MACHINE LEARNING ASSIGNMENT # 03

Binary Classification

Submitted by:

Name: Khuram Shahzad

Roll No: p218742

Subject: Machine Learning (MS DS)

Submitted to:

Dr. Muhammad Taimoor Khan

JUNE 5, 2022

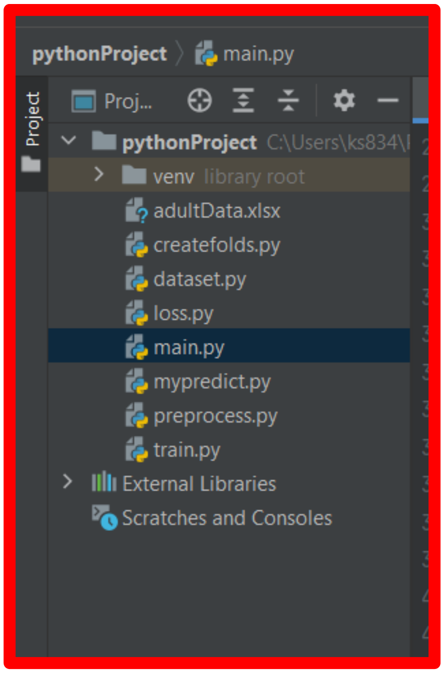
FAST NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES – PESHAWAR CAMPUS
Jamrud Road 160 Industrial Estate Road, Peshawar - PK

1 TABLE OF CONTENTS

2 Assignment Question:

Use an IDE with a somewhat similar directory/file structure as given below. Try different possibilities to improve accuracy as much as possible. Report accuracies, precision, recall, f1-scores for 5-fold and 10-fold cross validation. Its a binary class classification problem with labels as individuals earning above 50K per year or below.

3 Directory/File structure (IDE PyCharm)



```
pythonProject  C:\Users\ks834\
  venv  library root
  adultData.xlsx
  createfolds.py
  dataset.py
  loss.py
  main.py
  mypredict.py
  preprocess.py
  train.py
External Libraries
Scratches and Consoles
```

4 About Data (discussion):

1. Training and testing Data are separate. Note: I have set 0 for <=50k and 1 for >50k in excel file before loading into this project.

5 Steps (discussion):

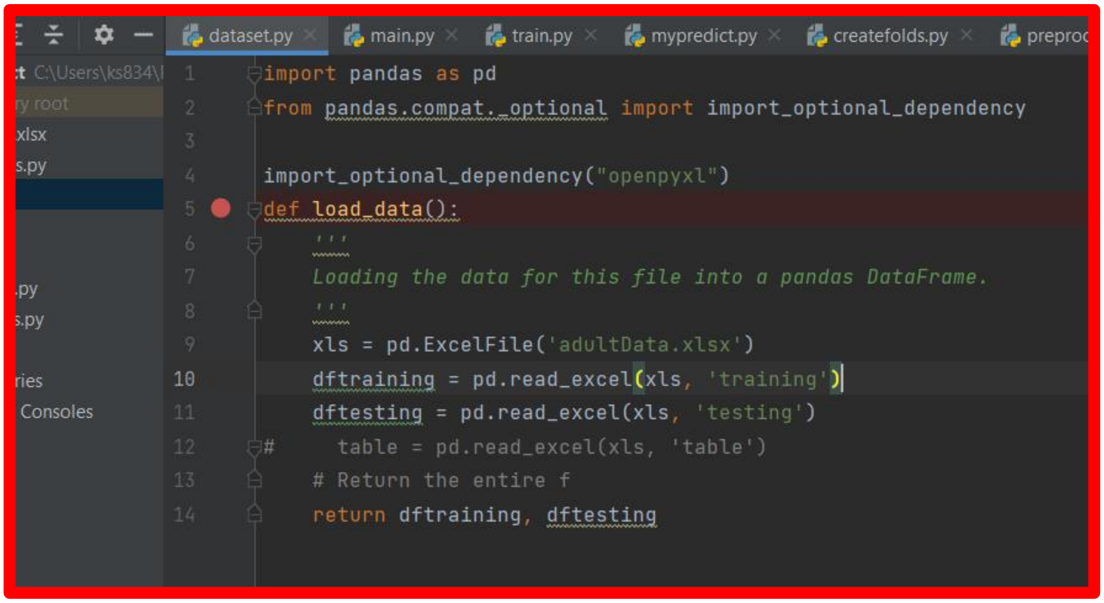
It's to be noted that we have defined a function for each step and we have made the generalized code.

5.1 Importing Libraries

In this step we just imported the all libraries which is being used in assignment

5.2 Loading the Data (dataset.py)

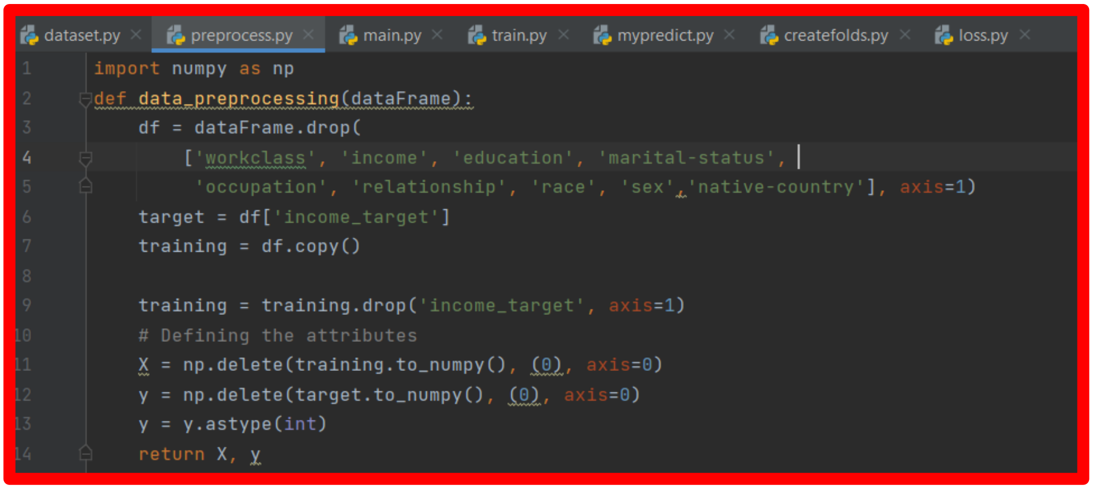
Here we just imported / read our xlsx file.



```
import pandas as pd
from pandas.compat._optional import import_optional_dependency

import_optional_dependency("openpyxl")
def load_data():
    '''
    Loading the data for this file into a pandas DataFrame.
    '''
    xls = pd.ExcelFile('adultData.xlsx')
    dftraining = pd.read_excel(xls, 'training')
    dftesting = pd.read_excel(xls, 'testing')
#     table = pd.read_excel(xls, 'table')
    # Return the entire f
    return dftraining, dftesting
```

5.3 Data Preprocessing (preprocessing.py)

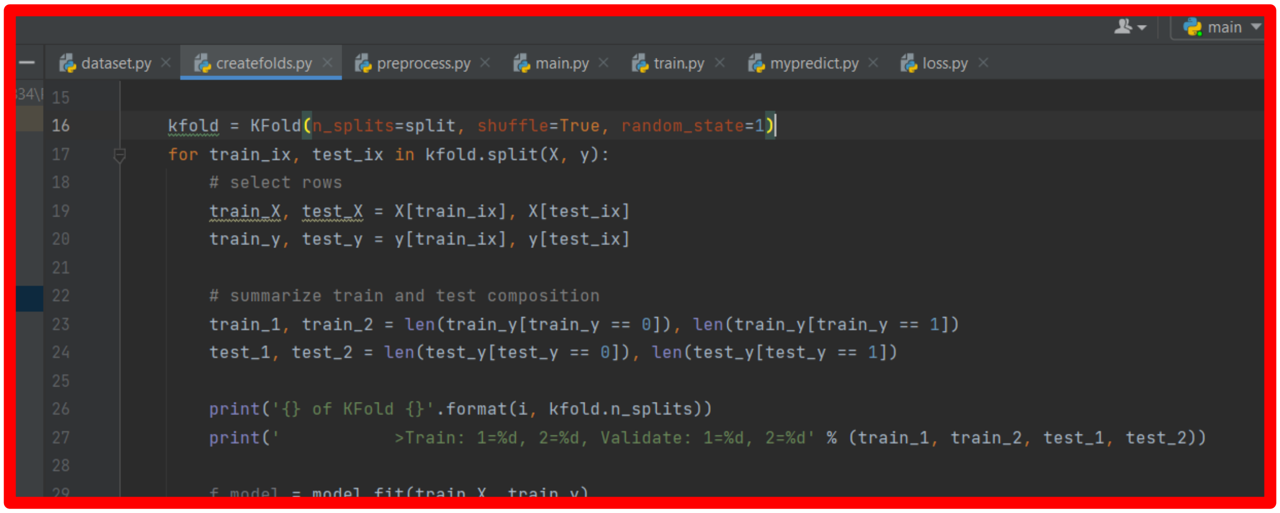
In Data Preprocessing we just convert our data into NumPy Array and then we have just changed our data label is 0 for <=50k and 1 for >50k respectively for smooth classification and evaluation.



```
import numpy as np
def data_preprocessing(dataFrame):
    df = dataFrame.drop(
        ['workclass', 'income', 'education', 'marital-status',
         'occupation', 'relationship', 'race', 'sex','native-country'], axis=1)
    target = df['income_target']
    training = df.copy()

    training = training.drop('income_target', axis=1)
    # Defining the attributes
    X = np.delete(training.to_numpy(), (0), axis=0)
    y = np.delete(target.to_numpy(), (0), axis=0)
    y = y.astype(int)
    return X, y
```

5.4 Create Folds (createfolds.py)

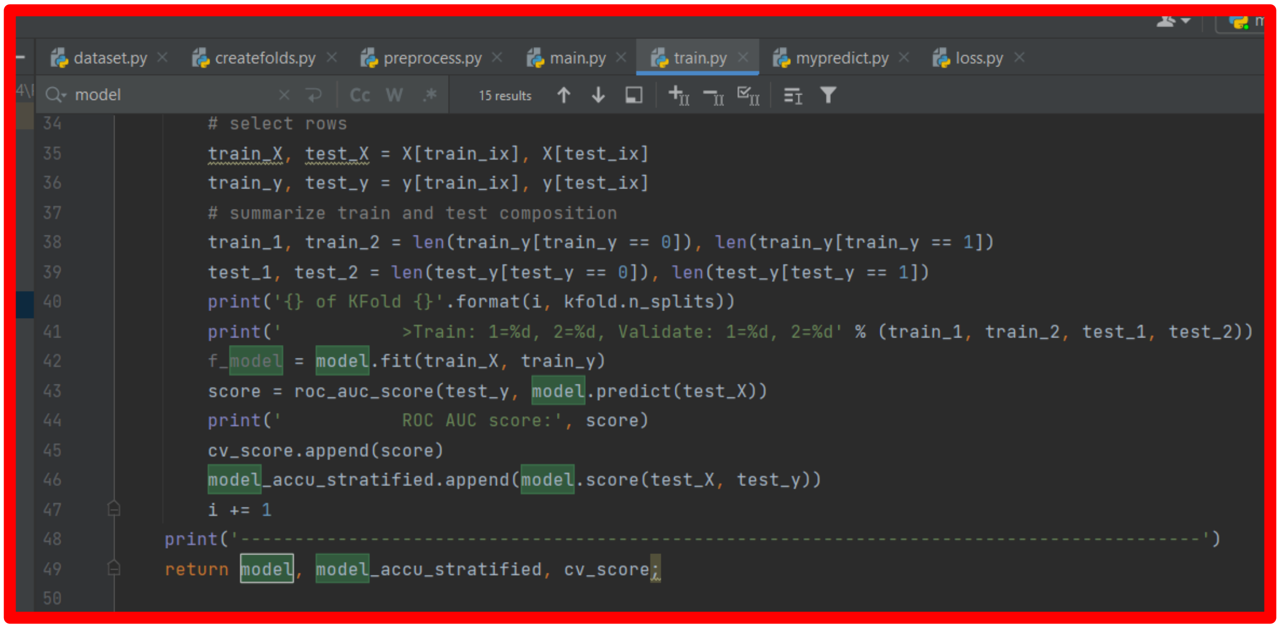
Here we have defined our training and testing generalized function/file for classifier.



```
    kfold = KFold(n_splits=split, shuffle=True, random_state=1)
    for train_ix, test_ix in kfold.split(X, y):
        # select rows
        train_X, test_X = X[train_ix], X[test_ix]
        train_y, test_y = y[train_ix], y[test_ix]

        # summarize train and test composition
        train_1, train_2 = len(train_y[train_y == 0]), len(train_y[train_y == 1])
        test_1, test_2 = len(test_y[test_y == 0]), len(test_y[test_y == 1])

        print('{} of KFold {}'.format(i, kfold.n_splits))
        print('            >Train: 1=%d, 2=%d, Validate: 1=%d, 2=%d' % (train_1, train_2, test_1, test_2))

        f_model = model.fit(train_X, train_y)
```

5.5 Training (train.py)

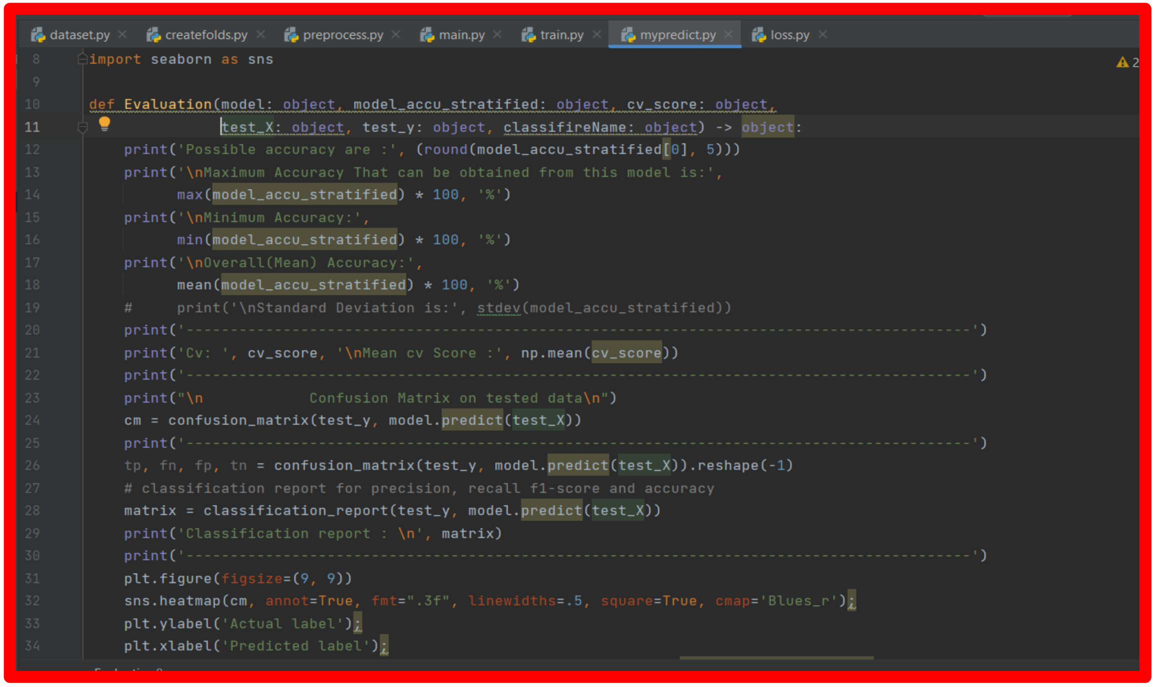
Here we have defined our training and testing generalized function for classifier.



```
        # select rows
        train_X, test_X = X[train_ix], X[test_ix]
        train_y, test_y = y[train_ix], y[test_ix]
        # summarize train and test composition
        train_1, train_2 = len(train_y[train_y == 0]), len(train_y[train_y == 1])
        test_1, test_2 = len(test_y[test_y == 0]), len(test_y[test_y == 1])
        print('{} of KFold {}'.format(i, kfold.n_splits))
        print('            >Train: 1=%d, 2=%d, Validate: 1=%d, 2=%d' % (train_1, train_2, test_1, test_2))
        f_model = model.fit(train_X, train_y)
        score = roc_auc_score(test_y, model.predict(test_X))
        print('            ROC AUC score:', score)
        cv_score.append(score)
        model_accu_stratified.append(model.score(test_X, test_y))
        i += 1
    print('-----------------------------------------------------------------------------------------')
    return model, model_accu_stratified, cv_score;
```

5.6 TESTING (MYPREDICT.PY)

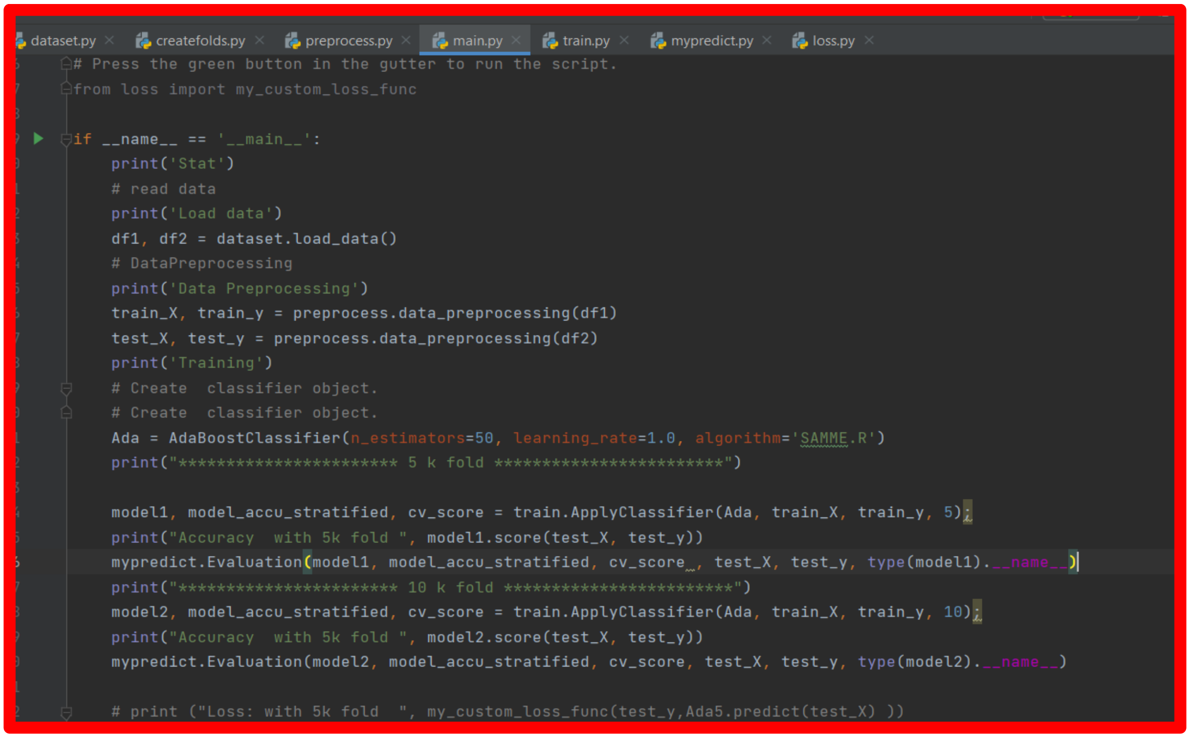
Here we have defined testing function for classifier in my predict file. Here we have a function for evaluation report



```python
import seaborn as sns

def Evaluation(model: object, model_accu_stratified: object, cv_score: object,
               test_X: object, test_y: object, classifireName: object) -> object:
    print('Possible accuracy are :', (round(model_accu_stratified[0], 5)))
    print('\nMaximum Accuracy That can be obtained from this model is:',
          max(model_accu_stratified) * 100, '%')
    print('\nMinimum Accuracy:',
          min(model_accu_stratified) * 100, '%')
    print('\nOverall(Mean) Accuracy:',
          mean(model_accu_stratified) * 100, '%')
    #     print('\nStandard Deviation is:', stdev(model_accu_stratified))
    print('------------------------------------------------------------------------------------------')
    print('Cv: ', cv_score, '\nMean cv Score :', np.mean(cv_score))
    print('------------------------------------------------------------------------------------------')
    print("\n            Confusion Matrix on tested data\n")
    cm = confusion_matrix(test_y, model.predict(test_X))
    print('------------------------------------------------------------------------------------------')
    tp, fn, fp, tn = confusion_matrix(test_y, model.predict(test_X)).reshape(-1)
    # classification report for precision, recall f1-score and accuracy
    matrix = classification_report(test_y, model.predict(test_X))
    print('Classification report : \n', matrix)
    print('------------------------------------------------------------------------------------------')
    plt.figure(figsize=(9, 9))
    sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square=True, cmap='Blues_r');
    plt.ylabel('Actual label');
    plt.xlabel('Predicted label');
```

5.7 DRIVER CLASS (_ _MAIN_ _.PY)

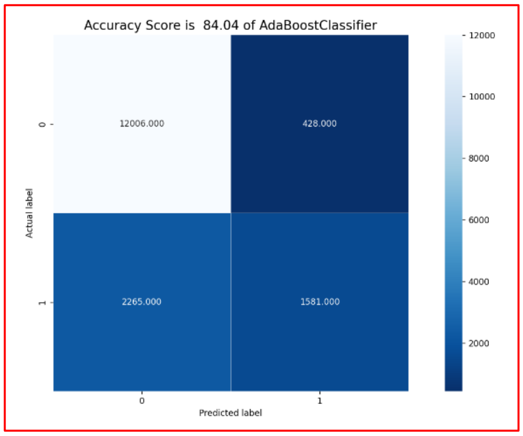
Finally, here the driver file from where we will drive the whole code.



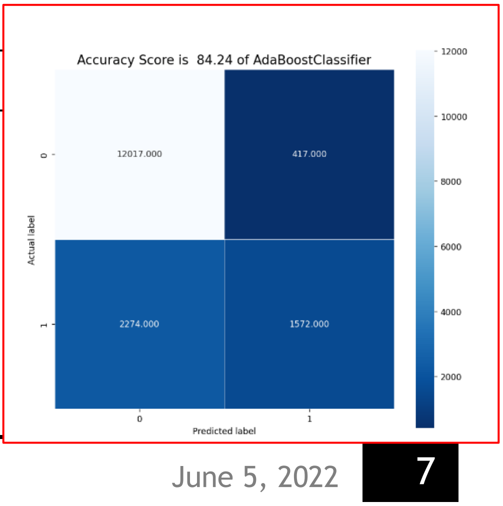
```python
# Press the green button in the gutter to run the script.
from loss import my_custom_loss_func

if __name__ == '__main__':
    print('Stat')
    # read data
    print('Load data')
    df1, df2 = dataset.load_data()
    # DataPreprocessing
    print('Data Preprocessing')
    train_X, train_y = preprocess.data_preprocessing(df1)
    test_X, test_y = preprocess.data_preprocessing(df2)
    print('Training')
    # Create  classifier object.
    # Create  classifier object.
    Ada = AdaBoostClassifier(n_estimators=50, learning_rate=1.0, algorithm='SAMME.R')
    print("*********************** 5 k fold ************************")

    model1, model_accu_stratified, cv_score = train.ApplyClassifier(Ada, train_X, train_y, 5);
    print("Accuracy  with 5k fold ", model1.score(test_X, test_y))
    mypredict.Evaluation(model1, model_accu_stratified, cv_score_, test_X, test_y, type(model1).__name__)
    print("*********************** 10 k fold ************************")
    model2, model_accu_stratified, cv_score = train.ApplyClassifier(Ada, train_X, train_y, 10);
    print("Accuracy  with 5k fold ", model2.score(test_X, test_y))
    mypredict.Evaluation(model2, model_accu_stratified, cv_score, test_X, test_y, type(model2).__name__)

    # print ("Loss: with 5k fold  ", my_custom_loss_func(test_y,Ada5.predict(test_X) ))
```

6 ADA BOOST CLASSIFIER (TESTING):

```
-------------------------------------------------------------------------
Classifier Name:  AdaBoostClassifier
-------------------------------------------------------------------------
        *********************** 5 k fold ***********************
1 of KFold 5
           >Train: 1=19729, 2=6319, Validate: 1=4990, 2=1522
           ROC AUC score: 0.6988561880660138
2 of KFold 5
           >Train: 1=19798, 2=6250, Validate: 1=4921, 2=1591
           ROC AUC score: 0.7092361639485262
3 of KFold 5
           >Train: 1=19806, 2=6242, Validate: 1=4913, 2=1599
           ROC AUC score: 0.6917904369041968
4 of KFold 5
           >Train: 1=19766, 2=6282, Validate: 1=4953, 2=1559
           ROC AUC score: 0.6915442620543306
5 of KFold 5
           >Train: 1=19777, 2=6271, Validate: 1=4942, 2=1570
           ROC AUC score: 0.6980985804761991

Accuracy with 5k fold 0.8345823095823096
Possible accuracy is: 0.84045

Maximum Accuracy can be obtained from this model is: 84.0448402948403 %
Minimum Accuracy: 82.83169533169533 %

Overall (Mean) Accuracy: 83.41523341523342 %
-------------------------------------------------------------------------
Cv: [0.6988561880660138, 0.7092361639485262, 0.6917904369041968, 0.6915442
620543306, 0.6980985804761991]
Mean cv Score: 0.6979051262898534
-------------------------------------------------------------------------
            Confusion Matrix on tested data
-------------------------------------------------------------------------
Classification report:
               precision    recall  f1-score   support

           0       0.84      0.97      0.90     12434
           1       0.79      0.41      0.54      3846
    accuracy                           0.83     16280
   macro avg       0.81      0.69      0.72     16280
weighted avg       0.83      0.83      0.81     16280
```

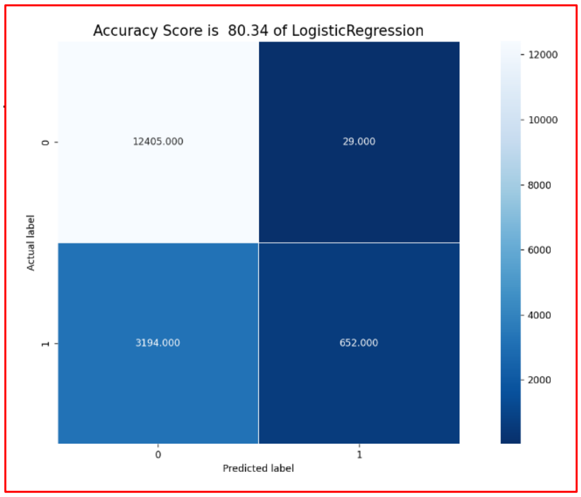


```
----------------------------------------------------------------------------
Classifier Name:  AdaBoostClassifier
----------------------------------------------------------------------------
        ********************** 10 k fold ***********************

1 of KFold 10
           >Train: 1=22212, 2=7092, Validate: 1=2507, 2=749
           ROC AUC score: 0.6906576139546253
2 of KFold 10
           >Train: 1=22236, 2=7068, Validate: 1=2483, 2=773
           ROC AUC score: 0.7040996499352128
3 of KFold 10
           >Train: 1=22256, 2=7048, Validate: 1=2463, 2=793
           ROC AUC score: 0.6979109227666564
4 of KFold 10
           >Train: 1=22261, 2=7043, Validate: 1=2458, 2=798
           ROC AUC score: 0.6849033690817768
5 of KFold 10
           >Train: 1=22264, 2=7040, Validate: 1=2455, 2=801
           ROC AUC score: 0.7183815546249469
6 of KFold 10
           >Train: 1=22261, 2=7043, Validate: 1=2458, 2=798
           ROC AUC score: 0.6851067864943072
7 of KFold 10
           >Train: 1=22228, 2=7076, Validate: 1=2491, 2=765
           ROC AUC score: 0.695922576176195
8 of KFold 10
           >Train: 1=22257, 2=7047, Validate: 1=2462, 2=794
           ROC AUC score: 0.6895102791652257
9 of KFold 10
           >Train: 1=22260, 2=7044, Validate: 1=2459, 2=797
           ROC AUC score: 0.6915035184299807
10 of KFold 10
           >Train: 1=22236, 2=7068, Validate: 1=2483, 2=773
           ROC AUC score: 0.7047037578691636
Accuracy  with 10k fold  0.8347051597051597
Possible accuracy are : 0.83937
Maximum Accuracy can be obtained from this model is: 84.24447174447175 %
Minimum Accuracy: 82.7088452088452 %
Overall(Mean) Accuracy: 83.49201474201475 %
------------------------------------------------------------------------
Cv:  [0.6906576139546253, 0.7040996499352128, 0.6979109227666564, 0.684903369
0817768, 0.7183815546249469, 0.6851067864943072, 0.695922576176195, 0.6895102
791652257, 0.6915035184299807, 0.7047037578691636]
Mean cv Score : 0.696270002849809
------------------------------------------------------------------------
            Confusion Matrix on tested data
------------------------------------------------------------------------
Classification report:
               precision    recall  f1-score   support

           0       0.84      0.97      0.90     12434
           1       0.79      0.41      0.54      3846
    accuracy                           0.83     16280
   macro avg       0.82      0.69      0.72     16280
weighted avg       0.83      0.83      0.81     16280
```

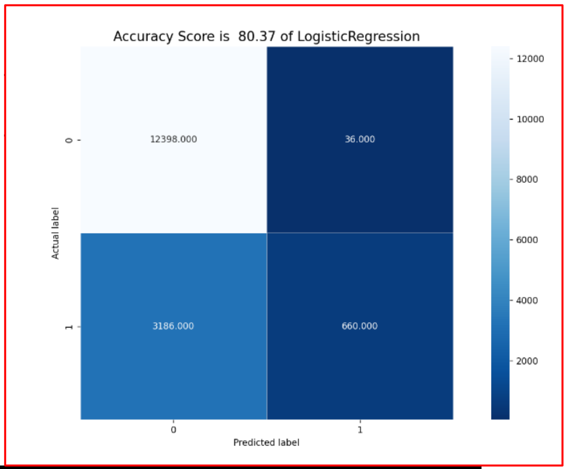


7 LOGISTIC REGRESSION (TESTING):

```
-----------------------------------------------------------------------------
Classifier Name:  LogisticRegression
-----------------------------------------------------------------------------

         *********************** 5 k fold ***********************
1 of KFold 5
           >Train: 1=19729, 2=6319, Validate: 1=4990, 2=1522
           ROC AUC score: 0.6142044930860406
2 of KFold 5
           >Train: 1=19798, 2=6250, Validate: 1=4921, 2=1591
           ROC AUC score: 0.617316389654211
3 of KFold 5
           >Train: 1=19806, 2=6242, Validate: 1=4913, 2=1599
           ROC AUC score: 0.6127285817629505
4 of KFold 5
           >Train: 1=19766, 2=6282, Validate: 1=4953, 2=1559
           ROC AUC score: 0.6150461548303896
5 of KFold 5
           >Train: 1=19777, 2=6271, Validate: 1=4942, 2=1570
           ROC AUC score: 0.6114479555196973
-----------------------------------------------------------------------------
------------
Accuracy  with 5k fold  0.802027027027027
Possible accuracy are : 0.80344
Maximum Accuracy That can be obtained from this model is: 80.34398034398035 %
Minimum Accuracy: 79.43796068796068 %
Overall(Mean) Accuracy: 79.71130221130221 %
-----------------------------------------------------------------------------
------------
Cv:  [0.6142044930860406, 0.617316389654211, 0.6127285817629505, 0.6150461548
303896, 0.6114479555196973]
Mean cv Score : 0.6141487149706577
-----------------------------------------------------------------------------
            Confusion Matrix on tested data
-----------------------------------------------------------------------------
Classification report :
               precision    recall  f1-score   support
           0       0.80      1.00      0.89     12434
           1       0.96      0.17      0.29      3846
    accuracy                           0.80     16280
   macro avg       0.88      0.58      0.59     16280
weighted avg       0.83      0.80      0.74     16280
-----------------------------------------------------
```



```
---------------------------------------------------------------------------
Classifier Name:  LogisticRegression
---------------------------------------------------------------------------
        *********************** 10 k fold ************************
1 of KFold 10
           >Train: 1=22212, 2=7092, Validate: 1=2507, 2=749
            ROC AUC score: 0.5859039282798552
2 of KFold 10
           >Train: 1=22236, 2=7068, Validate: 1=2483, 2=773
           ROC AUC score: 0.6254731397305038
3 of KFold 10
           >Train: 1=22256, 2=7048, Validate: 1=2463, 2=793
            ROC AUC score: 0.6132186370899655
4 of KFold 10
           >Train: 1=22261, 2=7043, Validate: 1=2458, 2=798
           ROC AUC score: 0.6198949366907912
5 of KFold 10
           >Train: 1=22264, 2=7040, Validate: 1=2455, 2=801
           ROC AUC score: 0.624957601368961
6 of KFold 10
           >Train: 1=22261, 2=7043, Validate: 1=2458, 2=798
           ROC AUC score: 0.602774226045178
7 of KFold 10
           >Train: 1=22228, 2=7076, Validate: 1=2491, 2=765
           ROC AUC score: 0.615721171380368
8 of KFold 10
           >Train: 1=22257, 2=7047, Validate: 1=2462, 2=794
           ROC AUC score: 0.6085404956343986
9 of KFold 10
           >Train: 1=22260, 2=7044, Validate: 1=2459, 2=797
           ROC AUC score: 0.6050676515175094
10 of KFold 10
           >Train: 1=22236, 2=7068, Validate: 1=2483, 2=773
           ROC AUC score: 0.6223121886004651
---------------------------------------------------------------------------
Accuracy with 10k fold 0.8020884520884521
Possible accuracy are: 0.79484
Maximum Accuracy can be obtained from this model is: 80.37469287469288 %
Minimum Accuracy: 78.83906633906635 %
Overall (Mean) Accuracy: 79.60995085995086 %
---------------------------------------------------------------------------
Cv:  [0.5859039282798552, 0.6254731397305038, 0.6132186370899655, 0.619894936
6907912, 0.624957601368961, 0.602774226045178, 0.615721171380368, 0.608540495
6343986, 0.6050676515175094, 0.6223121886004651]
Mean cv Score: 0.6123863976337995
-----------------------------------------------------------------
            Confusion Matrix on tested data
-----------------------------------------------------------------
Classification report:
               precision    recall  f1-score   support

           0       0.80      1.00      0.89     12434
           1       0.95      0.17      0.29      3846
    accuracy                           0.80     16280
   macro avg       0.87      0.58      0.59     16280
weighted avg       0.83      0.80      0.74     16280
```


Accuracy Score is 80.37 of LogisticRegression
Actual label
Predicted label