Logo

Description automatically generated

Ain Shams University, Faculty of Engineering

Milestone 1 Report

**27 December 2022**

Prepared by: Abdulrahman Ayman, Ahmed Sameh, ElSaeed Ahmed ElSaeed

Contents

[1)Function Implementation: 3](#_Toc123076842)

[Naïve Bayes Classifier: 3](#_Toc123076843)

[Naïve Bayes Tuning: 5](#_Toc123076844)

[KNN Classifier: 8](#_Toc123076845)

[KNN Tuning: 10](#_Toc123076846)

[2)Running Project: 12](#_Toc123076847)

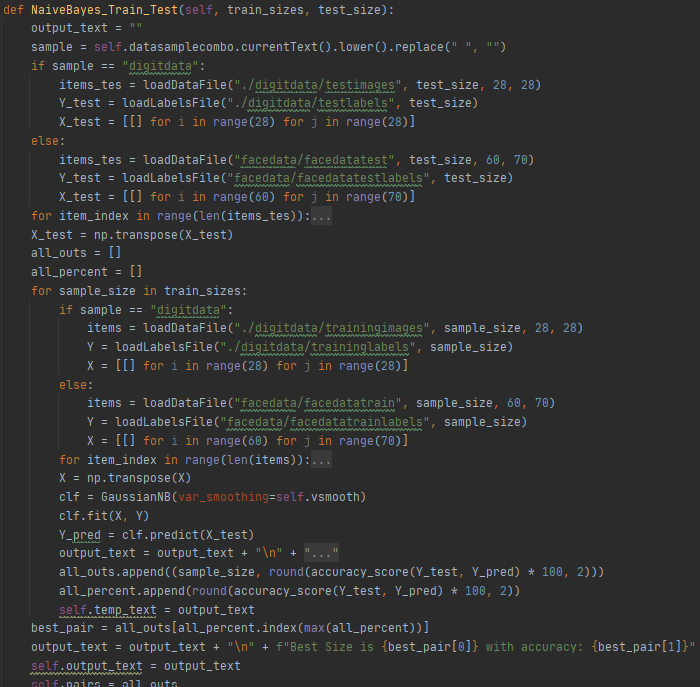
[Main Screen: 12](#_Toc123076848)

[Naïve Bayes Functions: 13](#_Toc123076849)

[KNN Functions: 17](#_Toc123076850)

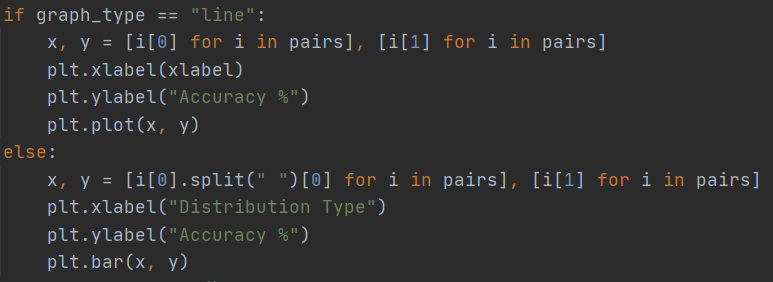
# 1)Function Implementation:

## Naïve Bayes Classifier:



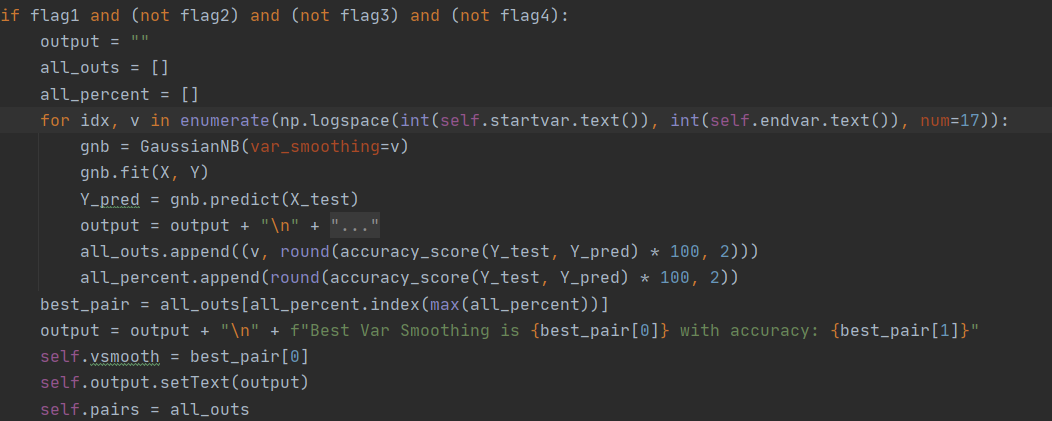
Steps:

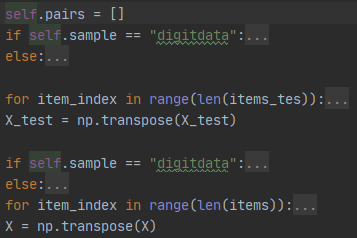
1. Read either digitdata or facedata to get X\_test and Y\_test using loadDataFile & LoadLabelsFile provided in the samples.py file
2. Cycle through all sample sizes that are given each time solving classification
3. Once inside for loop load training data either digitdata or facedata to get X and Y using loadDataFile & LoadLabelsFile provided in the samples.py file
4. Create input equal to size of rows \* columns and process the loaded data through it
5. Initialize a Gaussian Naïve Bayes Using The implemented version from the SkLearn Library
6. Fit X and Y to the GaussianNB and predict Y\_Pred using X\_test
7. Check it against Y\_test to get both accuracy and precision using function provided by the SkLearn Library
8. Append all outputs to array to get the best one in the end



1. Graph the result using matplotlib Library

## Naïve Bayes Tuning:

Text

Description automatically generated 

Steps:

1. Read either digitdata or facedata to get X\_test and Y\_test using loadDataFile & LoadLabelsFile provided in the samples.py file (same as all other functions)
2. load training data either digitdata or facedata to get X and Y using loadDataFile & LoadLabelsFile provided in the samples.py file (same as all other functions)
3. If we are tuning only gaussian distribution, then we cycle through all v smoothing values that are given each time solving classification
   1. Initialize a GaussianNB Using The implemented version from the SkLearn Library with var smoothing the current var smoothing
   2. Fit X and Y to the GaussianNB and predict Y\_Pred using X\_test
   3. Check it against Y\_test to get both accuracy and precision using function provided by the SkLearn Library
   4. Append all outputs to array to get the best one in the end with the best one being the new var smoothing value
4. If we are comparing distributions
   1. we Initialize Distributions for each one checked Using The implemented version from the SkLearn Library
   2. Fit X and Y for all to the distributions and predict Y\_Pred using X\_test
   3. Check it against Y\_test to get both accuracy and precision for each one using function provided by the SkLearn Library
   4. Append all outputs to array to get the best one in the end with the best one being the best distribution

Text

Description automatically generated

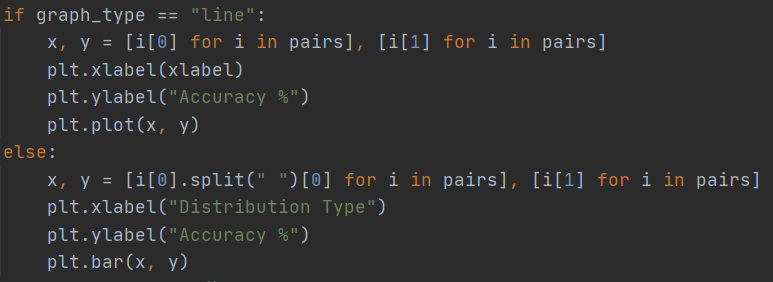
1. Graph the result using matplotlib Library

## KNN Classifier:



Steps:

1. Read either digitdata or facedata to get X\_test and Y\_test using loadDataFile & LoadLabelsFile provided in the samples.py file
2. Cycle through all sample sizes that are given each time solving classification
3. Once inside for loop load training data either digitdata or facedata to get X and Y using loadDataFile & LoadLabelsFile provided in the samples.py file
4. Create input equal to size of rows \* columns and process the loaded data through it
5. Initialize a KNeighborsClassifier Using The implemented version from the SkLearn Library with default K = 2 & Euclidean metric
6. Fit X and Y to the KNeighborsClassifier and predict Y\_Pred using X\_test
7. Check it against Y\_test to get both accuracy and precision using function provided by the SkLearn Library
8. Append all outputs to array to get the best one in the end



Graph the result using matplotlib Library

## KNN Tuning:

Text

Description automatically generated

Steps:

1. Read either digitdata or facedata to get X\_test and Y\_test using loadDataFile & LoadLabelsFile provided in the samples.py file
2. load training data either digitdata or facedata to get X and Y using loadDataFile & LoadLabelsFile provided in the samples.py file
3. Cycle through all K neighbours that are given each time solving classification
4. Initialize a KNeighborsClassifier Using The implemented version from the SkLearn Library with K the current K & Euclidean metric
5. Fit X and Y to the KNeighborsClassifier and predict Y\_Pred using X\_test
6. Check it against Y\_test to get both accuracy and precision using function provided by the SkLearn Library
7. Append all outputs to array to get the best one in the end with the best one being the new K value

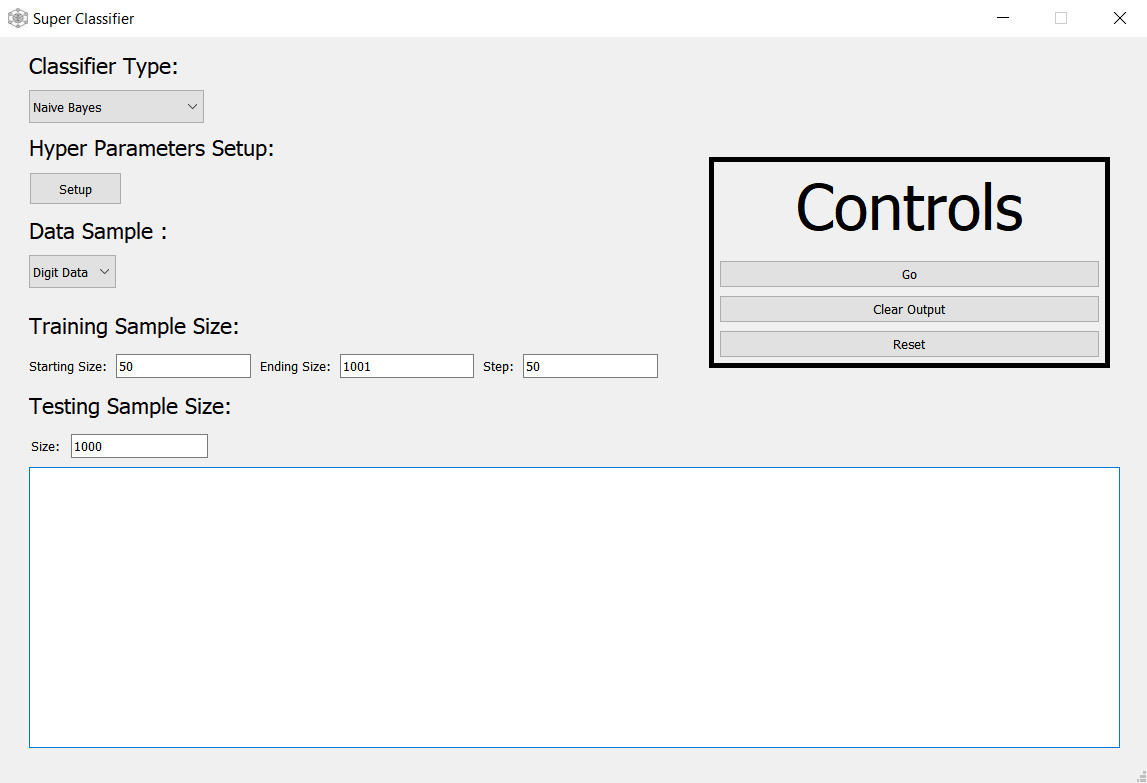
Text

Description automatically generated

Graph the result using matplotlib Library

# 2)Running Project:

## Main Screen:



Here we have the main screen:

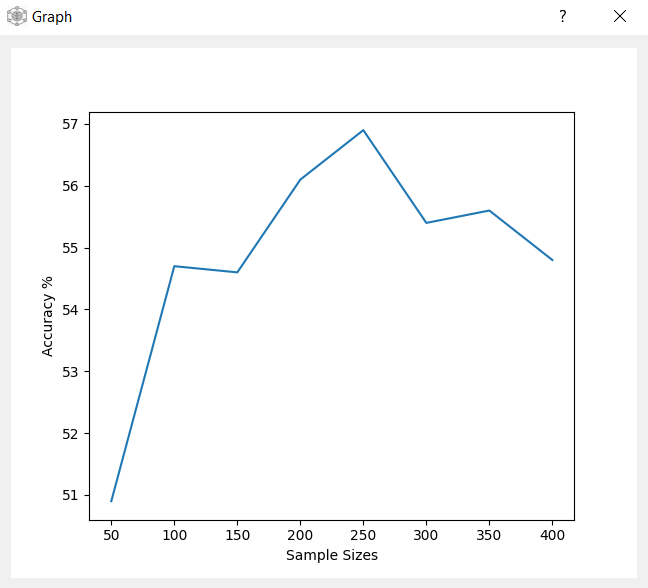
* we can choose classifier type from comb box
* we can setup hyperparameter from setup button
* we can choose which sample to use from data sample
* we can set variable training size by giving start, end and step in the text boxes which will be transferred to format range(start, end, step) in code
* we can set variable testing size in the text box

## Naïve Bayes Functions:

Graphical user interface, text, application, email

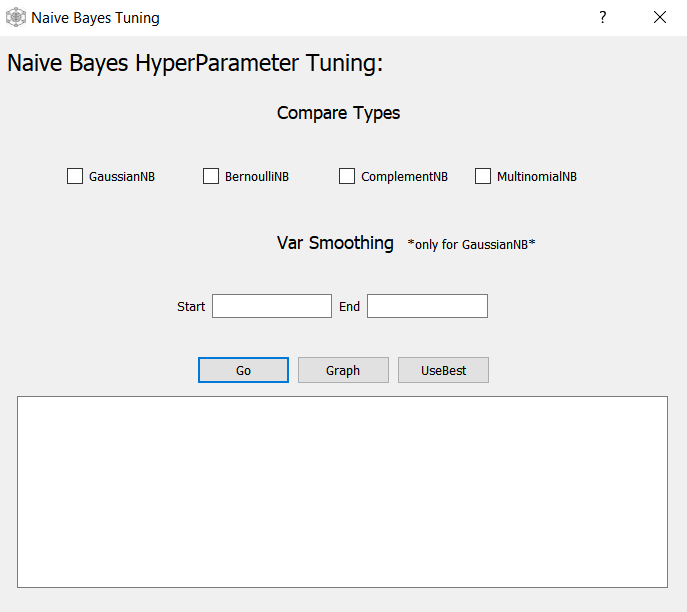
Description automatically generated

* By pressing Go we get output of Gaussian Naïve bayes for all training sizes we have set:

 Text

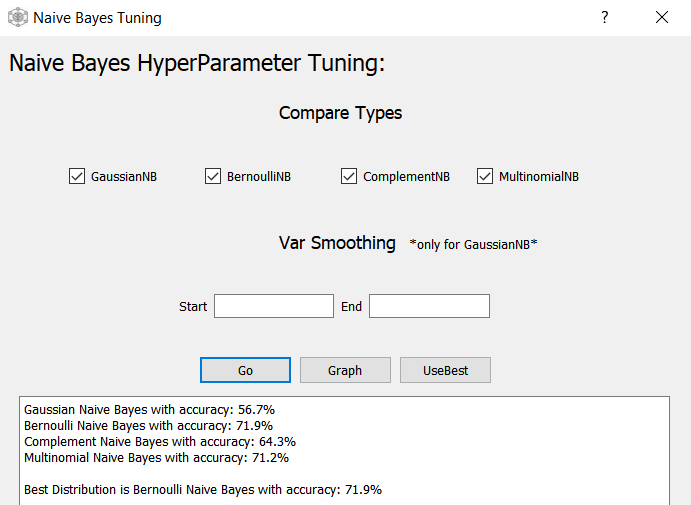
Description automatically generated

* By pressing Clear we clear output & by pressing reset will reset all variables and all textboxes
* By pressing Hyper parameter Setup Button:



We get the following screen where we can choose different distribution types to compare their accuracy you can either:

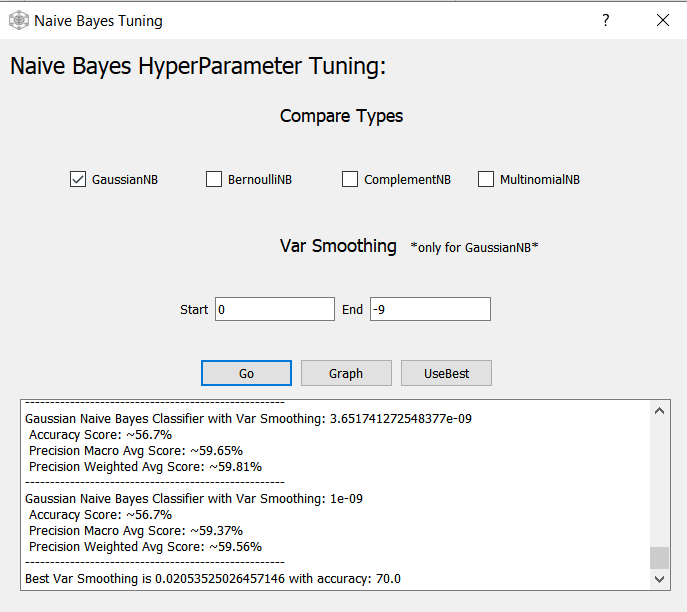
1. Compare by checking multiple distributions

 Chart, bar chart

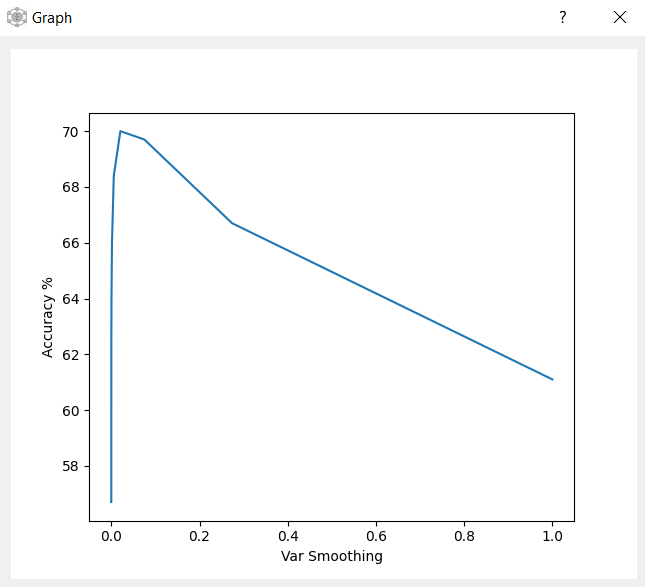
Description automatically generated

Note: if you want to graph press go first then press graph

1. Tune the var smoothing parameter by choosing GaussianNB and entering start and end

 Table

Description automatically generated



Note: By pressing UseBest Button you will return to main screen and the best var smoothing will be applied

Before:

Chart, line chart

Description automatically generated Text

Description automatically generated

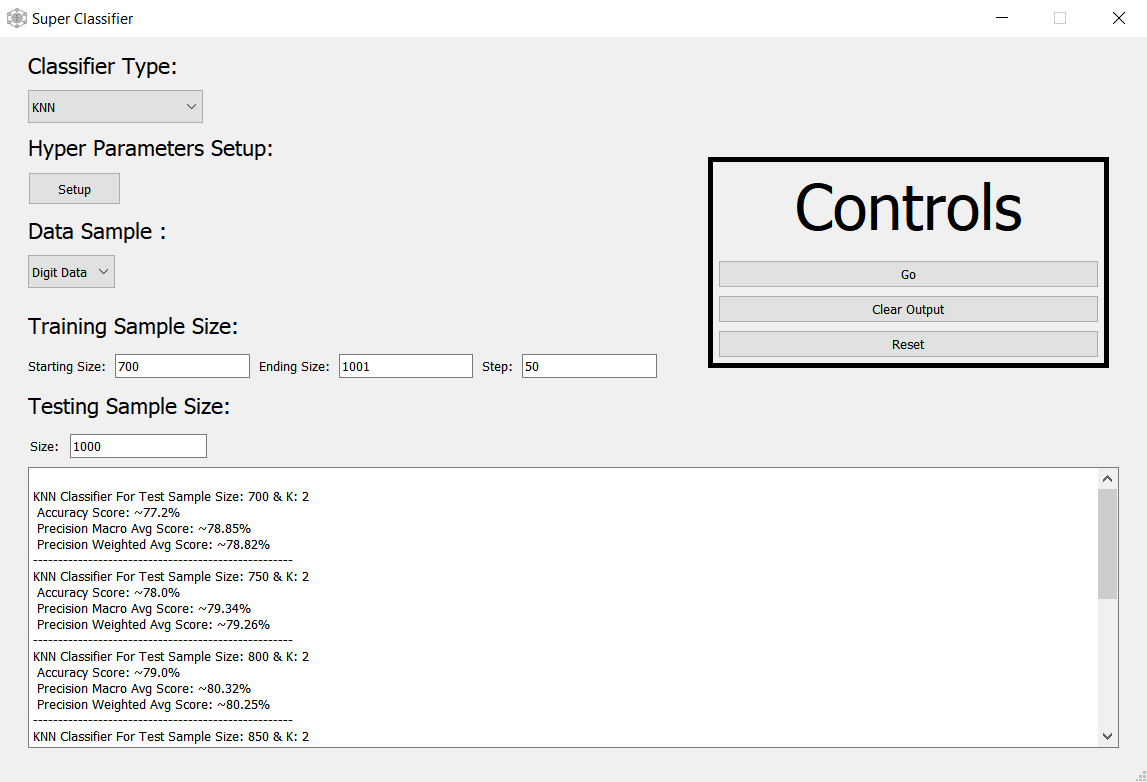
After:

Chart, line chart

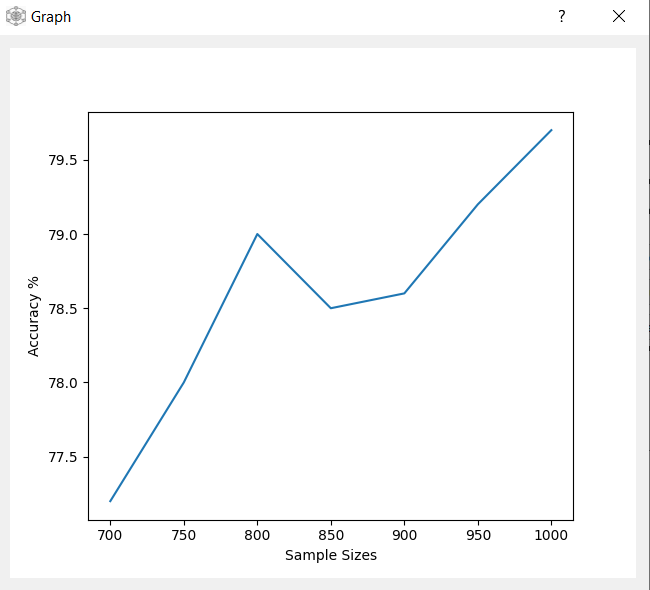
Description automatically generated Text

Description automatically generated

## KNN Functions:

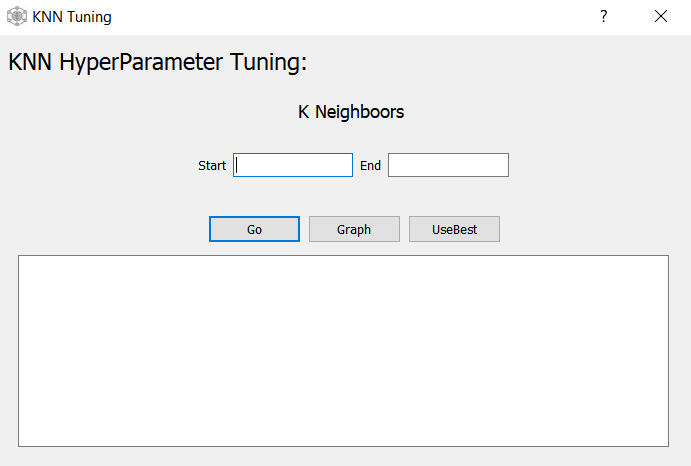


* By pressing Go we get output of KNN Classification results for all training sizes we have set and a default K value of 2:

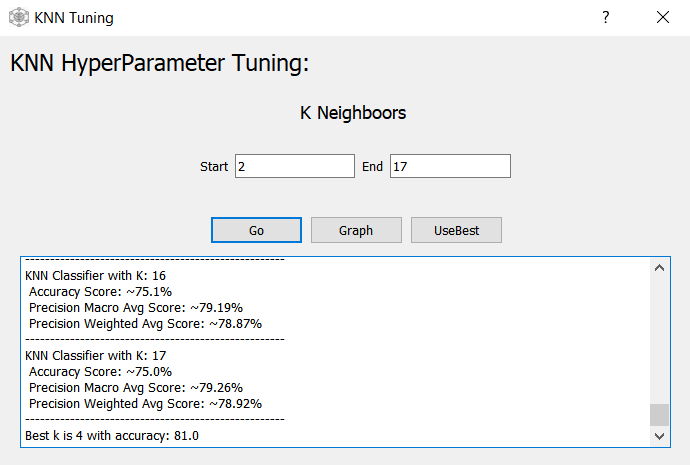
 Text

Description automatically generated

* By pressing Clear we clear output & by pressing reset will reset all variables and all textboxes
* By pressing Hyper parameter Setup Button:



We get the following screen where we test a range of values for the nearest neighbour hyper parameter by entering start and end then pressing

 Chart, line chart

Description automatically generated

Note: By pressing UseBest Button you will return to main screen and the best K will be applied

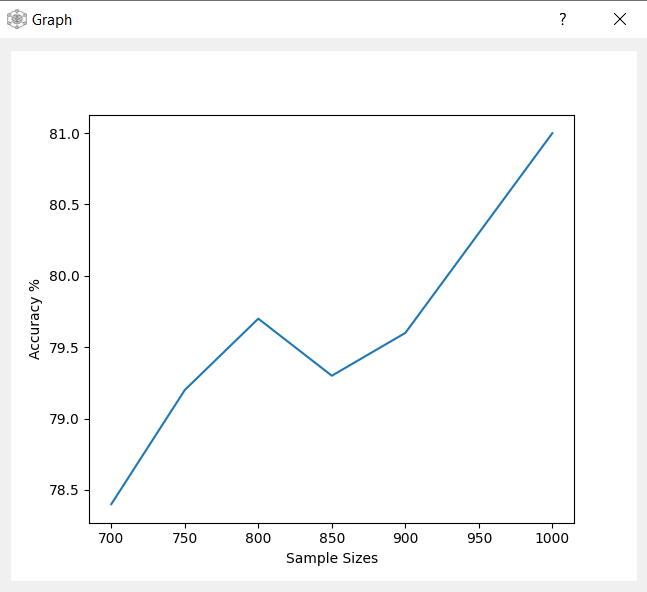
Before:

Chart, line chart

Description automatically generated Text

Description automatically generated

After:

 Text

Description automatically generated