Assignment 2: Binary KNN classifier with 10-Fold Cross Validation

Sourav Sarker (201892985) & Isaac Adeuwon (200925352)

Date: 16 February 2019

Introduction: In this assignment, we used libraries named sys, pandas & numpy for implementing cross-validation from scratch. For the KNN, performance matrix and graphical representations for the performance of the KNN, we used different sci-kit learn libraries and matplotlib. To execute the program from the command prompt:

```
$ python A2_t2.py A2_t2_dataset.tsv
```

Data Handling: Our approach was to select features to eradicate low variance from the dataset with the "`VarianceThreshold`" feature selection function where we used `.6 * (1 - .6)` as a threshold parameter.

We have split the dataset as follows:

1: Load the dataset into → data_frame

2: a ← from data_frame slice dataset based on '0' class

3: b ← from data_frame slice dataset based on '1' class

4: np.array_split ('a' into 10 parts where axis = 0)

yes_1	yes_2	...	yes_10

5: np.array_split ('b' into 10 parts where axis = 0)

no_1	no_2	...	no_10

6: get testing set by concatenating using np.vstack as:

yes_1	no_1	→	yes_no_1

...

yes_10	no_10	→	yes_no_10

7: get training set by concatenating using np.vstack as:

yes_no_2			
yes_no_3			
yes_no_4			
yes_no_5			
yes_no_6	yes_no_1	→	fold_1
yes_no_7			
yes_no_8			
yes_no_9			
yes_no_10			

...

yes_no_1			
yes_no_2			
yes_no_3			
yes_no_4			
yes_no_5	yes_no_10	→	fold_10
yes_no_6			
yes_no_7			
yes_no_8			
yes_no_9			

Accuracy Calculation of KNN:

For the function **cross_fold**
Requires: training_set, test_set, k

1: x_train ← remove the last column from training_set

2: y_train ← keep only the last column from training_set

3: x_test ← remove the last column from test_set

4: y_test ← keep only the last column from test_set

5: classifier ← neighbors.KNeighborsClassifier(value of k, 2 is for binary classification)

6: classifier ← classifier.fit with x_train,y_train

7: y_p ← classifier.predict using x_test

8: score ← get classifier.score from x_test & y_test

9: return metric.accuracy_score from y_test and y_p

For a value of K call the cross_fold function with each fold:

1: get the metic.accuracy_score & append a list knn_score[]

2: get the mean score of knn_score[]

Model Selection: For the binary classification on a highly imbalanced data to select a better model can be tricky, where average accuracy score of KNN is not the key factor. Therefore, we have decided to use other metrics for our assignment as follows: performance metric, per-class precision and recall, log loss function and roc curve.

Binary Class Performance Matric: For building this matric, we used the libraries named `'pandas.crosstab'` and `'pandas.m1'` which requires **`y_predicted, actual_y`**.

```
For the function y_prediction
Requires: training_set, test_set, k

    1:  x_train ← pandas slicing to remove the
        last column from training_set
    2:  y_train ← pandas slicing to keep only
        the last column from training_set
    3:  x_test ← slice to remove the last
        column from test_set
    4:  classifier ←
        neighbors.KNeighborsClassifier(value
        of k, 2 is for binary classification)
    5:  classifier ← classifier.fit with
        x_train,y_train
    6:  classifier ← classifier.predict using
        x_train and y_train
    7:  y_pred ← get classifier.predict from
        x_test
    8:  return y_pred
For a value of K call the y_prediction
function with each fold

        1:  get the y_pred & append a list
            predicted_y[]

For the function y_actual
Requires: dataset
    1:  y_dataset ← keep only last column from
        dataset
    2:  return y_dataset

From our dataset

    1:  get y_dataset
    2:  store class dataset → actual_y[]
    3:  for the value k:
            a.  use pd.crosstab with actual_y
                & predicted_y, store confusion
                matric → confusion_mat_list[]
            b.  use classification_report with
                actual_y & predicted_y → print
                performance matric

This performance metrics consists Precision
and Recall value for each class.
```

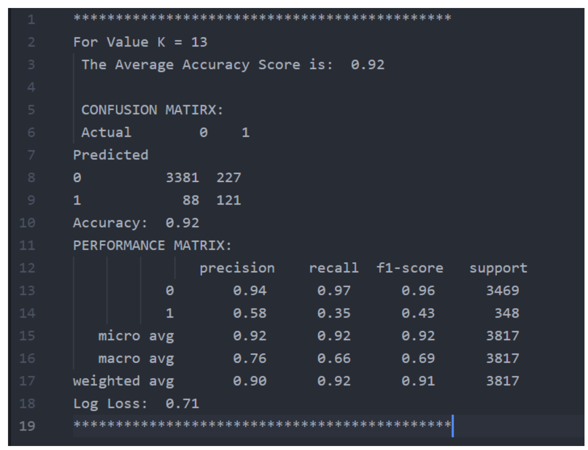
Log Loss function & ROC Curve: We chose the logarithmic loss function for performance measurement for this binary classification model. A higher value of log loss means the divergence from the actual class to predicted class. So, we required a ration predicted probability of a label within all the original label.

```
For the function predict_proba
Requires: training_set,test_set,k
    1:  x_train ← remove the last column from
        training_set
    2:  y_train ← keep only the last column
        from training_set
    3:  x_test ← remove the last column from
        test_set
    4:  y_test ← keep only the last column from
        test_set
    5:  classifier ←
        neighbors.KNeighborsClassifier(value
        of k, 2 is for binary classification)
    6:  classifier ← classifier.fit with
        x_train,y_train
    7:  pre_proba ← classifier.predict_proba
        with x_test
    8:  pre_proba ← from pre_proba keep all
        row and 1st column
    9:  return pre_proba

With training_set,test_set & k call the
predict_proba

    1:  get predict_proba
    2:  calculate log_loss with each index
        actual_y[] and proba
    3:  store value → t_logloss[]
    4:  calculate mean value of t_logloss[] &
        print.
```

Output:



```
********************************************
For Value K = 13
 The Average Accuracy Score is:  0.92

 CONFUSION MATIRX:
 Actual        0    1
Predicted
0           3381  227
1             88  121
Accuracy:  0.92
PERFORMANCE MATRIX:
               precision    recall  f1-score   support
           0       0.94      0.97      0.96      3469
           1       0.58      0.35      0.43       348
   micro avg       0.92      0.92      0.92      3817
   macro avg       0.76      0.66      0.69      3817
weighted avg       0.90      0.92      0.91      3817
Log Loss:  0.71
********************************************
```

Fig-1: Output

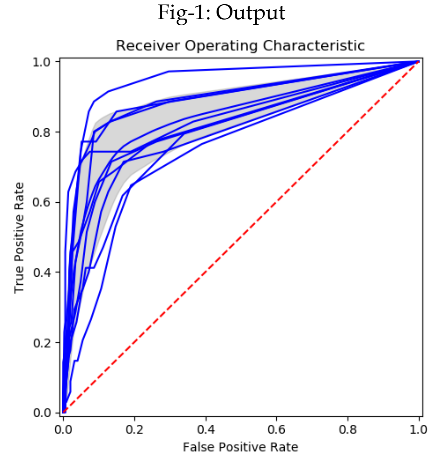


Fig-2: ROC Curve

Conclusion: From our observation, for K=13 we got better performance metric as well as in the ROC curve.