Model Building

Date	01 November 2022
Team ID	PNT2022TMID12860
Project Name	Smart Lender - Applicant Credibility Prediction for Loan Approval

DECISION TREE MODEL

- A function named decision tree is created and train and test data are passed as the parameters. Inside the function, the DecisionTreeClassifier algorithm is initialized and training data is passed to the model with .fit() function.
- Test data is predicted with the .predict() function and saved in the new variable. For evaluating the model, a confusion matrix and classification report are done.

```
[20] def decisionTree(x_train, x_test, y_train, y_test):
         dt=DecisionTreeClassifier()
         dt.fit(x_train,y_train)
         yPred = dt.predict(x_test)
         print('***DecisionTreeClassifier***')
         print('Confusion matrix')
         print(confusion_matrix(y_test,yPred))
         print('Classification report')
         print(classification_report (y_test,yPred))
         print("score")
         print(dt.score(x_test,y_test))
```

RANDOM FOREST MODEL

- A function named randomForest is created and train and test data are passed as the parameters. Inside the function, the RandomForestClassifier algorithm is initialized and training data is passed to the model with the .fit() function.
- Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report are done.

```
def randomForest(x_train, x_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(x_train,y_train)
    yPred = rf.predict(x_test)
    print('***RandomForestClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
    print("score")
    print(rf.score(x_test,y_test))
```

KNN MODEL

- A function named KNN is created and train and test data are passed as the parameters. Inside the function, the KNeighborsClassifier algorithm is initialized and training data is passed to the model with the .fit() function.
- Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
[22]  def KNN(x_train, x_test, y_train, y_test):
          knn = KNeighborsClassifier()
          knn.fit(x_train,y_train)
          yPred = knn.predict(x_test)
          print('***KNeighborsClassifier***')
          print('Confusion matrix')
          print(confusion_matrix(y_test,yPred))
          print('Classification report')
          print(classification_report(y_test,yPred))
          print("score")
          print(knn.score(x_test,y_test))
```

XGBOOST MODEL

- A function named xgboost is created and train and test data are passed as the parameters. Inside the function, the GradientBoostingClassifier algorithm is initialized and training data is passed to the model with .fit() function.
- Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report are done.
- Multivariate analysis is to find the relation between multiple features. Here we have used a swarm plot from the seaborn package.

```
def xgboost(x_train, x_test, y_train, y_test):
    xg = GradientBoostingClassifier()
    xg.fit(x_train,y_train)
    yPred = xg.predict(x_test)
    print('***Gradient BoostingClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
    print("score")
    print(xg.score(x_test,y_test))
```

COMPARE THE MODELS

- For comparing the above four models, the Model function is defined.
- After calling the function, the results of models are displayed as output. From the four models Xgboost is performing well. From the below image, We can see the accuracy of the model. Xgboost is giving the accuracy of 94.7% with training data , 81.1% accuracy for the testing data.so we are considering xgboost and deploying this model.

```
decisionTree(x_train, x_test, y_train, y_test)

***DecisionTreeClassifier***
Confusion matrix
[[50 17]
 [16 54]]
Classification report
              precision    recall  f1-score   support

           0       0.76      0.75      0.75        67
           1       0.76      0.77      0.77        70

    accuracy                           0.76       137
   macro avg       0.76      0.76      0.76       137
weighted avg       0.76      0.76      0.76       137

score
0.7591240875912408
```

```
randomForest(x_train, x_test, y_train, y_test)

***RandomForestClassifier***
Confusion matrix
[[51 16]
 [ 9 61]]
Classification report
              precision    recall  f1-score   support

           0       0.85      0.76      0.80        67
           1       0.79      0.87      0.83        70

    accuracy                           0.82       137
   macro avg       0.82      0.82      0.82       137
weighted avg       0.82      0.82      0.82       137

score
0.8175182481751825
```

```
KNN(x_train, x_test, y_train, y_test)

***KNeighborsClassifier***
Confusion matrix
[[49 18]
 [21 49]]
Classification report
              precision    recall  f1-score   support

           0       0.70      0.73      0.72        67
           1       0.73      0.70      0.72        70

    accuracy                           0.72       137
   macro avg       0.72      0.72      0.72       137
weighted avg       0.72      0.72      0.72       137

score
0.7153284671532847
```

```
xgboost(x_train, x_test, y_train, y_test)

***Gradient BoostingClassifier***
Confusion matrix
[[48 19]
 [ 7 63]]
Classification report
              precision    recall  f1-score   support

           0       0.87      0.72      0.79        67
           1       0.77      0.90      0.83        70

    accuracy                           0.81       137
   macro avg       0.82      0.81      0.81       137
weighted avg       0.82      0.81      0.81       137

score
0.8102189781021898
```

EVALUATING PERFORMANCE OF THE MODEL AND SAVING THE MODEL

- From sklearn, cross_val_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by pickle.dump().

```
[34] from sklearn.model_selection import cross_val_score
     rf = RandomForestClassifier()
     rf.fit(x_train,y_train)
     yPred = rf.predict(x_test)
     f1_score(yPred,y_test, average='weighted')
     cv = cross_val_score(rf,x,y,cv=5)
     np.mean(cv)

     0.8036925719192787
```

```
[35] pickle.dump(rf,open('rdf.pkl','wb'))
     pickle.dump(sc,open("scalar.pkl","wb"))
```

Code is uploaded in the following drive link:

https://colab.research.google.com/drive/10YDz5VLr60QmNikdFTSWBxKUMaqxPi2w