AI

Dr.Marwan Torki

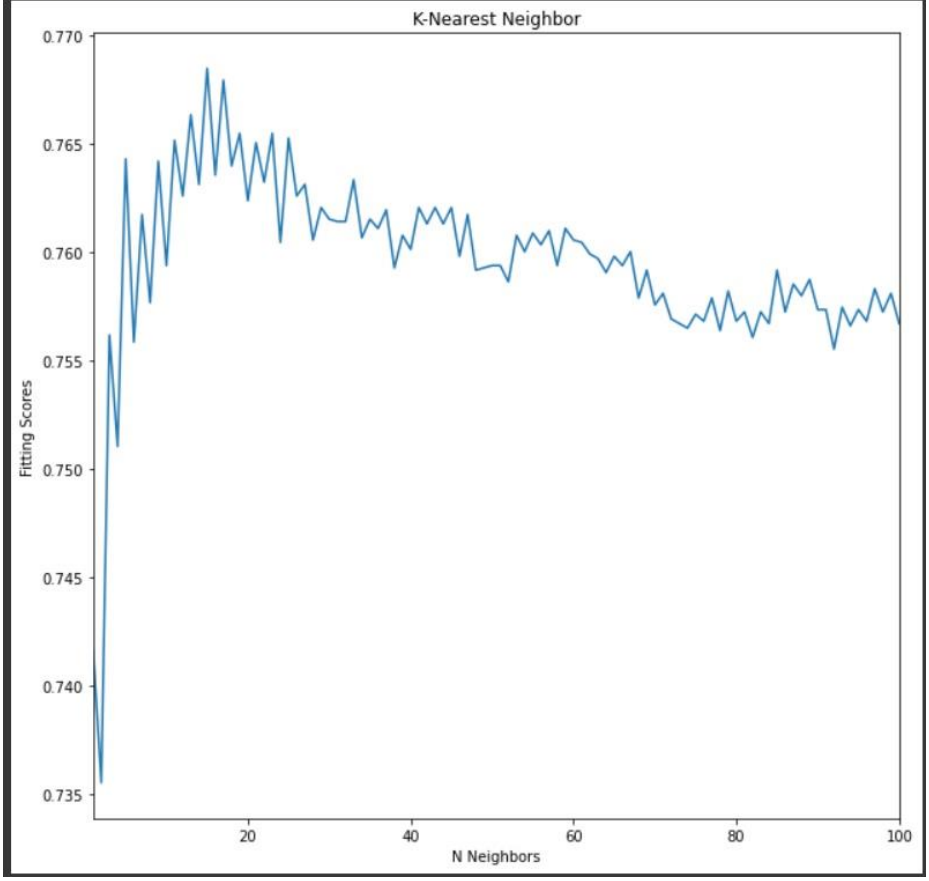
Data Classification

Amr Ashraf Ibrahim	**6301**
Mohamed Kamal	**6331**
Mahmoud ElBanna	**6407**
Raphael Rafik	**6171**

1) K-NN:

After applying grid search on our train data, we have found that the best value for K neighbors to have the best accuracy is equal to 15 neighbors with accuracy = 76.8% (Train Data), 77.10% (Test Data)
The training time = 1 minute and 40 sec
Comparing this accuracy with the training time, The K-NN model is very reasonable to classify this data.

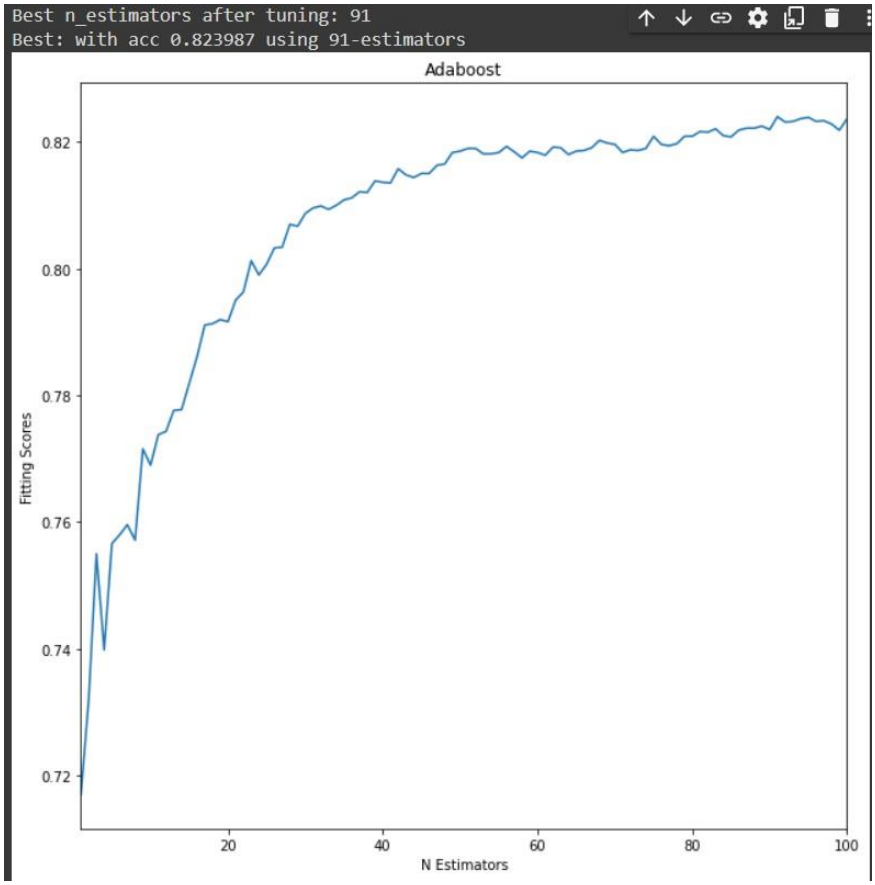
```
Best k_neighbors after tuning: 15
Best: with acc 0.768449 using 15-neighbors
```




```
K-Nearest Neighbor Misplaced Labels for our testing dataset:  919
K-Nearest Neighbor Accuracy for our testing dataset: 77.10%
K-Nearest Neighbor Confusion Matrix for our testing dataset:
                  Actual g     Actual h
Predicted g     1738    268
Predicted h     651             1356
K-Nearest Neighbor Report (F-measure,Recall,Precision) for our testing dataset:
              precision    recall  f1-score   support

           g       0.73      0.87      0.79      2006
           h       0.83      0.68      0.75      2007

    accuracy                           0.77      4013
   macro avg       0.78      0.77      0.77      4013
weighted avg       0.78      0.77      0.77      4013

Running Time :  99.56080794334412
```

2) Adaboost:

After applying grid search on our train data, we have found that the best value for the number of estimators to have the best accuracy is equal to 91 estimators with accuracy = 82.39% (Train Data), 82.18% (Test Data)
The training time = 15 minutes and 9 sec
Comparing The Adaboost model with the other models, Adaboost accuracy is one of the best accuracies (2nd one) while comparing it to the other models but one of its disadvantages it takes a long time to train.

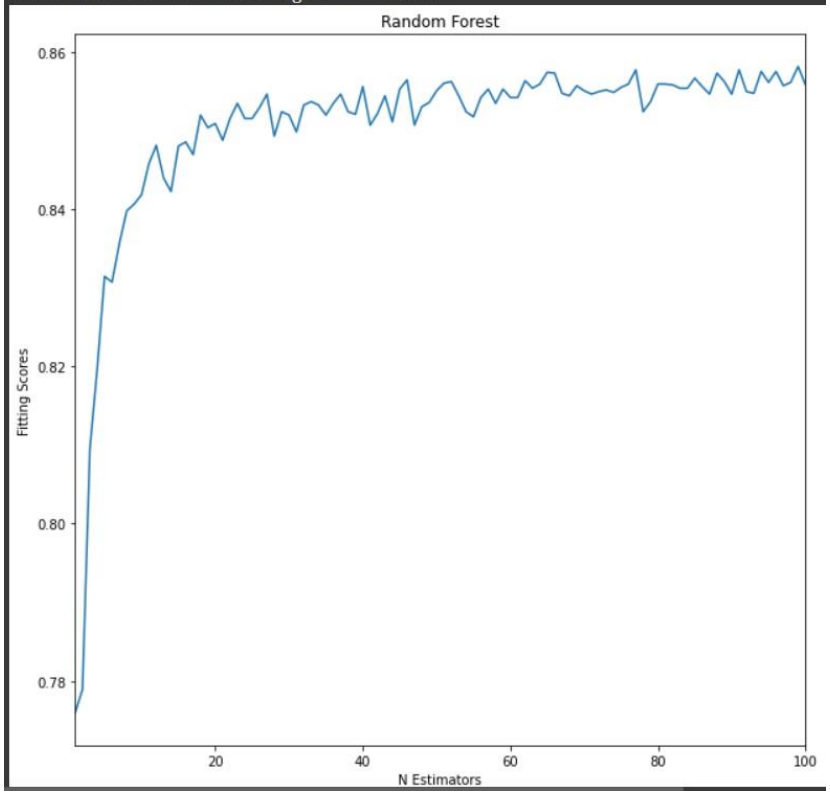



```
Adaboost Misplaced Labels for our testing dataset:  715
Adaboost Accuracy for our testing dataset: 82.18%
Adaboost Confusion Matrix for our testing dataset:
                 Actual g     Actual h
Predicted g     1664    342
Predicted h     373             1634
Adaboost Report (F-measure,Recall,Precision) for our testing dataset:
              precision    recall  f1-score   support

           g       0.82      0.83      0.82      2006
           h       0.83      0.81      0.82      2007

    accuracy                           0.82      4013
   macro avg       0.82      0.82      0.82      4013
weighted avg       0.82      0.82      0.82      4013

Running Time :  905.6547486782074
```

3) Random Forest:

After applying grid search on our train data, we have found that the best value for the number of estimators to have the best accuracy is equal to 99 estimators with accuracy = 85.81% (Train Data), 85.65% (Test Data)
The training time = 23 minutes and 14 sec
Comparing The Random Forest model with the other models, Random Forest accuracy has the best accuracy while comparing it to the other models but has also the longest time for training.

```
Best n_estimators after tuning: 99
Best: with acc 0.858164 using 99-estimators
```




```
Random Forest Misplaced Labels for our testing dataset:  576
Random Forest Accuracy for our testing dataset: 85.65%
Random Forest Confusion Matrix for our testing dataset:
                Actual g     Actual h
Predicted g     1771    235
Predicted h     341             1666
Random Forest Report (F-measure,Recall,Precision) for our testing dataset:
              precision    recall  f1-score   support

           g       0.84      0.88      0.86      2006
           h       0.88      0.83      0.85      2007

    accuracy                           0.86      4013
   macro avg       0.86      0.86      0.86      4013
weighted avg       0.86      0.86      0.86      4013

Running Time :  1388.3461196422577
```

4) Naïve Bayes:

While Naïve Bayes model does not need hyperparameter tuning it is the fastest model 55ms but also with the lowest accuracy = 64.29%
Comparing it to decision tree model the last one is much better than the Naïve Bayes because it has a higher accuracy with the same approx. time.

```
Naive Bayes Misplaced Labels for our testing dataset:  1433
Naive Bayes Accuracy for our testing dataset: 64.29%
Naive Bayes Confusion Matrix for our testing dataset:
                Actual g     Actual h
Predicted g     1775    231
Predicted h     1202            805
Naive Bayes Report (F-measure,Recall,Precision) for our testing dataset:
              precision    recall  f1-score   support

           g       0.60      0.88      0.71      2006
           h       0.78      0.40      0.53      2007

    accuracy                           0.64      4013
   macro avg       0.69      0.64      0.62      4013
weighted avg       0.69      0.64      0.62      4013

Running Time :  0.055501699447631836
```

5) Decision Tree:

While Decision Tree model does not need hyperparameter tuning it is one of the fastest models 198ms and with a very reasonable accuracy = 79.34% this accuracy exceeds K-NN model.
Comparing it to the other this is one of the best models regarding the accuracy and execution time.

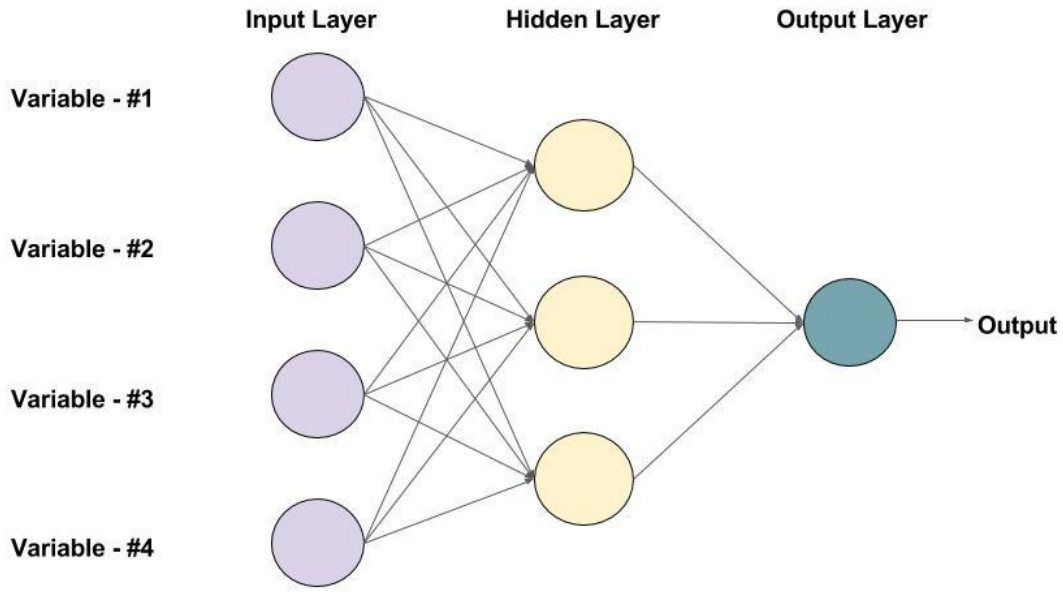
```
Decision Tree Misplaced Labels for our testing dataset:  829
Decision Tree Accuracy for our testing dataset: 79.34%
Decision Tree Confusion Matrix for our testing dataset:
                Actual g     Actual h
Predicted g     1611    395
Predicted h     434             1573
Decision Tree Report (F-measure,Recall,Precision) for our testing dataset:
              precision    recall  f1-score   support

           g       0.79      0.80      0.80      2006
           h       0.80      0.78      0.79      2007

    accuracy                           0.79      4013
   macro avg       0.79      0.79      0.79      4013
weighted avg       0.79      0.79      0.79      4013

Running Time :  0.1981360912322998
```

6) Comparison Table:

Model	Accuracy (Test Data)	Execution Time (Sec)
Random Forest	85.65%	1388.35
Adaboost	82.18%	905.65
Decision Tree	79.34%	0.198
K-NN	77.10%	99.5
Naïve Bayes	64.29%	0.055

7) Bonus Part:

We used input layer with 10 neurons (to match the 10 features) and in the hidden layer we used 64 neurons and # of generations = 50
Accuracy = 87.29% (Train Data), 85% (Test Data)
Execution time = 15 sec



An example of a Feed-forward Neural Network with one hidden layer (with 3 neurons)

```
--------------Start Training------------
Epoch 001: | Loss: 0.42776 | Acc: 80.347
Epoch 002: | Loss: 0.38160 | Acc: 82.796
Epoch 003: | Loss: 0.36862 | Acc: 83.293
Epoch 004: | Loss: 0.35927 | Acc: 83.748
Epoch 005: | Loss: 0.34844 | Acc: 84.490
Epoch 006: | Loss: 0.34499 | Acc: 84.776
Epoch 007: | Loss: 0.33936 | Acc: 84.925
Epoch 008: | Loss: 0.33428 | Acc: 85.177
Epoch 009: | Loss: 0.33743 | Acc: 84.789
Epoch 010: | Loss: 0.33394 | Acc: 85.204
Epoch 011: | Loss: 0.32933 | Acc: 85.102
Epoch 012: | Loss: 0.32816 | Acc: 85.490
Epoch 013: | Loss: 0.33098 | Acc: 85.456
Epoch 014: | Loss: 0.32602 | Acc: 85.007
Epoch 015: | Loss: 0.32360 | Acc: 85.687
Epoch 016: | Loss: 0.32171 | Acc: 85.905
Epoch 017: | Loss: 0.31998 | Acc: 85.571
Epoch 018: | Loss: 0.32028 | Acc: 85.891
Epoch 019: | Loss: 0.31984 | Acc: 86.082
Epoch 020: | Loss: 0.31179 | Acc: 86.170
Epoch 021: | Loss: 0.31951 | Acc: 85.796
Epoch 022: | Loss: 0.31226 | Acc: 86.122
Epoch 023: | Loss: 0.31527 | Acc: 86.170
Epoch 024: | Loss: 0.30812 | Acc: 86.211
Epoch 025: | Loss: 0.31250 | Acc: 85.878
Epoch 026: | Loss: 0.30971 | Acc: 86.435
Epoch 027: | Loss: 0.30769 | Acc: 86.286
Epoch 028: | Loss: 0.30696 | Acc: 86.197
Epoch 029: | Loss: 0.30289 | Acc: 86.633
Epoch 030: | Loss: 0.30593 | Acc: 86.361
Epoch 031: | Loss: 0.29887 | Acc: 87.027
Epoch 032: | Loss: 0.30417 | Acc: 86.497
Epoch 033: | Loss: 0.29618 | Acc: 86.912
Epoch 034: | Loss: 0.30023 | Acc: 86.442
Epoch 035: | Loss: 0.30486 | Acc: 86.381
Epoch 036: | Loss: 0.30112 | Acc: 86.585
Epoch 037: | Loss: 0.29969 | Acc: 86.769
Epoch 038: | Loss: 0.29684 | Acc: 86.952
Epoch 039: | Loss: 0.29356 | Acc: 87.354
Epoch 040: | Loss: 0.29365 | Acc: 86.980
Epoch 041: | Loss: 0.29394 | Acc: 87.265
Epoch 042: | Loss: 0.29746 | Acc: 86.673
Epoch 043: | Loss: 0.29512 | Acc: 87.122
Epoch 044: | Loss: 0.29594 | Acc: 86.816
Epoch 045: | Loss: 0.28908 | Acc: 87.245
Epoch 046: | Loss: 0.29204 | Acc: 87.429
Epoch 047: | Loss: 0.28832 | Acc: 87.293
Epoch 048: | Loss: 0.29124 | Acc: 86.966
Epoch 049: | Loss: 0.29111 | Acc: 87.027
Epoch 050: | Loss: 0.28862 | Acc: 87.299
```

----------Classification Report----------

	precision	recall	f1-score	support
0	0.88	0.81	0.85	2007
1	0.83	0.89	0.86	2006
accuracy			0.85	4013
macro avg	0.85	0.85	0.85	4013
weighted avg	0.85	0.85	0.85	4013

----------Confusion Matrix----------