        V21       0
        V22       0
        V23       0
        V24       0
        V25       0
        V26       0
        V27       0
        V28       0
        Amount    0
        Class     0
        dtype: int64
```

Figure 9: Checking for Null Values



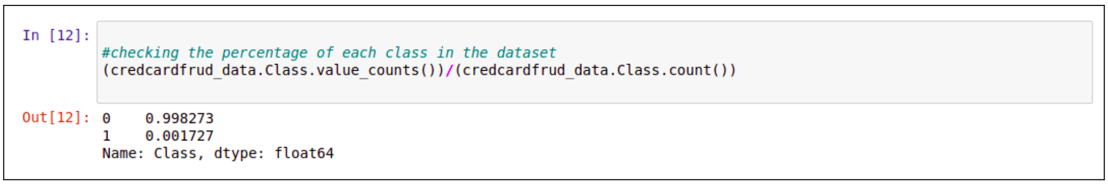
```
In [7]: credcardfrud_data['Class'].nunique()

Out[7]: 2

In [8]: credcardfrud_data.Class.value_counts()

Out[8]: 0    284315
        1       492
        Name: Class, dtype: int64
```

Figure 10: Different Class



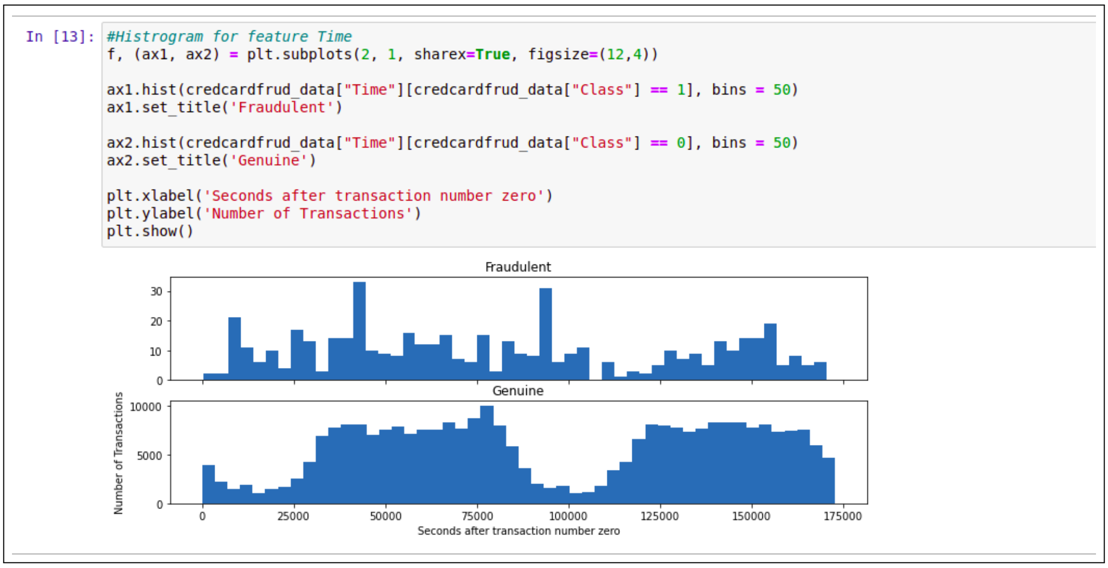
```
In [10]:
#Before sampling (PCA is performed for visualization only)
ccf_pca= PCA(n_components=2)
credcardfrud_2d= pd.DataFrame(ccf_pca.fit_transform(credcardfrud_data.iloc[:,0:30]))
credcardfrud_2d= pd.concat([credcardfrud_2d, credcardfrud_data['Class']], axis=1)
credcardfrud_2d.columns= ['x', 'y', 'Class']
sns.lmplot(x='x', y='y', data=credcardfrud_2d, fit_reg=False, hue='Class')

Out[10]: <seaborn.axisgrid.FacetGrid at 0x7f192fd32550>
```

Figure 11: PCA transformation



```
In [12]:
#checking the percentage of each class in the dataset
(credcardfrud_data.Class.value_counts())/(credcardfrud_data.Class.count())

Out[12]: 0    0.998273
         1    0.001727
         Name: Class, dtype: float64
```

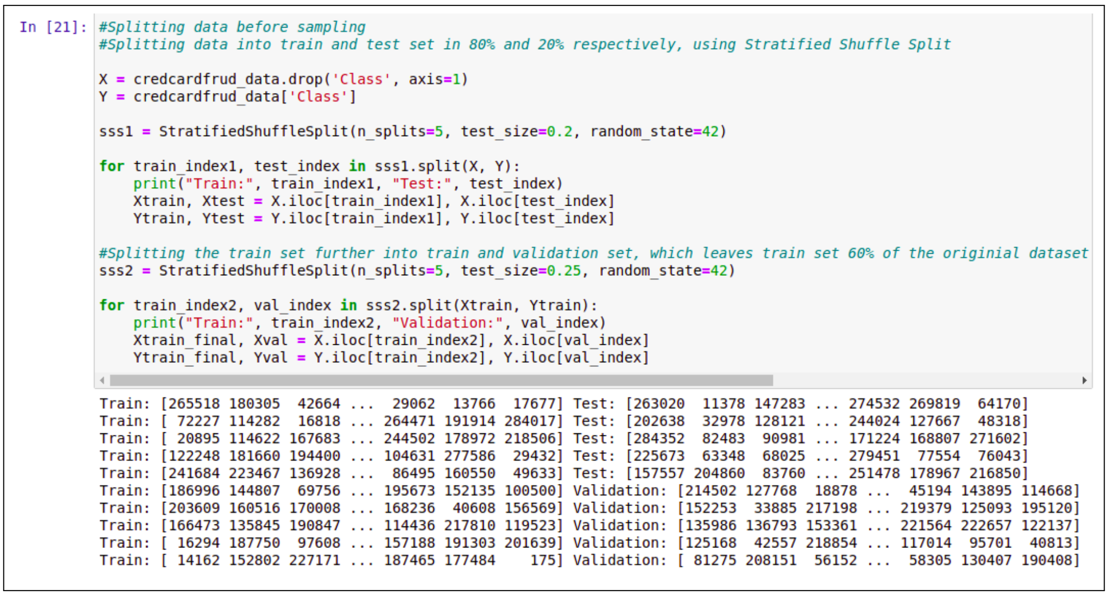
Figure 12: Percentage of each Class



In [13]:

```
#Histrogram for feature Time
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(12,4))

ax1.hist(credcardfrud_data["Time"][credcardfrud_data["Class"] == 1], bins = 50)
ax1.set_title('Fraudulent')

ax2.hist(credcardfrud_data["Time"][credcardfrud_data["Class"] == 0], bins = 50)
ax2.set_title('Genuine')

plt.xlabel('Seconds after transaction number zero')
plt.ylabel('Number of Transactions')
plt.show()
```

Figure 13: Histogram of Fraudulent and Genuine Transactions

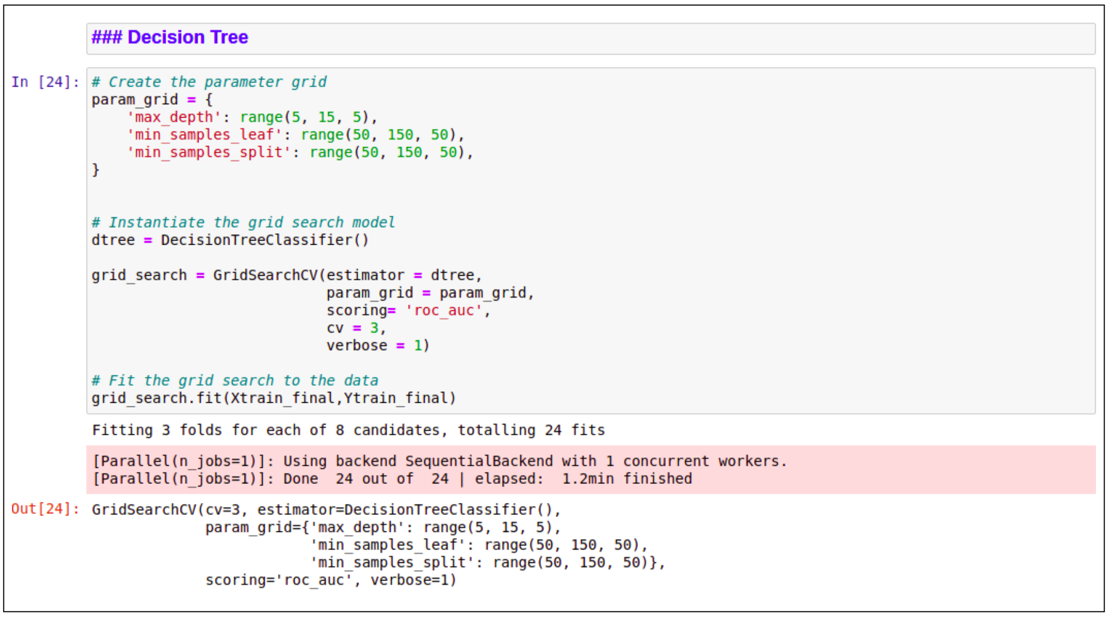


In [21]:

```
#Splitting data before sampling
#Splitting data into train and test set in 80% and 20% respectively, using Stratified Shuffle Split

X = credcardfrud_data.drop('Class', axis=1)
Y = credcardfrud_data['Class']

sss1 = StratifiedShuffleSplit(n_splits=5, test_size=0.2, random_state=42)

for train_index1, test_index in sss1.split(X, Y):
    print("Train:", train_index1, "Test:", test_index)
    Xtrain, Xtest = X.iloc[train_index1], X.iloc[test_index]
    Ytrain, Ytest = Y.iloc[train_index1], Y.iloc[test_index]

#Splitting the train set further into train and validation set, which leaves train set 60% of the originial dataset
sss2 = StratifiedShuffleSplit(n_splits=5, test_size=0.25, random_state=42)

for train_index2, val_index in sss2.split(Xtrain, Ytrain):
    print("Train:", train_index2, "Validation:", val_index)
    Xtrain_final, Xval = X.iloc[train_index2], X.iloc[val_index]
    Ytrain_final, Yval = Y.iloc[train_index2], Y.iloc[val_index]
```

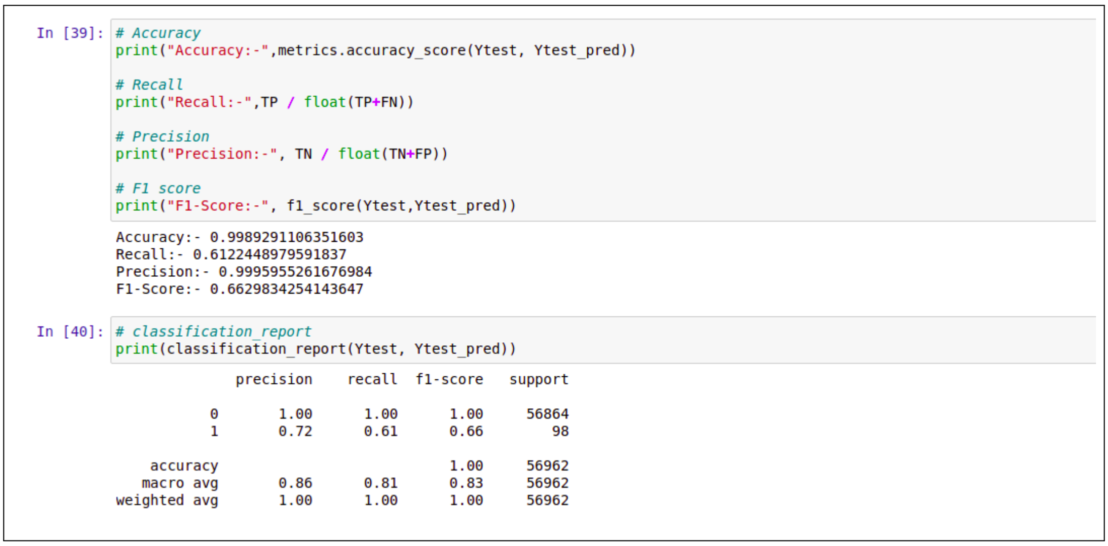
```
Train: [265518 180305  42664 ...  29062  13766  17677] Test: [263020  11378 147283 ... 274532 269819  64170]
Train: [ 72227 114282  16818 ... 264471 191914 284017] Test: [202638  32978 128121 ... 244024 127667  48318]
Train: [ 20895 114622 167683 ... 244502 178972 218506] Test: [284352  82483  90981 ... 171224 168807 271602]
Train: [122248 181660 194400 ... 104631 277586  29432] Test: [225673  63348  68025 ... 279451  77554  76043]
Train: [241684 223467 136928 ...  86495 160550  49633] Test: [157557 204860  83760 ... 251478 178967 216850]
Train: [186996 144807  69756 ... 195673 152135 100500] Validation: [214502 127768  18878 ...  45194 143895 114668]
Train: [203609 160516 170008 ... 168236  40608 156569] Validation: [152253  33885 217198 ... 219379 125093 195120]
Train: [166473 135845 190847 ... 114436 217810 119523] Validation: [135986 136793 153361 ... 221564 222657 122137]
Train: [ 16294 187750  97608 ... 157188 191303 201639] Validation: [125168  42557 218854 ... 117014  95701  40813]
Train: [ 14162 152802 227171 ... 187465 177484    175] Validation: [ 81275 208151  56152 ...  58305 130407 190408]
```

Figure 14: Splitting of Data



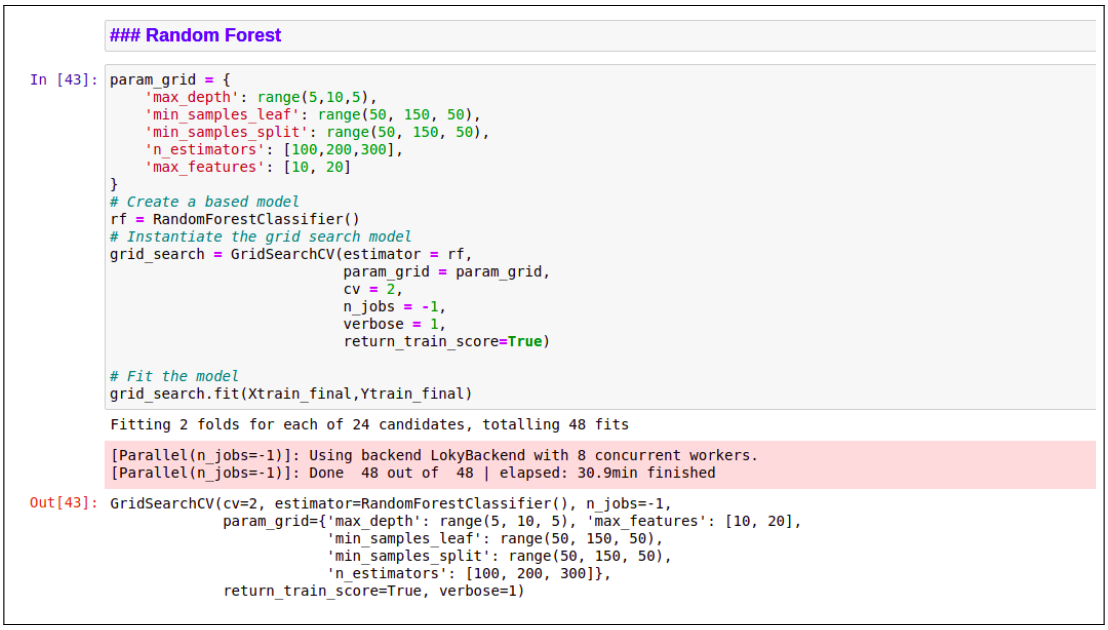
```
### Decision Tree

In [24]: # Create the parameter grid
         param_grid = {
             'max_depth': range(5, 15, 5),
             'min_samples_leaf': range(50, 150, 50),
             'min_samples_split': range(50, 150, 50),
         }


         # Instantiate the grid search model
         dtree = DecisionTreeClassifier()

         grid_search = GridSearchCV(estimator = dtree,
                                    param_grid = param_grid,
                                    scoring= 'roc_auc',
                                    cv = 3,
                                    verbose = 1)

         # Fit the grid search to the data
         grid_search.fit(Xtrain_final,Ytrain_final)

         Fitting 3 folds for each of 8 candidates, totalling 24 fits

         [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
         [Parallel(n_jobs=1)]: Done  24 out of  24 | elapsed:  1.2min finished

Out[24]: GridSearchCV(cv=3, estimator=DecisionTreeClassifier(),
                      param_grid={'max_depth': range(5, 15, 5),
                                  'min_samples_leaf': range(50, 150, 50),
                                  'min_samples_split': range(50, 150, 50)},
                      scoring='roc_auc', verbose=1)
```

Figure 15: Decision Tree



```
In [39]: # Accuracy
         print("Accuracy:-",metrics.accuracy_score(Ytest, Ytest_pred))

         # Recall
         print("Recall:-",TP / float(TP+FN))

         # Precision
         print("Precision:-", TN / float(TN+FP))

         # F1 score
         print("F1-Score:-", f1_score(Ytest,Ytest_pred))

         Accuracy:- 0.9989291106351603
         Recall:- 0.6122448979591837
         Precision:- 0.9995955261676984
         F1-Score:- 0.6629834254143647

In [40]: # classification_report
         print(classification_report(Ytest, Ytest_pred))

                       precision    recall  f1-score   support

                    0       1.00      1.00      1.00     56864
                    1       0.72      0.61      0.66        98

             accuracy                           1.00     56962
            macro avg       0.86      0.81      0.83     56962
         weighted avg       1.00      1.00      1.00     56962
```

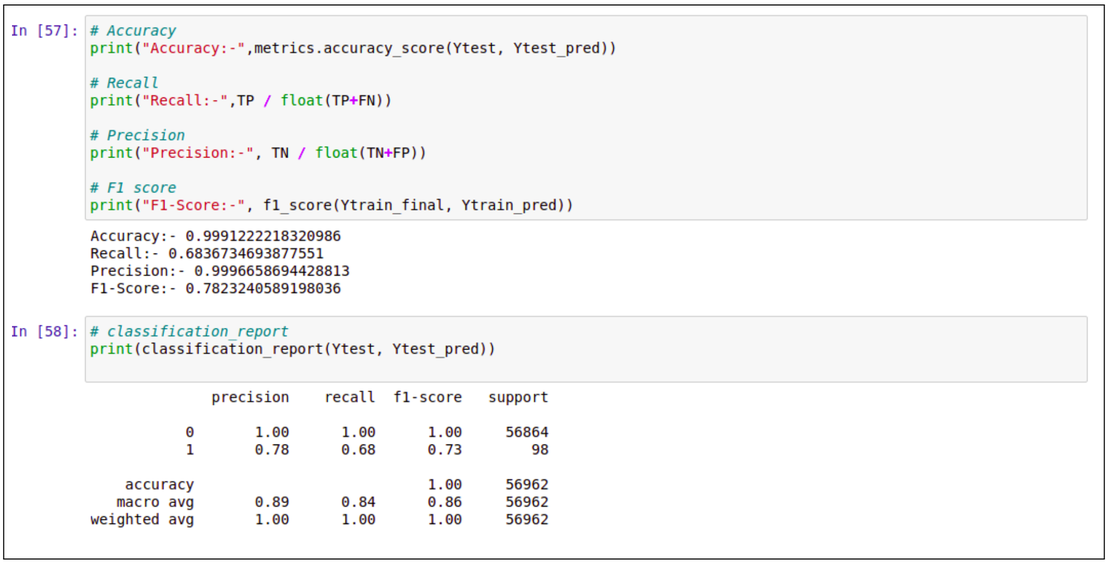
Figure 16: Decision Tree Evaluation metrics



Random Forest

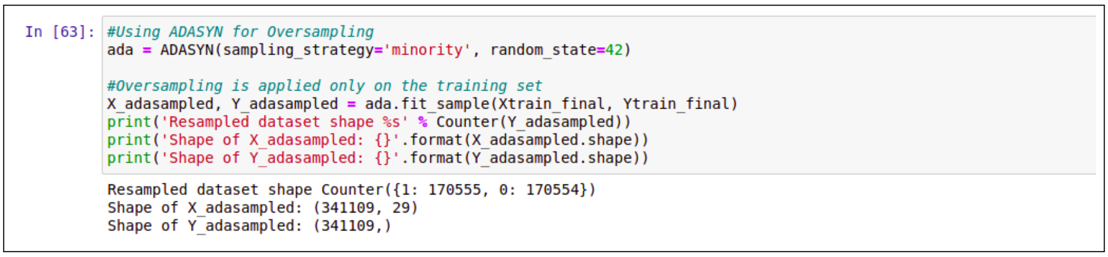
```
In [43]: param_grid = {
             'max_depth': range(5,10,5),
             'min_samples_leaf': range(50, 150, 50),
             'min_samples_split': range(50, 150, 50),
             'n_estimators': [100,200,300],
             'max_features': [10, 20]
         }
         # Create a based model
         rf = RandomForestClassifier()
         # Instantiate the grid search model
         grid_search = GridSearchCV(estimator = rf,
                                    param_grid = param_grid,
                                    cv = 2,
                                    n_jobs = -1,
                                    verbose = 1,
                                    return_train_score=True)

         # Fit the model
         grid_search.fit(Xtrain_final,Ytrain_final)

         Fitting 2 folds for each of 24 candidates, totalling 48 fits

         [Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
         [Parallel(n_jobs=-1)]: Done  48 out of  48 | elapsed: 30.9min finished

Out[43]: GridSearchCV(cv=2, estimator=RandomForestClassifier(), n_jobs=-1,
                      param_grid={'max_depth': range(5, 10, 5), 'max_features': [10, 20],
                                  'min_samples_leaf': range(50, 150, 50),
                                  'min_samples_split': range(50, 150, 50),
                                  'n_estimators': [100, 200, 300]},
                      return_train_score=True, verbose=1)
```

Figure 17: Random Forest



```
In [57]: # Accuracy
         print("Accuracy:-",metrics.accuracy_score(Ytest, Ytest_pred))

         # Recall
         print("Recall:-",TP / float(TP+FN))

         # Precision
         print("Precision:-", TN / float(TN+FP))

         # F1 score
         print("F1-Score:-", f1_score(Ytrain_final, Ytrain_pred))

         Accuracy:- 0.9991222218320986
         Recall:- 0.6836734693877551
         Precision:- 0.9996658694428813
         F1-Score:- 0.7823240589198036

In [58]: # classification_report
         print(classification_report(Ytest, Ytest_pred))

                       precision    recall  f1-score   support

                    0       1.00      1.00      1.00     56864
                    1       0.78      0.68      0.73        98

             accuracy                           1.00     56962
            macro avg       0.89      0.84      0.86     56962
         weighted avg       1.00      1.00      1.00     56962
```

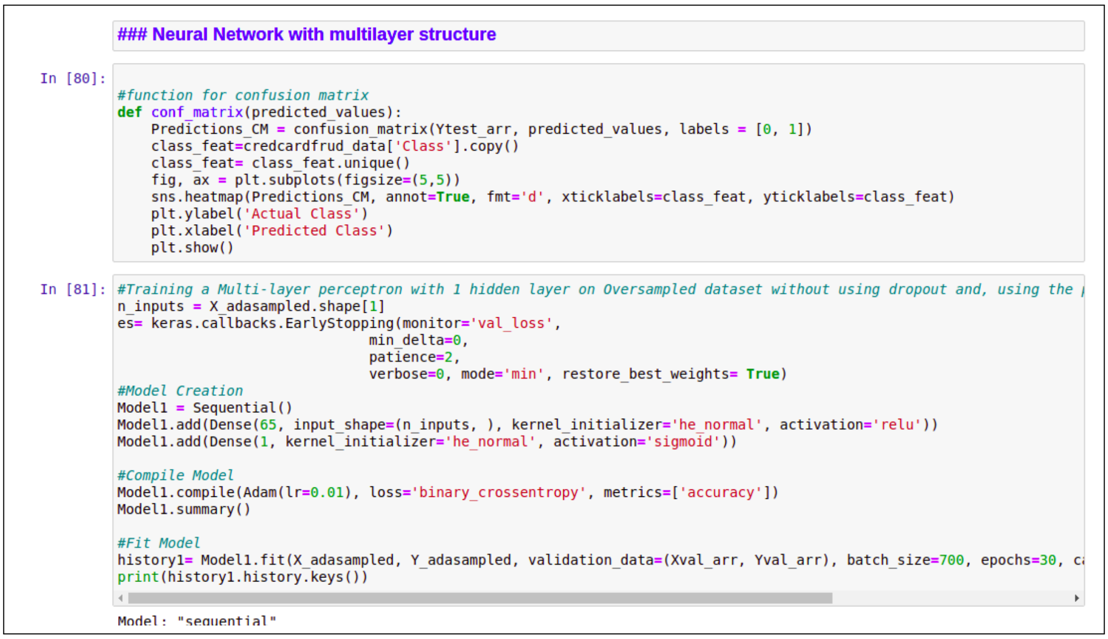
Figure 18: Decision Tree Evaluation metrics



```
In [63]: #Using ADASYN for Oversampling
         ada = ADASYN(sampling_strategy='minority', random_state=42)

         #Oversampling is applied only on the training set
         X_adasampled, Y_adasampled = ada.fit_sample(Xtrain_final, Ytrain_final)
         print('Resampled dataset shape %s' % Counter(Y_adasampled))
         print('Shape of X_adasampled: {}'.format(X_adasampled.shape))
         print('Shape of Y_adasampled: {}'.format(Y_adasampled.shape))

         Resampled dataset shape Counter({1: 170555, 0: 170554})
         Shape of X_adasampled: (341109, 29)
         Shape of Y_adasampled: (341109,)
```

Figure 19: ADASYN Oversampling technique

4.4 Implementation of Neural Network with multiple hidden layers



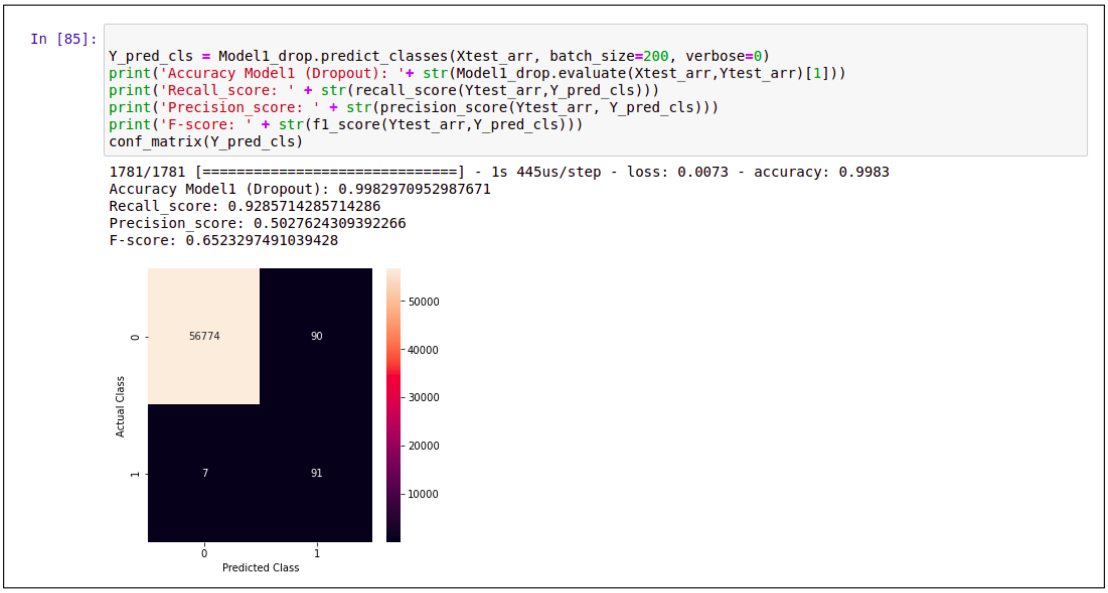
Neural Network with multilayer structure

```
In [80]:
#function for confusion matrix
def conf_matrix(predicted_values):
    Predictions_CM = confusion_matrix(Ytest_arr, predicted_values, labels = [0, 1])
    class_feat=credcardfrud_data['Class'].copy()
    class_feat= class_feat.unique()
    fig, ax = plt.subplots(figsize=(5,5))
    sns.heatmap(Predictions_CM, annot=True, fmt='d', xticklabels=class_feat, yticklabels=class_feat)
    plt.ylabel('Actual Class')
    plt.xlabel('Predicted Class')
    plt.show()
```

```
In [81]: #Training a Multi-layer perceptron with 1 hidden layer on Oversampled dataset without using dropout and, using the
n_inputs = X_adasampled.shape[1]
es= keras.callbacks.EarlyStopping(monitor='val_loss',
                              min_delta=0,
                              patience=2,
                              verbose=0, mode='min', restore_best_weights= True)
#Model Creation
Model1 = Sequential()
Model1.add(Dense(65, input_shape=(n_inputs, ), kernel_initializer='he_normal', activation='relu'))
Model1.add(Dense(1, kernel_initializer='he_normal', activation='sigmoid'))

#Compile Model
Model1.compile(Adam(lr=0.01), loss='binary_crossentropy', metrics=['accuracy'])
Model1.summary()

#Fit Model
history1= Model1.fit(X_adasampled, Y_adasampled, validation_data=(Xval_arr, Yval_arr), batch_size=700, epochs=30, c
print(history1.history.keys())
```

```
Model: "sequential"
```

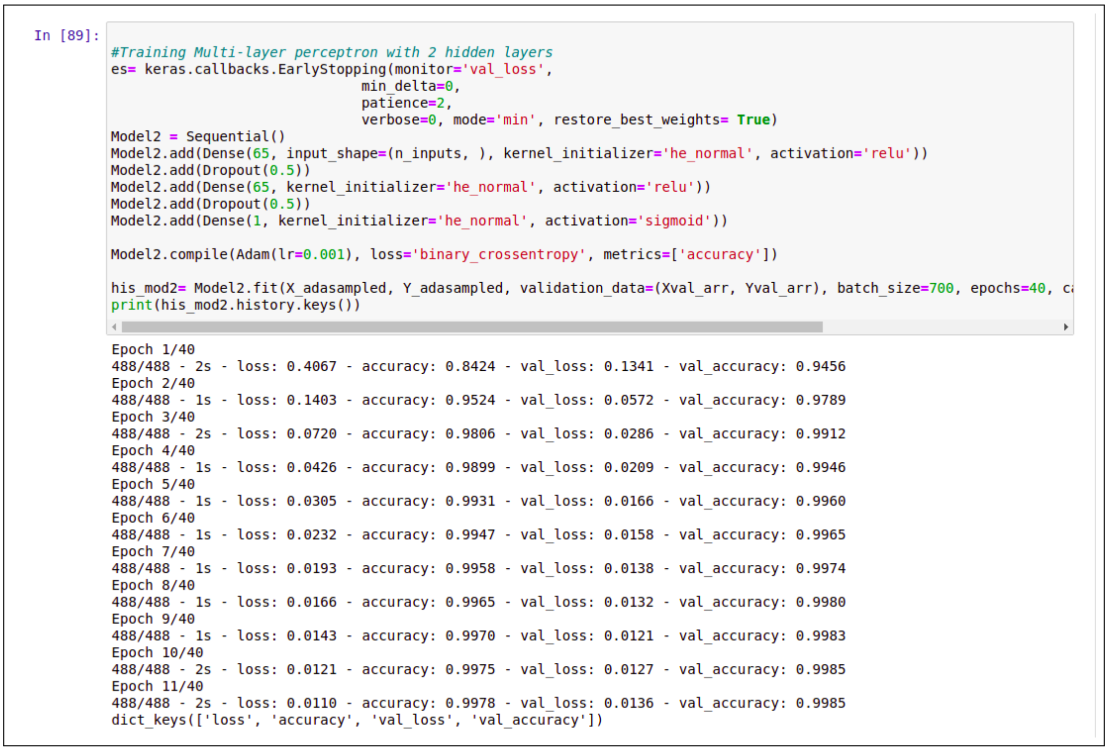
Figure 20: Training MLP with one hidden layer



```
In [85]:
Y_pred_cls = Model1_drop.predict_classes(Xtest_arr, batch_size=200, verbose=0)
print('Accuracy Model1 (Dropout): '+ str(Model1_drop.evaluate(Xtest_arr,Ytest_arr)[1]))
print('Recall_score: ' + str(recall_score(Ytest_arr,Y_pred_cls)))
print('Precision_score: ' + str(precision_score(Ytest_arr, Y_pred_cls)))
print('F-score: ' + str(f1_score(Ytest_arr,Y_pred_cls)))
conf_matrix(Y_pred_cls)
```

```
1781/1781 [==============================] - 1s 445us/step - loss: 0.0073 - accuracy: 0.9983
Accuracy Model1 (Dropout): 0.9982970952987671
Recall_score: 0.9285714285714286
Precision_score: 0.5027624309392266
F-score: 0.6523297491039428
```

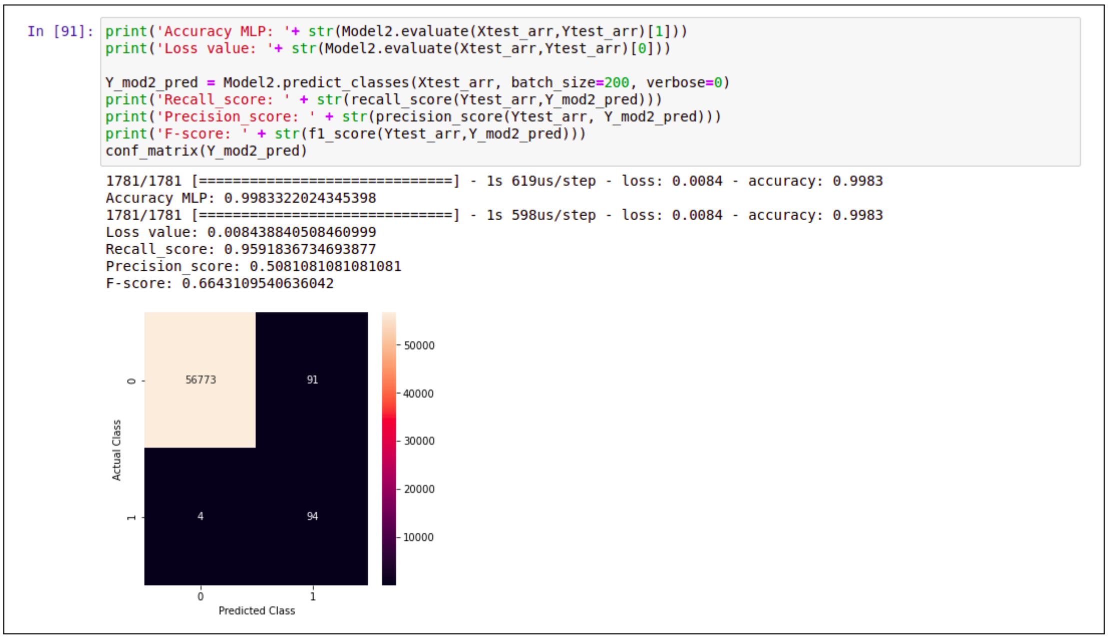
Figure 21: Evaluation metrics of Training MLP with one hidden layer



```
In [89]: #Training Multi-layer perceptron with 2 hidden layers
         es= keras.callbacks.EarlyStopping(monitor='val_loss',
                                           min_delta=0,
                                           patience=2,
                                           verbose=0, mode='min', restore_best_weights= True)
         Model2 = Sequential()
         Model2.add(Dense(65, input_shape=(n_inputs, ), kernel_initializer='he_normal', activation='relu'))
         Model2.add(Dropout(0.5))
         Model2.add(Dense(65, kernel_initializer='he_normal', activation='relu'))
         Model2.add(Dropout(0.5))
         Model2.add(Dense(1, kernel_initializer='he_normal', activation='sigmoid'))

         Model2.compile(Adam(lr=0.001), loss='binary_crossentropy', metrics=['accuracy'])

         his_mod2= Model2.fit(X_adasampled, Y_adasampled, validation_data=(Xval_arr, Yval_arr), batch_size=700, epochs=40, ca
         print(his_mod2.history.keys())
```

```
Epoch 1/40
488/488 - 2s - loss: 0.4067 - accuracy: 0.8424 - val_loss: 0.1341 - val_accuracy: 0.9456
Epoch 2/40
488/488 - 1s - loss: 0.1403 - accuracy: 0.9524 - val_loss: 0.0572 - val_accuracy: 0.9789
Epoch 3/40
488/488 - 2s - loss: 0.0720 - accuracy: 0.9806 - val_loss: 0.0286 - val_accuracy: 0.9912
Epoch 4/40
488/488 - 1s - loss: 0.0426 - accuracy: 0.9899 - val_loss: 0.0209 - val_accuracy: 0.9946
Epoch 5/40
488/488 - 1s - loss: 0.0305 - accuracy: 0.9931 - val_loss: 0.0166 - val_accuracy: 0.9960
Epoch 6/40
488/488 - 1s - loss: 0.0232 - accuracy: 0.9947 - val_loss: 0.0158 - val_accuracy: 0.9965
Epoch 7/40
488/488 - 1s - loss: 0.0193 - accuracy: 0.9958 - val_loss: 0.0138 - val_accuracy: 0.9974
Epoch 8/40
488/488 - 1s - loss: 0.0166 - accuracy: 0.9965 - val_loss: 0.0132 - val_accuracy: 0.9980
Epoch 9/40
488/488 - 1s - loss: 0.0143 - accuracy: 0.9970 - val_loss: 0.0121 - val_accuracy: 0.9983
Epoch 10/40
488/488 - 2s - loss: 0.0121 - accuracy: 0.9975 - val_loss: 0.0127 - val_accuracy: 0.9985
Epoch 11/40
488/488 - 2s - loss: 0.0110 - accuracy: 0.9978 - val_loss: 0.0136 - val_accuracy: 0.9985
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

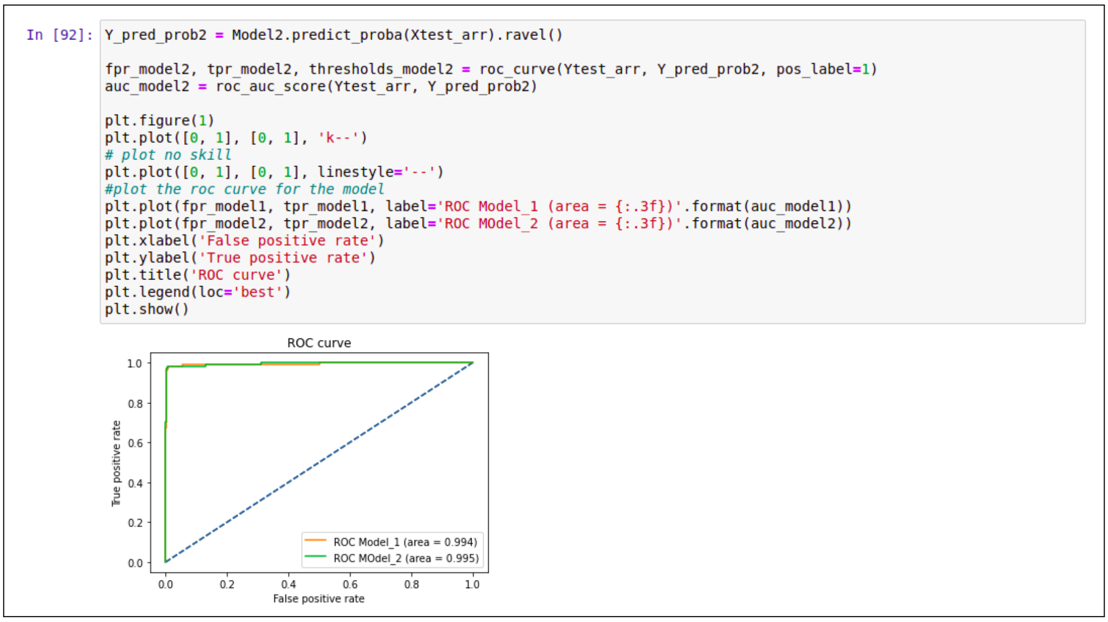
Figure 22: Training MLP with two hidden layer



```
In [91]: print('Accuracy MLP: '+ str(Model2.evaluate(Xtest_arr,Ytest_arr)[1]))
         print('Loss value: '+ str(Model2.evaluate(Xtest_arr,Ytest_arr)[0]))

         Y_mod2_pred = Model2.predict_classes(Xtest_arr, batch_size=200, verbose=0)
         print('Recall_score: ' + str(recall_score(Ytest_arr,Y_mod2_pred)))
         print('Precision_score: ' + str(precision_score(Ytest_arr, Y_mod2_pred)))
         print('F-score: ' + str(f1_score(Ytest_arr,Y_mod2_pred)))
         conf_matrix(Y_mod2_pred)
```

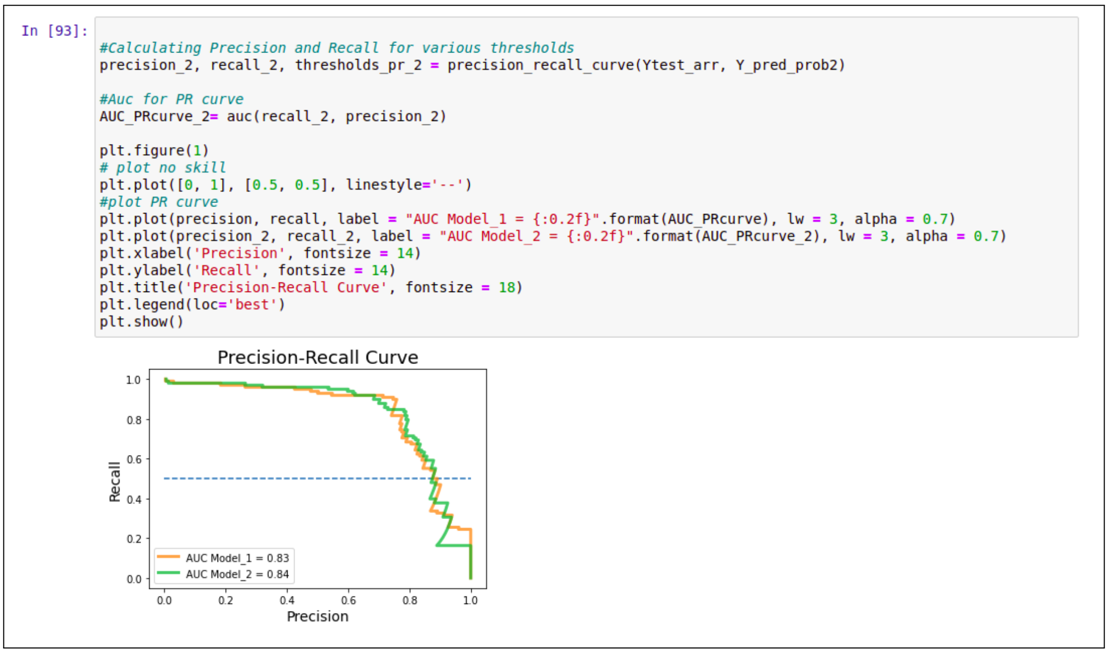
```
1781/1781 [==============================] - 1s 619us/step - loss: 0.0084 - accuracy: 0.9983
Accuracy MLP: 0.9983322024345398
1781/1781 [==============================] - 1s 598us/step - loss: 0.0084 - accuracy: 0.9983
Loss value: 0.008438840508460999
Recall_score: 0.9591836734693877
Precision_score: 0.5081081081081081
F-score: 0.6643109540636042
```

Figure 23: Evaluation metrics of Training MLP with one hidden layer



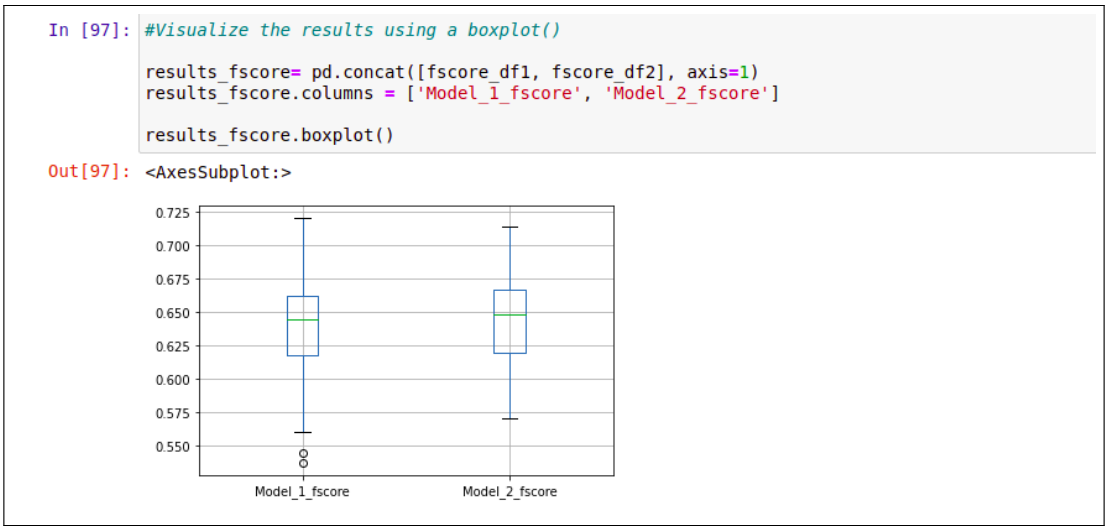
```
In [92]: Y_pred_prob2 = Model2.predict_proba(Xtest_arr).ravel()

fpr_model2, tpr_model2, thresholds_model2 = roc_curve(Ytest_arr, Y_pred_prob2, pos_label=1)
auc_model2 = roc_auc_score(Ytest_arr, Y_pred_prob2)

plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
#plot the roc curve for the model
plt.plot(fpr_model1, tpr_model1, label='ROC Model_1 (area = {:.3f})'.format(auc_model1))
plt.plot(fpr_model2, tpr_model2, label='ROC MOdel_2 (area = {:.3f})'.format(auc_model2))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```

Figure 24: ROC Curve for Model 1 and Model 2



```
In [93]: #Calculating Precision and Recall for various thresholds
precision_2, recall_2, thresholds_pr_2 = precision_recall_curve(Ytest_arr, Y_pred_prob2)

#Auc for PR curve
AUC_PRcurve_2= auc(recall_2, precision_2)

plt.figure(1)
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
#plot PR curve
plt.plot(precision, recall, label = "AUC Model_1 = {:0.2f}".format(AUC_PRcurve), lw = 3, alpha = 0.7)
plt.plot(precision_2, recall_2, label = "AUC Model_2 = {:0.2f}".format(AUC_PRcurve_2), lw = 3, alpha = 0.7)
plt.xlabel('Precision', fontsize = 14)
plt.ylabel('Recall', fontsize = 14)
plt.title('Precision-Recall Curve', fontsize = 18)
plt.legend(loc='best')
plt.show()
```

Figure 25: AUC Curve for Model 1 and Model 2



```
In [97]: #Visualize the results using a boxplot()

         results_fscore= pd.concat([fscore_df1, fscore_df2], axis=1)
         results_fscore.columns = ['Model_1_fscore', 'Model_2_fscore']

         results_fscore.boxplot()

Out[97]: <AxesSubplot:>
```

Figure 26: Box plot for Model and Model 2

References

Agrawal, A., Kumar, S. and Mishra, A. K. (2015). Implementation of novel approach for credit card fraud detection, *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 1–4.