Machine Learning

Shervin Halat

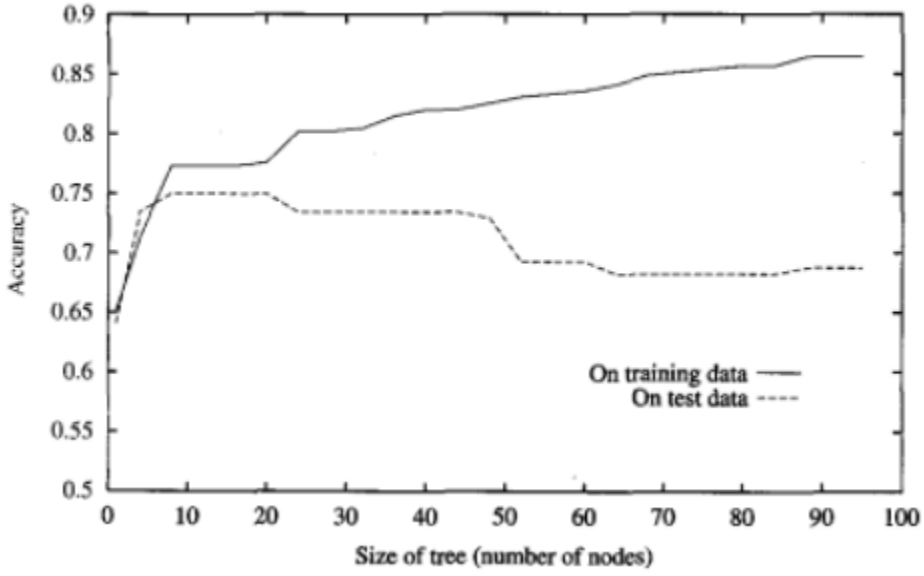
98131018

Second Homework

First Part

1.

Decision Tree in basically a recursive algorithm which will keep splitting the Training Data until it ends up with pure and perfect subsets and what that means is that the decision tree will always classify training examples perfectly. Because decision trees will keep splitting until it gets its perfect and until the accuracy is 100%. Now in many cases this means splitting until we have singleton subsets so it will keep splitting until the leaves have one example each. This is not a good thing because it has not guaranteed the testing and future data. This is what is called overfitting. So, as the decision tree keeps splitting the data, the tree gets bigger and bigger and as it gets bigger it becomes more and more accurate on the training data but it will become less accurate on the future and test data. In the following, a corresponding graph to the mentioned subject is shown.



So in order to overcome the issue of mentioned overfitting, a method called **Pruning** is suggested to decrease the size of the Decision Tree which both maximizes the accuracy and minimizes the overfitting at the same time. Therefore, pruning to a certain and optimum limit as can be figured out from the graph above would decrease the overfitting of the Decision Tree.

In order to implement pruning, it is needed to split the data set into two training and validation sets. Then the Tree is grown on the training data portion to its full depth (highest variance). Then pruning will be triggered on the validation data portion. In each iteration of the pruning method the branches (nodes) of the tree which seems that do not work well on the validation data will be removed.

2.

a. The feature “B” seems to be the best feature for classification as each training data label can be obtained by evaluating this feature.

b. Using following equations:

Entropy of a feature:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Information Gain of a feature:

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

Using equations above, the obtained Information Gain of each feature is as follows:

Gain(T, A) = E(T) – E(T, A) = E(T) – (¼ E(A_0) + ¾ E(A_1))

Gain(T, B) = E(T) – E(T, B) = E(T) – (½ E(B_0) + ½ E(B_1))

Gain(T, C) = E(T) – E(T, C) = E(T) – (½ E(C_0) + ½ E(C_1))

First we are to calculate E(A_0), E(A_1), E(B_0), E(B_1), E(C_0), E(C_1):

E(A_0) = -(1 log(1) + 0 log(0)) = 0

E(A_1) = -(2/3 log(2/3) + 1/3 log(1/3)) = 0.276

E(B_0) = -(1 log(1) + 0 log(0)) = 0

E(B_1) = -(1 log(1) + 0 log(0)) = 0

E(C_0) = -(½ log(½) + ½ log(½)) = 1

E(C_1) = -(½ log(½) + ½ log(½)) = 1

And now the E(S):

E(T) = -(½ log(½) + ½ log(½)) = 1

Therefore, we have:

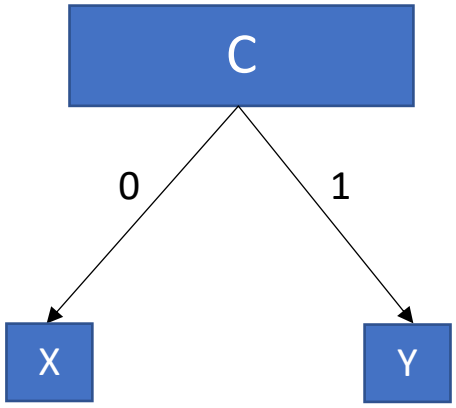
Gain(T, A) = 0.303

Gain(T, B) = 1

Gain(T, C) = 0

c.

Decision Tree



3.

According to the mentioned below article (Random Forests and Decision Trees), beside benefits accrued from Decision Tree algorithm there are some drawbacks one of which is the sorting of all numerical attributes when the tree decides to split a node. Such split on sorting all numerical attributes becomes costly i.e. efficiency or running time and memory size, especially if Decision Trees are set on data the size of which is large i.e. it has more number of instances. In 2001 the idea of Random Forests was presented which perform well as compared with other classifiers including Support Vector Machines, Neural Networks and Discriminant Analysis, and overcomes the over fitting problem. Every Decision Tree is made by randomly selecting the data from the available data. For example a Random Forest for each Decision Tree (as in Random Subspaces) can be built by randomly sampling a feature subset, and/or by the random sampling of a training data subset for each Decision Tree (the concept of Bagging). In a

Random Forest, the features are randomly selected in each decision split. The correlation between trees is reduces by randomly selecting the features which improves the prediction power and results in higher efficiency. As such the advantages of Random Forest are:

- Overcoming the problem of over fitting
- In training data, they are less sensitive to outlier data
- Parameters can be set easily and therefore, eliminates the need for pruning the trees
- variable importance and accuracy is generated automatically

Random Forest not only keeps the benefits achieved by the Decision Trees but through the use of bagging on samples, its voting scheme through which decision is made and a random subsets of variables, it most of the time achieves better results than Decision Trees.

what's more, the Random Forest is appropriate for high dimensional data modeling because it can handle missing values and can handle continuous, categorical and binary data. As mentioned before Random Forest is strong enough to overcome the problems of over fitting and hence there is no need to prune the trees.

The main conclusions of the reviewed paper:

- It can be concluded that the Random Forest achieves increased classification performance and yields results that are accurate and precise in the cases of large number of instances.
- Random Forest also covers missing values problem in the datasets and thus besides accuracy, it also overcomes the

over-fitting problem generated due to missing values in the datasets.

Article Citation:

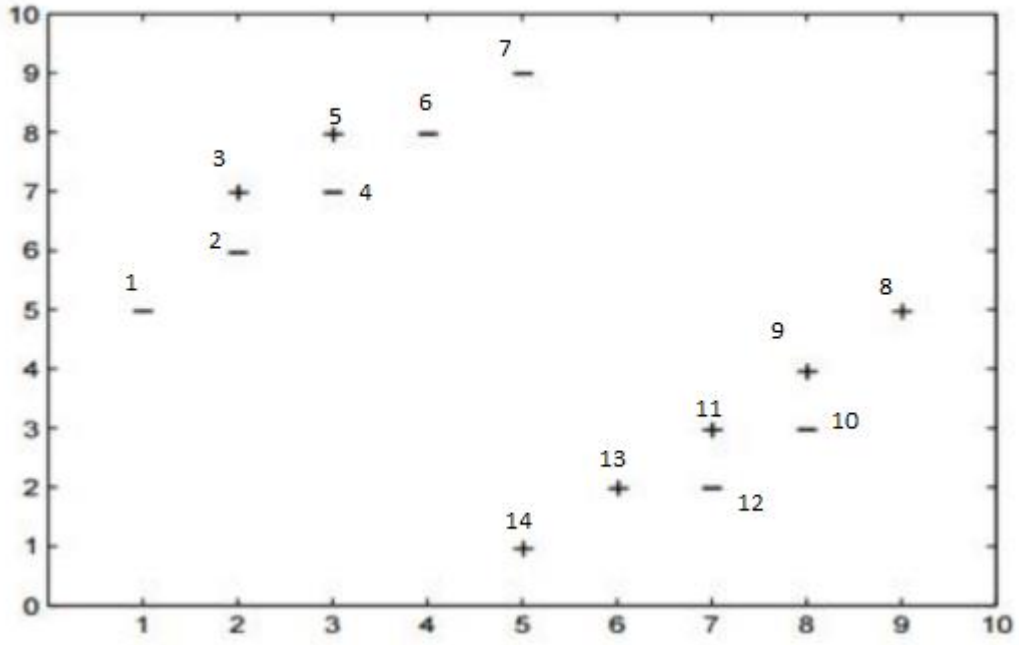
Ali, Jehad & Khan, Rehanullah & Ahmad, Nasir & Maqsood, Imran. (2012). Random Forests and Decision Trees. International Journal of Computer Science Issues(IJCSI). 9.

https://www.researchgate.net/publication/259235118_Random_Forests_and_Decision_Trees

Second Part

1.

a. Best “k” value using LOOCV method:
As there are 14 sample data, in order to implement LOOCV method, for each iteration, a new sample will be considered as Test Sample and remained 13 sample data will be considered as Training Samples. In the following the mentioned method (LOOCV) has been used to evaluate the best “K” value in KNN algorithm among values of 3, 5, 7:



K = 1:

1: - 2: + 3: - 4: + 5: - 6: + 7: - 8: + 9: -
10: + 11: - 12: + 13: - 14: +

K = 3:

1: - 2: - 3: - 4: + 5: - 6: - 7: - 8: + 9:+
10: + 11: - 12: + 13: + 14: +

K = 5:

1: - 2: - 3: - 4: - 5: - 6: - 7: - 8: + 9: +
10: + 11: + 12: + 13: + 14: +

K-value	1	3	5
Error	10/14	6/14	4/14

Hence, the best K-value is '5'.

b.

Precision of KNN algorithm for this dataset using K = 5:

1: - 2: - 3: - 4: - 5: - 6: - 7: - 8: + 9: + 10: +
11: + 12: + 13: + 14: +

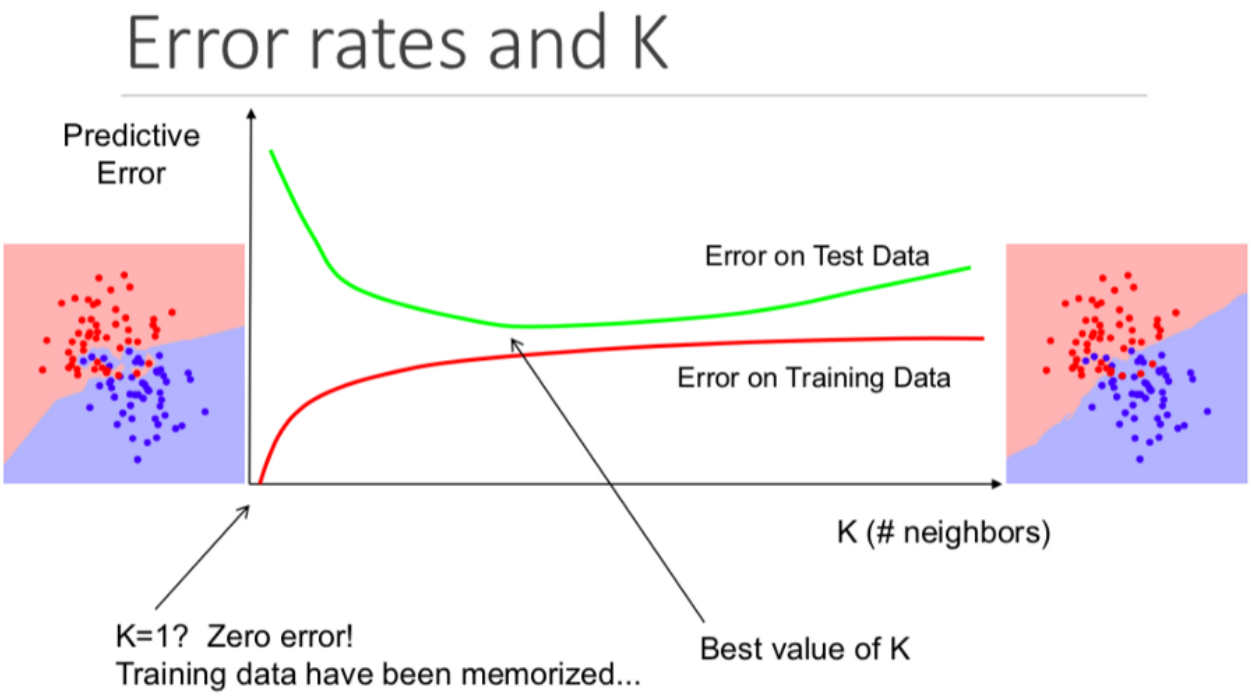
Precision = 10/14

2.

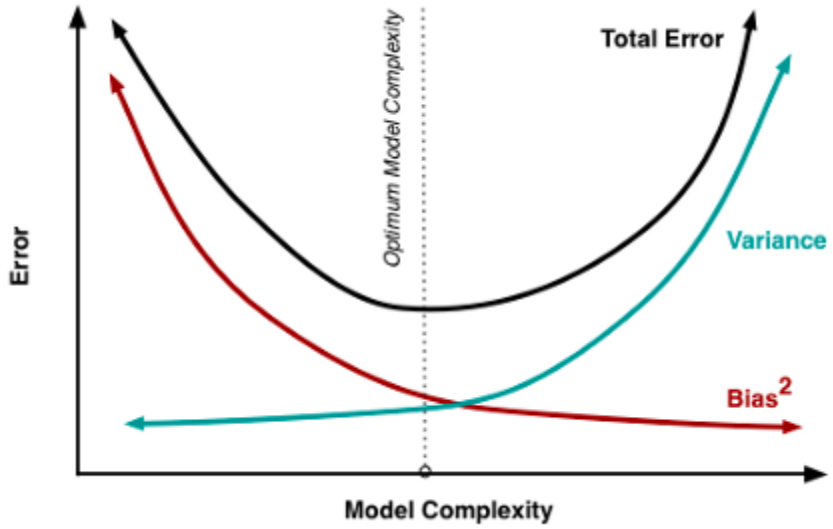
As we know KNN is a lazy learning algorithm. By considering an extreme case K = 1, the training data will perfectly predicted and hence the bias will become 0 when k = 1 but when it comes to new data (in test set) it has higher chance to be an error which leads to high variance. When we increase K the training error will increase (increase bias), but test error may decrease at the same time (decrease variance).

Therefore, for k = 1 we can say that the complexity of the KNN model reaches its peak and by increasing it the complexity decreases.

Now by splitting the data into training and validation set the following graph will be generated for different values of K.

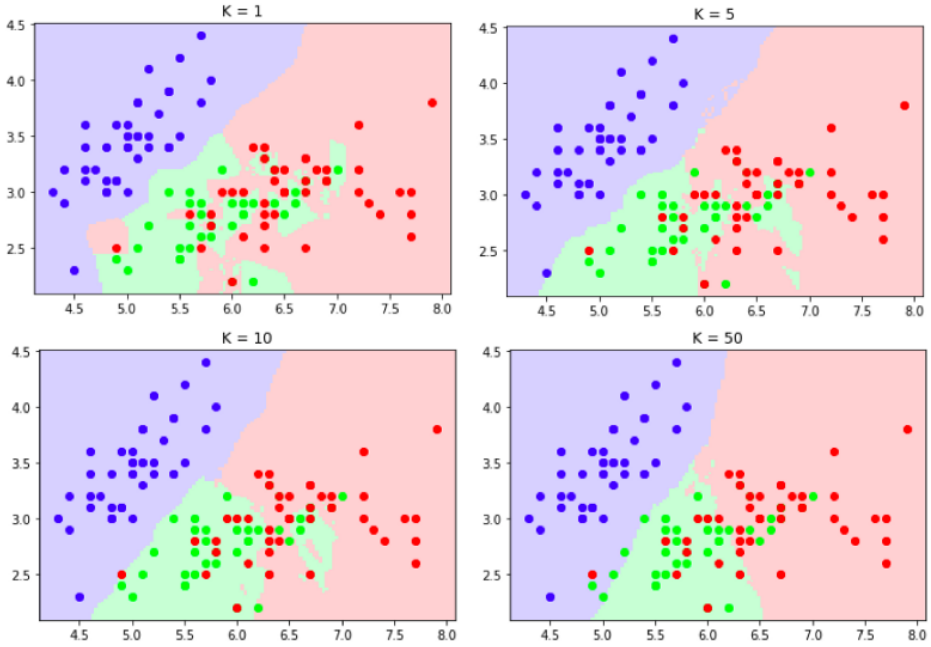


As it can be figured out from the figure above, as we increase value of K from 1, the Error of Training Data keeps constantly increasing but the Error of Test Data decreases to an optimum value of K (complexity) but afterward it starts to increase as K increases. In the following the effect of model complexity (K value in other words) in KNN algorithm on Total Error, Variance and Bias is shown:



As it can be figured out from the graph above there is an optimum value for K (or model complexity) in KNN algorithm.

Also it should be noted that, as the chosen K is closer to the optimum value, the boundaries of the classifier becomes more consistent and reasonable at the same time which can be seen in the following graphs.



Visualization of the results based on different K for KNN algorithm

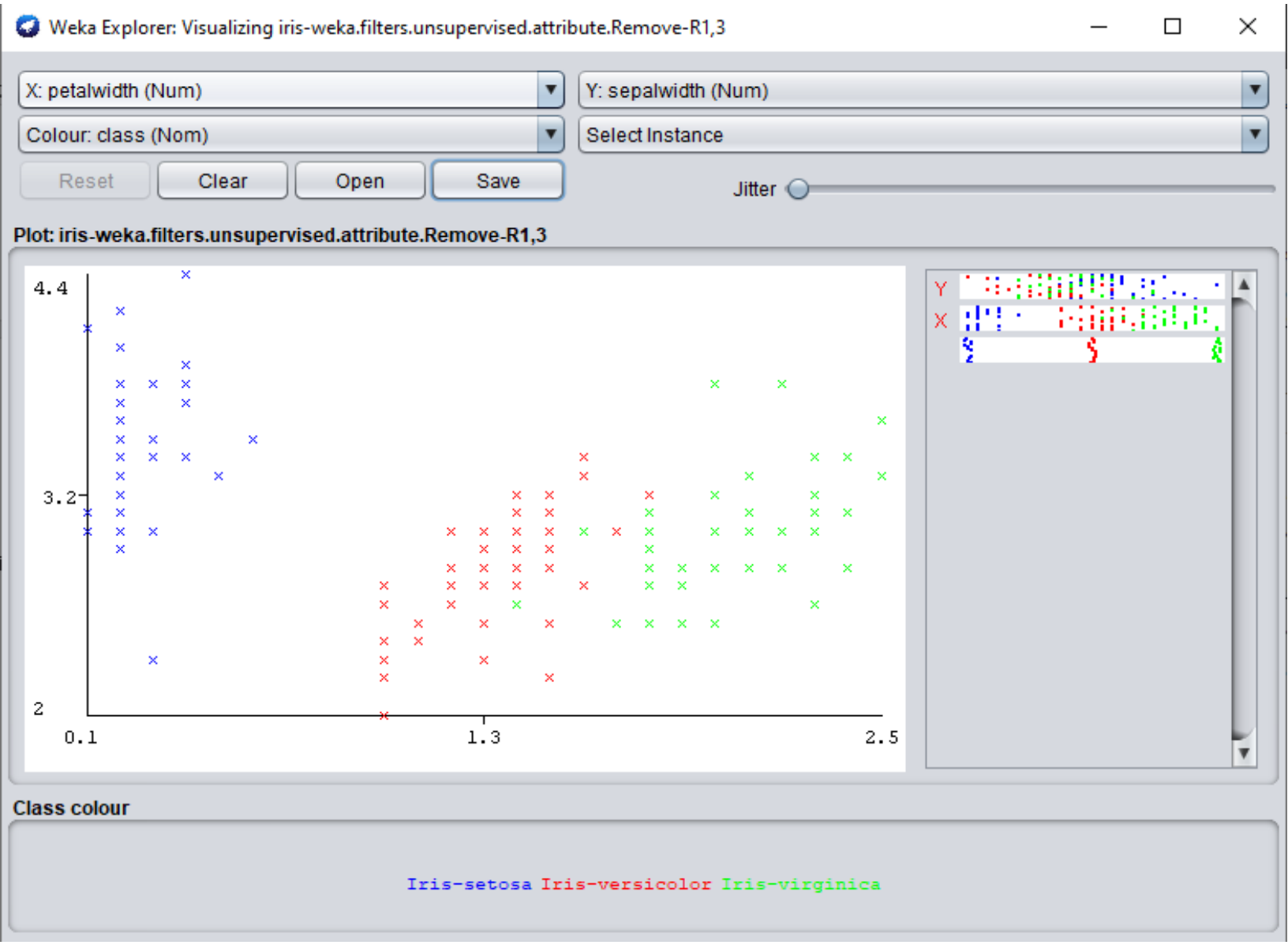
3. As it was mentioned and as it can be figured out from the graphs above, for the low values of “K”, the model is more prone to overfitting and the outlier data will be more considered in the model and therefore the model will be more sensitive to the noise data and the model won’t be reliable. To overcome this hurdle it is suggested to increase the value of “K”. Since as we increase the “K” value the model, the outlier data (noise data) would become the minority portion of the data considered for classifying. It should be noted that increasing K value may result in high bias of the model therefore we have to choose a trade off between variance and bias as mentioned in the previous part.

4. By considering what’s just mentioned in previous part, as it can be noticed from the graph in the left side, the model is too sensitive to outlier data and as we move to the right this sensitivity lowers. Consequently we can say that the graph in the left has the lowest “K” value and it increases as we move to the right.

Third Part

1.

a. Visualization of the Iris data considering sepal and petal width:



Sepalwidth vs petalwidth graph of Iris data using weka (whitout noise)

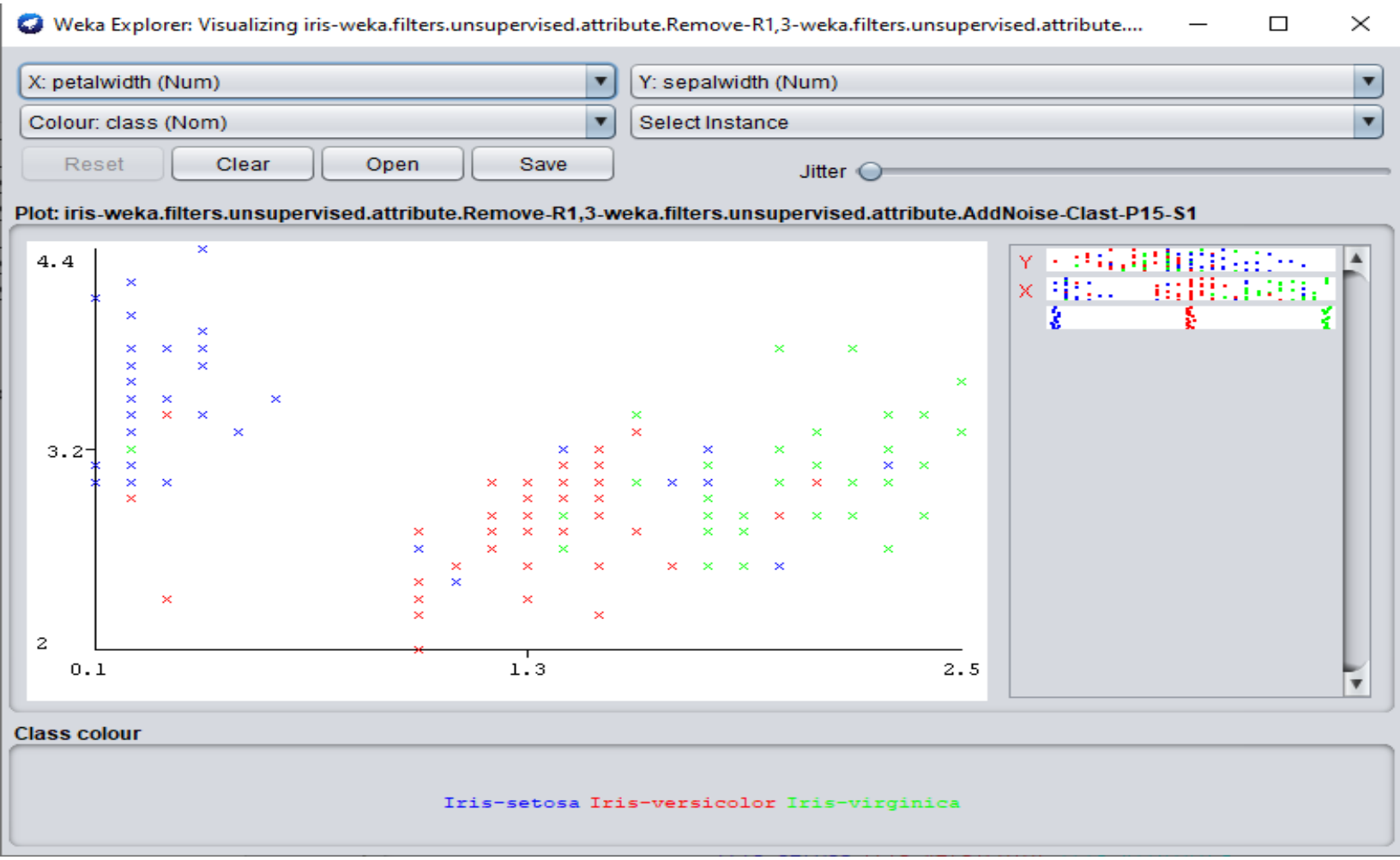
b.

Accuracy of KNN algorithm on mentioned data for K values of 1 to 5:

K-value	Accuracy
1	92.6667 %
2	94.6667 %
3	96 %
4	95.3333 %
5	95.3333 %

c.

Visualization of the data after adding 15 percent noise data:



Sepalwidth vs petalwidth graph of Iris data using weka (after 15% noise added)

As it can be seen in the graph above, some noise data has been added comparing to the previous graph.

Now we evaluate accuracy of KNN algorithm implemented for “K” values of 1 to 5:

K-value	Accuracy
1	74 %
2	76 %
3	79.3333 %
4	80.6667 %
5	80.6667 %

d.

Considering first table in part “a” as K-value increases the accuracy of the model also increases to a specific limit and it starts to lower afterwards which shows us there is an optimum value for K-value. Considering the second table in part “c”, it can be noticed that although the accuracy of the model has decreased generally, but as we increase the k-value, the Accuracy increases simultaneously which is consistent with what had just been mentioned in previous parts.

2.

a. for Vote data set we choose J48 algorithm and the confusion matrix is as follows:

with pruning:

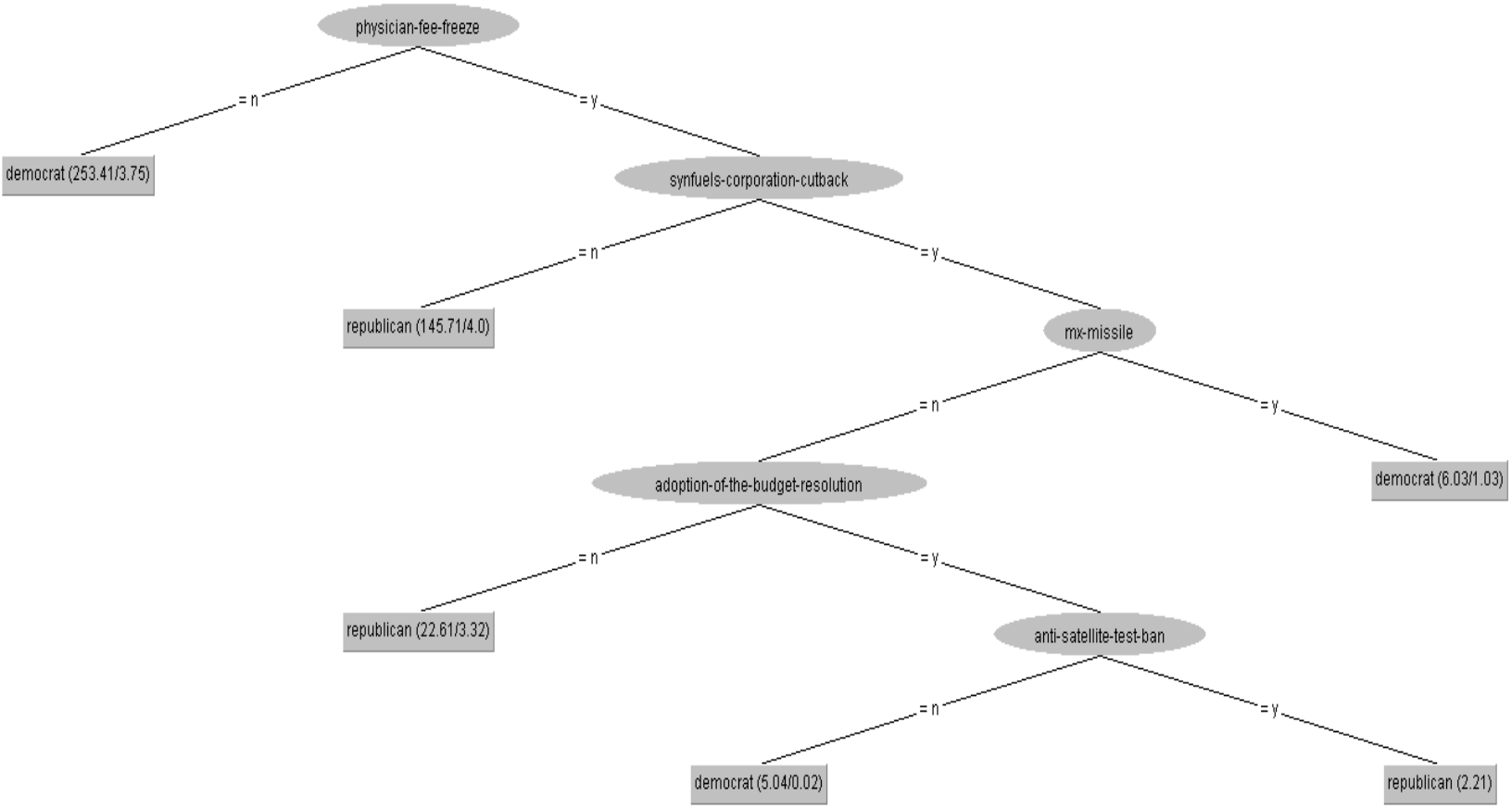
Classifier output

```
Size of the tree :      11


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         419               96.3218 %
Incorrectly Classified Instances        16                3.6782 %
Kappa statistic                          0.9224
Mean absolute error                      0.0611
Root mean squared error                  0.1748
Relative absolute error                 12.887  %
Root relative squared error             35.9085 %
Total Number of Instances              435

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.970    0.048    0.970      0.970    0.970      0.922    0.971     0.965     democrat
                 0.952    0.030    0.952      0.952    0.952      0.922    0.971     0.947     republican
Weighted Avg.    0.963    0.041    0.963      0.963    0.963      0.922    0.971     0.958

=== Confusion Matrix ===

   a   b   <-- classified as
 259   8 |   a = democrat
   8 160 |   b = republican
```

Pruned Confusion Matrix:

```
=== Confusion Matrix ===

   a   b   <-- classified as
 259   8 |   a = democrat
   8 160 |   b = republican
```

Pruned Tree Visualization



Pruned Tree Visualization for Vote Data by Weka

b.

without pruning:

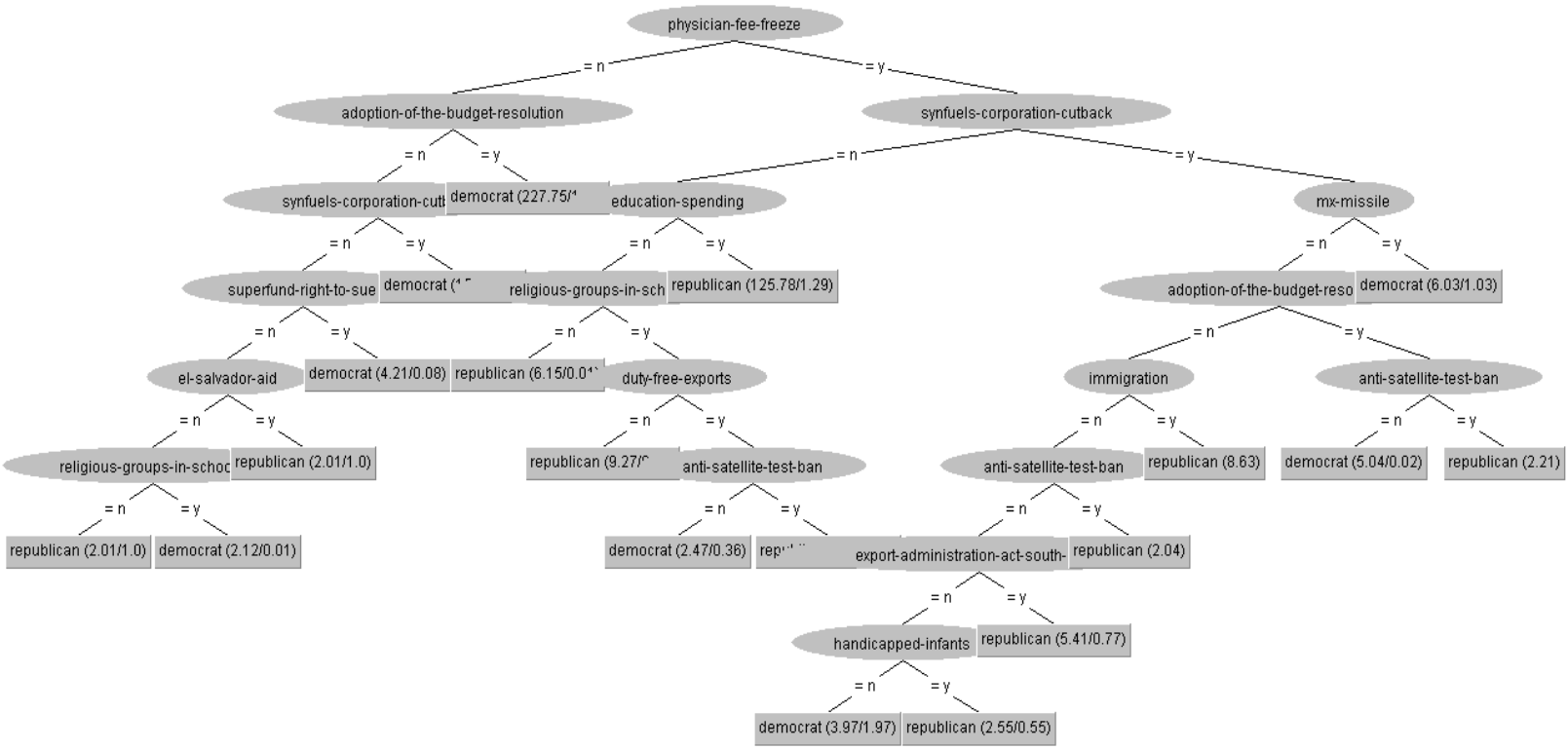
Classifier output

```
Size of the tree :      37


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         419               96.3218 %
Incorrectly Classified Instances        16                3.6782 %
Kappa statistic                          0.9224
Mean absolute error                      0.0552
Root mean squared error                  0.1748
Relative absolute error                 11.6378 %
Root relative squared error             35.8923 %
Total Number of Instances              435

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.970    0.048    0.970      0.970    0.970      0.922    0.978     0.980     democrat
                 0.952    0.030    0.952      0.952    0.952      0.922    0.977     0.955     republican
Weighted Avg.    0.963    0.041    0.963      0.963    0.963      0.922    0.978     0.971

=== Confusion Matrix ===

   a   b   <-- classified as
 259   8 |   a = democrat
   8 160 |   b = republican
```

Pruned Confusion Matrix:

```
=== Confusion Matrix ===

   a   b   <-- classified as
 259   8 |   a = democrat
   8 160 |   b = republican
```

Unpruned Tree Visualization



Unpruned Tree Visualization for Vote Data by Weka

d.

As we can see in the pictures above, the accuracy of the algorithm in pruned and unpruned hasn't remarkably changed. This shows that although we pruned the tree and in other words changed it to a more simple and faster model but the accuracy didn't fall. Therefore, pruning a Decision Tree works wonders to us.

Programming Problems:

1.

a.

For normalized features:

The best value of K is 5 according to the comparison of K-values' accuracies. The code (p1.a) output is shown below:

```
accuracy of KNN algorithm for K = 1 is 0.737374
accuracy of KNN algorithm for K = 2 is 0.707071
accuracy of KNN algorithm for K = 3 is 0.767677
accuracy of KNN algorithm for K = 4 is 0.727273
accuracy of KNN algorithm for K = 5 is 0.787879
accuracy of KNN algorithm for K = 6 is 0.767677
accuracy of KNN algorithm for K = 7 is 0.757576
accuracy of KNN algorithm for K = 10 is 0.737374
accuracy of KNN algorithm for K = 15 is 0.737374
so the best K-value would be 5 with the accuray of 0.787879
```

The accuracy of training and test data for K-value of 5 is as following according to the code (p1.a) output:

```
accuracy of training data for K-value of 5 is 0.868687
accuracy of test data for K-value of 5 is 0.787879
```

Confusion Matrix of Training Data:

predicted / actual	Posetive(1)	Negative(0)
Posetive(1)	74	16
Negative(0)	10	98

Confusion Matrix of Test Data:

predicted / actual	Posetive(1)	Negative(0)
Posetive(1)	30	17
Negative(0)	4	48

b.

For unnormalized features:

Again all the steps that were taken in previous part has been taken but this time **for unnormalized features**:

The results of the corresponding code (p1.b) are as follows:

```
accuracy of KNN algorithm for K = 1 is 0.626263
accuracy of KNN algorithm for K = 2 is 0.595960
accuracy of KNN algorithm for K = 3 is 0.666667
accuracy of KNN algorithm for K = 4 is 0.626263
accuracy of KNN algorithm for K = 5 is 0.707071
accuracy of KNN algorithm for K = 6 is 0.626263
accuracy of KNN algorithm for K = 7 is 0.636364
accuracy of KNN algorithm for K = 10 is 0.676768
accuracy of KNN algorithm for K = 15 is 0.656566
so the best K-value would be 5
accuracy of training data for K-value of 5 is 0.797980
accuracy of test data for K-value of 5 is 0.707071
```

As it was expected, it can be noticed form the results above that all accuracies for different values of “K” has been decreased for both Training and Test Data. More importantly. the best K-value is still 5. Therefore, in KNN algorithm normalizing features may boost accuracy of the algorithm to a remarkable level but it has no effect on optimum K-value.

In the following the Confusion Matrixes of Training and Test Data for unnormalized features is shown:

Confusion Matrix of Training Data: (according to code p1.a)

predicted / actual	Posetive(1)	Negative(0)
Posetive(1)	65	25
Negative(0)	15	93

Confusion Matrix of Training Data

Confusion Matrix of Test Data: (according to code p1.a)

predicted / actual	Posetive(1)	Negative(0)
Posetive(1)	27	20
Negative(0)	9	43

Confusion Matrix of Test Data

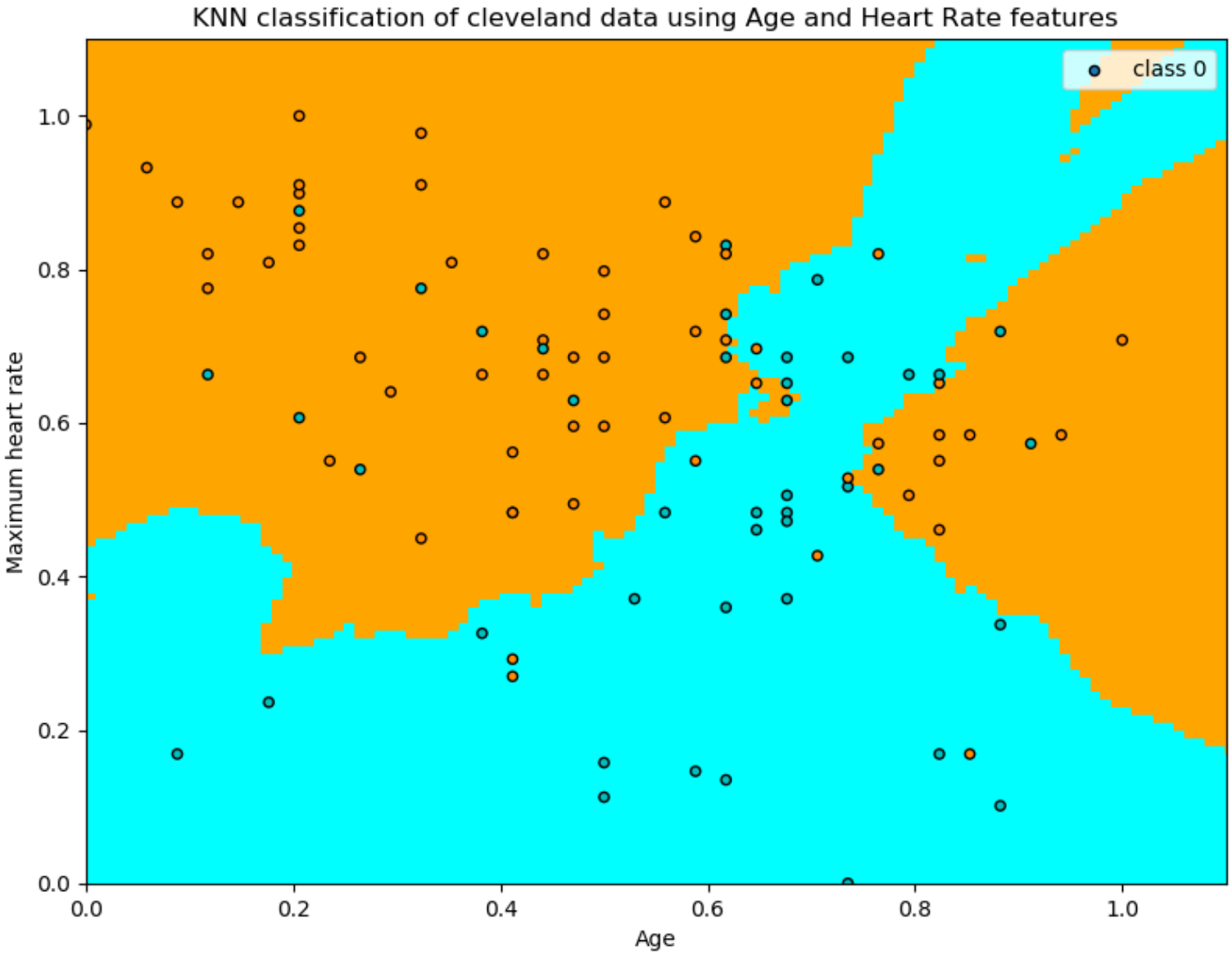
Here as it was expected too, the values of True-Positive and True-Negative for both of the Confusion Matrix Tables for unnormalized features has decreased compared to normalized one.

c.

Considering only two features of “Age” and “Maximum Heart Rate” and comparing K-values of 1 to 30 the best K-value would be **13**. Hence, for the rest of the solution the K-value is allocated 13. (according to the code p1.c):

```
accuracy of KNN algorithm for K = 1 is 0.621212
accuracy of KNN algorithm for K = 2 is 0.646465
accuracy of KNN algorithm for K = 3 is 0.661616
accuracy of KNN algorithm for K = 4 is 0.641414
accuracy of KNN algorithm for K = 5 is 0.666667
accuracy of KNN algorithm for K = 6 is 0.671717
accuracy of KNN algorithm for K = 7 is 0.681818
accuracy of KNN algorithm for K = 8 is 0.676768
accuracy of KNN algorithm for K = 9 is 0.696970
accuracy of KNN algorithm for K = 10 is 0.696970
accuracy of KNN algorithm for K = 11 is 0.707071
accuracy of KNN algorithm for K = 12 is 0.702020
accuracy of KNN algorithm for K = 13 is 0.712121
accuracy of KNN algorithm for K = 14 is 0.696970
accuracy of KNN algorithm for K = 15 is 0.702020
accuracy of KNN algorithm for K = 16 is 0.702020
accuracy of KNN algorithm for K = 17 is 0.696970
accuracy of KNN algorithm for K = 18 is 0.712121
accuracy of KNN algorithm for K = 19 is 0.691919
accuracy of KNN algorithm for K = 20 is 0.707071
accuracy of KNN algorithm for K = 21 is 0.691919
accuracy of KNN algorithm for K = 22 is 0.666667
accuracy of KNN algorithm for K = 23 is 0.666667
accuracy of KNN algorithm for K = 24 is 0.641414
accuracy of KNN algorithm for K = 25 is 0.671717
accuracy of KNN algorithm for K = 26 is 0.651515
accuracy of KNN algorithm for K = 27 is 0.666667
accuracy of KNN algorithm for K = 28 is 0.641414
accuracy of KNN algorithm for K = 29 is 0.671717
accuracy of KNN algorithm for K = 30 is 0.656566
so the best K-value would be 13
```

Using code p1.c the colored map for KNN with K-value of 13 would be:



d.

Manhattan-Distance:

The best value of K is 3 according to the comparison of K-values' accuracies. The code (p1.d) output is shown below:

```
accuracy of KNN algorithm for K = 1 is 0.767677
accuracy of KNN algorithm for K = 2 is 0.707071
accuracy of KNN algorithm for K = 3 is 0.828283
accuracy of KNN algorithm for K = 4 is 0.767677
accuracy of KNN algorithm for K = 5 is 0.787879
accuracy of KNN algorithm for K = 6 is 0.777778
accuracy of KNN algorithm for K = 7 is 0.777778
accuracy of KNN algorithm for K = 10 is 0.757576
accuracy of KNN algorithm for K = 15 is 0.767677
so the best K-value would be 3
```

The accuracy of training and test data for K-value of 3 is as following according to the code (p1.d) output:

```
accuracy of training data for K-value of 3 is 0.924242
accuracy of test data for K-value of 3 is 0.828283
```

Confusion Matrix of Training Data:

predicted / actual	Posetive(1)	Negative(0)
Posetive(1)	83	7
Negative(0)	8	100

Confusion Matrix of Training Data

Confusion Matrix of Test Data:

predicted / actual	Posetive(1)	Negative(0)
Posetive(1)	34	13
Negative(0)	4	48

Confusion Matrix of Test Data

Chebyshev-Distance:

The best value of K is 5 according to the comparison of K-values' accuracies. The code (p1.d) output is shown below:

```
accuracy of KNN algorithm for K = 1 is 0.696970
accuracy of KNN algorithm for K = 2 is 0.656566
accuracy of KNN algorithm for K = 3 is 0.717172
accuracy of KNN algorithm for K = 4 is 0.676768
accuracy of KNN algorithm for K = 5 is 0.727273
accuracy of KNN algorithm for K = 6 is 0.707071
accuracy of KNN algorithm for K = 7 is 0.696970
accuracy of KNN algorithm for K = 10 is 0.686869
accuracy of KNN algorithm for K = 15 is 0.676768
so the best K-value would be 5
```

The accuracy of training and test data for K-value of 5 is as following according to the code (p1.d) output:

```
accuracy of training data for K-value of 5 is 0.843434
accuracy of test data for K-value of 5 is 0.727273
```

Confusion Matrix of Training Data:

predicted / actual	Posetive(1)	Negative(0)
Posetive(1)	76	14
Negative(0)	17	91

Confusion Matrix of Training Data

Confusion Matrix of Test Data:

predicted / actual	Posetive(1)	Negative(0)
Posetive(1)	30	17
Negative(0)	10	42

Confusion Matrix of Test Data

2.

a.

For solving this problem no feature has been normalized.

For the first step the best K-value among values of 1 to 30 were examained. The results are as following:

(These results obtained after splitting data into training , evaluatioin and test sets as asked in the question. Evaluation set was used in order to find accuracies.)

```
accuracy of KNN algorithm for K = 1 is 0.980501
accuracy of KNN algorithm for K = 2 is 0.974930
accuracy of KNN algorithm for K = 3 is 0.974930
accuracy of KNN algorithm for K = 4 is 0.977716
accuracy of KNN algorithm for K = 5 is 0.969359
accuracy of KNN algorithm for K = 6 is 0.974930
accuracy of KNN algorithm for K = 7 is 0.974930
accuracy of KNN algorithm for K = 8 is 0.977716
accuracy of KNN algorithm for K = 9 is 0.966574
accuracy of KNN algorithm for K = 10 is 0.966574
accuracy of KNN algorithm for K = 11 is 0.966574
accuracy of KNN algorithm for K = 12 is 0.966574
accuracy of KNN algorithm for K = 13 is 0.963788
accuracy of KNN algorithm for K = 14 is 0.958217
accuracy of KNN algorithm for K = 15 is 0.963788
accuracy of KNN algorithm for K = 16 is 0.963788
accuracy of KNN algorithm for K = 17 is 0.952646
accuracy of KNN algorithm for K = 18 is 0.952646
accuracy of KNN algorithm for K = 19 is 0.944290
accuracy of KNN algorithm for K = 20 is 0.944290
accuracy of KNN algorithm for K = 21 is 0.947075
accuracy of KNN algorithm for K = 22 is 0.949861
accuracy of KNN algorithm for K = 23 is 0.949861
accuracy of KNN algorithm for K = 24 is 0.947075
accuracy of KNN algorithm for K = 25 is 0.947075
accuracy of KNN algorithm for K = 26 is 0.947075
accuracy of KNN algorithm for K = 27 is 0.947075
accuracy of KNN algorithm for K = 28 is 0.947075
accuracy of KNN algorithm for K = 29 is 0.947075
accuracy of KNN algorithm for K = 30 is 0.947075
so the best K-value would be 1
```

According to the results above **the best K-value would be 1 !**

And the accuracies of training, evaluation and test sets would be as follows respectively:

```
accuracy of training data for K-value of 1 is 1.000000
accuracy of evaluation data for K-value of 1 is 0.980501
accuracy of test data for K-value of 1 is 0.947222
```

Now we will evaluate the best Disctance Method among three Euclidean, Manhattan and Chebyshev Methods:

Euclidean:

The accuray of three sets using Euclidean are as follows:

```
accuracy of training data for K-value of 1 is 1.000000
accuracy of evaluation data for K-value of 1 is 0.980501
accuracy of test data for K-value of 1 is 0.947222
```

Manhattan:

The accuray of three sets using Manhattan are as follows:

```
accuracy of training data for K-value of 1 is 1.000000
accuracy of evaluation data for K-value of 1 is 0.969359
accuracy of test data for K-value of 1 is 0.930556
```

Chebyshev:

The accuray of three sets using Chebyshev are as follows:

```
accuracy of training data for K-value of 1 is 1.000000
accuracy of evaluation data for K-value of 1 is 0.955432
accuracy of test data for K-value of 1 is 0.933333
```

By comparing results of three Distance Methods we found that **Euclidean Distance is the best distance calculator method** as it reached to the highest accuracies for both evaluation and test sets.

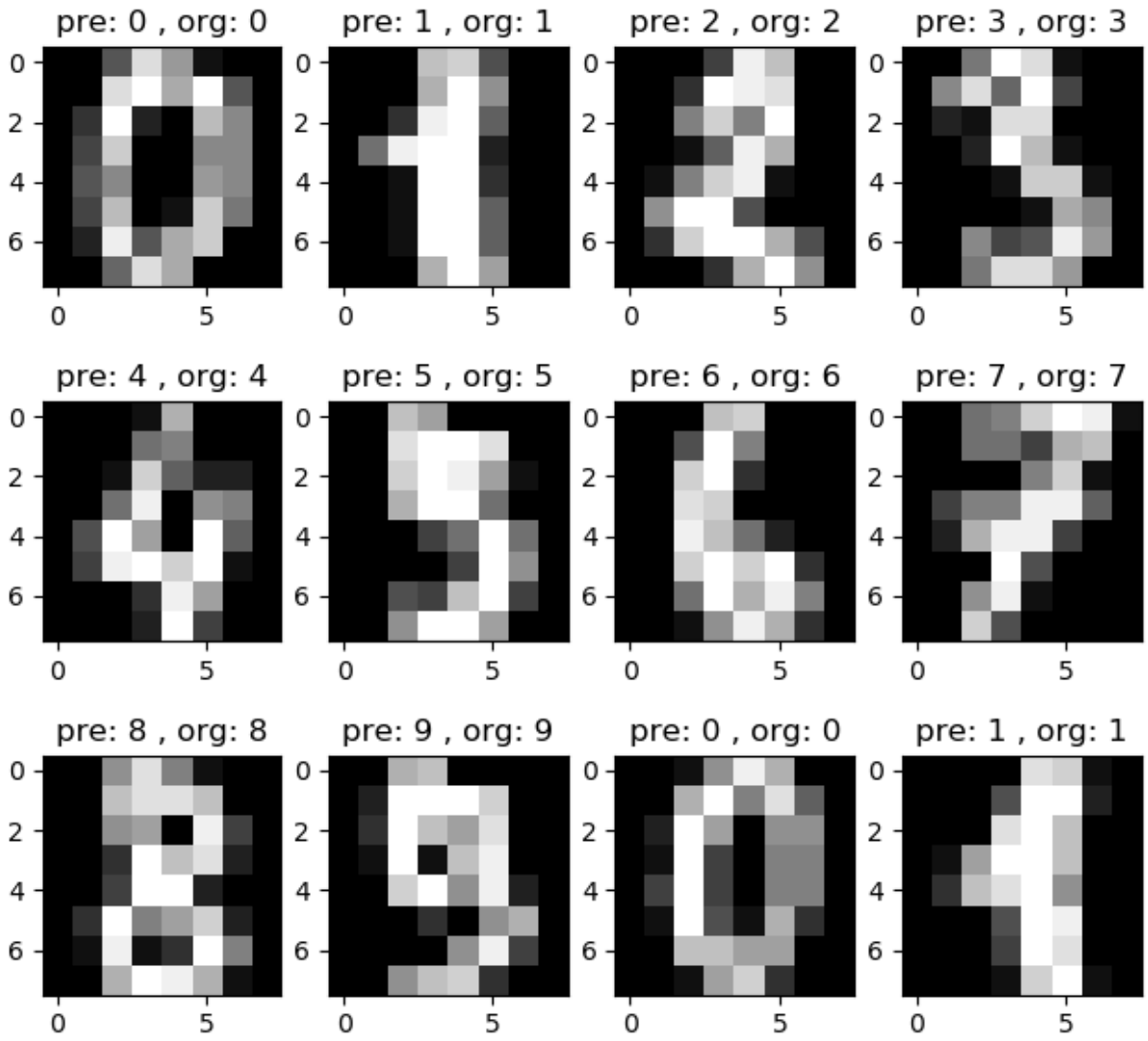
Therefore, by choosing Euclidean Distance and K-value of 1, the Confusion Matrix of Test Set would be as follows:

(Columns: 0 to 10 // Rows: 0 to 10)

```
[[34.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0. 36.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0. 34.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. 31.  0.  2.  0.  1.  2.  1.]
 [ 0.  0.  0.  0. 34.  0.  0.  0.  0.  3.]
 [ 0.  0.  0.  0.  0. 37.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0. 37.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. 36.  0.  0.]
 [ 0.  4.  0.  0.  0.  0.  0.  0. 28.  1.]
 [ 0.  0.  0.  1.  0.  2.  0.  0.  0. 34.]]
```

Confusion Matrix of Test set (K = 1, Euclidean Distance)

Now we are going to show 12 first photos with its real and predicted values as photos upper labels using code (p2.a.image):



As we can see the predicted and actual labels for all photos are exactly the same! Which was predictable due to the high value of accuracy obtained in previous part.

b.

The Script of this part is as the name of **p2.b** in the uploaded files which is another version of part “a” but this time with using scikit learn libraries!

```
accuracy of test set using K-value of 1 set is 0.988889
accuracy of test set using K-value of 2 set is 0.988889
accuracy of test set using K-value of 3 set is 0.983333
accuracy of test set using K-value of 4 set is 0.980556
accuracy of test set using K-value of 5 set is 0.986111
accuracy of test set using K-value of 6 set is 0.983333
accuracy of test set using K-value of 7 set is 0.986111
accuracy of test set using K-value of 8 set is 0.983333
accuracy of test set using K-value of 9 set is 0.983333
accuracy of test set using K-value of 10 set is 0.986111
accuracy of test set using K-value of 11 set is 0.986111
accuracy of test set using K-value of 12 set is 0.975000
accuracy of test set using K-value of 13 set is 0.972222
accuracy of test set using K-value of 14 set is 0.969444
accuracy of test set using K-value of 15 set is 0.972222
accuracy of test set using K-value of 16 set is 0.972222
accuracy of test set using K-value of 17 set is 0.966667
accuracy of test set using K-value of 18 set is 0.963889
accuracy of test set using K-value of 19 set is 0.963889
accuracy of test set using K-value of 20 set is 0.966667
the best K-value by comparison is 1
```

Again like previous part the best K-value is 1 with the accuracy of 98.88 percent!