Logo

Description automatically generated

Ain Shams University, Faculty of Engineering

Milestone 2 Report

**9 January 2023**

Prepared by: Abdulrahman Ayman 19P6458

Ahmed Sameh 19P5861

ElSaeed Ahmed ElSaeed 19P1087

Contents

[**1)Function Implementation:** 3](#_Toc124189652)

[**MLP Classifier:** 3](#_Toc124189653)

[**MLP Tuning:** 4](#_Toc124189654)

[**SVM Classifier:** 5](#_Toc124189655)

[**SVM Tuning:** 6](#_Toc124189656)

[**Decision Tree Classifier:** 7](#_Toc124189657)

[**Decision Tree Tuning:** 8](#_Toc124189658)

[**2)Running Project:** 10](#_Toc124189659)

[**MLP Functions:** 10](#_Toc124189660)

[**SVM Functions:** 13](#_Toc124189661)

[**Decision Tree Functions:** 16](#_Toc124189662)

[**3)GitHub Link:** 20](#_Toc124189663)

# **1)Function Implementation:**

## **MLP Classifier:**

Text

Description automatically generated

Steps:

1. Load data using self.load\_test\_data() function same as Milestone 1
2. Initialize a MLPClassifier Using The implemented version from the SkLearn Library
3. Fit X and Y to the MLPClassifier and predict Y\_Pred using X\_test
4. Check it against Y\_test to get both accuracy and precision using function provided by the SkLearn Library
5. Append all outputs to array to get the best one in the end
6. Graph the result using matplotlib Library same as Milestone 1

## **MLP Tuning:**

Text

Description automatically generated

Text

Description automatically generated

Steps:

1. Load data using self.load\_test\_data() function same as Milestone 1
2. By checking certain checkboxes user adds more hyperparameters to tune
3. We then preform an exhaustive search on these parameters to find the best combination using multithreading for faster computation
   1. Initialize a MLPClassifier Using The implemented version from the SkLearn Library with current parameters
   2. Fit X and Y to the MLPClassifier 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 hyperparameters
4. Graph the result using matplotlib Library same as Milestone 1

## **SVM Classifier:**

Text

Description automatically generated

Steps:

1. Load data using self.load\_test\_data() function same as Milestone 1
2. Initialize a SVC Classifier Using The implemented version from the SkLearn Library
3. Fit X and Y to the SVC and predict Y\_Pred using X\_test
4. Check it against Y\_test to get both accuracy and precision using function provided by the SkLearn Library
5. Append all outputs to array to get the best one in the end
6. Graph the result using matplotlib Library same as Milestone 1

## **SVM Tuning:**

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Steps:

1. Load data using self.load\_test\_data() function same as Milestone 1
2. By checking certain checkboxes user adds more hyperparameters to tune
3. We then preform an exhaustive search on these parameters to find the best combination using multithreading for faster computation
   1. Initialize an SVC Classifier Using The implemented version from the SkLearn Library with current parameters
   2. Fit X and Y to the SVC 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 hyperparameters
4. Graph the result using matplotlib Library same as Milestone 1

## **Decision Tree Classifier:**

Text

Description automatically generated

Steps:

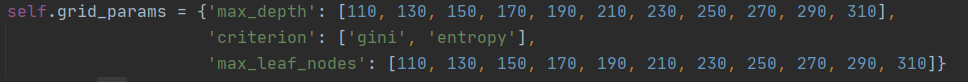
1. Load data using self.load\_test\_data() function same as Milestone 1
2. Initialize a DecisionTreeClassifier Using The implemented version from the SkLearn Library
3. Fit X and Y to the DecisionTreeClassifier and predict Y\_Pred using X\_test
4. Check it against Y\_test to get both accuracy and precision using function provided by the SkLearn Library
5. Append all outputs to array to get the best one in the end
6. Graph the result using matplotlib Library same as Milestone 1 for accuracy and using sklearn’s plot\_tree() function for plotting the decision tree

Text

Description automatically generated

## **Decision Tree Tuning:**

Text

Description automatically generatedText

Description automatically generated

Steps:

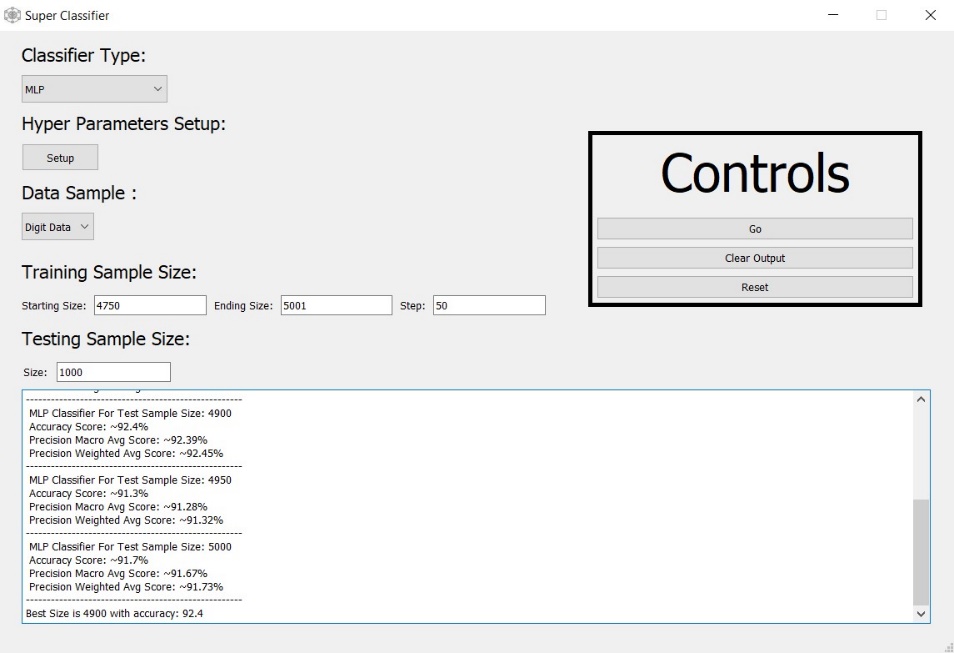
1. Load data using self.load\_test\_data() function same as Milestone 1
2. By inputting specific starts and ends for hyper parameters user can use manual tuning with their choosing values and pressing manual tune, Or by pressing auto tune directly and tuning using pre-existing values
3. We then preform an exhaustive search on these parameters to find the best combination using multithreading for faster computation
   1. Initialize an DecisionTreeClassifier Using The implemented version from the SkLearn Library with current parameters
   2. Fit X and Y to the DecisionTreeClassifier 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 hyperparameters
4. Graph the result using matplotlib Library same as Milestone 1 for accuracy and using sklearn’s plot\_tree() function for plotting the decision tree

Text

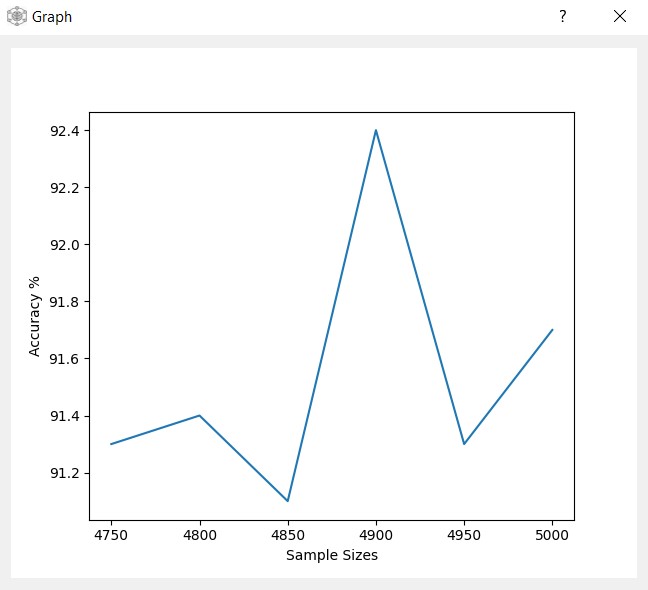
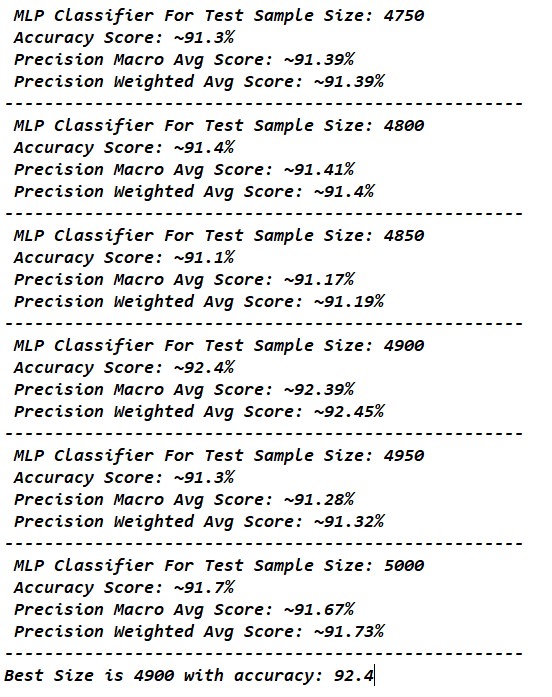
Description automatically generated

# **2)Running Project:**

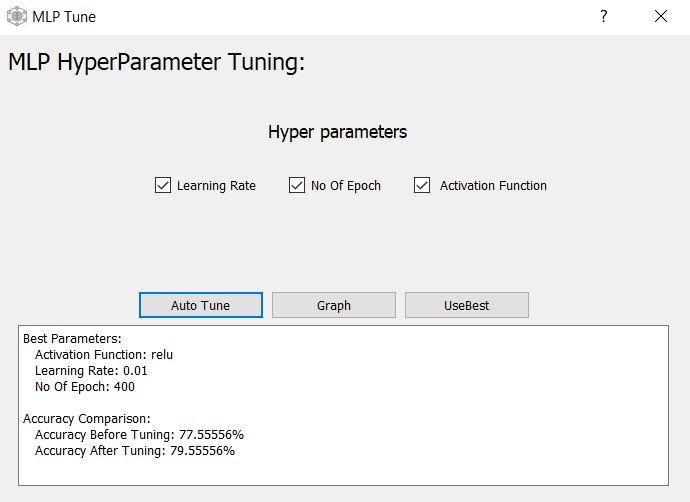
## **MLP Functions:**



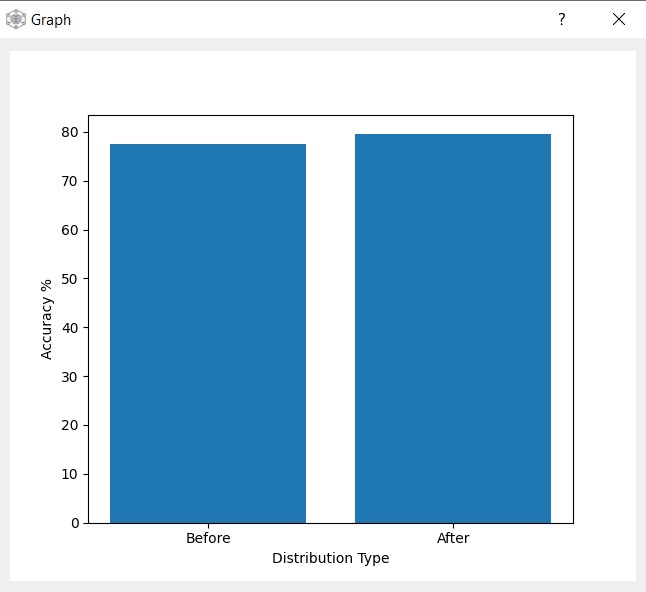
* By pressing Go we get output of MLP Classification results for all training sizes we have set and default hyperparameters:

* By pressing Hyper parameter Setup Button:

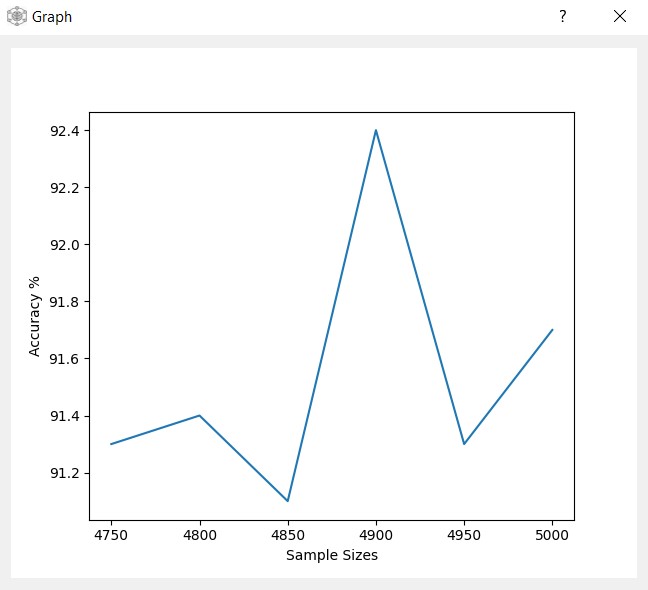


We get the following screen where we can auto tune MLP using list of values defined in code to get best parameters



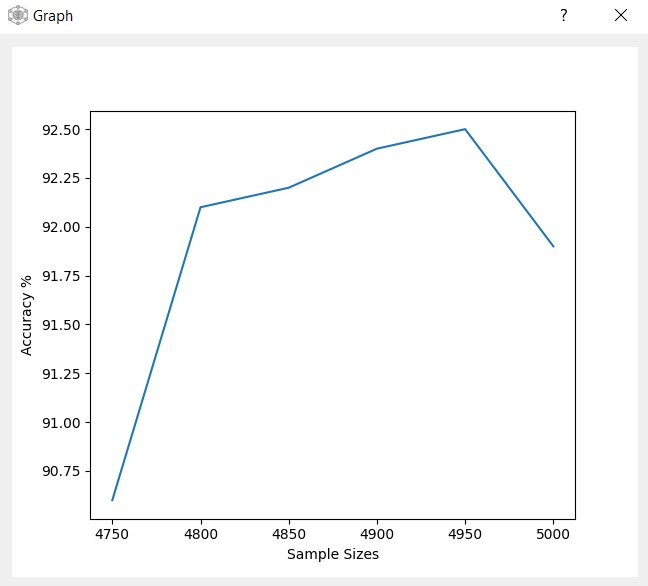
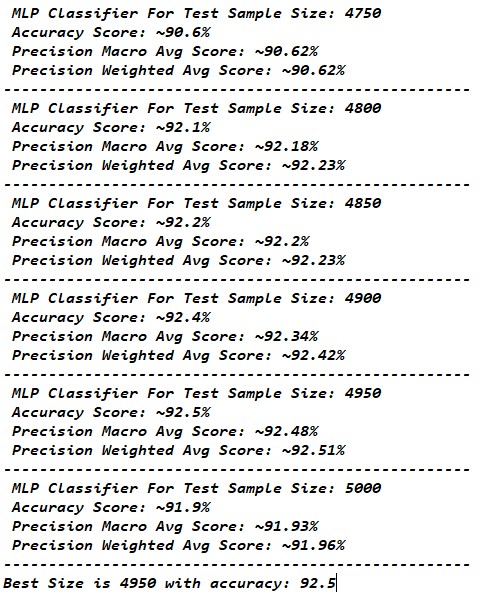
Note: By pressing UseBest Button you will return to main screen and the best parameters will be applied

Before:

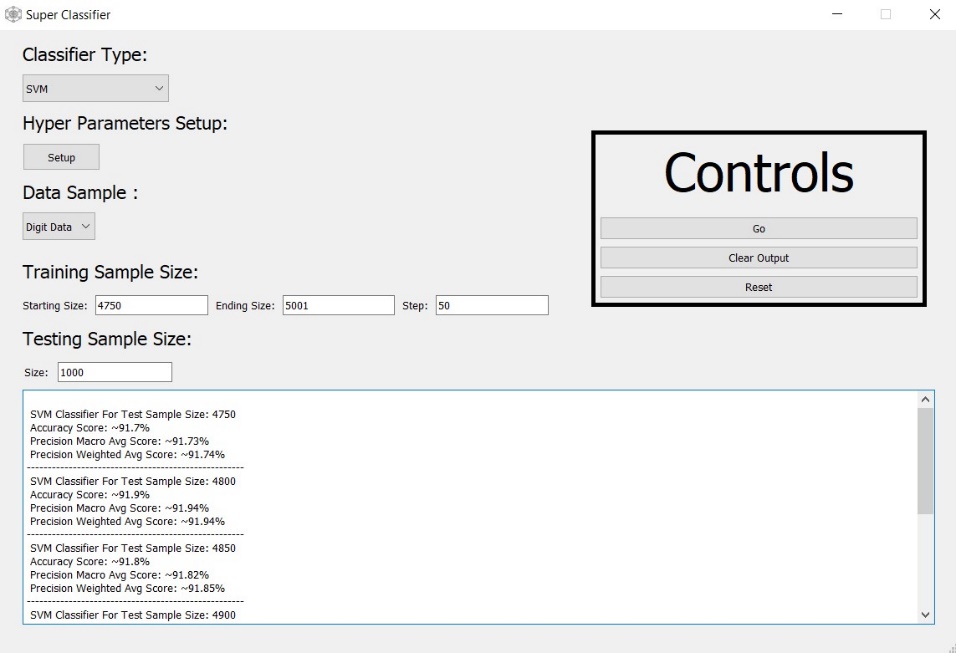
 Text

Description automatically generated

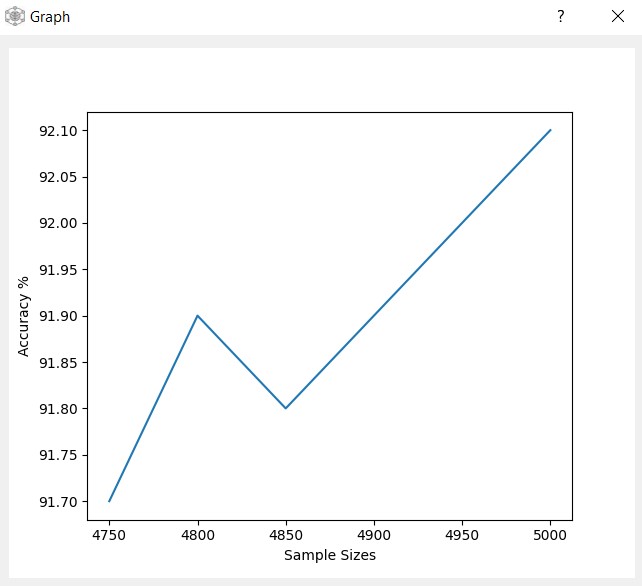
After:

## **SVM Functions:**

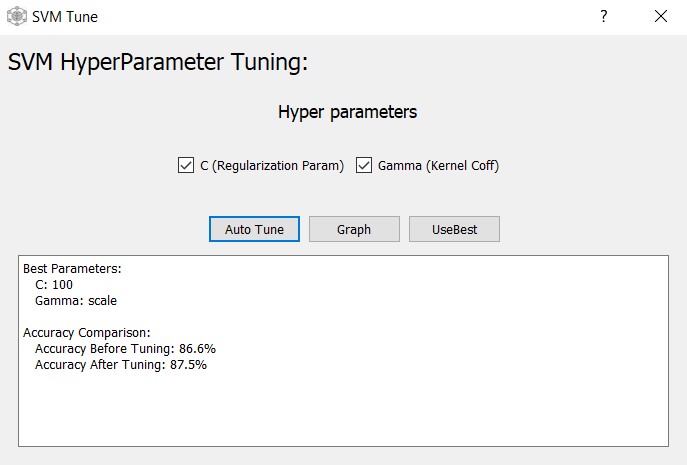


* By pressing Go we get output of SVM Classification results for all training sizes we have set and default hyperparameters:

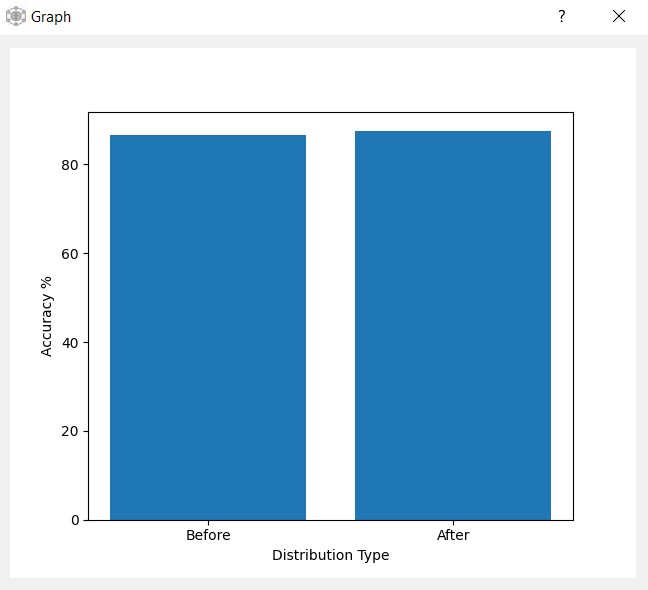
 Text

Description automatically generated

* By pressing Hyper parameter Setup Button:

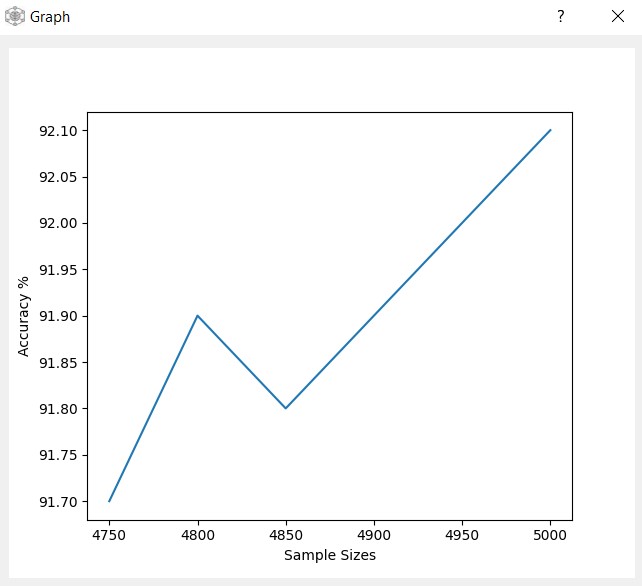


We get the following screen where we can auto tune MLP using list of values defined in code to get best parameters



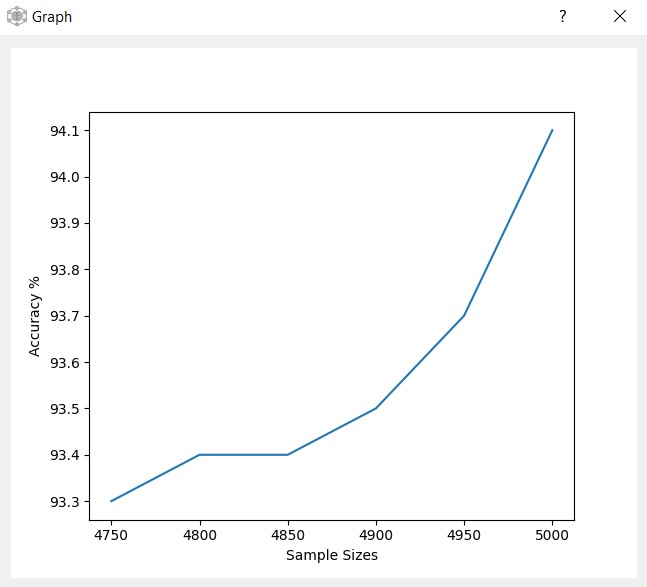
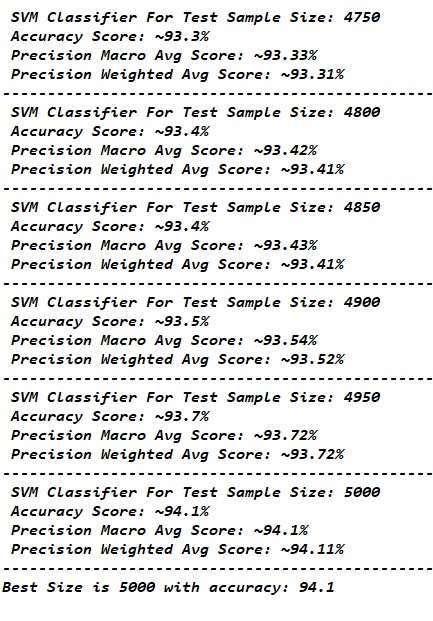
Note: By pressing UseBest Button you will return to main screen and the best parameters will be applied

Before:

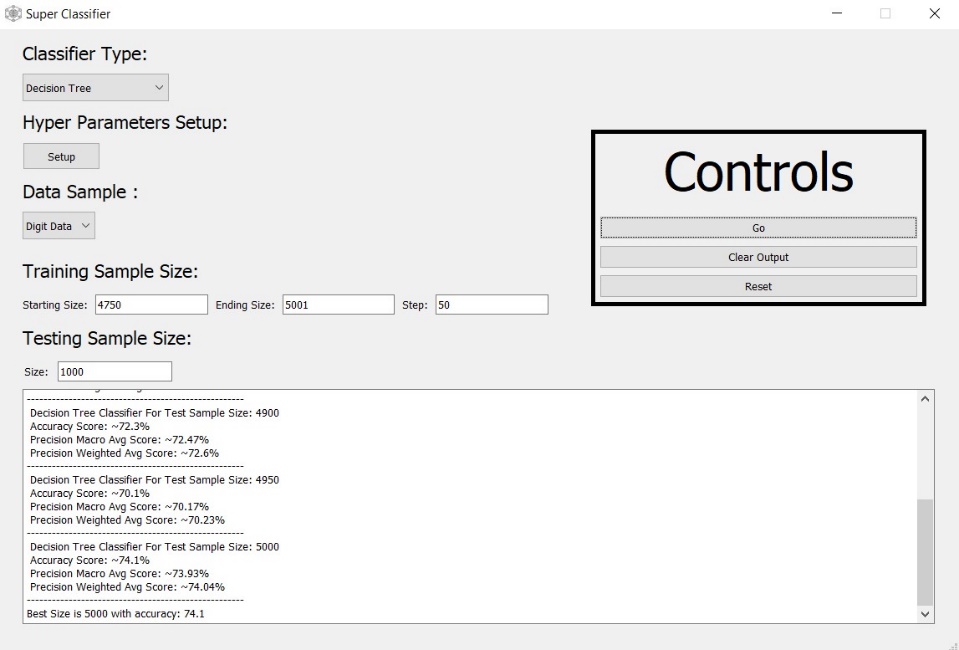
 Text

Description automatically generated

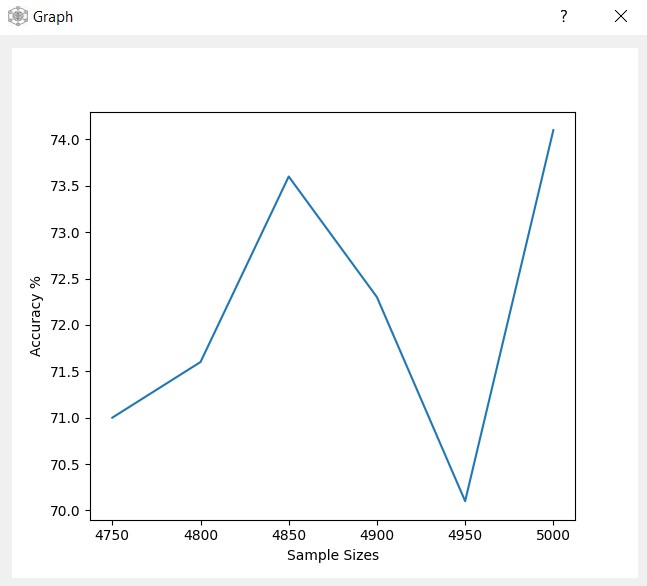
After:

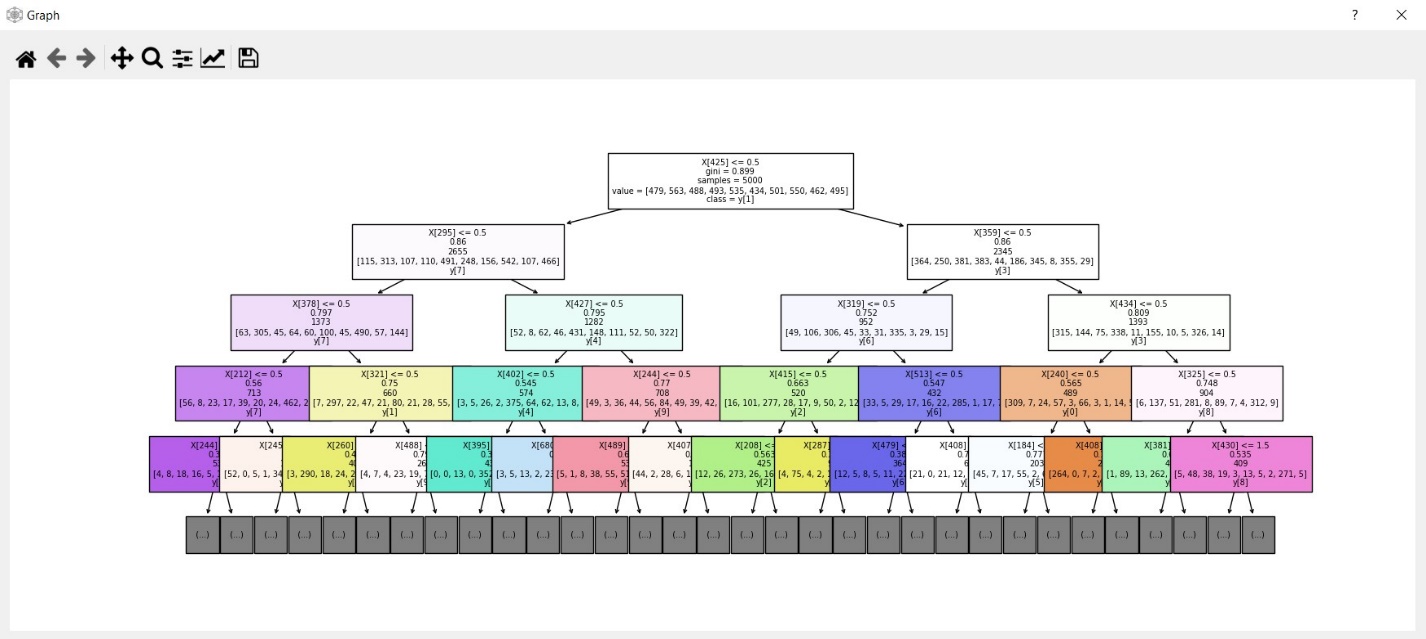
## **Decision Tree Functions:**



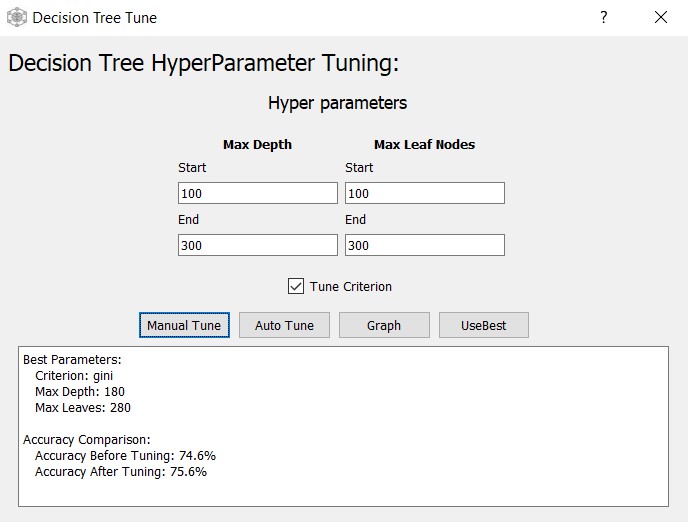
* By pressing Go we get output of Decision Tree Classification results for all training sizes we have set and default hyperparameters:

 Text

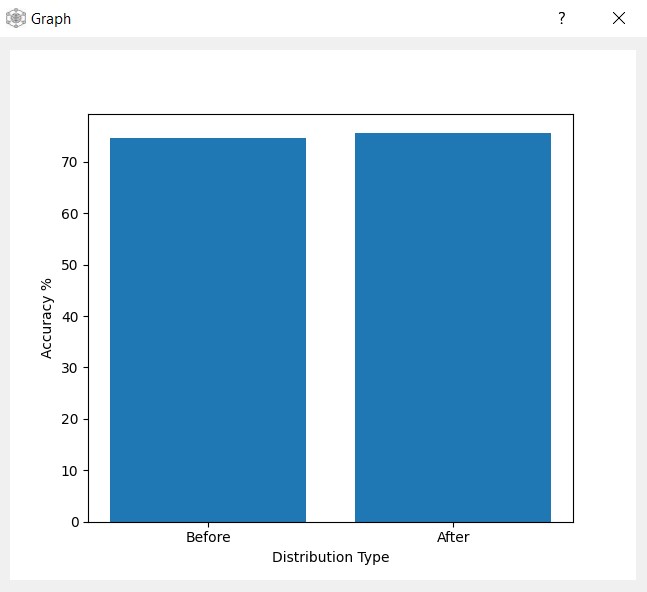
Description automatically generated

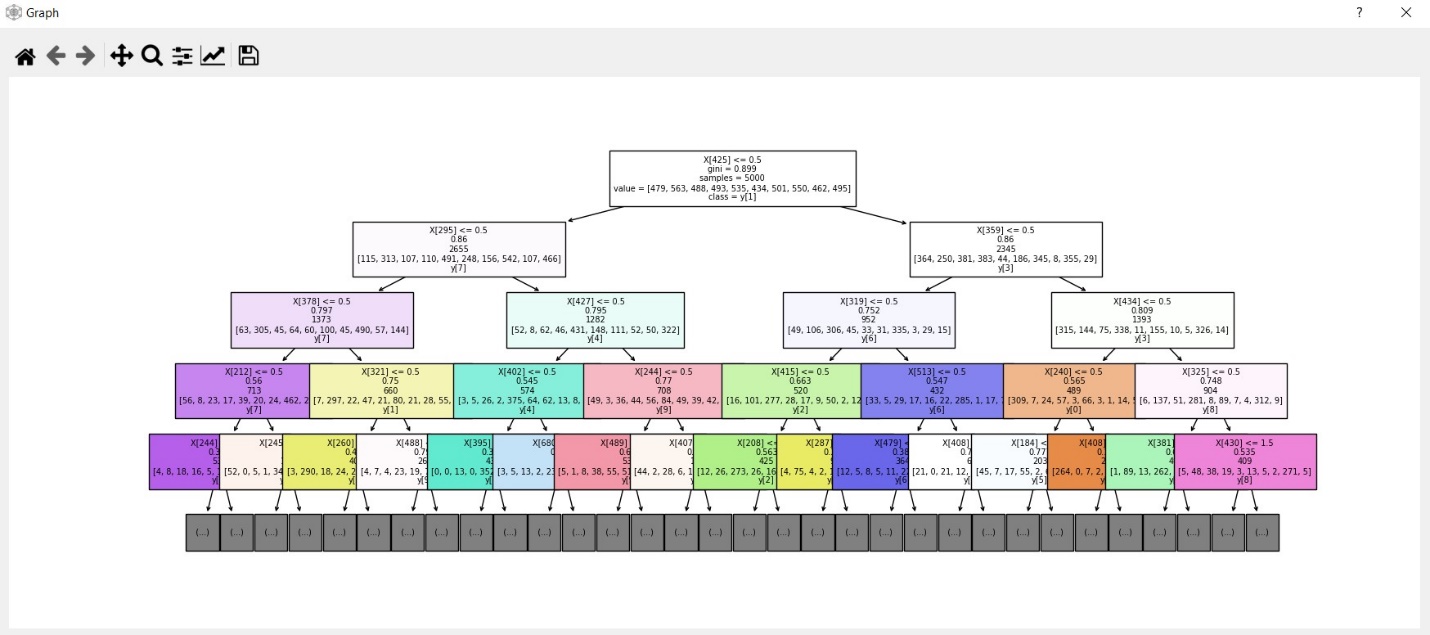


* By pressing Hyper parameter Setup Button:



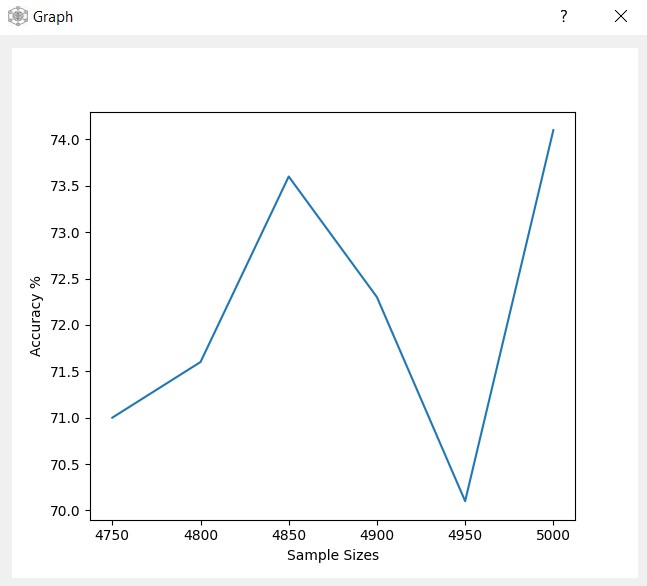
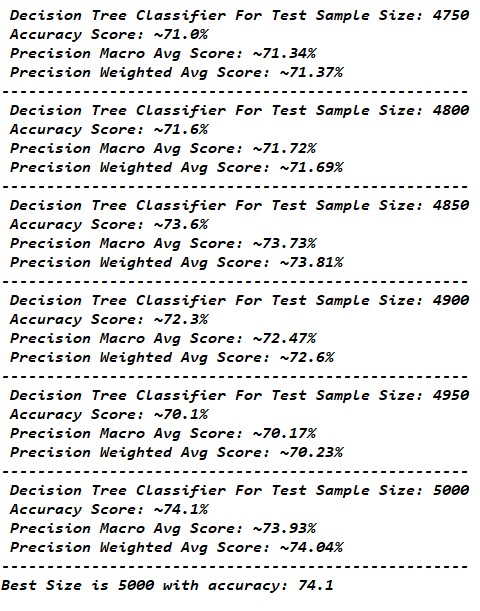
We get the following screen where we can auto tune Decision Tree either using list of values defined in code to get best parameters or using manually inputted values



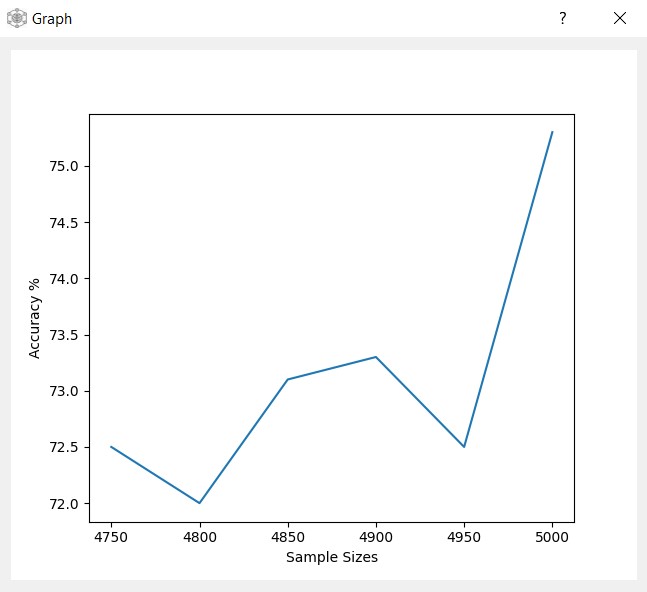
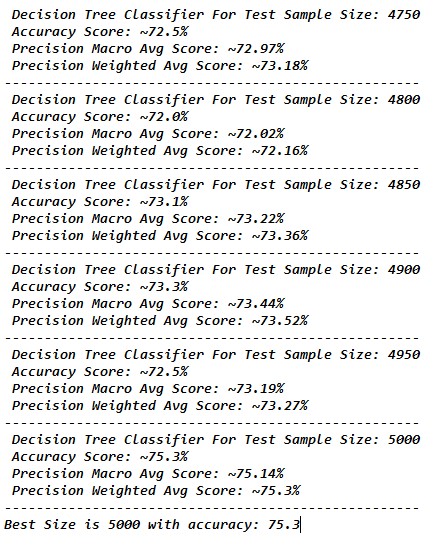


Note: By pressing UseBest Button you will return to main screen and the best parameters will be applied

Before:

After:

# **3)GitHub Link:**

<https://github.com/Abduaws/Machine-Learning-Algorithms>