Project 2 CS 205. Artificial Intelligence

Kamalika Poddar Instructor: Dr. Eamonn Keogh

[kpodd001@ucr.edu](mailto:kpodd001@ucr.edu)

22-March-2018

In completing this homework, I consulted…

* https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/ (for k-Nearest Neighbor)
* https://python4astronomers.github.io/files/asciifiles.html/ (loading data)
* https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/
* https://en.wikipedia.org/wiki/Feature\_scaling (feature scaling)
* https://stackoverflow.com/questions/12525722/normalize-data-in-pandas (normalizing data)
* https://github.com/ktang012/Feature\_Selection/blob/master/nearest\_neighbor.py (structure of code)
* https://github.com/ader003/cs170-FSwNN/blob/master/featSelecwNN.py (forward and backward selection)
* https://en.wikipedia.org/wiki/Standard\_score
* I also referred to the project sample report-1 for making this report.

Apart from this I imported libraries like

* Sklearn- to preprocess data for min-max normalization
* Scipy- for z-score normalization
* matplotlib- to plot data
* math- mathematical computations
* numpy- to load data
* pandas –to load data for preprocessing

**CS-205/ ARTIFICIAL INTELLIGENCE**

**PROJECT REPORT-2**

**INTRODUCTION**

This is my second project under Professor Eamonn Keogh for CS 205- Artificial Intelligence. This project concentrates on Feature selection using Nearest Neighbor Algorithm. I used Python as the programming language to complete the project. I tried to select best features using Forward Selection, Backward Elimination and my own algorithm (search algorithm). Below is the detailed report of the work done for this project.

**NORMALIZATION TECHNIQUES**

**Z-Score Normalization**

Here we first calculate the mean, and then subtract it from dataset and divide by standard deviation. Z score is normally negative below the mean and positive above the mean. This we do for all instances in the data set which helps us remove outliers from the data. For this project, I am using an in-built z-score function from ‘scipy’ library.

**Feature Scaling (Min-Max Normalization)**

In this type of normalization, the entire set of observed values from the dataset is transformed between ‘0’ to ‘1’ where ‘0’ is the minimum value of the dataset and ‘1’ is the maximum. Suppose ‘x’ is the observed set of values then after normalization, y= (x-min)/(max-min), where min and max depends on x.

**CROSS-VALIDATION TECHNIQUE**

**Leave One Out Cross Validation Technique**

Leave one out cross validation technique is a particular type of a technique where 9 out of 10 instances are used as training dataset and the 10th instance is used as the test dataset. This process is repeated for 10 times and the accuracy is predicted. The predicted accuracy should match the actual accuracy of the test data.

**K-NEAREST NEIGHBOR(KNN)**

k-Nearest Neighbor works well for prediction analysis. If we have a dataset and we want to divide it into classes with similar features then kNN model will search the whole dataset for similar features and will return the classified dataset. Euclidean distance is used for real valued data and hamming distance for binary data. K-NN belongs to instance based (includes data instances in the form of row), competitive learning (try to compare between elements) and lazy learning algorithms (it builds a model after evaluating all the unseen data).

**ALGORITHMS**

**Forward selection**

Suppose we have 10 features in a dataset, first we take one feature at a time and try to compute the accuracy. The feature with highest accuracy is kept and for the next level again all the features are checked one at a time. This combination goes on for all the 10 features. When the highest accuracy is reached after combining the set of features, those set of features are considered to be the best features in the dataset. This type of combining features by increasing one feature at a time is known as Forward Selection.

**Backward elimination**

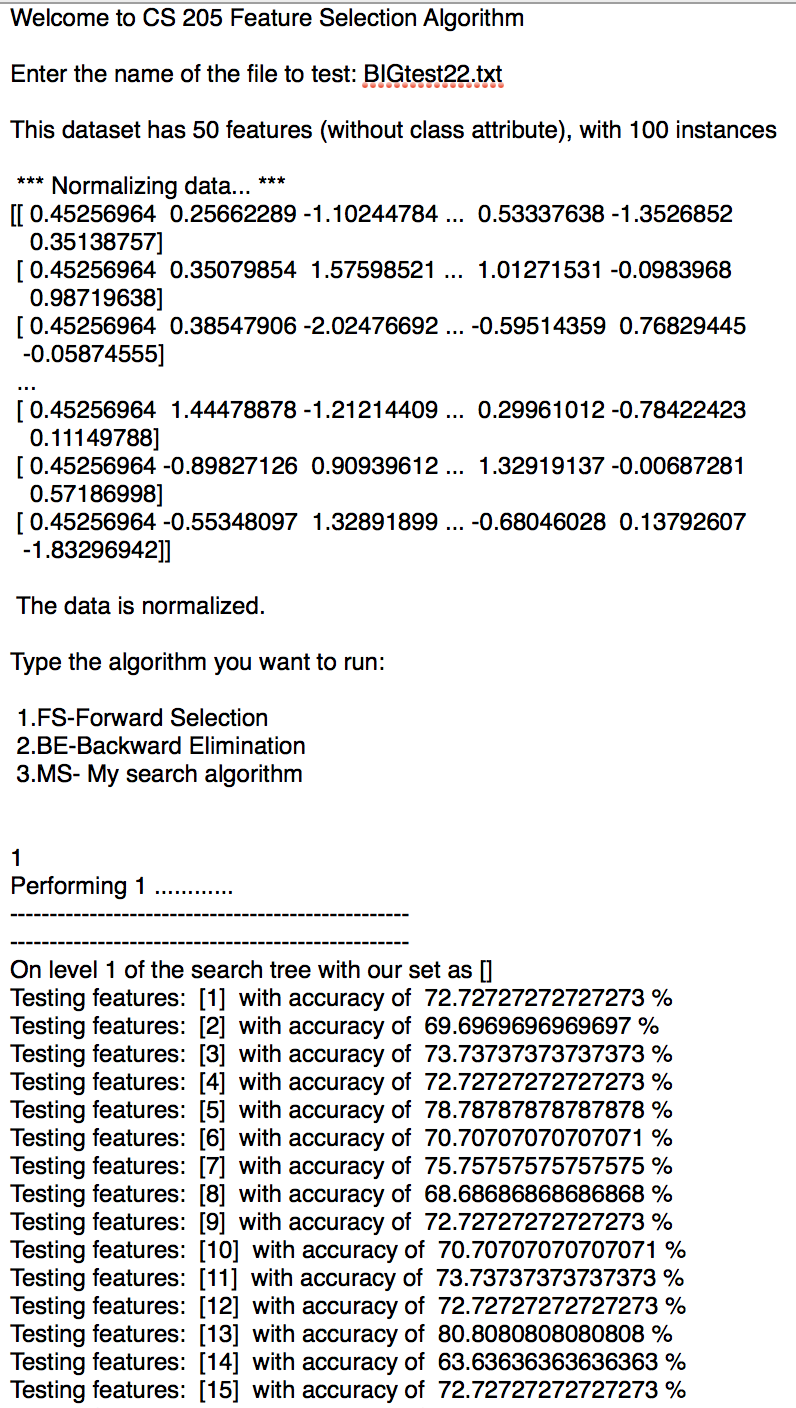
In this feature extraction techniques, first we take all the features say 10 features, then with each iteration we remove one feature at a time from the list and compare the accuracy. We do this for all the features till the current set of features reaches one. At this level, we keep updating the global accuracy, after all the features being tested, we print out the best set of features with highest accuracy.

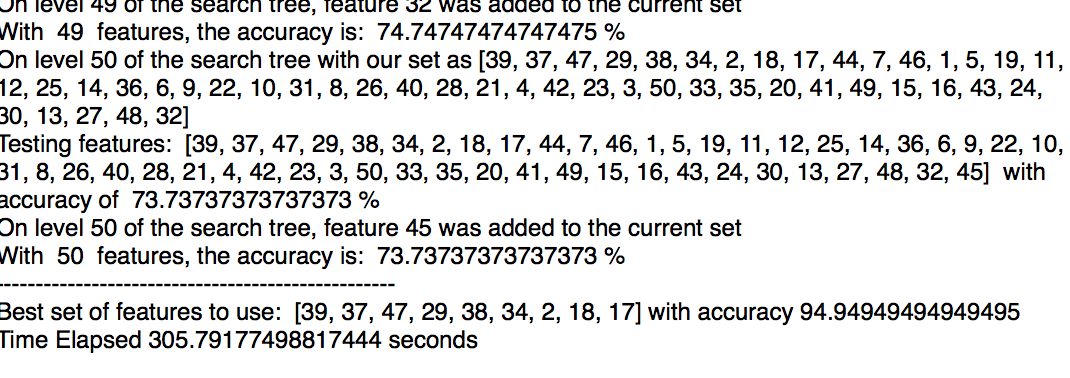
**Search algorithm**

In Forward selection, we keep on adding features till we reach the current set of features which includes all the features i.e. it checks all the features even if we have reached the highest accuracy. This is same for backward elimination where it keeps on removing the features even if after some iteration the accuracy tends to decrease. In this algorithm, the unnecessary computation after the decrease in accuracy is avoided. This algorithm has two parts- 1. It trims off the data while using forward selection till the point where its accuracy starts decreasing, 2. It performs backward elimination till a point where it encounters a decrease in accuracy and then it stops. This algorithm computes best features at a much lesser time for bigger dataset and number of features collected is also less indicating the importance of features.

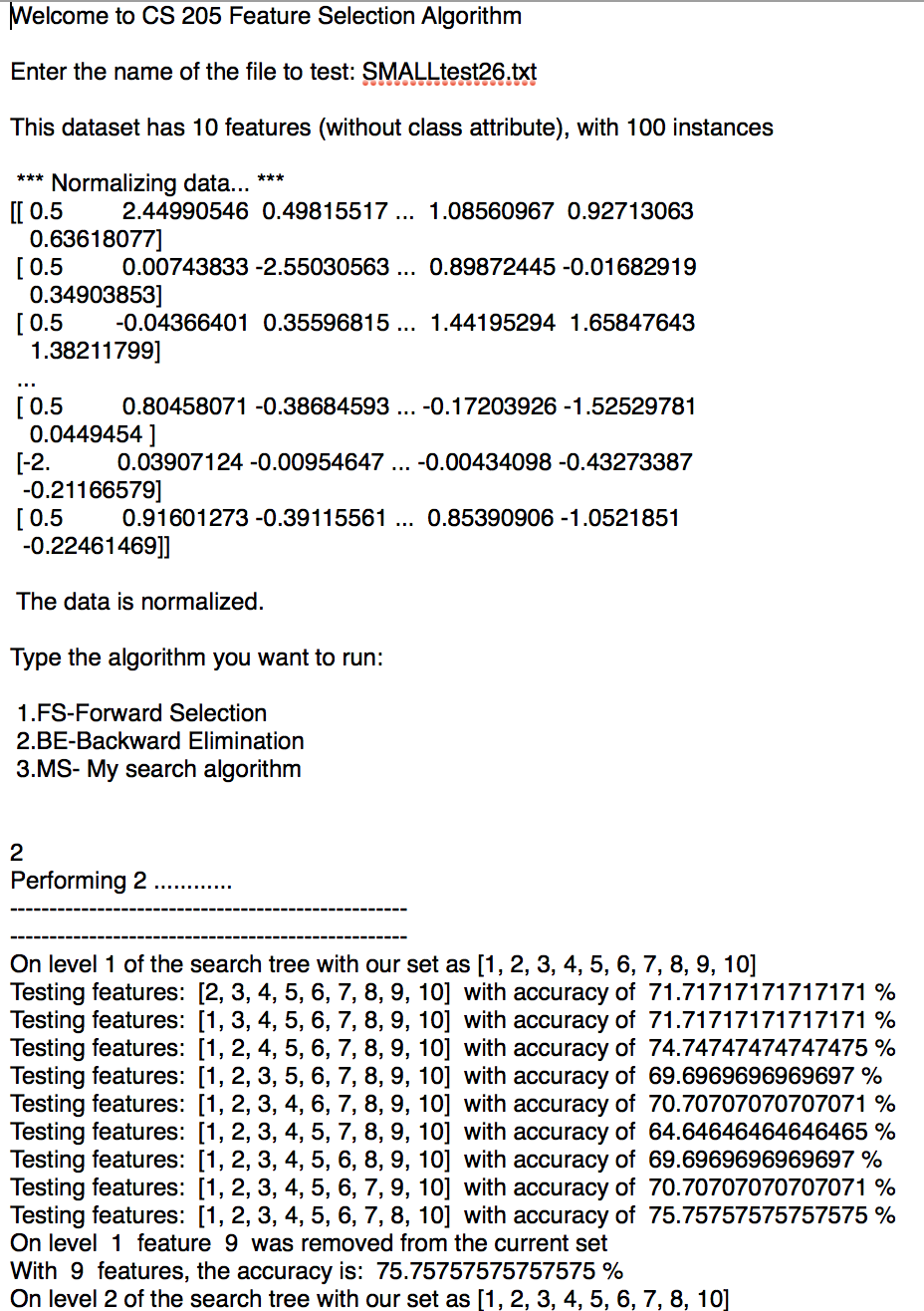
**OUTPUT**

* Below shows the output for Bigdataset22.txt using forward selection which gives best features as [39, 37, 47, 29, 38, 34, 2, 18, 17] with an accuracy of 94.94%.

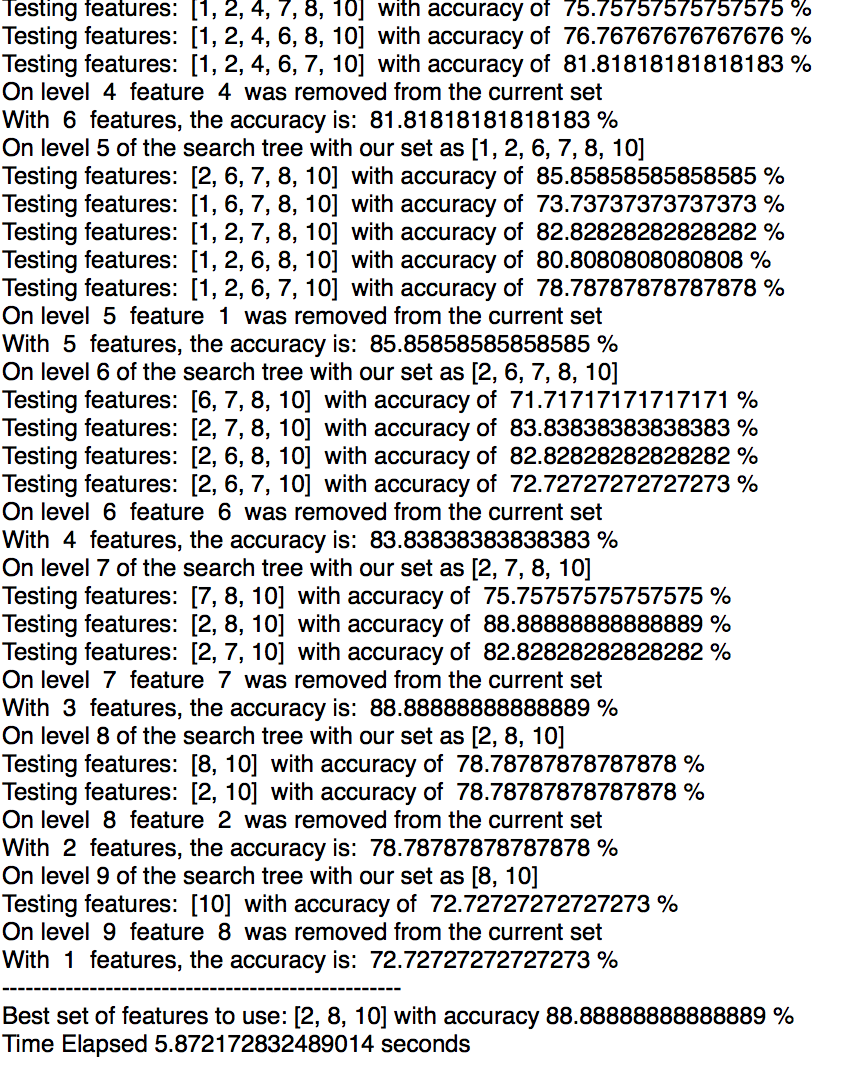




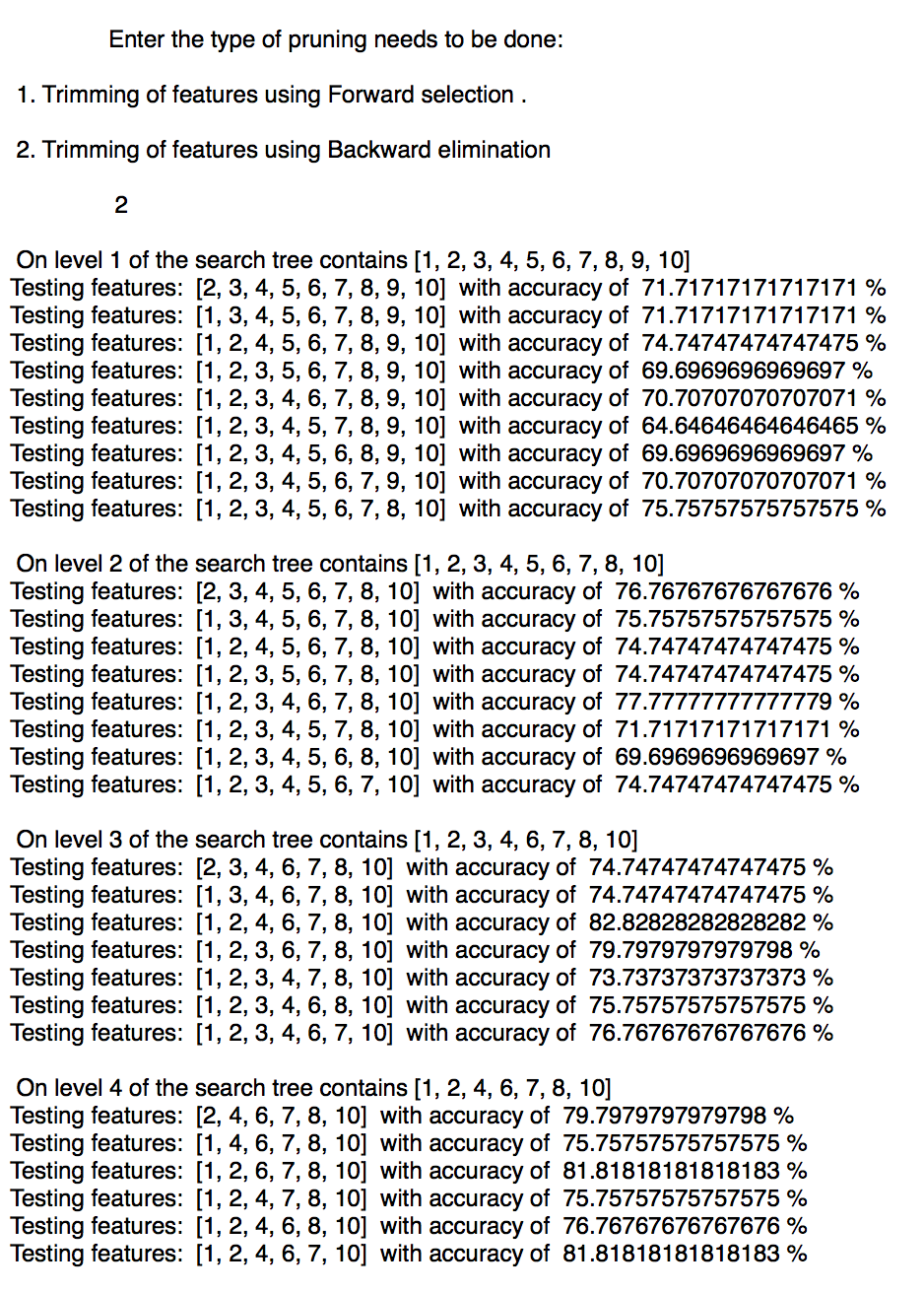
* Below shows the output for Smalldataset26.txt using Backward Elimination which gives best features as [2,8,10] with an accuracy of 88.88%



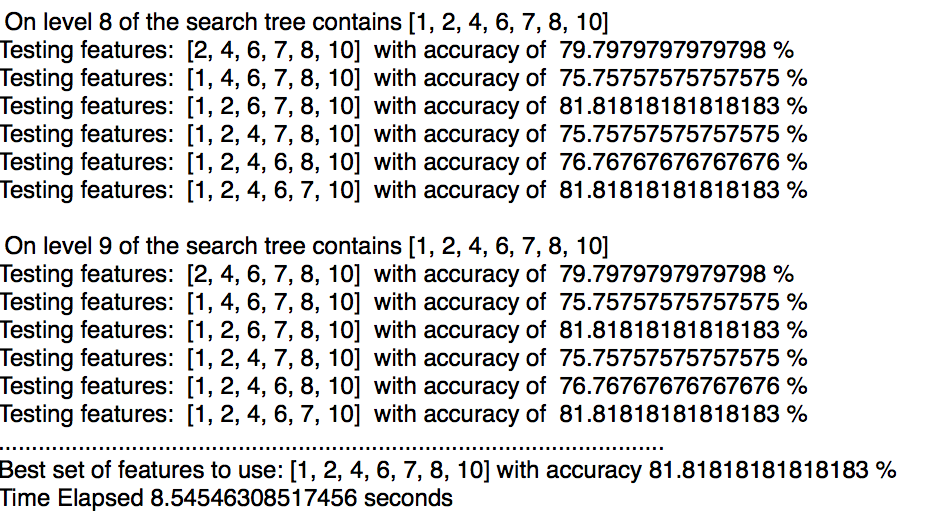
**……………………………………………………………**

****

* Below shows the output for Smalldataset26.txt using Search Algorithm-Backward trimming after decreasing in accuracy. Results gives best features as [1,2,4,6,7,8,10] with an accuracy of 81.81%

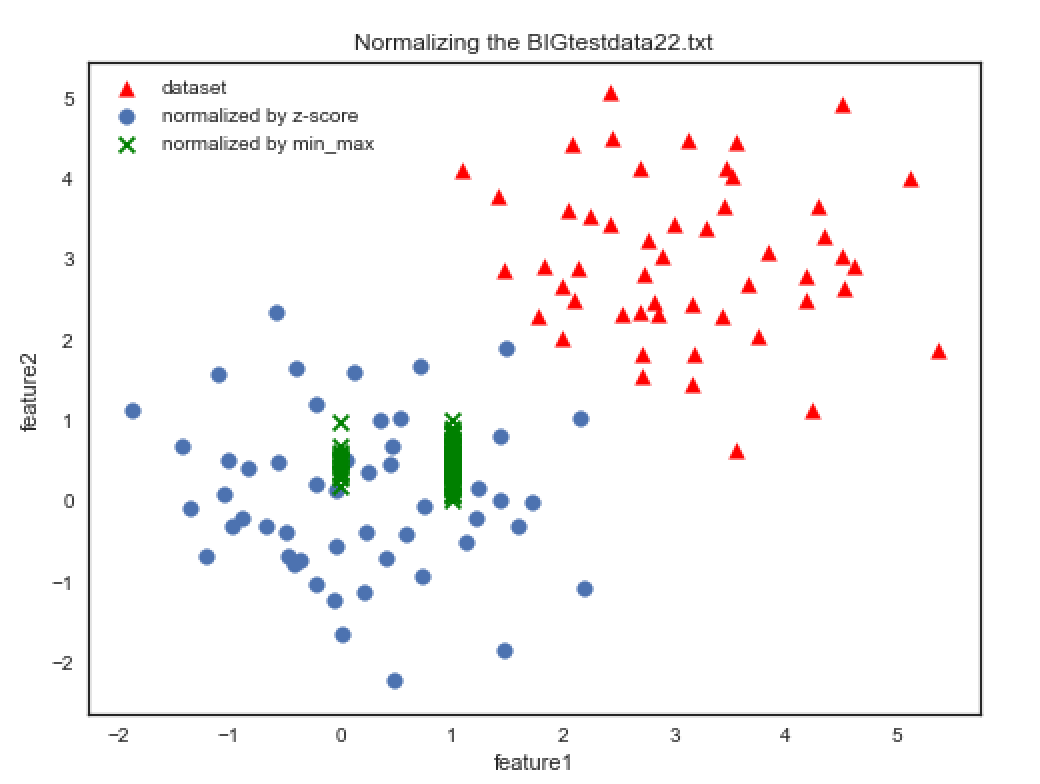
****

**……………………**

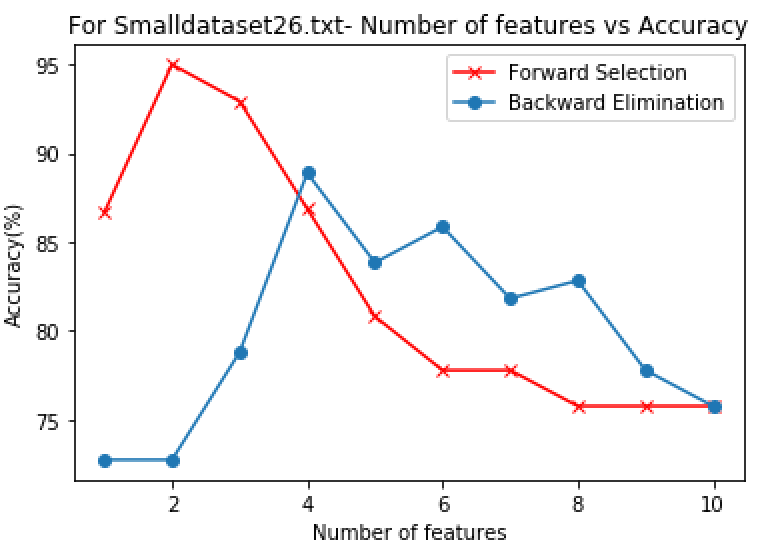
****

**Graphs**

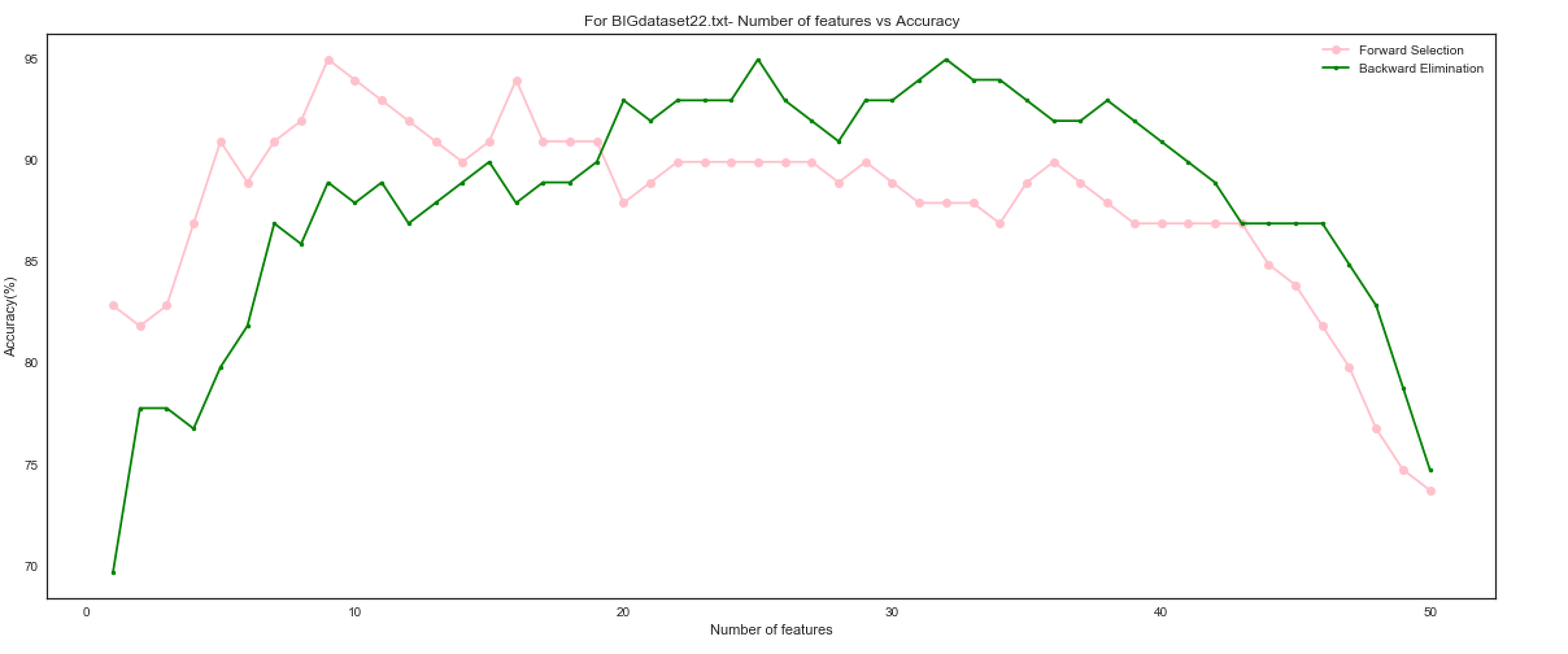
* Below graph shows the normalization of features for the dataset by z-score and min-max normalization.



* Below graph shows the accuracy of forward selection and backward elimination with the number of features for Smalldataset26.txt.

****

* Below graph shows the accuracy of forward selection and backward elimination with the number of features for BIGdataset22.txt.

****

**RESULTS**

Below is the table showing findings of accuracy, best features, time elapsed for forward selection, backward elimination and search algorithm. The results for the small and the big dataset are compared with Professor Keogh’s key.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **Forward Selection** | | | **Backward Elimination** | | |
|  | **Time**  **(seconds)** | **Accuracy** | **Best feature** | **Time**  **(seconds)** | **Accuracy** | **Best feature** |
| **Test22.txt** | 305.791 | 94.94% | [39, 37, 47, 29, 38, 34, 2, 18, 17] | 494.090 | 94.94% | [1, 3, 6, 7, 9, 11, 14, 15, 17, 20,22, 25, 27, 31,33, 37, 38, 39, 40, 42,44, 47, 48, 49, 50] |
| **Test26.txt** | 6.0198 | 94.94% | [2,8] | 5.872 | 88.88% | [2,8,10] |

**Applying the search algorithm:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Puzzle** | **Forward Trimming** | | | **Backward trimming** | | |
|  | **Time**  **(seconds)** | **Accuracy** | **Best feature** | **Time**  **(seconds)** | **Accuracy** | **Best feature** |
| **Test22.txt** | 7.8163 | 81.81% | [39] | 1183.83 | 86.86% | [1, 2, 3, 4, 5, 6,7, 8, 9, 10, 11, 14, 15,17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50] |
| **Test26.txt** | 2.31 | 92.92% | [2,8] | 8.545 | 81.81% | [1, 2, 4, 6, 7, 8, 10] |

Professor Keogh’s Key:

Big Dataset22.txt - [47,27,17]

Small dataset26.txt- [2,5,8]

According to the algorithms in the code, the Backward Elimination performs better than the forward selection as it includes all the features from the professor’s key whereas forward selection eliminates the key features.

**CONCLUSION**

* According to the results above, we observe that Backward Elimination takes more time than forward selection for big dataset22.txt. And for the smalldataset26.txt backward elimination takes lesser time than forward selection but accuracy decreases from 94% to 88%.
* For the search algorithm, the time taken by forward trimming is less for both the dataset but the accuracy decreases.
* For Backward trimming search algorithm, time elapsed is more and the accuracy also decreases which is not desired.
* The graph obtained from the bigger dataset shows less variation in accuracy than the one with smaller dataset, so it concludes the algorithm works better for large dataset than small dataset.
* In the search algorithm, the accuracy decreases because it is not able to explore the entire tree and exits the loop after it finds a decrease in accuracy.

This algorithm does not work when we need to explore all the features, but it helps to eliminate excess features after a certain amount of time. Another reason that this algorithm takes more time can be because of the insertion of branching statements in the algorithm. I hope to improve the accuracy of the algorithm in the future.

**ACKNOWLEDGEMENT**

Thank you, Professor, for giving us the opportunity to explore the wide arena of Artificial Intelligence. Not only I became aware of the concepts but I could also understand how the knowledge can be used for implementation of real life problems. This course was indeed a learning experience for me and I hope to use my knowledge in implementing other concepts in the future.

**ABOUT THE CODE**

The below code consists of 193 lines including whitespaces and comments. Firstly, we load the data from the dataset then we use z-score normalization to remove outliers and scale the data around mean. Three algorithms are used to find out the best feature set in the data. A function Leave one out cross validation is used to train the data by taking (n-1) instances and test on the nth instance. The output of the result is obtained at the console.

**First page of the code as instructed by the instructor**

import pandas as pd

import numpy as np

from sklearn import preprocessing

from scipy import stats

import time

import math

import matplotlib.pyplot as plt

#Normalizing data by zscore normalization

def Normalize\_zscore(instances):

dataset = stats.zscore(instances)

return dataset

#Normalizing data using min\_max normalization

def Normalize\_minmax(instances):

df = pd.DataFrame(instances)

min\_max\_scaler = preprocessing.MinMaxScaler()

np\_scaled = min\_max\_scaler.fit\_transform(df)

normalized = pd.DataFrame(np\_scaled)

return normalized

#plotting the normalized datasets

def plot(instances,normalized,normalized1):

plt.figure(figsize=(8,6))

plt.scatter(instances[0], instances[1], s=60, c='red',

marker='^',label="dataset")

plt.scatter(normalized[0],normalized[1],

s=60,marker='o',label="normalized by z-score")

plt.scatter(normalized1[0],normalized1[1], s=60,c=

'green',marker='x',label="normalized by min\_max")

plt.title('Normalizing the BIGtestdata22.txt')

plt.xlabel('first instance')

plt.ylabel('second instance')

plt.legend()

plt.show()

return

#Performing Leave One Out Cross Validation on the normalized data

def LeaveOneOut\_CV(current\_features,normalized,my\_feature, parameter):

feature\_set=list(current\_features)

if parameter==1:#selecting parameter to perform the desired algorithm

feature\_set.append(my\_feature)

if parameter ==2:

feature\_set.remove(my\_feature)

correct = 0

nearest\_neighbor\_distance = float('inf')

result=0

for i in range(0,len(normalized)):

one\_out=i

nearest\_neighbor\_distance = float('inf')

for h in range(0,len(normalized)):

if not np.array\_equal(h,i):

**continue. . . .**

**Last page of the code as instructed by the instructor**

**Continue...**

#This search algorithm works for both forward and backward trimming of data when

#the accuracy of level decreases from its previous level then it stops the search and exit the loop.

def SearchAlgorithm(normalized,num\_features):

print("." \* 100)

global\_acc = 0

best\_feature\_set=[]

print("." \* 100)

feature\_to\_remove = 0

feature\_to\_add = 0

curr1 = []

curr = [i for i in range(1, num\_features+1)]

prun=input("""

Enter the type of trimming needs to be done:

\n 1. Trimming of features using Forward selection .

\n 2. Trimming of features using Backward elimination \n

""")

if (prun=="1"):

parameter=1

feature\_to\_prun=feature\_to\_add

current\_set\_of\_features=curr1

features=len(normalized[0])

if (prun=="2"):

parameter=2

feature\_to\_prun=feature\_to\_remove

current\_set\_of\_features=curr

features=num\_features

start\_time = time.time()

for i in range(1, features):

print("\n On level %d of the search tree" % (i),"contains",

current\_set\_of\_features)

local\_acc = 0.0

for j in range(1,features):

if (prun=="1"):

if (j not in current\_set\_of\_features):

accuracy = LeaveOneOut\_CV(current\_set\_of\_features,

normalized,j,parameter)

if accuracy > local\_acc:

local\_acc = accuracy

feature\_to\_prun = j

if (prun=="2"):

if (j in current\_set\_of\_features):

accuracy = LeaveOneOut\_CV(current\_set\_of\_features,

normalized,j,parameter)

if accuracy > local\_acc:

local\_acc = accuracy

feature\_to\_prun = j

if local\_acc <= global\_acc:

if j == num\_features: # checks if addition of any feature results in

increase in accuracy, if not then it breaks the loop

break

if local\_acc > global\_acc: # check for decrease in accuracy

if prun=="1":

current\_set\_of\_features.append(feature\_to\_prun)# adds feature selected

by inner for loop

if prun=="2":

current\_set\_of\_features.remove(feature\_to\_prun) # remove feature

selected by inner for loop

global\_acc = local\_acc

best\_feature\_set= list(current\_set\_of\_features)

end\_time = time.time()

print("." \* 100)

print("Best set of features to use:", best\_feature\_set,"with accuracy", local\_acc

\* 100, "%")

print("Time Elapsed",end\_time - start\_time,"seconds")

def main():

print("Welcome to CS 205 Feature Selection Algorithm")

file\_name = input("Enter the name of the file to test: ")

instances=np.loadtxt(file\_name)

num\_instances=len(instances)

print("\n \*\*\* Normalizing data... \*\*\*")

normalized = Normalize\_zscore(instances)

normalized1 = Normalize\_minmax(instances)

print (normalized)

print("\n The data is normalized.")

num\_features= len(normalized[0])-1

print ("\nThis dataset has "+ str(num\_features)+ " features (without class

attribute), with "+str(num\_instances)+ " instances")

plot(instances,normalized,normalized1)

algo=input("Type the algorithm you want to run:\n \n 1.FS-Forward Selection\n

2.BE-Backward Elimination\n 3.MS- My search algorithm \n \n \n")

print ("Performing "+ str(algo)+ " ..........")

if (algo == "1"):

ForwardSelection(normalized,num\_features)

elif (algo == "2"):

BackwardElimination(normalized,num\_features)

else:

SearchAlgorithm(normalized,num\_features)

if \_\_name\_\_ == '\_\_main\_\_':

main()