C S 487/519 Applied Machine Learning I

Fall 2018

Project 3: Compare classifiers in scikit-learn library

Joanna Augustine

800656114

Objective: To understand and compare several classification algorithms that are provided by the Python scikit-learn library.

Perceptron:

Firstly the perceptron is implemented using the scikit-library.This is firstly done for the digits dataset.

The prediction of the various digits is achieved.

Accuracy:96% is achieved with the help of perceptron and the total running time is 25.69 seconds.I have used both command prompt and Pycharm to compare the time and analyze it accordingly.

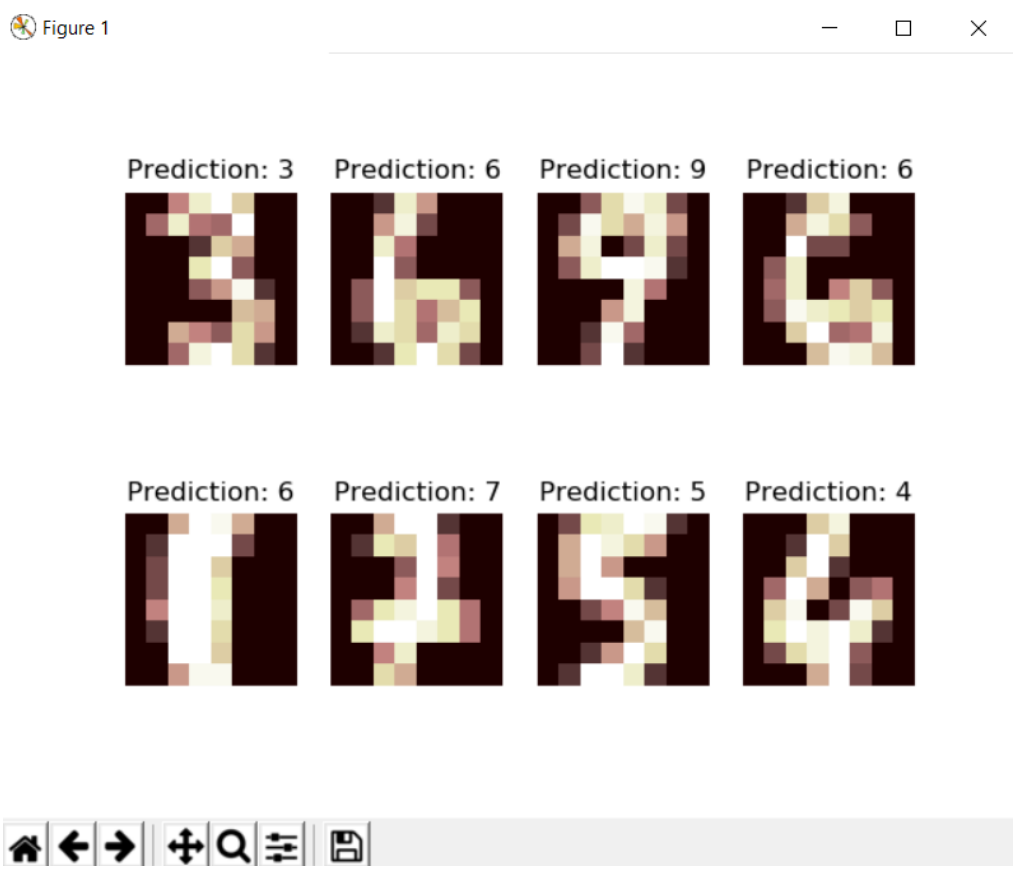


Fig1: Prediction of the various digits

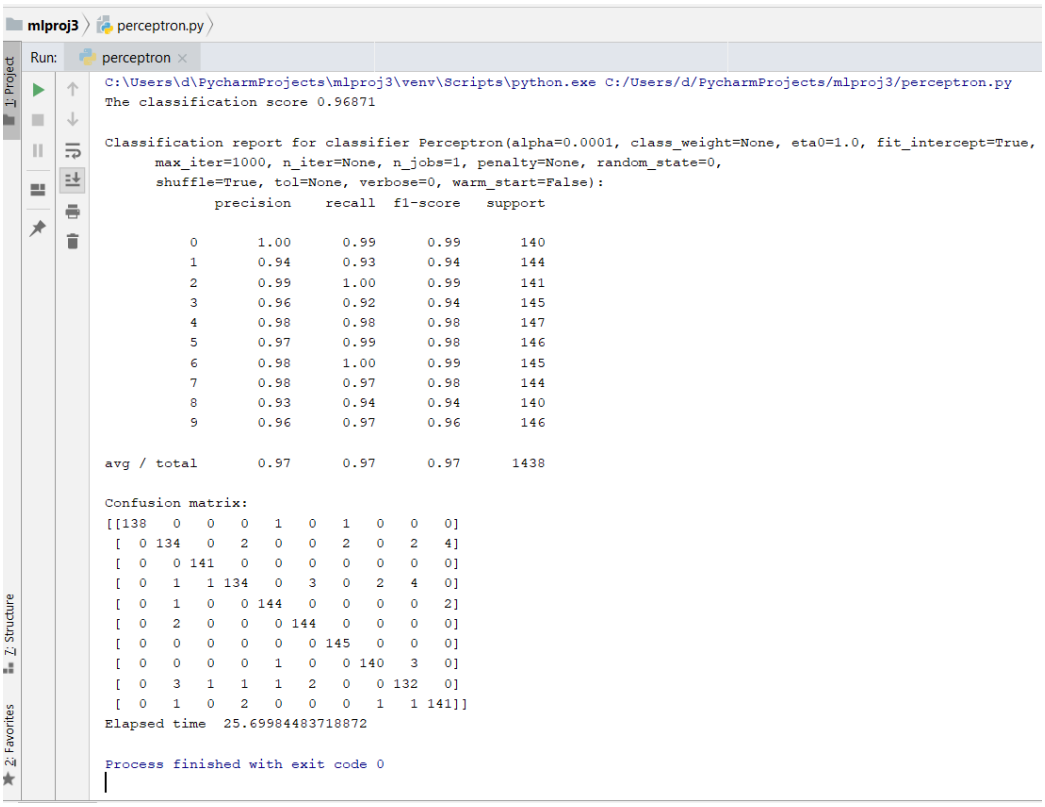


Fig 1.2: The confusion matrix is printed along with the accuracy and the elapsed time

```
allen grad13/jaugusti> time python ./perceptron.py
The classification score 0.96871

Classification report for classifier Perceptron(alpha=0.0001, class_weight=None,
 eta0=1.0, fit_intercept=True,
      max_iter=1000, n_iter=None, n_jobs=1, penalty=None, random_state=0,
      shuffle=True, tol=None, verbose=0, warm_start=False):
             precision    recall  f1-score   support

          0       1.00      0.99      0.99       140
          1       0.94      0.93      0.94       144
          2       0.99      1.00      0.99       141
          3       0.96      0.92      0.94       145
          4       0.98      0.98      0.98       147
          5       0.97      0.99      0.98       146
          6       0.98      1.00      0.99       145
          7       0.98      0.97      0.98       144
          8       0.93      0.94      0.94       140
          9       0.96      0.97      0.96       146

avg / total       0.97      0.97      0.97      1438

Confusion matrix:
[[138   0   0   0   1   0   1   0   0   0]
 [  0 134   0   2   0   0   2   0   2   4]
 [  0   0 141   0   0   0   0   0   0   0]
 [  0   1   1 134   0   3   0   2   4   0]
 [  0   1   0   0 144   0   0   0   0   2]
 [  0   2   0   0   0 144   0   0   0   0]
 [  0   0   0   0   0   0 145   0   0   0]
 [  0   0   0   0   1   0   0 140   3   0]
 [  0   3   1   1   1   2   0   0 132   0]
 [  0   1   0   2   0   0   0   1   1 141]]
2.370u 0.590s 0:02.51 117.9%    0+0k 0+32io 0pf+0w
allen grad13/jaugusti>
```

Fig 1.3 : The run time is compared with the command prompt in linux system

K-nearest neighbor

This is used to separate the datapoints into several different classes.The KNN achieves an accuracy of 98% and the running time is roughly 15 seconds.

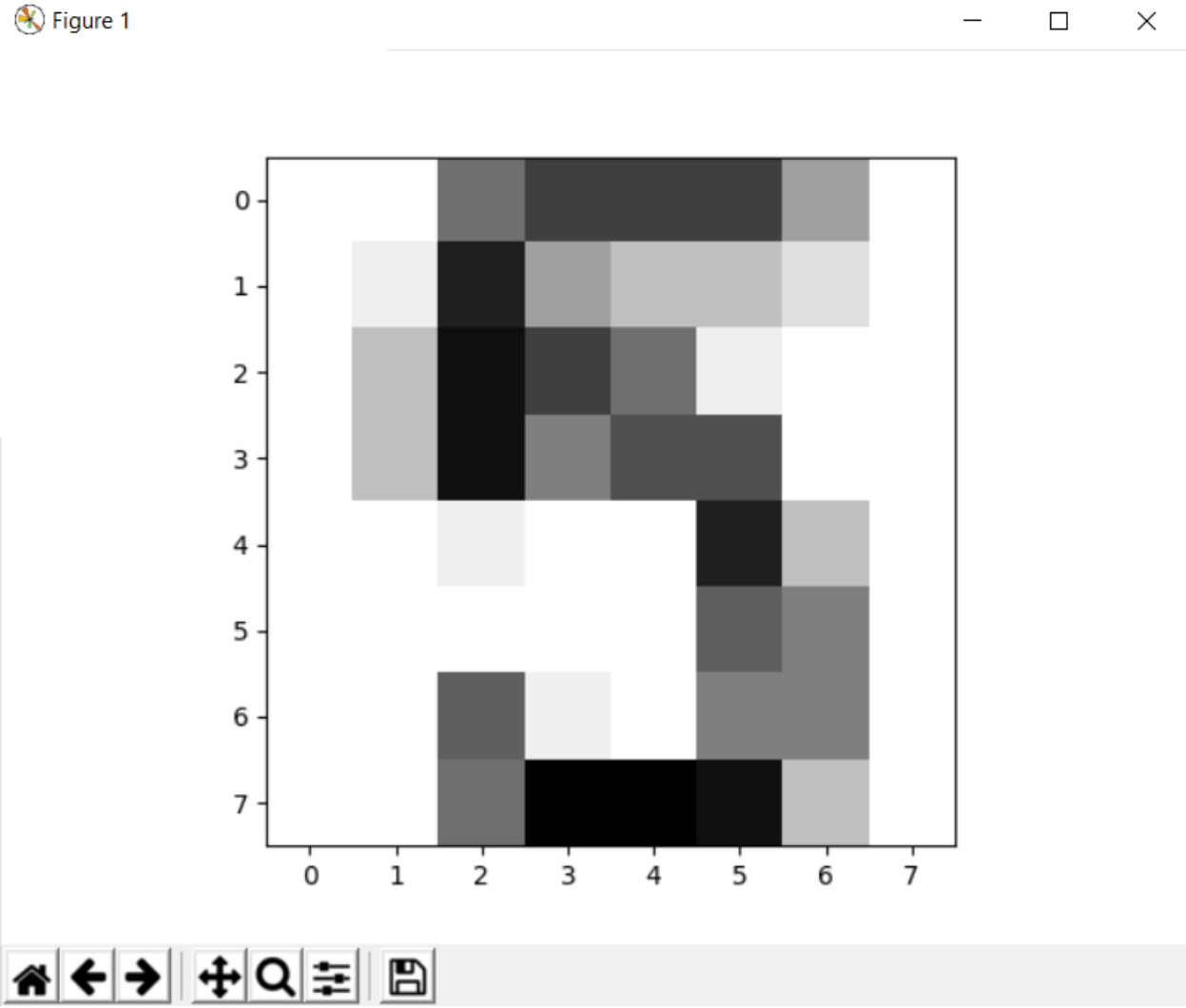


Fig 2 : The separation of a particular digit from the dataset

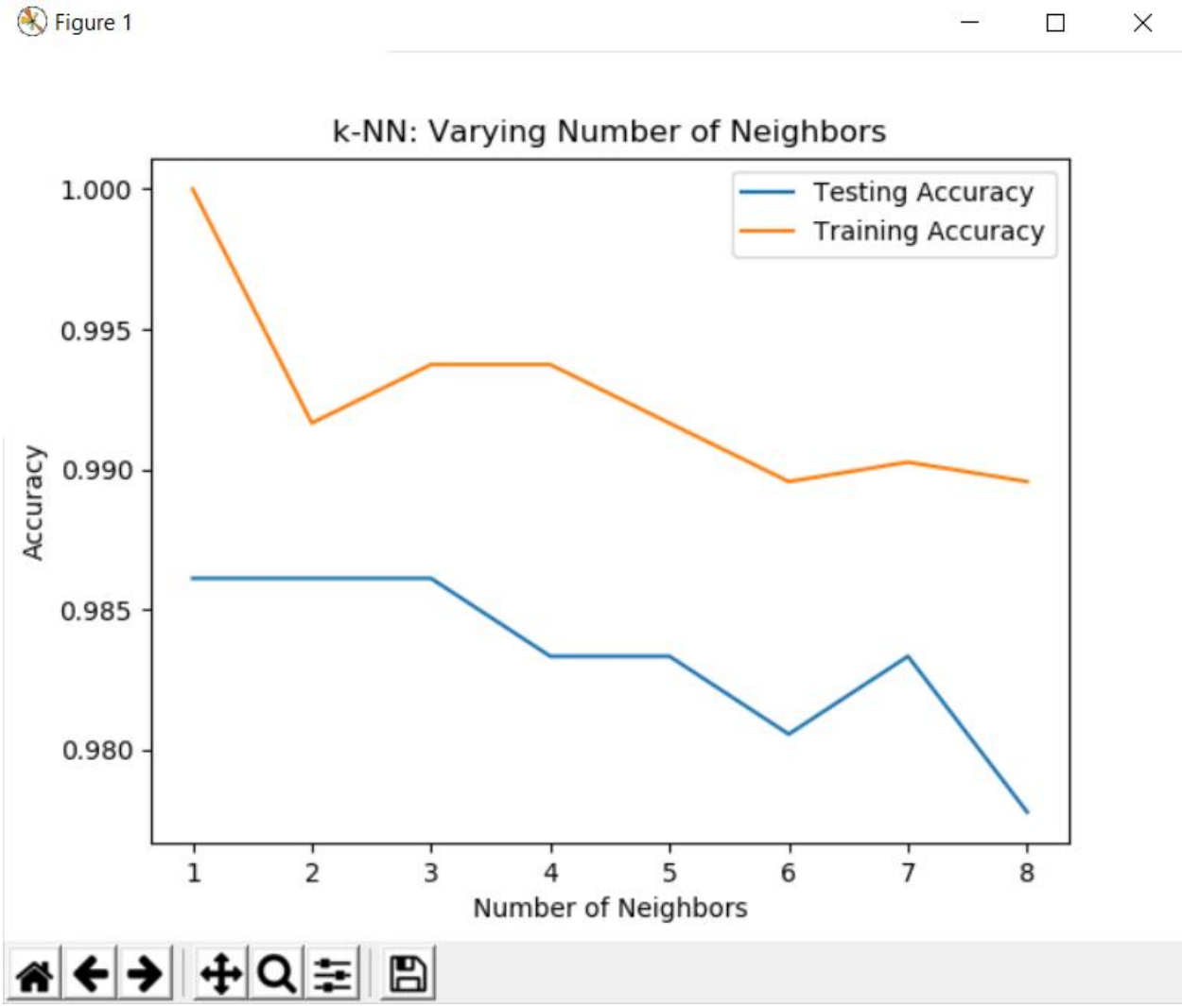


Fig 2.1: The testing and training data is plotted as given above

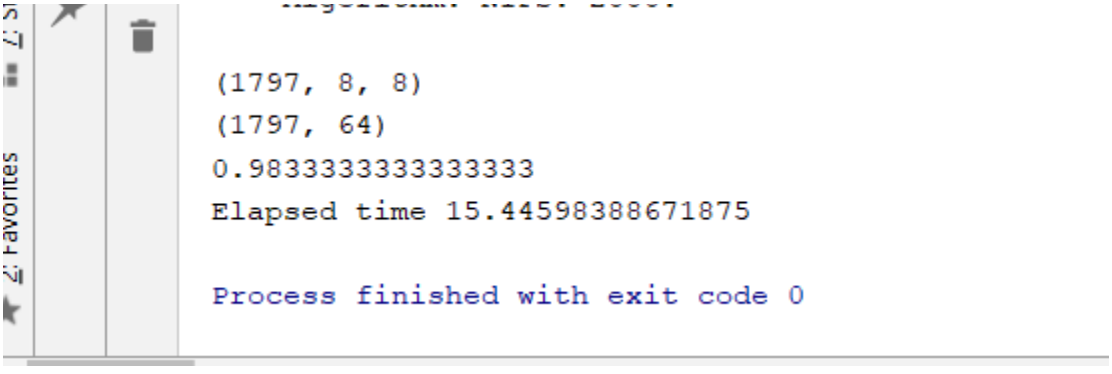


Fig 2.2 : The elapsed running time is 15 seconds and is shown above

```
(1797, 8, 8)
(1797, 64)
0.9833333333333333
2.651u 0.449s 0:02.77 111.5%    0+0k 0+32io 0pf+0w
allen grad13/jaugusti>
```

Fig 2.3: Comparison of running time with the linux command line

Support Vector Machine (linear and non-linear using Radial Basis Function (RBF) kernel

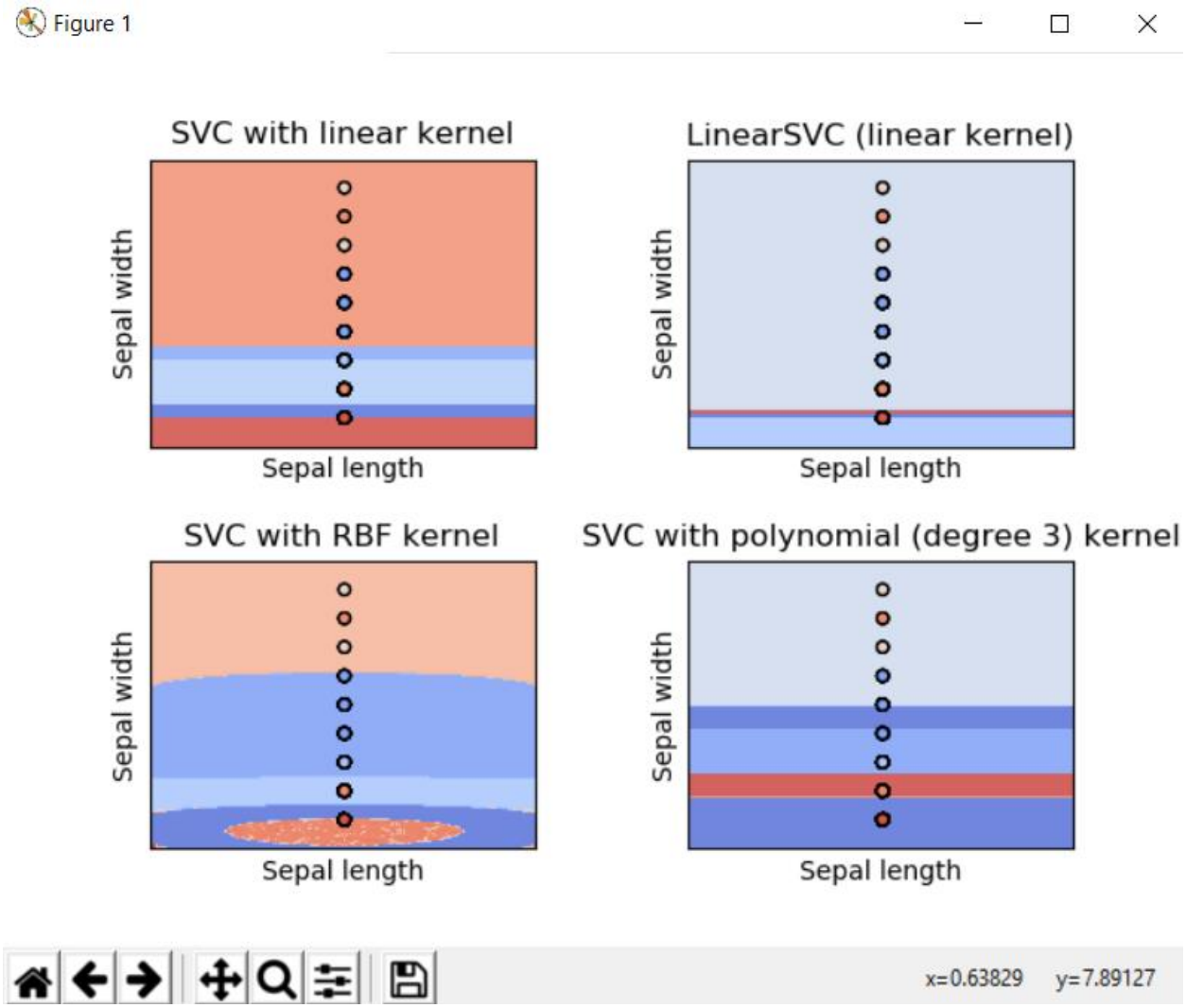


Fig 3: SVM with linear and non linear kernel

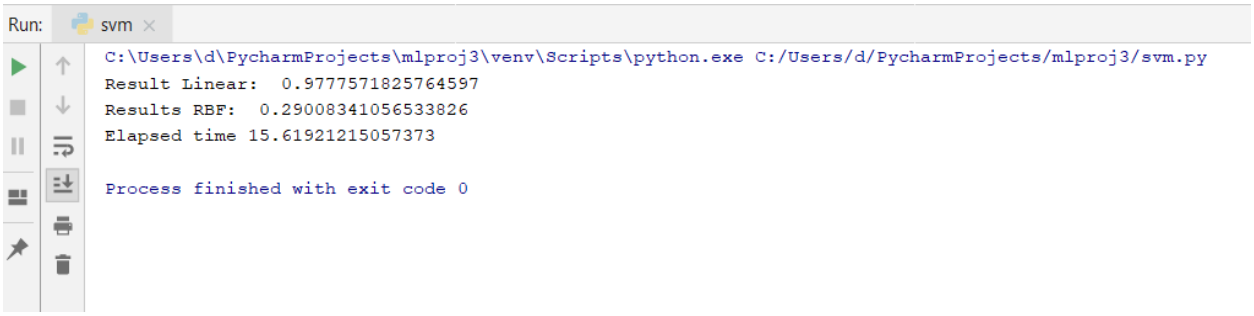


Fig 3.1: Linear and RBF Kernel for digits dataset

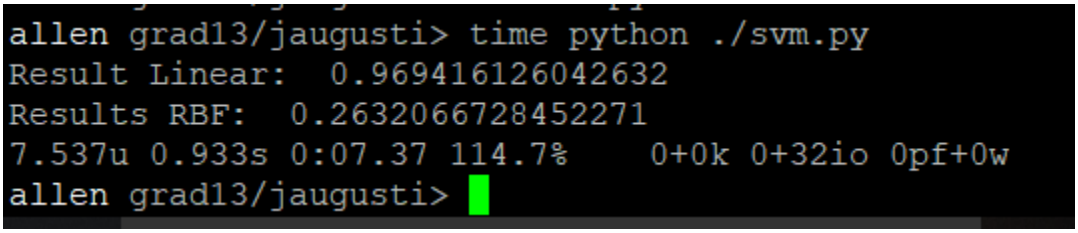


Fig 3.2:Running time compared with linux system

Decision tree

The two strategies that were implemented in the decision tree are max_depth and min_samples_leaf.The min_samples_leaf was done as a result of the post pruning methodology.Max_depth is given to be 10 and min_samples_leaf is given as 2.

This code was implemented In line 26 of the decisiontree.py file.

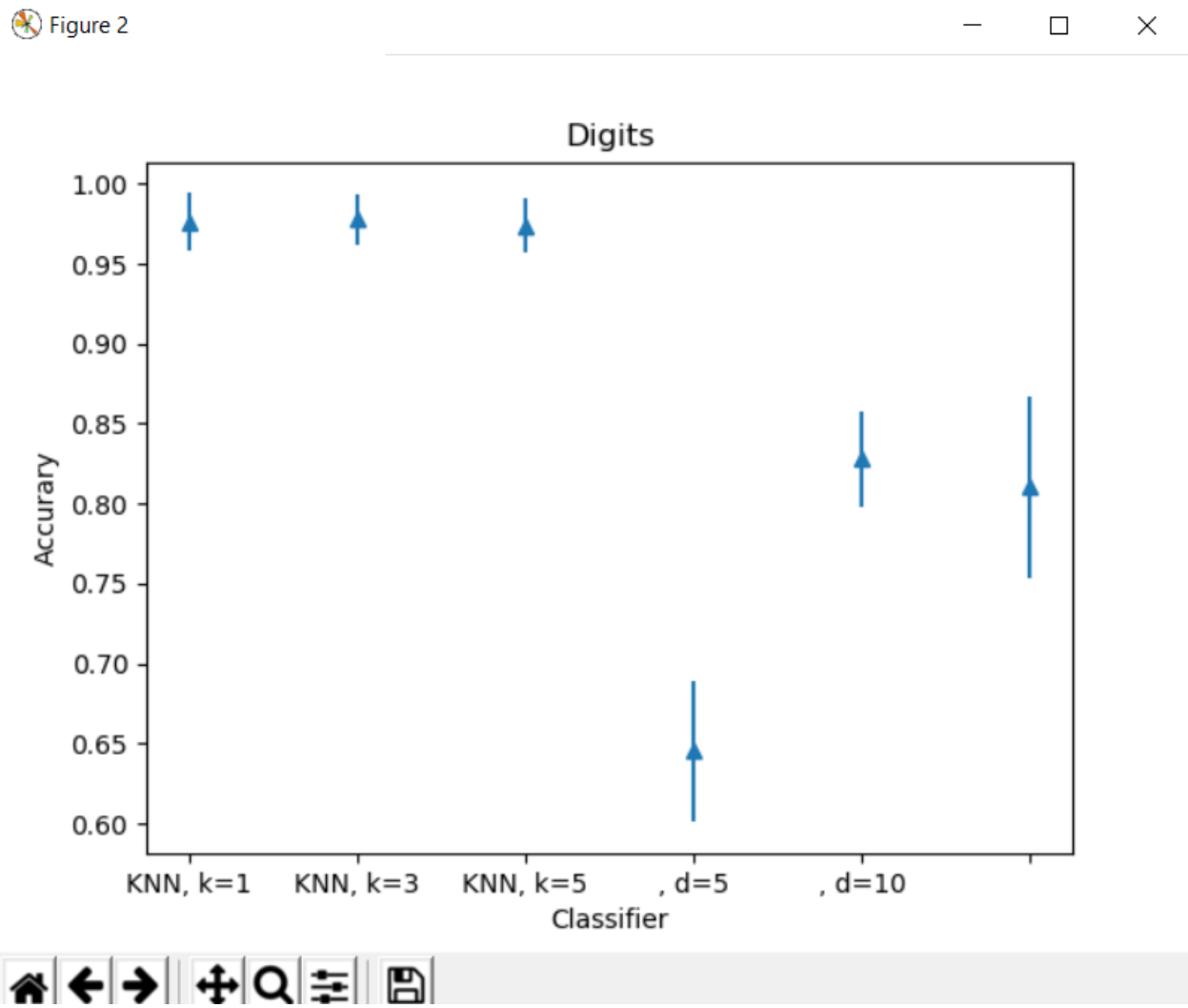


Fig 4: The prediction of accuracy for different values of KNN

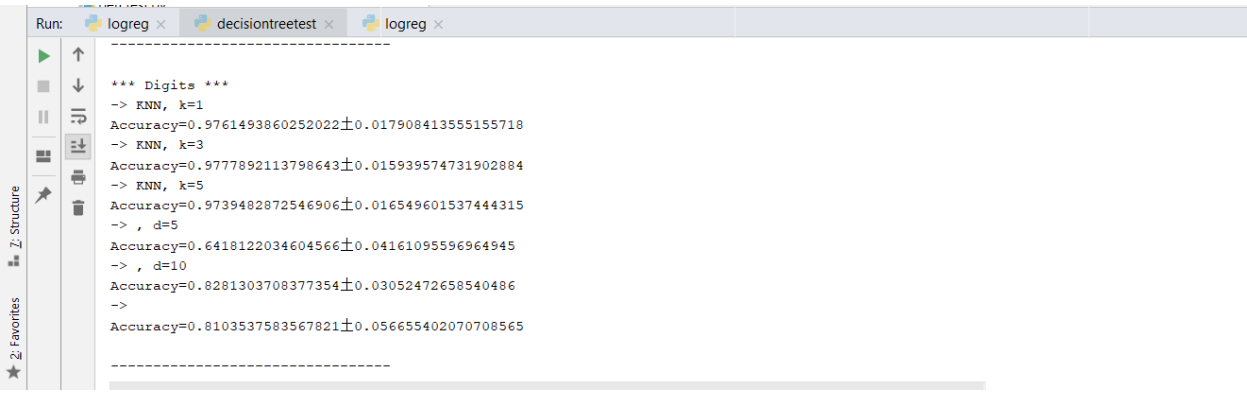


Fig 4.1 : Plotting of the different values of digits using KNN

```
Elapsed Time (s): 3.901808738708496
```

```
--------------------------------

2.502u 0.510s 0:02.64 114.0%    0+0k 0+32io 0pf+0w
allen grad13/jaugusti>
```

Fig 4.2 Time which is compared with the linux system

Logistic regression

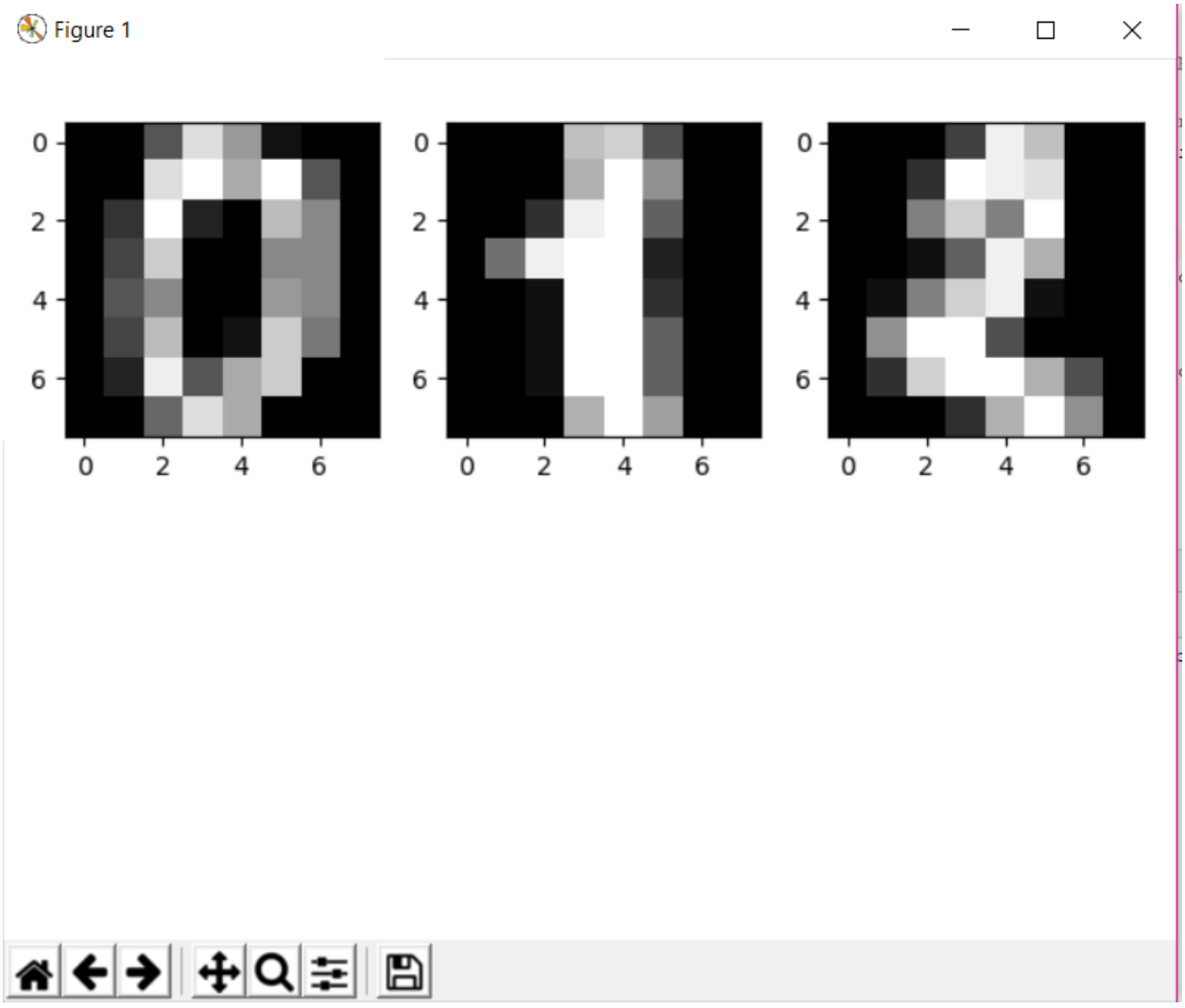


Fig 5:Digits using logistic regression

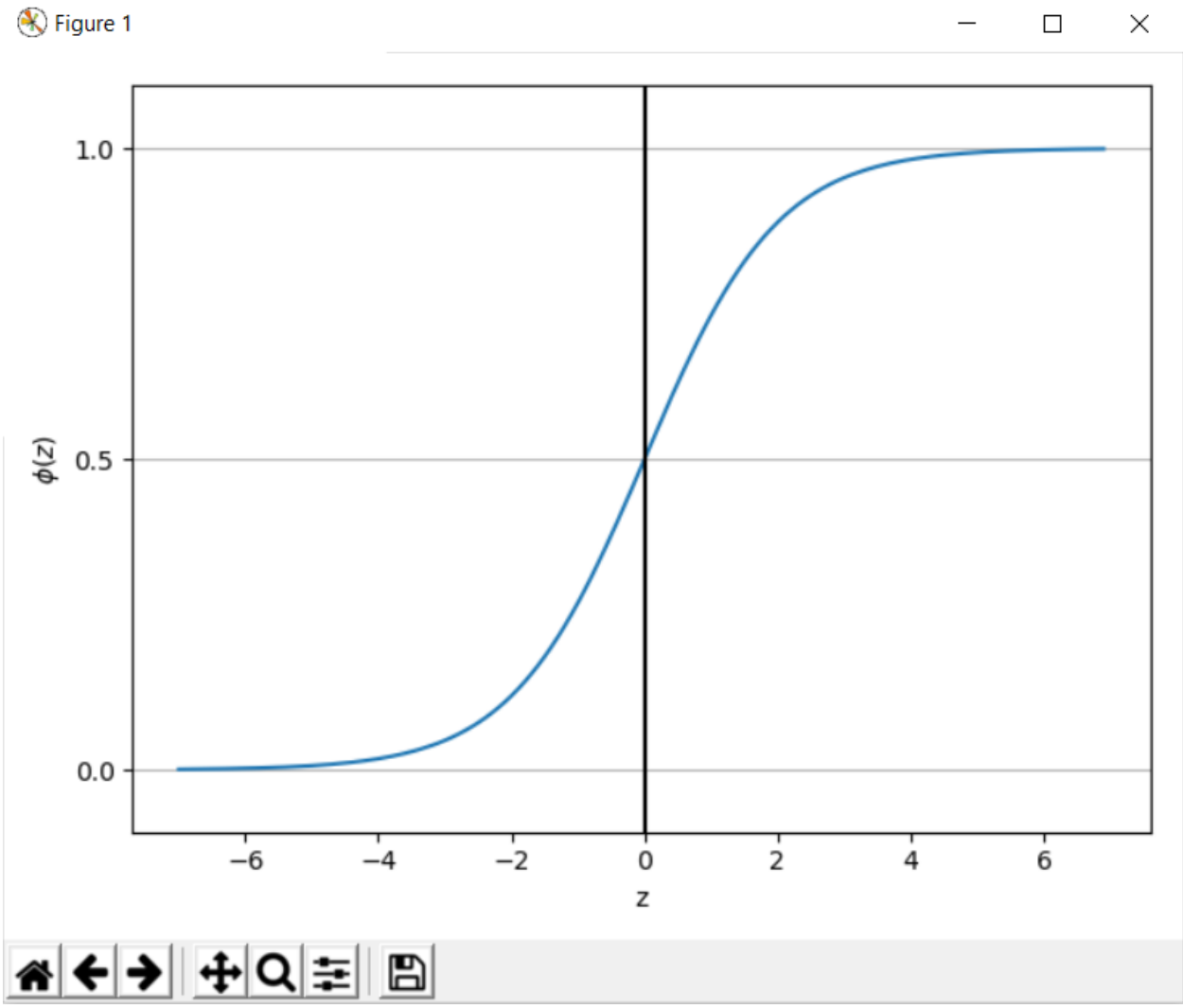


Fig 5.1 Cost function of logistic regression

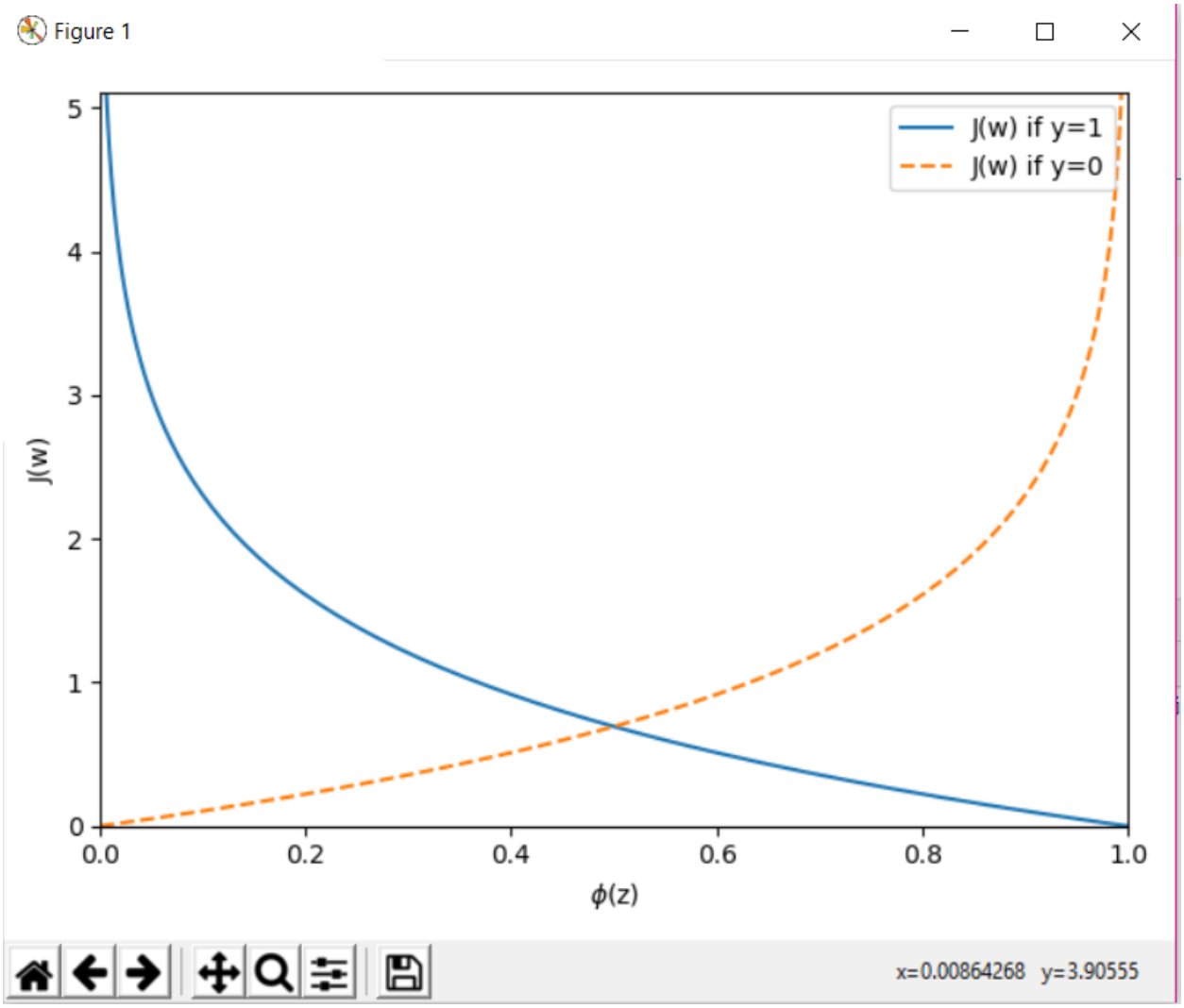


Fig 5.2 Values for y=1 and y=0

```
allen grad13/jaugusti> time python ./logreg.py
(1797, 64)
joanna
<function accuracy_score at 0x7fa38ce760d0>
Predicted :  [1]
Actual :  1
[[4.75035278e-18 9.99448149e-01 7.00780941e-10 3.72473931e-09
  2.15665416e-06 1.38992680e-09 5.71306379e-10 1.95589587e-13
  5.49687591e-04 5.64712749e-10]]
5.483u 0.760s 0:05.56 112.2%    0+0k 0+32io 0pf+0w
allen grad13/jaugusti>
```

Fig 5.3 Time is compared with the linux system

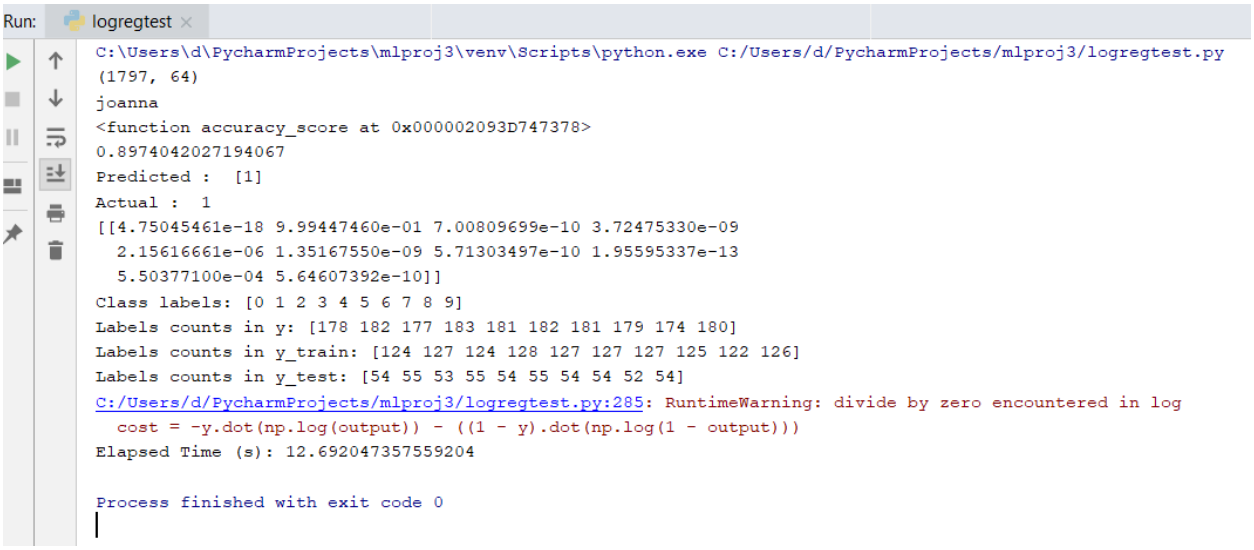


Fig 5.4 Accuracy and elapsed time is calculated

Dataset containing time series instances

The dataset which contains time series instances is the SONAR Dataset. This has been used as a dataset in the different classifiers to show the accuracy and the running time of the algorithms

Perceptron with Sonar Dataset

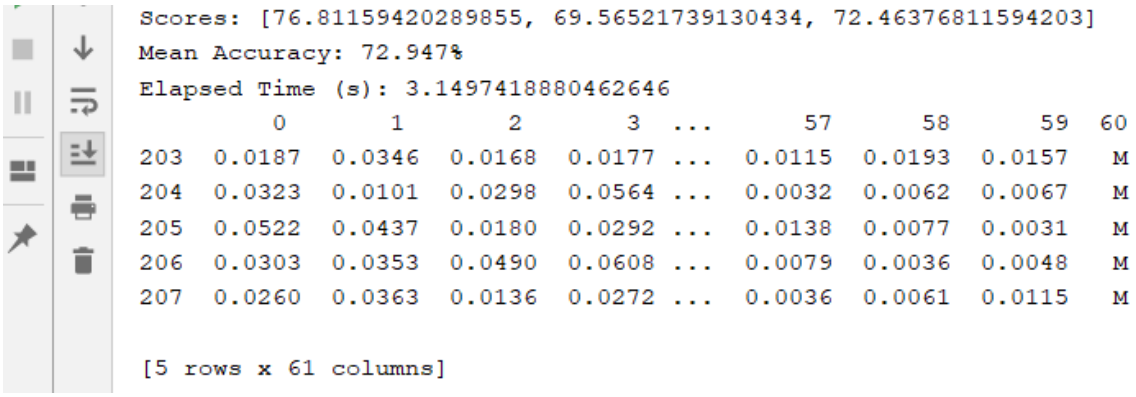


Fig 6: Accuracy and running time for perceptron using SONAR dataset

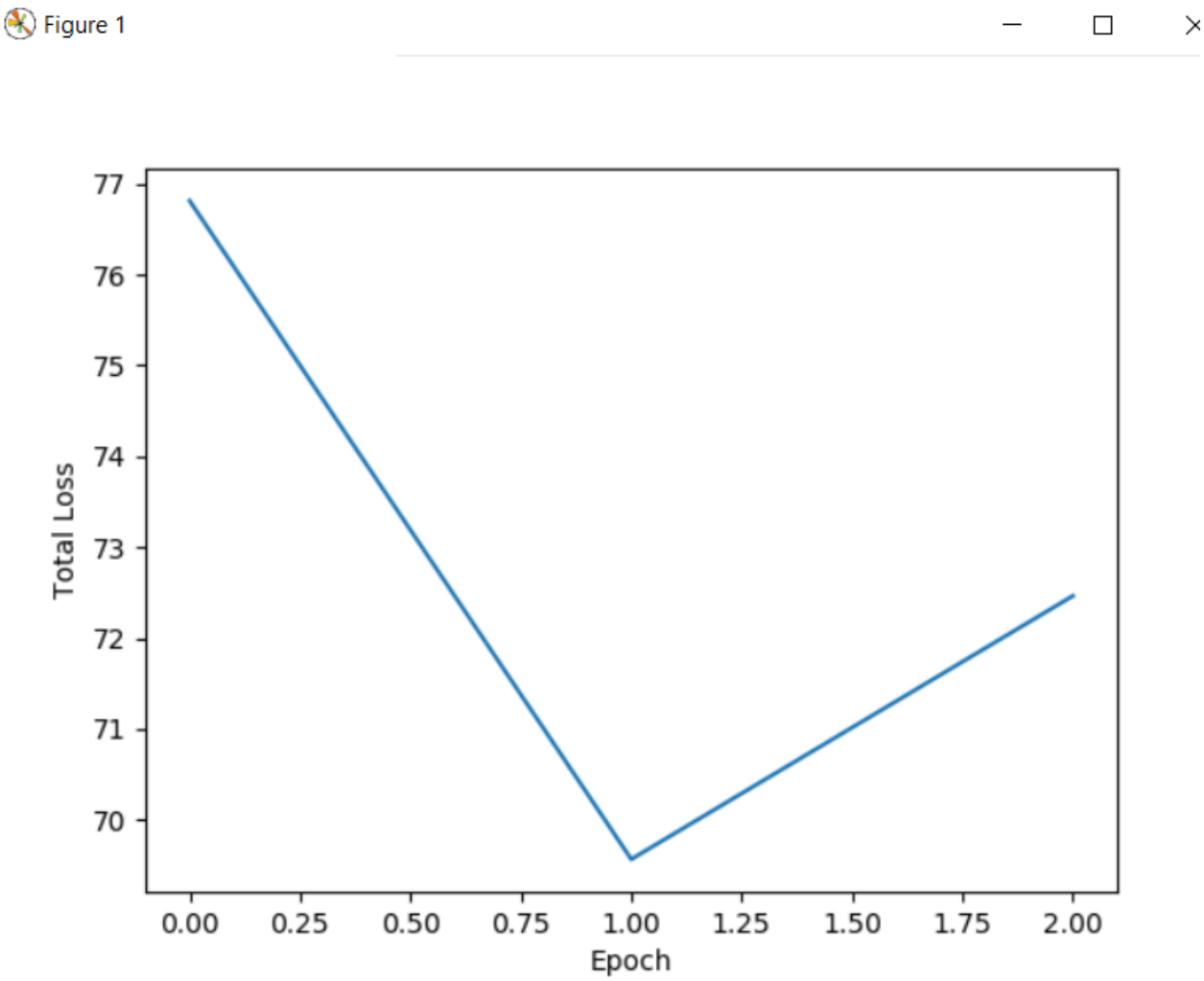


Fig 6.1 The epoch and the total loss is calculated as above

Knn with SONAR dataset

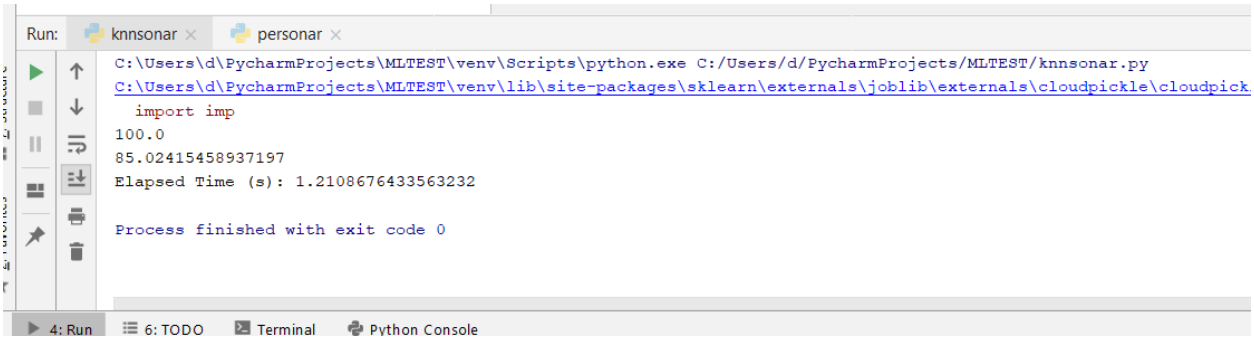


Fig 6.2: Accuracy and running time for KNN using SONAR dataset

Decision tree with SONAR dataset

```
Dataset Lenght::  208
Dataset Shape::  (208, 61)
['R' 'M' 'R' 'R' 'M' 'R' 'R' 'M' 'R' 'R' 'M' 'M' 'M' 'R' 'R' 'R' 'M' 'R'
 'R' 'M' 'R' 'M' 'M' 'R' 'R' 'R' 'R' 'R' 'M' 'R' 'M' 'M' 'M' 'R' 'M' 'M'
 'M' 'M' 'M' 'R' 'M' 'M' 'M' 'M' 'R' 'M' 'M' 'R' 'M' 'M' 'R' 'M' 'R' 'M'
 'R' 'M' 'M' 'R' 'M' 'M' 'R' 'R' 'M']
['M' 'M' 'R' 'R' 'M' 'R' 'M' 'M' 'M' 'M' 'R' 'M' 'M' 'R' 'R' 'R' 'M' 'M'
 'R' 'M' 'M' 'M' 'M' 'M' 'M' 'M' 'M' 'R' 'R' 'R' 'M' 'R' 'M' 'M' 'M' 'M'
 'M' 'M' 'M' 'M' 'R' 'R' 'M' 'R' 'R' 'M' 'R' 'R' 'R' 'M' 'R' 'M' 'M' 'M'
 'R' 'M' 'R' 'R' 'R' 'M' 'R' 'R' 'M']
Accuracy is  63.49206349206349
Elapsed Time (s): 0.0557398796081543

Process finished with exit code 0
```

Fig 6.3: Accuracy and running time for Decision Tree using SONAR dataset

Logistic Regression with SONAR dataset

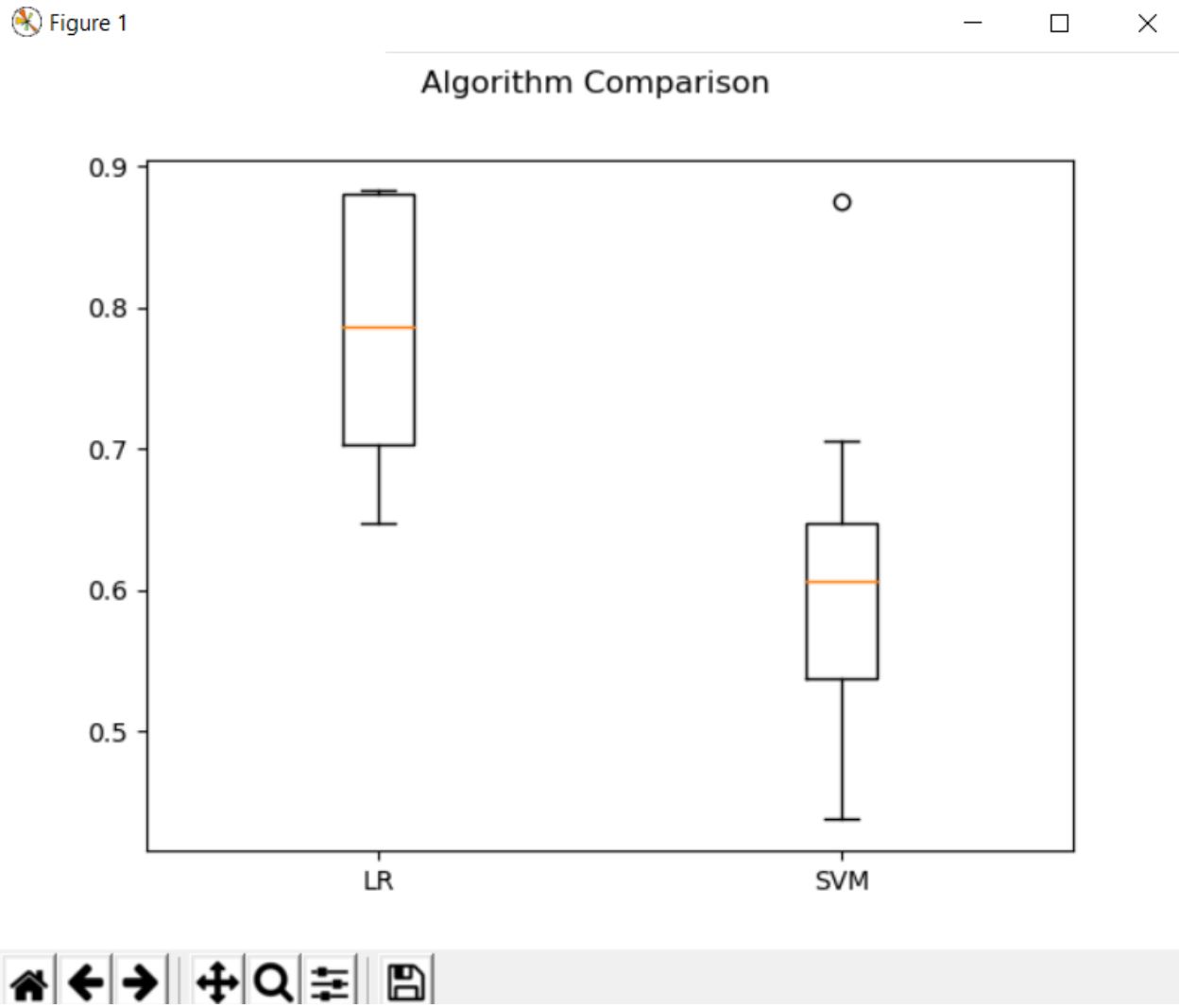
```
[8 rows x 60 columns]
60
M    111
R     97
dtype: int64
Elapsed Time for LR (s): 11.175235033035278

Accuracy for LR
0.7142857142857143
[[19  8]
 [ 4 11]]
              precision    recall  f1-score   support

           M       0.83      0.70      0.76        27
           R       0.58      0.73      0.65        15

   micro avg       0.71      0.71      0.71        42
   macro avg       0.70      0.72      0.70        42
weighted avg       0.74      0.71      0.72        42
```

Fig 6.3: Accuracy and running time for Logistic Regression using SONAR dataset




```
ScalerLR : 0.734191 (0.095885)
ScalerSVM : 0.836397 (0.088697)
```

Fig 6.3: Comparison of Logistic Regression and SVM using SONAR dataset for ScalerLR and ScalerSVM

SVM with sonar dataset

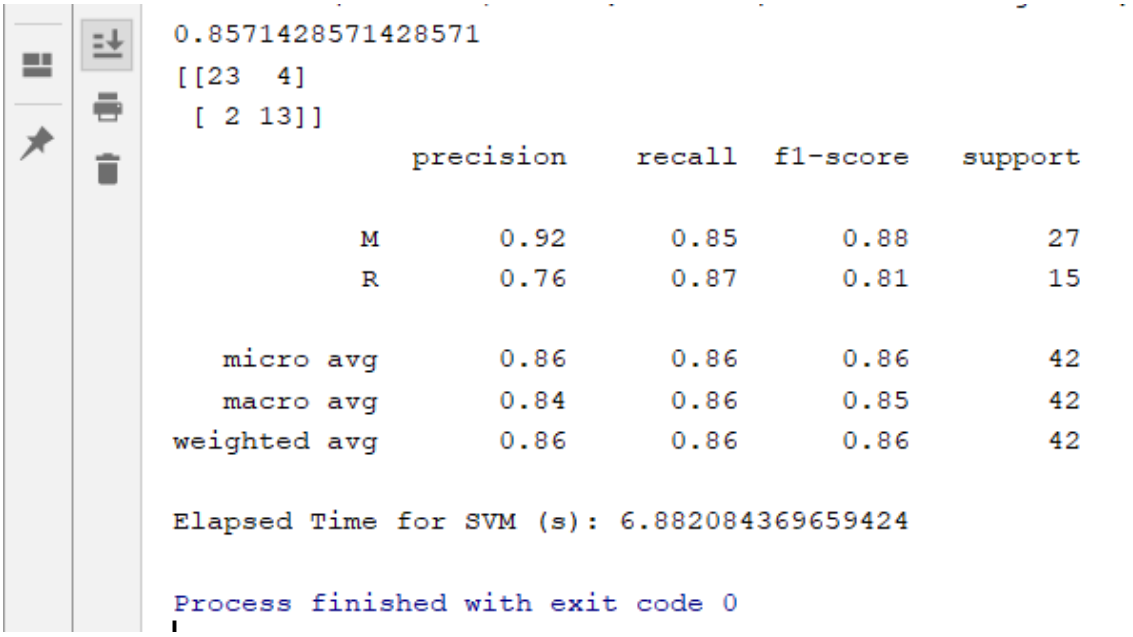


Fig 6.5: Accuracy and running time for Suppor Vector Machine using SONAR dataset

Analysis:

DIGITS DATASET

Sno	Classifier name	Accuracy	Runtime(sec)
1	Perceptron	96%	25.69
2	Decision Tree	97%	3.90
3	KNN	98%	15.44
4	SVM	97%(linear kernel) 29%(rbf kernel)	15.61
5	Logistic Regression	89%	12.69

According to my analysis the decision tree is more efficient in terms of performance and speed.

SONAR DATASET

Sno	Classifier name	Accuracy	Runtime(sec)
1	Perceptron	72%	3.14
2	Decision Tree	63%	0.05
3	KNN	85%	1.21
4	SVM	85%	6.88
5	Logistic Regression	71%	11.17

Here the KNN seems to perform better than the other classifiers

Results:

The various classifiers have been analyzed and classified according to their accuracies and running time.A brief analysis has been given after the implementation of the classifiers in the two datasets and based on their running time and accuracy their efficiency is computed.