**TASK 1:**

Here is how the training and testing data looks like:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 24 entries, 0 to 23

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 ID 24 non-null int64

1 Date 24 non-null object

2 Opponent 24 non-null object

3 Is\_Home\_or\_Away 24 non-null object

4 Is\_Opponent\_in\_AP25\_Preseason 24 non-null object

5 Media 24 non-null object

6 Label 24 non-null object

dtypes: int64(1), object(6)

memory usage: 1.4+ KB

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 12 entries, 0 to 11

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 ID 12 non-null int64

1 Date 12 non-null object

2 Opponent 12 non-null object

3 Is\_Home\_or\_Away 12 non-null object

4 Is\_Opponent\_in\_AP25\_Preseason 12 non-null object

5 Media 12 non-null object

6 Label 12 non-null object

dtypes: int64(1), object(6)

memory usage: 800.0+ bytes

Q1) Here are the definitions of Accuracy, Precision, Recall and F1 score:

Accuracy = (TP+TN)/(TP+FP+FN+TN)

Precision = TP/(TP+FP)

Recall = TP/(TP+FN)

F1 Score = 2\*(Recall \* Precision) / (Recall + Precision)

Where, TP is true positives, TN is true negatives, FP is false positives and FN is false negatives.

Based on the Naïve Bayes and KNN prediction, Accuracy, Precision, Recall and F1 score are calculated.

Naive Bayes Accuracy is: 0.833

KNN Accuracy is: 0.667

Naive Bayes Precision is: 1.000

KNN Precision is: 0.727

Naive Bayes Recall is: 0.778

KNN Recall is: 0.889

Naive Bayes F1\_score is: 0.875

KNN F1\_score is: 0.800

Q2) Naïve Bayes prediction output:

Given y labels: [1 0 1 1 1 1 1 1 1 0 1 0]

Naive Bayes: Predicted y labels: [1 0 1 1 1 0 0 1 1 0 1 0]

KNN prediction output:

Given y labels: [1 0 1 1 1 1 1 1 1 0 1 0]

KNN: Predicted y labels: [1 1 1 1 1 1 1 1 0 1 1 1]

In Naïve Bayes prediction, 2 values were predicted wrong. In KNN prediction, 4 values were predicted wrong.

**TASK 2:**

Here is how the training and testing data set looks like for the Titanic dataset:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 891 entries, 0 to 890

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 PassengerId 891 non-null int64

1 Survived 891 non-null int64

2 Pclass 891 non-null int64

3 Name 891 non-null object

4 Sex 891 non-null object

5 Age 714 non-null float64

6 SibSp 891 non-null int64

7 Parch 891 non-null int64

8 Ticket 891 non-null object

9 Fare 891 non-null float64

10 Cabin 204 non-null object

11 Embarked 889 non-null object

dtypes: float64(2), int64(5), object(5)

memory usage: 83.7+ KB

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 418 entries, 0 to 417

Data columns (total 11 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 PassengerId 418 non-null int64

1 Pclass 418 non-null int64

2 Name 418 non-null object

3 Sex 418 non-null object

4 Age 332 non-null float64

5 SibSp 418 non-null int64

6 Parch 418 non-null int64

7 Ticket 418 non-null object

8 Fare 417 non-null float64

9 Cabin 91 non-null object

10 Embarked 418 non-null object

dtypes: float64(2), int64(4), object(5)

memory usage: 36.0+ KB

Some data processing is done before prediction. Empty values of Embarked and Age feature in the training dataset are filled with mode values of the features. Similarly, empty values of the Fare feature in the testing dataset is filled with the mode value of the Fare feature. Cabin feature in both training and testing dataset is dropped since majority of the features are empty. Name and Ticket features from both the training set and the testing set are also dropped as there is no strong correlation with the Survived feature

Q1) The average accuracy, precision, recall and F1 score over five folds using naïve Bayes algorithm is:

Average metrics over five folds:

The average accuracy is: 0.7621

The average precision is: 0.6666

The average recall is: 0.7660

The average f1\_score is: 0.7112

The overall accuracy, precision, recall and F1 score for the entire dataset is:

NAIVE BAYES METRICS:

The accuracy on the entire model is: 0.8995

The precision on the entire model is: 0.8056

The recall on the entire model is: 0.9539

The f1 score on the entire model is: 0.8735

Based on my implementation of the Naïve Bayes algorithm on the football dataset and the Titanic dataset, it was observed that:

Naive Bayes for Football: Accuracy = 83.33%

Naive Bayes for Titanic: Accuracy = 89.95%

Even though the general idea is that Naïve Bayes performs better with smaller datasets, in my implementation better accuracy was found for the Titanic dataset compared to the football dataset. Hence, according to my implementation, Naïve Bayesian model performed better for the larger dataset.

Q2) KNN is implemented from scratch and is used to predict values on the titanic dataset. Five-fold cross validation is done on the Titanic training dataset and for different K values, average accuracies over all five-folds are plotted. Below are the accuracy values for different K values.

Graphs were plotted for K = 1 to 25, K = 1 to 50 and K = 1 to 100 and it was observed that the best K value is at K = 14 with an accuracy of 69.93%

1. K = 1 to 25

Chart, line chart

Description automatically generated

1. K = 1 to 50

Chart, line chart

Description automatically generated

K = 1 to 100:

Chart, line chart

Description automatically generated

The accuracy, precision, recall and F1 score on the entire dataset by taking K = 14 is as follows:

KNN METRICS:

The accuracy on the entire model is: 0.6411

The precision on the entire model is: 0.5116

The recall on the entire model is: 0.2895

The f1 score on the entire model is: 0.3697

Q3) According to my algorithm implementation and feature engineering, it was observed that the Naïve Bayes had better accuracy compared to KNN for the Titanic dataset.

Naive Bayes for Titanic: Accuracy = 89.95%  
KNN for Titanic: Accuracy = 61.72%

This could be due to a variety of reasons including feature engineering and the way data is massaged before passing it to these algorithms. Also, other reasons could include a) fact that KNN’s decision boundary can take on any form since KNN is non-parametric and it makes no assumption about the data distribution b) KNN doesn’t know which attributes are more important as each attribute normally weighs the same to the total Euclidean distance.

GitHub Link: