Table of Contents

---

1. ***Consider the dataset attached with this assignment. It has 450 data points on a 2-D plane. The first two columns contain the X and Y attribute values and the third column contains the class label. Perform the following tasks using this dataset and include the answers/results in your submission. Use Matlab or Scikit toolboxes (Python) to perform all the tasks and include the commands used to achieve each outcome/result.***

    a. ***(4) Display all the data points on a 2-D grid, distinguishing the points belonging to the two classes.***

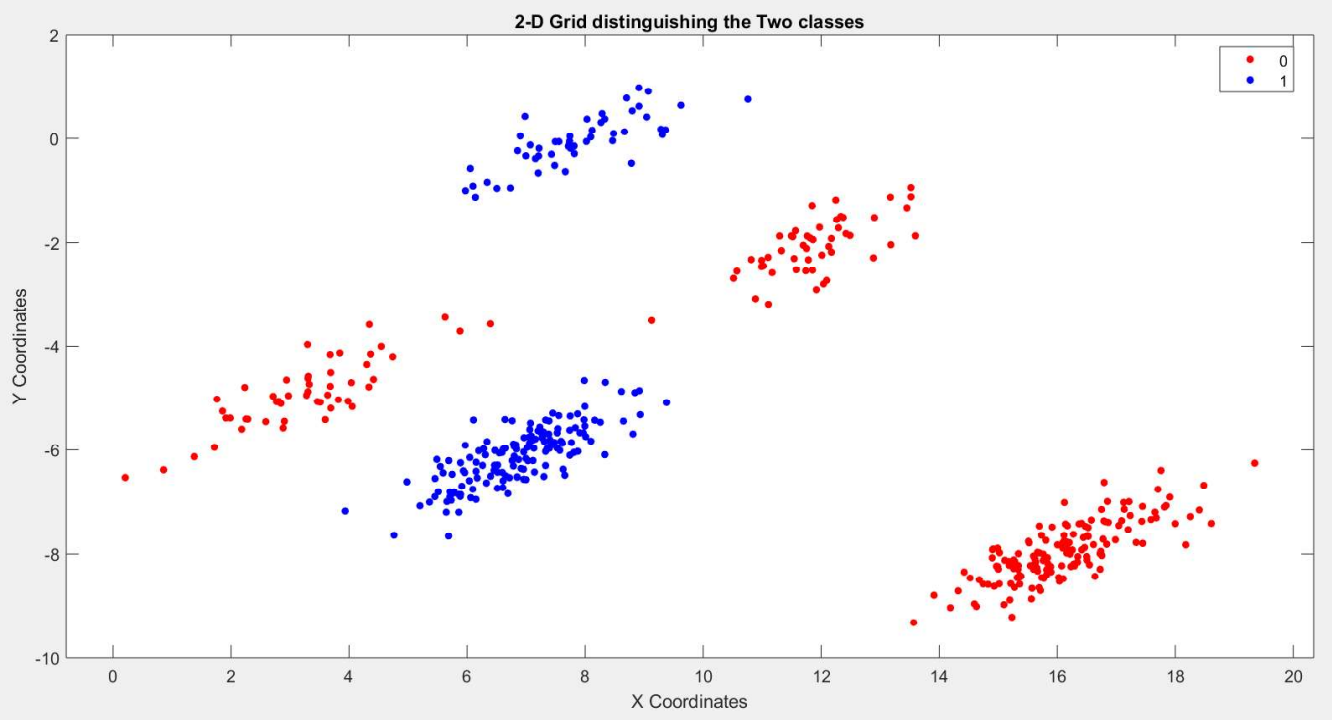
Solution of 1(a):

Matlab Commands Used:

Command Window

```
>> DataTable=readtable('HW2-Synth-Data.csv')
gscatter(DataTable.X,DataTable.Y,DataTable.class,'rb')
title('2-D Grid distinguishing the Two classes')
xlabel('X Coordinates')
ylabel('Y Coordinates')
```

Plot of 2-D grid displaying the points 2 classes:



Explanation 1(a) : The Graph depicts the 450 points plotted over a 2-D plane .The Red points indicate the points with class values '0' and the blue values indicate the points with class values '1'.

---

b. (10) Use 5-fold testing to the find best decision tree that you can fit to this dataset. Show all the parameter-value choices you made to get the best performance. You must try to tune the parameters until you get the best possible performance. Use all 450 data points as the test set to compute the confusion matrix, accuracy, and precision and recall values for each class.

Solution of 1(b):

In 5Fold technique the dataset is divided into 5 parts and 4 out of 5 parts are used for training the Data and the 5^{th} partition is used for testing the Data. Let's optimize the Error of the 5 partitions using '**Optimize Hyperparameters** ' parameter and it will imply the best decision tree possible. Since we have to use k fold method lets use another parameter **'HyperparameterOptimizationOptions'** for the optimization technique to be used as k-fold with value 5.By optimization of the parameters on min Leaf size we have found that the Tree with **'minLeafsize'** 1 is the best decision tree for this dataset with the code given below.

Parameters Selected:

1. **Optimize Hyperparameters, HyperparameterOptimizationOptions – k-fold with the value 5**

2. **minLeafsize with size '1'**

Matlab Code for tuning the best parameter and the technique with k-fold value=5 :

```
Command Window
>> Treec=fitctree(DataTable,'class','OptimizeHyperparameters','auto','HyperparameterOptimizationOptions',struct('kfold',5))
|=====================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   | MinLeafSize |
|      | result |             | runtime     | (observed)  | (estim.)    |             |
|=====================================================================================|
|    1 | Best   |        0.12 |    0.063898 |        0.12 |        0.12 |         120 |
|    2 | Best   |    0.022222 |    0.043906 |    0.022222 |    0.035505 |          29 |
|    3 | Best   |        0.02 |    0.040755 |        0.02 |    0.027137 |           2 |
|    4 | Accept |    0.024444 |    0.055904 |        0.02 |    0.020198 |           7 |
|    5 | Accept |        0.02 |    0.040093 |        0.02 |    0.020001 |           3 |
|    6 | Accept |    0.022222 |     0.04005 |        0.02 |    0.019999 |          17 |
|    7 | Best   |    0.017778 |    0.038031 |    0.017778 |    0.017783 |           1 |
|    8 | Accept |    0.017778 |    0.063302 |    0.017778 |     0.01778 |           1 |
|    9 | Accept |    0.017778 |    0.038319 |    0.017778 |    0.017779 |           1 |
|   10 | Accept |    0.017778 |    0.047794 |    0.017778 |    0.017779 |           1 |
|   11 | Accept |     0.44444 |    0.039353 |    0.017778 |    0.017779 |         225 |
|   12 | Accept |    0.068889 |    0.042226 |    0.017778 |    0.017779 |          60 |
|   13 | Accept |    0.026667 |    0.040147 |    0.017778 |    0.017779 |           5 |
|   14 | Accept |    0.022222 |    0.042075 |    0.017778 |    0.017779 |          11 |
|   15 | Accept |    0.022222 |    0.042334 |    0.017778 |    0.017779 |          23 |
|   16 | Accept |        0.12 |    0.043878 |    0.017778 |    0.017779 |          85 |
|   17 | Accept |    0.022222 |    0.044883 |    0.017778 |    0.017779 |          40 |
|   18 | Accept |    0.022222 |    0.040095 |    0.017778 |    0.017779 |           4 |
|   19 | Accept |    0.022222 |    0.040044 |    0.017778 |    0.017778 |           9 |
|   20 | Accept |    0.022222 |    0.040065 |    0.017778 |    0.017778 |          14 |
|=====================================================================================|
```

Optimization is performed on the parameter min leaf size for the Dataset and the optimal tree is formed.

Verifying the k-fold Loss individual and average for the dataset.

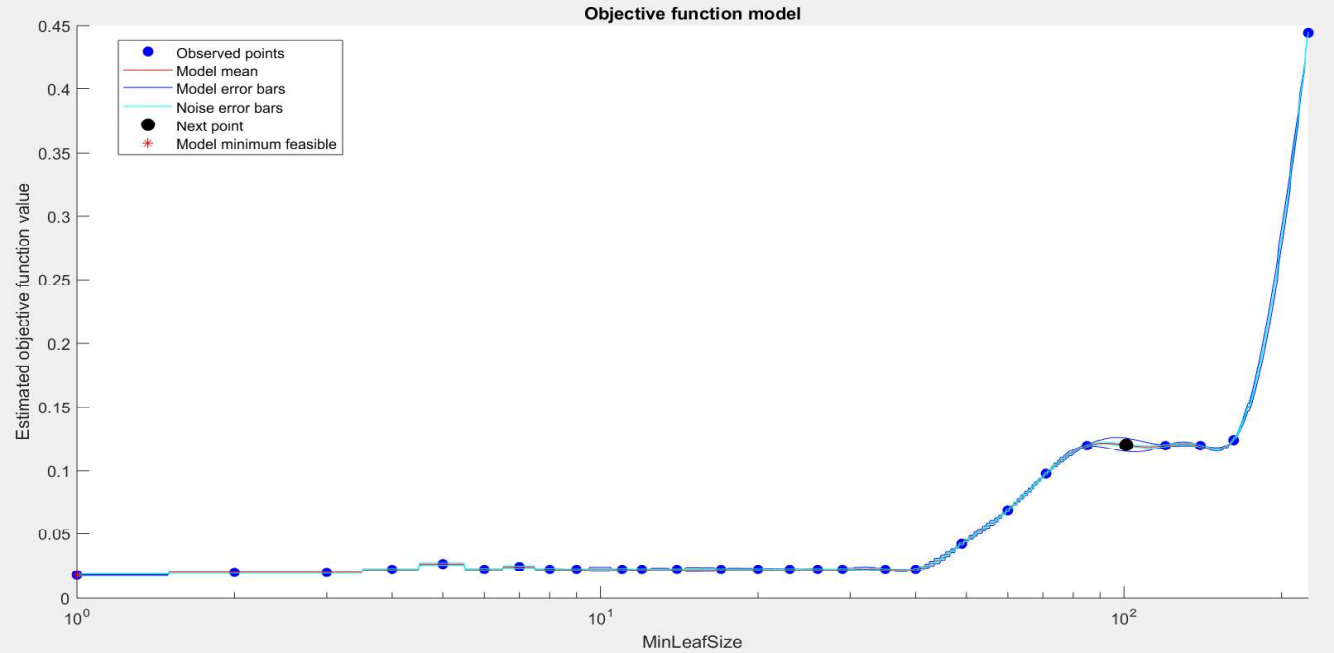
The k-fold Losses for the optimized 5 trees obtained are 0.0222,0,0.0222,0.0111,0.0222. This Indicates that the misclassifications for each k-fold run are 2,0,2,1,2.By calculating the average k-fold Loss for the Dataset we get it as 0.0156,which is very Low due to optimization of the 5 runs. After multiple repetitions, this parameters gave the minimum set of k-fold values. Hence, this parameter is selected as the best parameter.

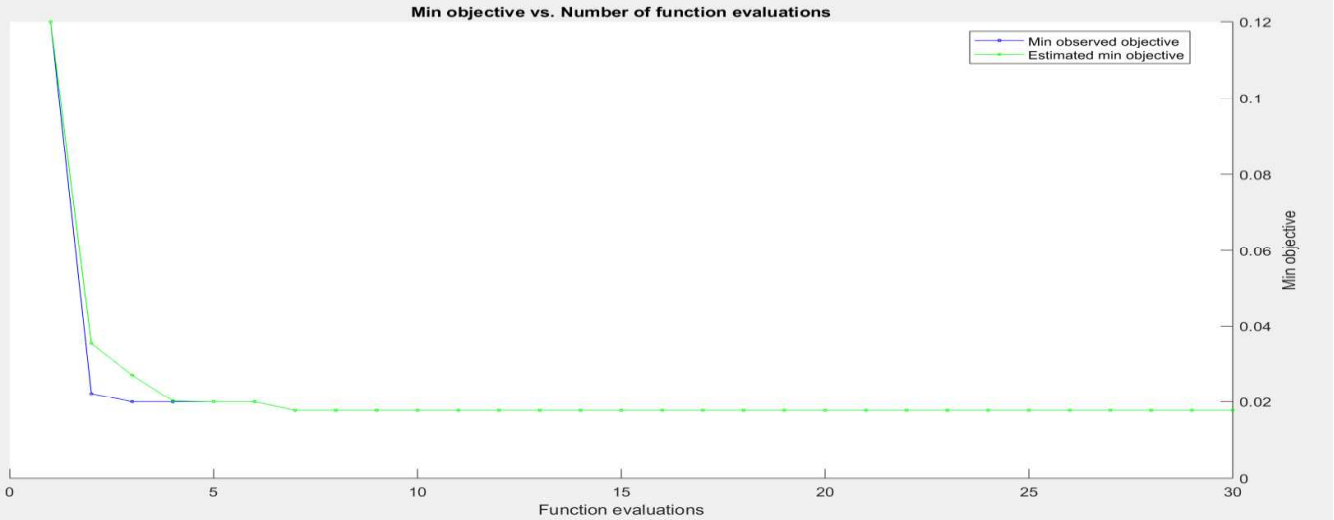
Matlab code for calculating the k-fold Loss of individual and average runs:

```
Command Window
>> Lc_ind=kfoldLoss(CVTreec,'mode','individual')
Lc_avg=kfoldLoss(CVTreec)

Lc_ind =

    0.0222
         0
    0.0222
    0.0111
    0.0222


Lc_avg =

    0.0156
```

Plot of Objective function model for all the parameters of minLeafsize and Min Objective vs Number of function evaluations:

By looking at the Objective function plot the Objective function is minimum at minLeafsize=1,which will give the maximum accuracy possible.



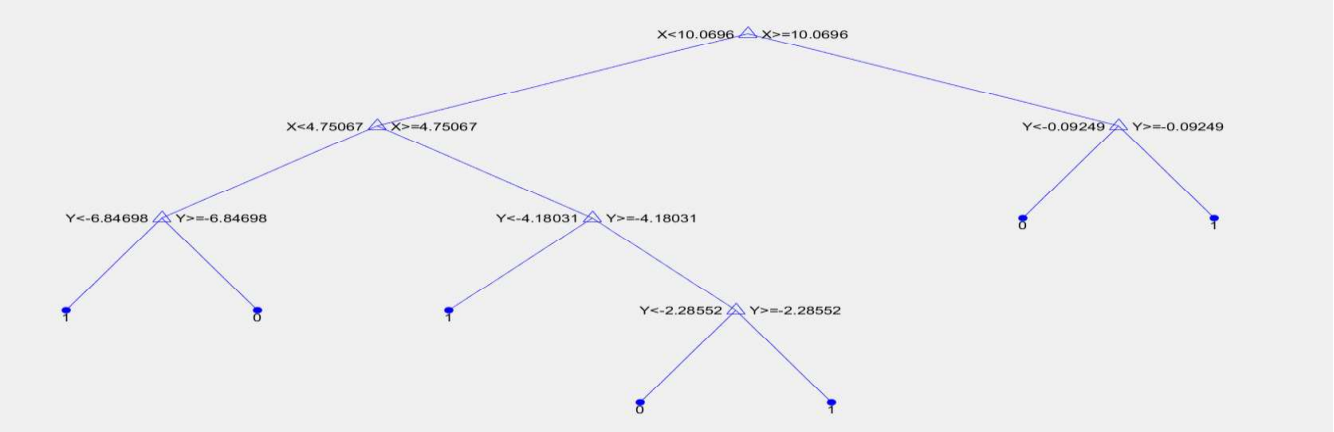


Model (or) Tree Formed using Optimization technique:

Matlab command for viewing the Tree

```
Command Window
fx >> view(Treec,'mode','graph')
```

Decision Tree:



Testing the Dataset with 450 Points and Computation of Accuracy, Precision, Recall Values:

Matlab commands for testing the Data and computation of accuracy, precision and recall values:

```
Command Window
>> predictionc=predict(Treec,DataTable)
C=confusionmat(DataTable.class,predictionc)
Accuracyc0=(C(1,1)+C(2,2))/(C(1,1)+C(1,2)+C(2,1)+C(2,2))
Precision0=(C(1,1))/(C(1,1)+C(2,1))
Recall0=(C(1,1))/(C(1,1)+C(1,2))
```

Results Obtained:

```
Command Window
C =

   250     0
     2   198


Accuracyc0 =

    0.9956


Precision0 =

    0.9921


Recall0 =

     1
```

Comments on Accuracy, Precision and Recall Values:

From the results, we see that the Accuracy obtained by optimizing the data is 0.9956 (or) 99.56% which is the maximum accuracy seen after tuning the data for 5 sets of training parameters .From the confusion matrix we can view that 2 values from the data are only misclassified as False 0's ,hence the precision is 99.21%.

The False Negative for the data is 0 and hence the recall for class '0' is 100%.This Model indicates the best optimized model for the given data, since the parameters accuracy, precision and recall are more than 99%.

Matlab Code for computation of Precision and recall for class '1':

```
Command Window
>> Precision1=(C(2,2))/(C(2,2)+C(1,2))
Recall1=(C(2,2))/(C(2,2)+C(2,1))

Precision1 =

     1


Recall1 =

    0.9900
```

As seen the precision and recall values for class are 100% and 99% respectively after tuning the decision tree.

c. ***(5) Draw the lines on the 2-D plot of the data to show the boundaries that your decision tree learned.***

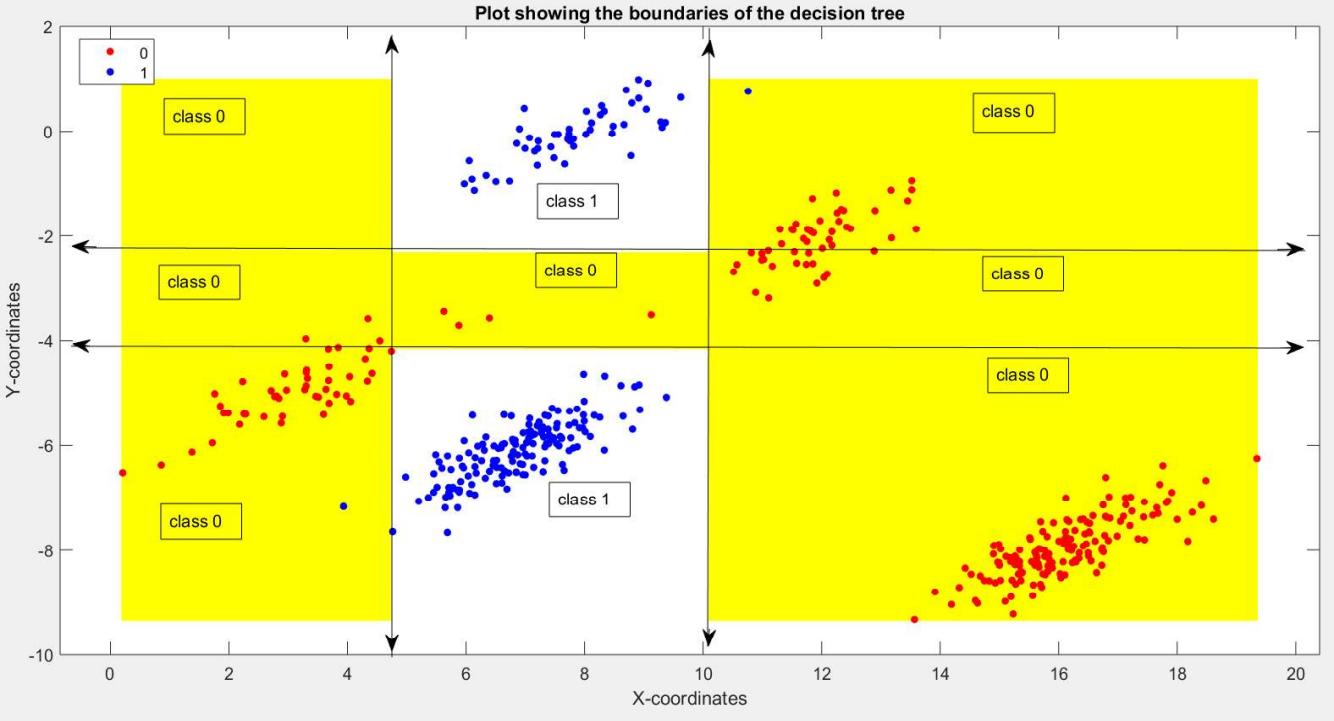
Solution for 1(c):

Matlab code for plotting the boundaries of the decision tree:

Command Window

```
>>
XRange=min(DataTable.X):.01:max(DataTable.X);
YRange=min(DataTable.Y):.01:max(DataTable.Y);
[xx1,xx2]=meshgrid(XRange,YRange);
XGrid=[xx1(:) xx2(:)];
predictgrid=predict(Treec,XGrid);
gscatter(xx1(:),xx2(:),predictgrid,'yw')
hold on
gscatter(DataTable.X,DataTable.Y,DataTable.class,'rb')
title('Plot showing the boundaries of the decision tree')
xlabel('X-coordinates')
ylabel('Y-coordinates')
```

For plotting the decision tree boundary, we first divide the x and y labels in the intervals of 0.01 and calculate a mesh grid using these ranges. Then we predict all these values using the decision tree formed .Then we plot all the xx1 and xx2 values of the mesh grid on a plot .We then draw a scatter with all the X and Y values of the data table to view all the correctly classified and misclassified examples.



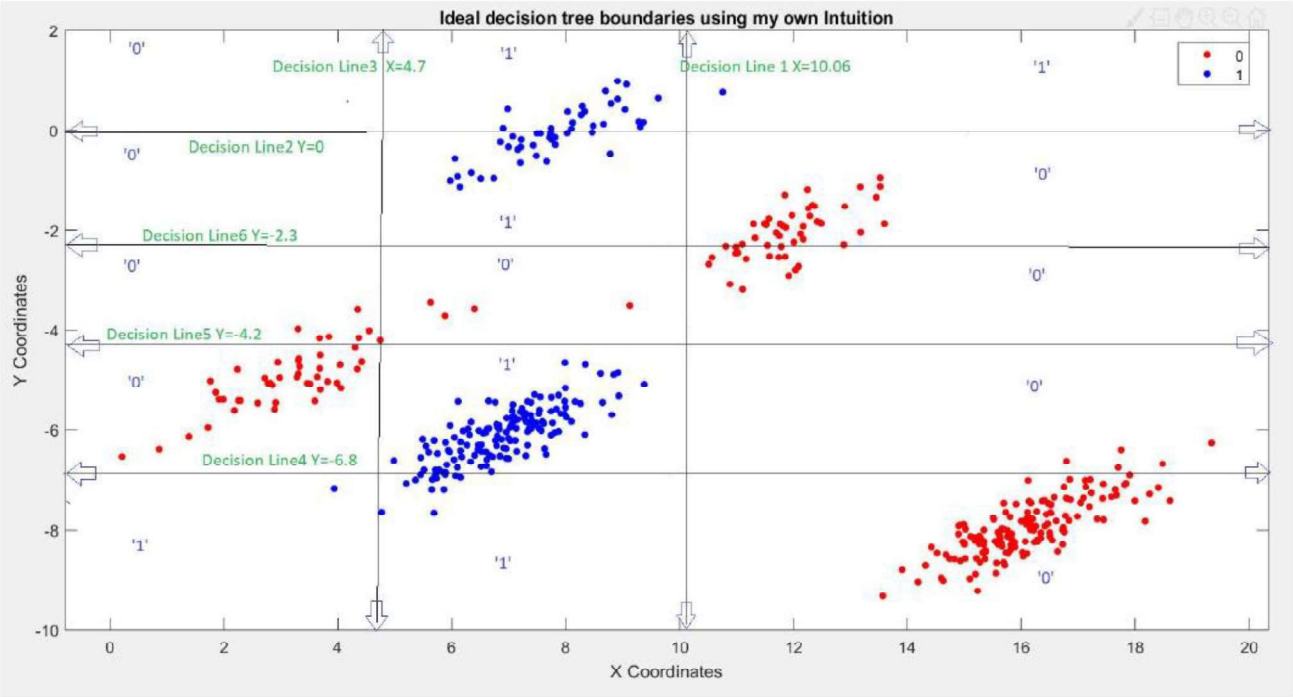
The plot indicates the boundaries of the decision tree with color ‘yellow’ showing the range of class ’0’ and color ‘white’ showing the range of class ‘1’.

From the plot, we can conclude that 2 points of class ‘1’ are misclassified which can be viewed from the confusion matrix. The 250 points of class ‘0’ and remaining 198 points of class ‘1’ are correctly classified by the optimized decision tree.

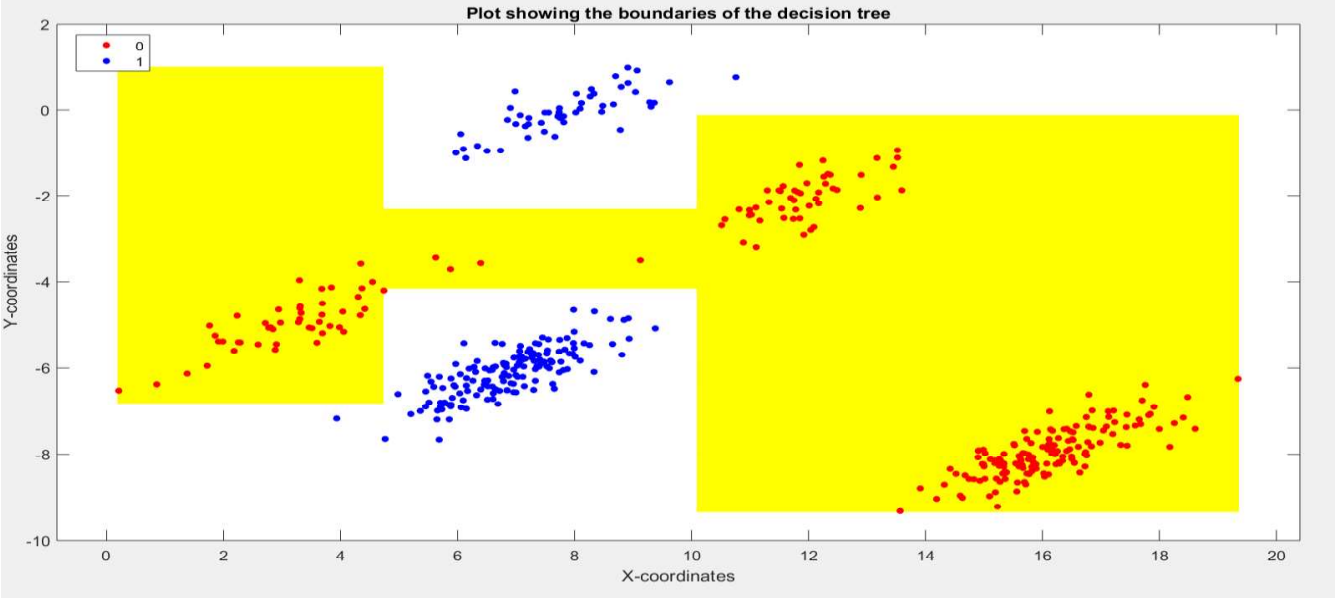
d. (10) Draw the best possible decision tree boundaries that you would draw if you were to use your intuition instead of the decision tree induction program. Show the accuracy, precision and recall values for this "ideal" decision tree.

Solution 1(d):

Best possible ideal decision tree using own intuition:



Plot showing the regions of the above plot or the ideal decision plot:



Yellow -Regions of class '0' White-Regions of class '1'

Computation of accuracy, precision, recall values for the ideal decision tree:

From the ideal (or) the decision boundary plotted through own intuition, we conclude the following.

Number of True ‘0’(T0) = 250

Number of False ‘0’(F0) = 0

Number of True ‘1’(T1) = 200

Number of False ‘1’(F1) = 0

Accuracy of dataset = T0+T1/(T0+F0+T1+F1) = 450/450 = 1

Precision of class ‘0’ = T0/T0+F0 =250/250 =1

Precision of class ‘1’ = T1/T1+F1 =200/200 =1

Recall of class ‘0’ = T0/T0+F1 =250/250 =1

Recall of class ‘1’ = T1/T1+F0 =200/200 =1

Comments on the Accuracy, Precision, Recall of the ideal decision tree :

By observing the plot, there are no misclassifications in the ideal decision tree. All the datapoints of class ‘0’ have been classified under the region ‘0’ and all the datapoints of class ‘1’ have been classified as the class ‘0’.Hence the accuracy, precision, recall for the ideal tree are all 100%,which has been showed through mathematical calculation as well.

---

e. ***(10) Repeat part (b) above for a linear SVM instead of the decision tree. Choose the parameter values that give you the best classification performance in terms of accuracy. State the chosen values of these parameters and give reasons in brief about why you think these values give the best performance. On a 2-D plot of the data show the support vectors learned by your SVM classifier. Using the support vectors draw an estimate of the boundary learned by your program.***

Solution of 1(e):

Choosing parameters for the SVM:

For choosing parameters for classification of the given dataset with SVM, we use ‘Box Constraint’ and ‘Kernel Scale’ as the data is not linearly separable. Let’s tune the data for the given dataset using ‘OptimizeHyperparameters’ using 5-fold validation technique.

Matlab Code for optimizing the given dataset and finding the best possible parameter values:

```
Command Window
>> svmc=fitcsvm(DataTable,'class','OptimizeHyperparameters','auto','HyperparameterOptimizationOptions',struct('kfold',5))
|=====================================================================================================|
| Iter | Eval   | Objective   | Objective    | BestSoFar    | BestSoFar    | BoxConstraint| KernelScale |
|      | result |             | runtime      | (observed)   | (estim.)     |              |             |
|=====================================================================================================|
|    1 | Best   |     0.27111 |     0.063453 |     0.27111 |     0.27111 |    0.0020175 |       3.755 |
|    2 | Accept |     0.44444 |     0.054377 |     0.27111 |     0.28526 |     0.093688 |       960.74 |
|    3 | Best   |     0.13111 |       2.1112 |     0.13111 |     0.13114 |       2.8158 |      0.22715 |
|    4 | Accept |     0.20222 |       17.359 |     0.13111 |     0.13114 |    0.0012168 |    0.0012776 |
|    5 | Accept |     0.13111 |      0.11384 |     0.13111 |     0.13106 |    0.0015143 |     0.071649 |
|    6 | Accept |     0.54444 |       18.307 |     0.13111 |     0.13113 |       58.436 |       0.1301 |
|    7 | Accept |     0.44444 |     0.051771 |     0.13111 |     0.13114 |       2.2235 |       872.57 |
|    8 | Accept |     0.13778 |     0.068815 |     0.13111 |     0.13113 |    0.0010014 |       0.3476 |
|    9 | Accept |     0.13333 |     0.080458 |     0.13111 |     0.13115 |    0.0010051 |     0.079112 |
|   10 | Accept |     0.16667 |     0.050702 |     0.13111 |     0.13114 |       959.22 |       979.22 |
|   11 | Accept |     0.17778 |     0.077913 |     0.13111 |     0.13114 |       358.32 |       997.73 |
|   12 | Accept |     0.44444 |     0.049728 |     0.13111 |     0.13114 |    0.0010029 |       918.57 |
|   13 | Accept |     0.13111 |     0.066875 |     0.13111 |     0.13111 |    0.0089496 |      0.15673 |
|   14 | Accept |     0.13111 |      0.14818 |     0.13111 |     0.13109 |      0.12869 |      0.25054 |
|   15 | Accept |     0.13111 |     0.066937 |     0.13111 |     0.13109 |      0.53691 |       1.4486 |
|   16 | Accept |     0.13111 |      0.28943 |     0.13111 |     0.13102 |      0.61886 |      0.33745 |
|   17 | Accept |     0.13333 |     0.053559 |     0.13111 |     0.13103 |     0.060469 |       1.4983 |
|   18 | Accept |     0.13111 |     0.060089 |     0.13111 |     0.13101 |     0.035233 |      0.40101 |
|   19 | Accept |     0.13111 |     0.066641 |     0.13111 |     0.13103 |       2.1878 |       2.8112 |
|   20 | Accept |     0.13333 |      0.06037 |     0.13111 |     0.13103 |       999.12 |         88.2 |
|=====================================================================================================|
```

```
Best observed feasible point:
    BoxConstraint    KernelScale
    _____________    ___________

       2.8158          0.22715

Observed objective function value = 0.13111
Estimated objective function value = 0.13096
Function evaluation time = 2.1112
```

Reasons for choosing the parameters and the Values:

Since this data is not linearly separable, the parameters Box Constraint and Kernel Scale parameters are suited for the given dataset . When the data is not perfectly separable, the training algorithm must allow some mis-classification in the training set. In this case it is applying a cost to the misclassification. The higher the box-constraint the higher the cost of the misclassified points, leading to a stricter separation of the data. Hence, we choose a Box constraint value of 2.81 as obtained in the optimized Function.

The Kernel scale parameter is called “gamma” Factor. If gamma is large, then this kernel will fall off rapidly as the point y moves away from x. As gamma decreases, the kernel will fall off less and less rapidly. Hence, we choose kernel scale factor to be 0.22715 after optimization.

Box Constraint C=2.81

Kernel scale (Gamma) = 0.227

Matlab Command for calculating 5-fold losses for the dataset:

```
Command Window
>> CVsvmc=crossval(svmc,'kfold',5)
Lc_ind=kfoldLoss(CVsvmc,'mode','individual')
Lc_avg=kfoldLoss(CVsvmc)
```

The Results obtained are:

```
Command Window
Lc_ind =

    0.1222
    0.1667
    0.1000
    0.1333
    0.1111


Lc_avg =

    0.1267
```

From the k-fold Losses the average loss for the support vector machine classifier is 0.1267,which is the optimal classifier.

Computation of Accuracy ,Precision and Recall Values of Model trained using SVM:

Matlab Code for computation of the above parameters:

```
Command Window
>> predictionsvmc=predict(svmc,DataTable)
Cv=confusionmat(DataTable.class,predictionsvmc)
Accuracycv=(Cv(1,1)+Cv(2,2))/(Cv(1,1)+Cv(1,2)+Cv(2,1)+Cv(2,2))
Precisioncv=(Cv(1,1))/(Cv(1,1)+Cv(2,1))
Recallcv=(Cv(1,1))/(Cv(1,1)+Cv(1,2))
```

Results Obtained:

```
Cv =

   195    55
     0   200


Accuracycv =

    0.8778


Precisioncv =

     1


Recallcv =

    0.7800
```

Comments on the Accuracy, Precision and Recall Values using SVM Classifier:

Using an optimal Linear SVM for this Dataset gives an Accuracy of 87.78 %,since the data is not linearly separable. Only once class i.e. class ‘1’ is separable and hence the Precision is very high which is 100%.But the Recall is Low due to misclassifications of another class ‘1’.

Matlab Code and Results for computation of Precision and recall of class ‘1’:

```
Command Window
>> Precisioncv1=(Cv(2,2))/(Cv(2,2)+Cv(1,2))
Recallcv1=(Cv(2,2))/(Cv(2,2)+Cv(2,1))

Precisioncv1 =

    0.7843


Recallcv1 =

     1
```

The Precision for class ‘1’ is very less ,since there are several false ‘1’s i.e. data of class ‘0’ Predicted as class ‘1’.

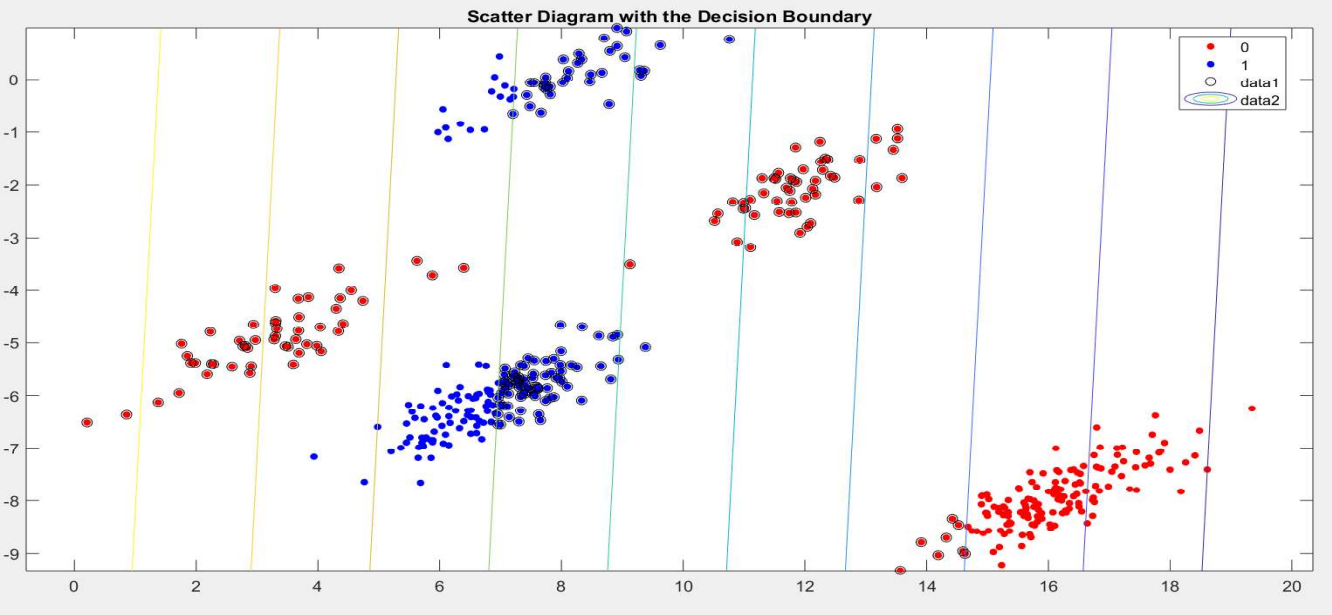
But the Recall is very high which is 100% since all the class ‘1’ data is perfectly classified into class ‘1’.

2-D Plot showing the support vectors learned by the SVM:

Matlab Code for showing support vectors learned by SVM:

```
Command Window
>> h(1:2)=gscatter(DataTable.X,DataTable.Y,DataTable.class,'rb');
hold on
h(3)=plot(X(svData,1),Y(svData,1),'ko');
contour(xx1,xx2,reshape(scores1(:,2),size(xx1,1),size(xx2,2)));
title('Scatter Diagram with the Decision Boundary')
hold off
```

Plot showing the support vectors obtained by the SVM:



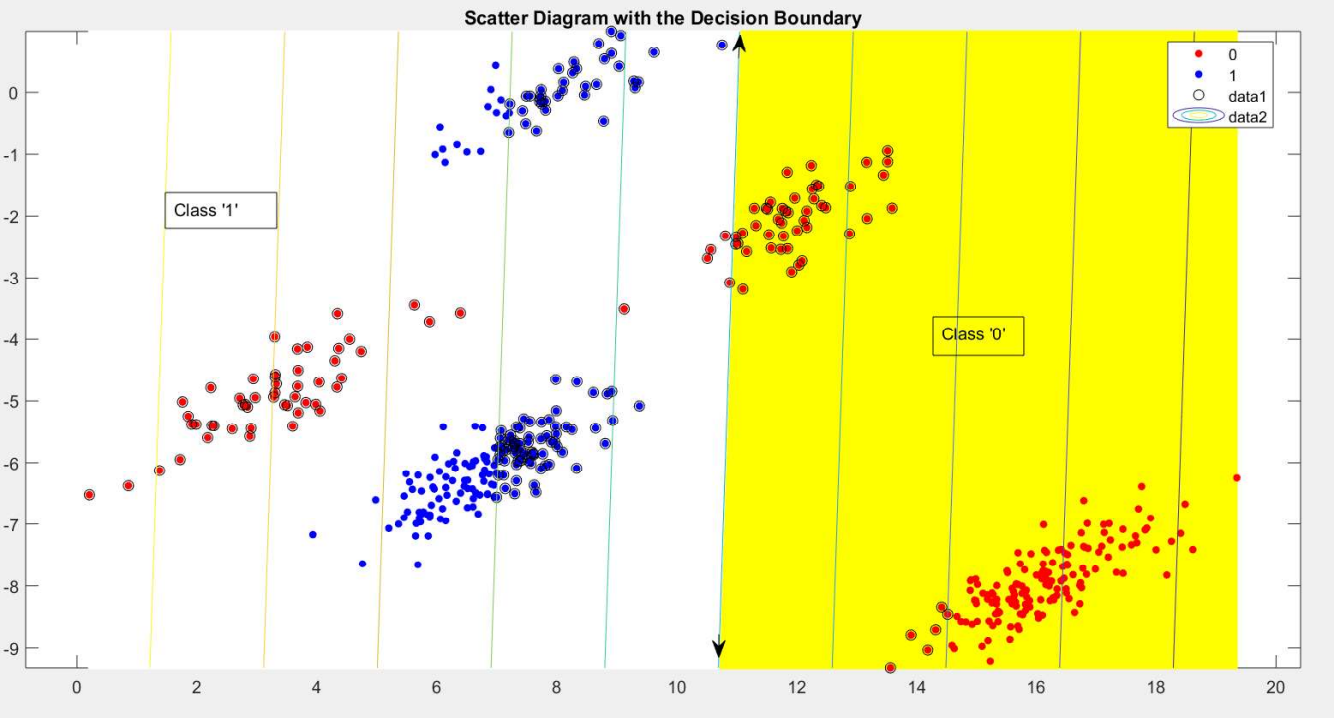
From the plot, the datapoints t are represented by circles, indicate the support vectors .These points are the ones which maximize the margin for the SVM classifier.

Estimate of the Boundary Learned by the Linear SVM:

Matlab Code for plotting the estimate of the boundary for SVM classifier:

```
Command Window
>>
svData=svmc.IsSupportVector;
predictsvmgrid=predict(svmc,XGrid);

gscatter(xx1(:),xx2(:),predictsvmgrid,'yw')
hold on
h(1:2)=gscatter(DataTable.X,DataTable.Y,DataTable.class,'rb');
hold on
h(3)=plot(X(svData,1),Y(svData,1),'ko');
contour(xx1,xx2,reshape(scores1(:,2),size(xx1,1),size(xx2,2)));
title('Scatter Diagram with the Decision Boundary')
hold off
```

Decision Boundary learned by SVM separating the two regions:



The decision boundary separating the two regions, the white region indicating class ‘1’ and the yellow region indicating class ‘0’ ,is the boundary which has been obtained for the optimal linear SVM classifier.

---

f. (10) Repeat part (e) above using an RBF kernel to learn a non-linear SVM.

Solution for 1(f):

Matlab Code for fitting a non-linear SVM using RBF kernel using 5fold validation:

Command Window

```
>> Treesvkf=fitcsvm(DataTable,'class','KernelFunction','rbf','kfold',5)
Lsvkf_ind=kfoldLoss(Treesvkf,'mode','individual')
Lsvkf_avg=kfoldLoss(Treesvkf)
```

Results for the k fold loss and optimization:

By tuning the non-linear SVM classifier we obtain a k fold loss of 0 as shown below.

Command Window

```
Lsvkf_ind =

     0
     0
     0
     0
     0


Lsvkf_avg =

     0
```

Computing Accuracy, precision and recall for the dataset with the radial basis kernel function:

Matlab Code and results of computation of accuracy, precision, recall parameters :

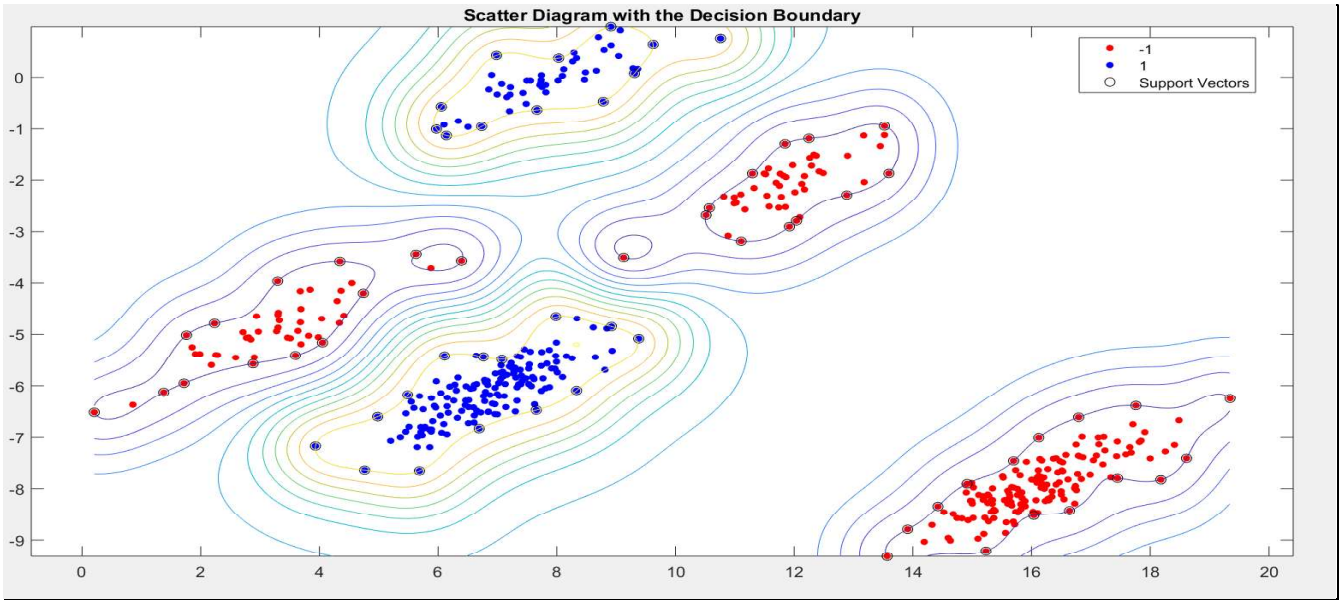
```
Command Window
>> predictionsvkf1=kfoldPredict(Treesvkf);
DataTableclass=DataTable.class;
Ck=confusionmat(DataTableclass,predictionsvkf1)
Accuracycsvkf1=(Ck(1,1)+Ck(2,2))/(Ck(1,1)+Ck(1,2)+Ck(2,1)+Ck(2,2))
Precisionsvkf1=(Ck(1,1))/(Ck(1,1)+Ck(2,1))
Recallsvkf1=(Ck(1,1))/(Ck(1,1)+Ck(1,2))

Ck =

   250     0
     0   200


Accuracycsvkf1 =

     1


Precisionsvkf1 =

     1


Recallsvkf1 =

     1
```

The Accuracy, precision and recall for the dataset with RBF kernel is 100%. Therefore, RBF kernel is the best classifier. Let's visualize this classification on a plot.

Matlab code for plotting an RBF classifier:

```
Command Window
>>
  [~,scoreskf]=predict(Treesvkf,XGrid);
  h(1:2)=gscatter(DataTable.X,DataTable.Y,DataTable.class,'rb');
  hold on
  h(3)=plot(X(svkfData,1),Y(svkfData,1),'ko');
  contour(xx1,xx2,reshape(scoreskf(:,2),size(xx1,1),size(xx2,2)));
  title('Scatter Diagram with the Decision Boundary')
  legend({'-1','1','Support Vectors'},'Location','Best');
  hold off
```

Plot of the Non-Linear RBF kernel classifier for the given dataset:



The Plot shows the RBF classifier for the given dataset. The support vectors encircled with yellow boundaries are transformed to one radial basis function (1) and the support vectors encircled with blue boundaries are transformed to another radial basis function (0).Hence all the data points are correctly classified . Hence all the parameters accuracy, precision and recall for the RBF kernel is the best.

Kernel uses a radial basis function to classify the data .It transforms the data points into a radial basis function and then classifies the new data points .By Kernel function the dot product between the vectors would be same as the dot product of transformed vectors. Hence it is possible to obtain a 100% accuracy with Kernel as the parameter.

---

g. (10) Compare and contrast the performance and decision boundaries arrived at in parts (c), (d), (e) and (f) above.

Solution of 1(g):

Comparison of performance of decision tree, ideal tree(self-intuition) ,Linear SVM and RBF kernel SVM:

The performance of the **RBF kernel** is the best since it uses a radial basis function to classify the data .Since the datapoints in our data are like bubbles ,this method would be the best suitable for test examples as well. Since the value of the RBF kernel decreases with distance and ranges between zero (in the limit) and one (when x = x'), it has a ready interpretation as a similarity measure. The feature space of the kernel has an infinite number of dimensions.

The next best performer in the context of accuracy is **the ideal or self-intuited decision tree**. This tree has the best accuracy because it is drawn by knowing the training examples. Hence ,this tree might be overfitting when performed on the test set. Hence , we can conclude that though the self-intuited tree has an accuracy of 100%,it is not the second-best possible tree if performed on test examples and might be overfitting, since it is not pruned.

The best linear classifier is the **decision tree** obtained which has got an accuracy of 99.56 .This tree has performed well on the training examples. But it may overfit on test data if it noisy as it is not pruned .But decision tree would be the best Linear classifier for this data.

Linear SVM hasn't performed well on this dataset, as we can observe that the dataset is not linearly separable.

The boundary obtained by linear SVM is also not optimal(as seen) since it has misclassified 55 examples on the Training data and may perform worse on the test data.

Comparison of decision boundaries of decision tree, ideal tree(self-intuition) ,Linear SVM and RBF kernel SVM:

The decision boundaries obtained for the **decision tree** are linear and has classified the space into 9 regions with each region indicating class '1' and class '0' .This classification followed a method of growing a tree by writing conditions on each attribute X and Y.

The decision boundaries obtained for the **RBF kernel** is radial and looks like bubbles. This model best suits the datapoints since the classifier is non-linear when compared to linear classifiers.

The decision boundaries obtained for **Linear SVM** is a line separating the plane into classes '0' and '1' .This decision boundary is obtained by optimizing the best model by tuning Box constraint C and Gamma Function.

The decision boundary obtained for **self-intuited decision tree** are linear boundaries with each region separated into a class '0' or '1'

2. ***(20) Consider a dataset containing six 4-D points followed by their class labels, given as follows: (3 4 2 5; C1), (5 9 3 10; C0), (1 8 11 6; C1), (6 1 6 9: C0), (12, -2 1 8; C0), (5 6 0 2; C1). Show two full epochs of the perceptron training algorithm with these data points. Use (3 4 5 6 7) as the initial weight vector.***

Solution of 2:

Two full epochs of perceptron training are shown below:

PERCEPTRON TRAINING

EPOCH 1

x1	x2	x3	x4	x5	w1	w2	w3	w4	w5	w.x	Error	w-x/w-x				
3	4	2	5	1	3	4	5	6	7	72	N	3	4	5	6	7
5	9	3	10	1	3	4	5	6	7	133	Y	-2	-5	2	-4	6
1	8	11	6	1	-2	-5	2	-4	6	-38	Y	-1	3	13	2	7
6	1	6	9	1	-1	3	13	2	7	100	Y	-7	2	7	-7	6
12	-2	1	8	1	-7	2	7	-7	6	-131	N	-7	2	7	-7	6
5	6	0	2	1	-7	2	7	-7	6	-31	Y	-2	8	7	-5	7

EPOCH 2

3	4	2	5	1	-2	8	7	-5	7	22	N	-2	8	7	-5	7
5	9	3	10	1	-2	8	7	-5	7	40	Y	-7	-1	4	-15	6
1	8	11	6	1	-7	-1	4	-15	6	-55	Y	-6	7	15	-9	7
6	1	6	9	1	-6	7	15	-9	7	-13	N	-6	7	15	-9	7
12	-2	1	8	1	-6	7	15	-9	7	-136	N	-6	7	15	-9	7
5	6	0	2	1	-6	7	15	-9	7	1	N	-6	7	15	-9	7

Therefore, the final weight vector observed is -6,7,15,-9,7

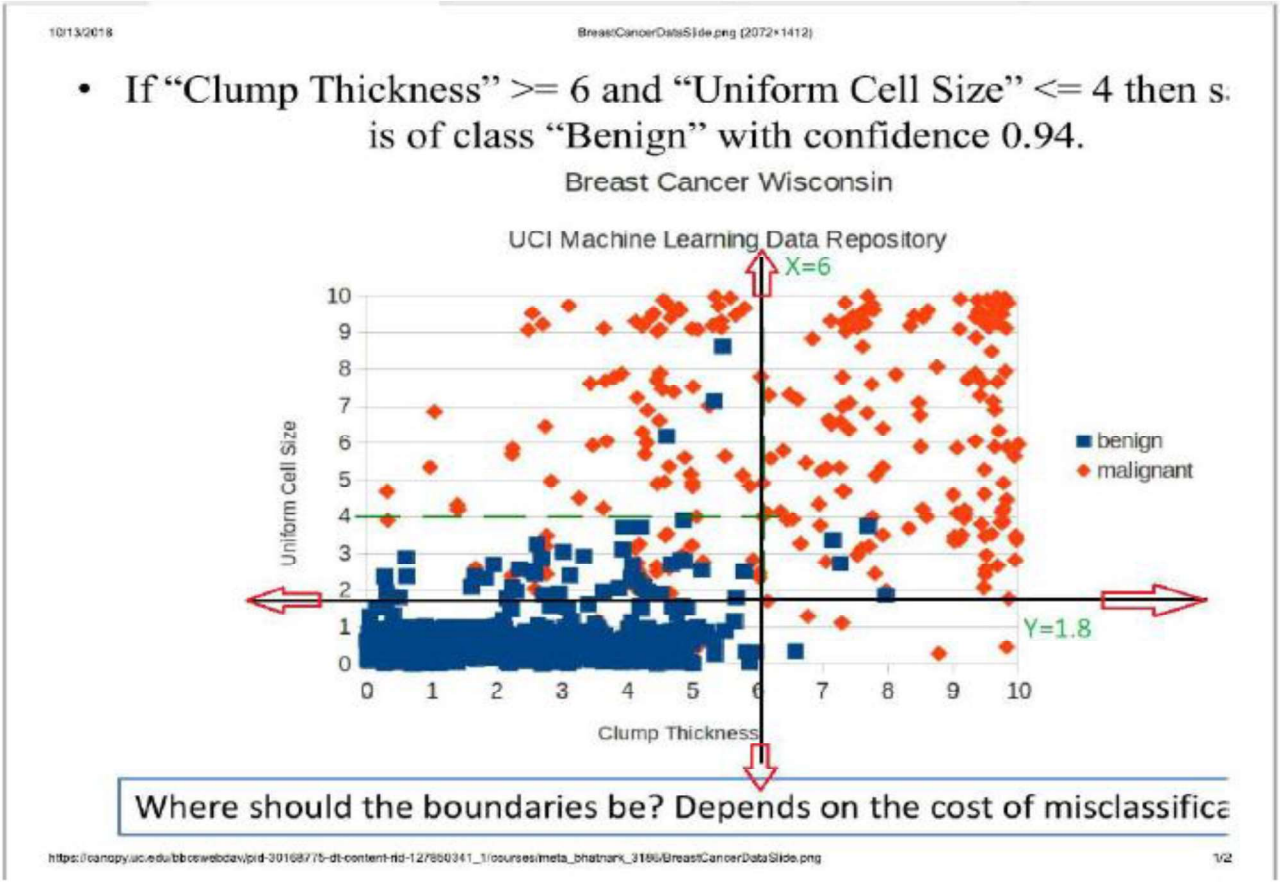
Explanation:

In the perceptron training, we append an extra column to all the examples with value 1,then we find the dot product of weight vector and each example. Then the Error on the Example is verified and if there is no error on the training example, we substitute wi+1 as wi, if there is an Error on the Example then we perform w+x or w-x based on the Error is '-ve' or '+ve' respectively. We continue training the Example till we reach all No Errors for the entire dataset. The Final weight vector is the weight vector of the dataset, trained using perceptron model.

3. ***Consider the attached image for a real Breast-Cancer dataset. It shows a decision tree fitted to the data to separate malignant cases from the benign cases. The decision tree rule for classifying the benign cases is included in the image. This decision tree minimizes the number of misclassifications (maximum accuracy). Answer the following questions in the context of this dataset/image.***
    a. ***(8) The cost of announcing a malignant case as benign is fifty times the cost of announcing a benign case as malignant. How would you adjust the boundaries of this decision tree? Show by drawing the new lines on data plot image and writing the new rule for classifying a case as benign. Explain briefly why you chose the new boundaries.***
    b. ***(8) Show how you will change the boundaries of the original decision tree in the image to increase the precision of the benign class. Draw the new boundary lines and briefly explain why this causes the desired effect. How does this change affect the precision of the malignant class?***

Solution of 3(a):

Data Plot Image by adjusting the boundaries of the decision Tree:



New rule for classifying a case as benign:

If clump Thickness<=6 and Uniform Cell Size<=1.8,then s is of class benign

The New Rule for classifying a case as benign is as shown in the figure which can be classified using the Decision Lines given below:

X=6 ,Y=1.8

Explanation for selecting the above Lines:

It is given that the cost of announcing a malignant case as benign is 50 times the cost of announcing the benign case as malignant. Therefore, no malignant case should be misclassified as benign to reduce the cost and maximize the accuracy for the tree. On the other hand, by redrawing the line from Y=4 to Y=1.8, we are misclassifying the benign cases as malignant, which will have a very little effect on the cost. But the overall cost by selecting this decision Tree would be the optimal. We have not moved the line X=6 since there are no misclassifications of malignant case as benign in the X direction. Hence the line X=6 is kept the same.

Therefore, the optimal rule of classifying benign and malignant class is possible by selecting clump thickness<=6 and Uniform cell size<=1.8

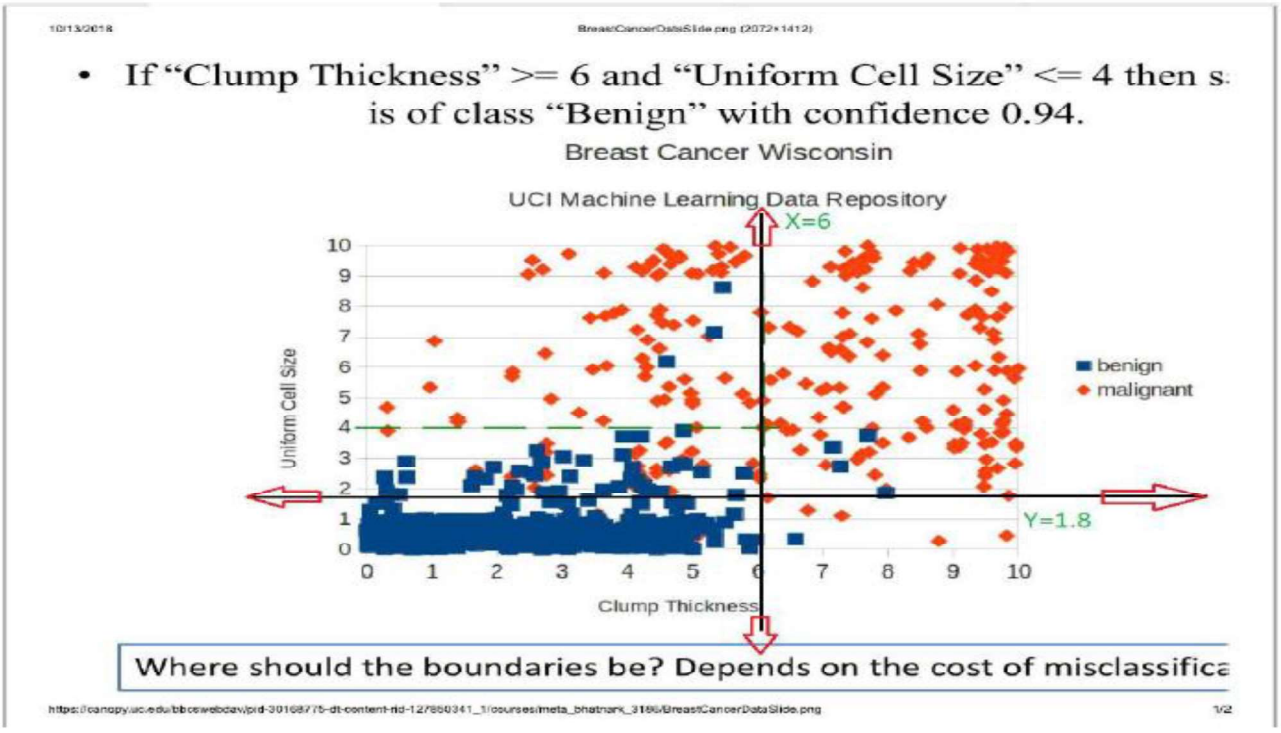
---

(8) b. Show how you will change the boundaries of the original decision tree in the image to increase the precision of the benign class. Draw the new boundary lines and briefly explain why this causes the desired effect. How does this change affect the precision of the malignant class?

Solution of 3(b):

The precision of benign class =True benign/(True benign+ False benign)

To increase the precision for the benign class, we must reduce the false benign and improve the true benign classifications .For increasing the precision we follow the same method that we have followed in the previous example. We reduce the false benign by moving the decision boundary to points where there are no false benign points or malignant points in the benign class. Let's visualize the data slide now.



New Boundaries : clump Thickness<=6 and Uniform Cell Size<=1.8
Precision Obtained for benign Class : 99% -100%

Explanation for the new decision boundaries to increase the precision:

By looking at the new benign and malignant class ,we can say that the precision of the benign class is very high(more than 99%).This has been achieved by reducing the number of false benign points inside the benign class. Therefore, by drawing the above decision boundaries, we can make all the benign class points to correctly classify. We know that in this case we have misclassified a few benign examples as malignant, but this will not impact the precision of the malignant class.

Affect on the precision of the malignant class of this decision boundary:

Precision Obtained for malignant Class would subsequently reduce as the Precision for benign class is increased

By looking at the new boundary ,we can say that the False malignant points have increased now. This will have an impact on the precision of the malignant class.

Precision of malignant class=True malignant/(True malignant +False malignant)

As the False malignant have increased ,the precision of the malignant class will subsequently reduce .Though the True malignant points have increased by small amount ,this term is present in both numerator and denominator and hence it will not a significant effect.

In the perspective of definition of precision, precision is the fraction of relevant examples among the retrieved examples. Therefore, in this case we are increasing count of relevant benign data points and reducing the irrelevant datapoints. Thereby, the precision becomes closer to 100%.