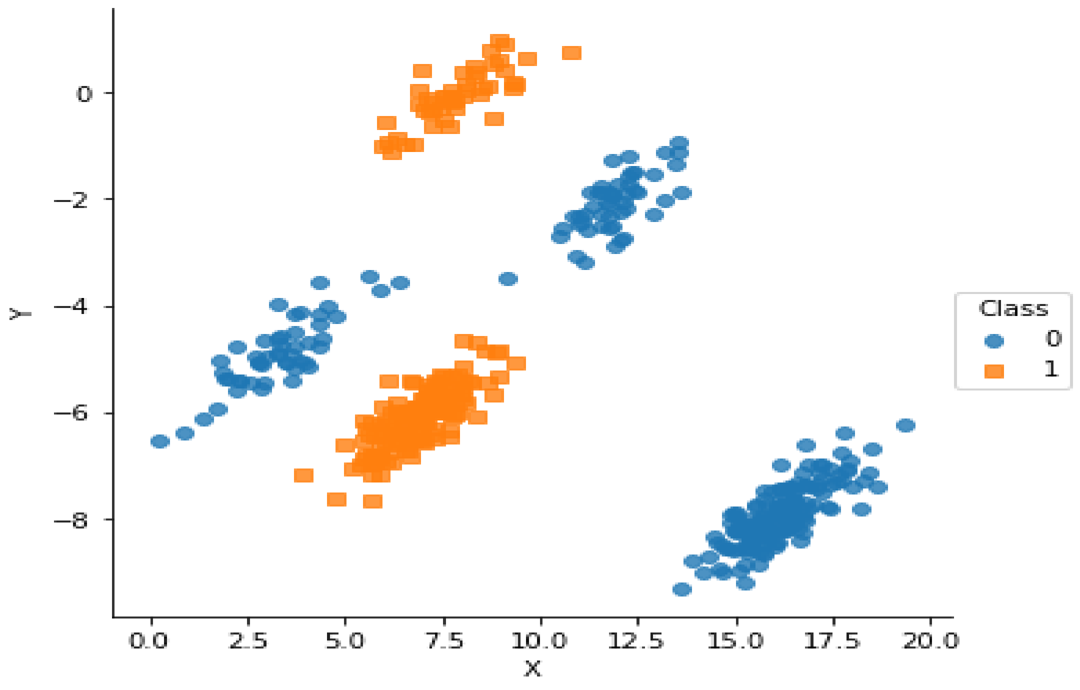
IDA Assignment-02
Tool Used: Python

Yandrapally, Aruna Harini (yandraai)

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



b) Use 5-fold testing to find the 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.

Steps Taken for all the 3 models (Decision tree, Linear SVM, RBF SVM):

- The k-fold cross validation is a procedure used to estimate the skill of the model on new data. Here we are using k=5 in which each iteration 4 sets would be used as training sets and one would be acting like test set.
- Steps done in each iteration:
  - Trained the model (Decision tree, Linear SVM, RBF SVM) on 4 sets of training data.
  - Tested on the left out testing data.
  - Calculated the accuracy, Precision and recall
- Found out the best model (Decision tree, Linear SVM, RBF SVM) with higher Accuracy, Precision and Recall values.
- Tested the best model on 450 data points and calculated Confusion matrix, Accuracy, precision and recall for each class

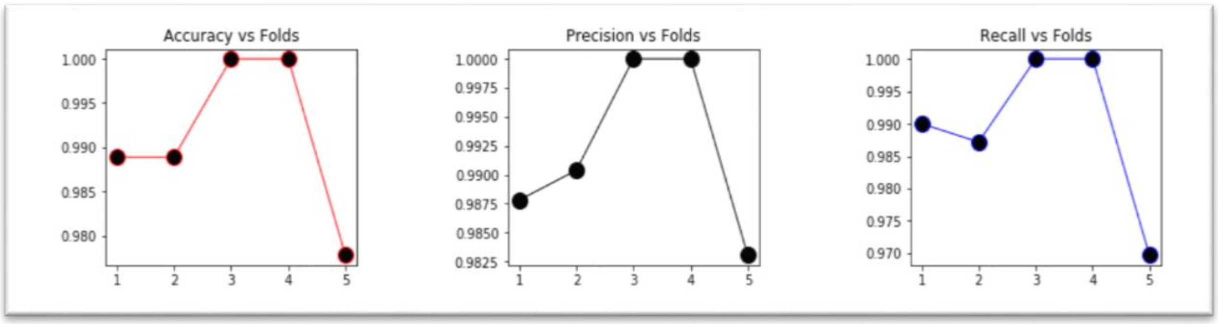
- Calculated the average accuracy, precision and recall of the model suggesting the performance model (Decision tree, Linear SVM, RBF SVM) on this dataset.

Parameters Considered:

- Validation of a Model is completely dependent on how well it is performing on test data set.
- As the best tree that is selected would be tested upon the entire 450 data points, one thing clear is that the data is similar to the data used on building the data. Therefore, choosing the tree that gives 100% accuracy, precision and recall on training and validation data sets during k-fold.
- As the data is small, the depth control parameters like **max_depth, min_samples_leaf,max_leaf_nodes** don't really make much of a difference.
- However, we have tuned the parameters to get the best possible accuracy as **Criteria = Entropy, Random_state =200, Min_samples_leaf = 1**(normally this is overfitting condition, but as our test set is the same data set we adopted this). Also I have used **Shuffle = true.**

Observations:

- Accuracy, Precision and recall for the five trees look like below:



- Hence 3rd tree is chosen (4th also gives the same) as the best tree for the next steps.
- Tested the best tree on the 450 data points and the following are obtained:
    - **Confusion Matrix**: `Predicted vs Actual`

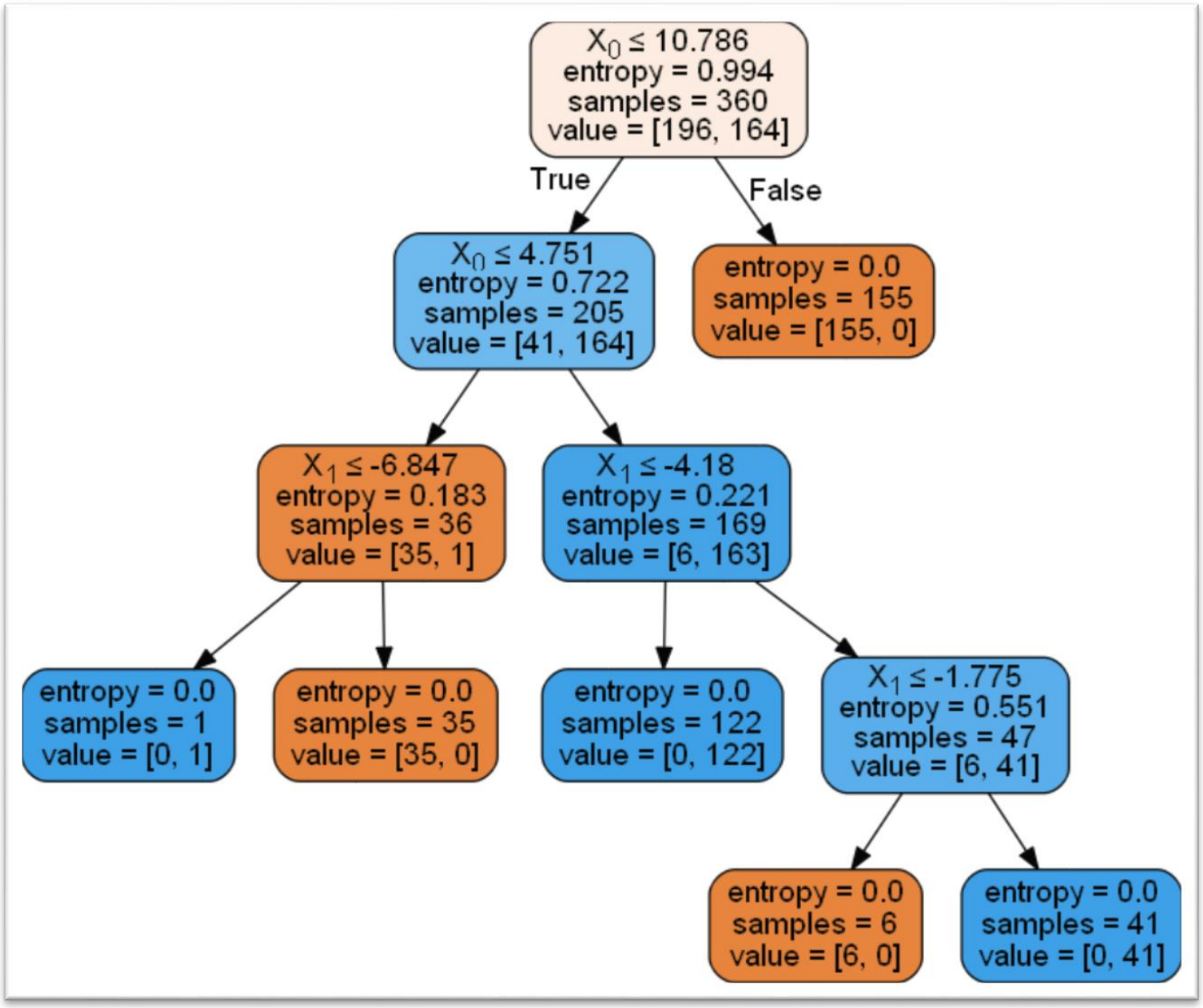
250	0
0	200

Making :: TP = 250 , TN = 200, FP =0, FN =0

- **Precision**: Precision is the fraction of retrieved documents that are relevant to the query. As it is the true positives among the true positives plus false positives, the precision for this hypothesis is 1 or 100% for both the classes.
- **Recall**: Recall is the fraction of the relevant documents that are successfully retrieved. As this is fraction between true positives and true positives plus false negatives, even this measure equals to 1 or 100% for both the classes.

- **Accuracy** : The number of correct predictions made divided by the total number of predictions made. As the correct predictions made is equal to the total number of predictions made, the accuracy is 1 or 100%

Viewing Best Tree:

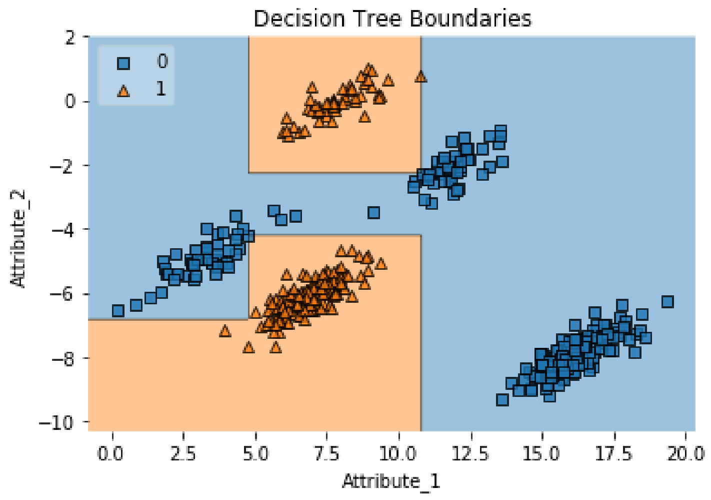


Average performance of Decision tree model on the dataset:

- K fold is used to find the estimate how well a model can fit the data. This is done analyzing or summarizing the metrics globally.
- **Average Accuracy**: Averaged over all the folds: **0.991 or 99.1%**
- **Average Precision**: Averaged over all the folds (both classes combined): **0.991 or 99.1%**
- **Average Precision**: Averaged over all the folds (both classes combined): **0.99 or 99.0%**

- The above summarized metrics suggest that decision tree would be a good model to be considered for the data set.

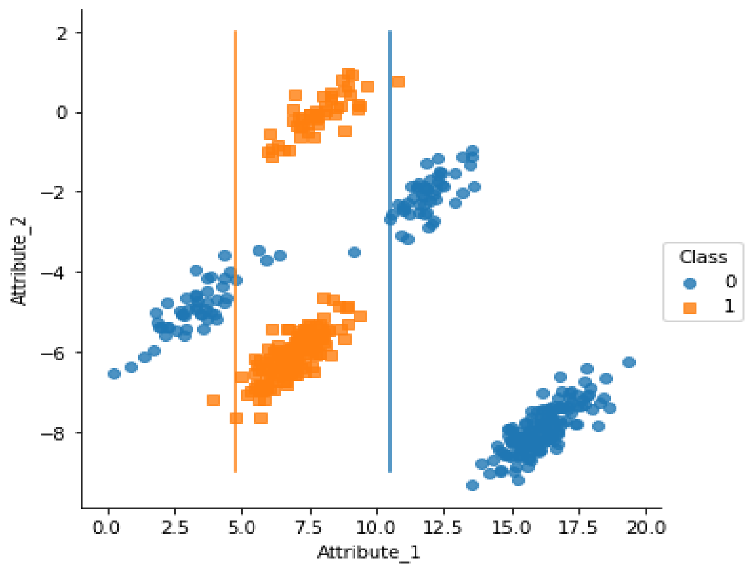
c) Draw the lines on the 2-D plot of the data to show the boundaries that your decision tree learned.



d) 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.

Boundaries of the ideal tree, I perceive:

- **Boundary_1: X(Attribute_1) = 10.6:** One Pure region(R1) on right side is formed with one misclassification data point for class 0.
- **Boundary_2: X(Attribute_1) = 4.751:** One pure region(R2) on its left side with one misclassification data for class 0. And pure region(R3) on the right with few misclassifications for class1.



Reasons for choosing the above boundaries:

- The hypothesis generated by the algorithm in the above step, gives out 100% accuracy because it satisfies the basic assumption of any modelling algorithm, that the training examples are representative to the new unseen examples. Therefore, the likelihood of overfitting was neglected.
- However, if we intuitively look into the data, there can be chances of overfitting if we use the above boundaries due to the following:
    - **Overfitting due to Noise**: For instance, the data point (3.9338,-7.1712) belonging to class 1 seems to be farther away from other data points belonging to class1. A new boundary has to be added to fit that point into the model which might be the case of overfitting if that data point is just some noise.
    - **Overfitting due to inadequate representative data**: The 6 data points out of 250 are farther away from their corresponding clusters pose a problem as, they can't be taken into new region without further enough data defining that region. They can be really belonging to class0 or they might also be just noisy entries. Only additional amount of training examples can give a further insight. Hence adding a boundary to just classify them would be overfitting on the training data.
- Therefore, I have avoided the above two boundaries in my ideal boundaries.
- Metrics for the ideal decision tree obtained:
    - **Misclassification points**:

Attribute_1	Attribute_2	Class
10.758778649	0.76045117	1
3.933819724	-7.171290403	1
6.394765411	-3.570080345	0
5.627318655	-3.441671818	0

5.879181203	-3.708981724	0
9.126270049	-3.504998386	0

- **Confusion Matrix**: Class 1 – positive class

  Actual vs Predicted

198	2
4	246

- **Precision**:
  - **Class 1**: Based on the above table, 4 data points from class 0 are being classified as class 1 and 2 are misclassified.
    So TP = 198, FP = 4 :: 198/198+4 = 0.98
  - **Class 0**: 2 data points belonging to class 1 are being classified as class 0 and 4 are misclassified.
    So TN = 246, FN = 2 :: 246/246+2 = 0.99
- **Recall**:
  - **Class 1**: Based on the confusion matrix: TP/TP+FN
    = 198/198+2 = 198/200 = 0.99
  - **Class 0**: Based on the confusion matrix: TN/TN+FP
    = 246/246+4 = 246/250 = 0.984
- **Accuracy**: Based on the confusion matrix: TP+TN/TP+FP+TN+FN
  = (198+246)/(198+2+246+4)=444/450
  =0.986 or 98.6%

d) 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.

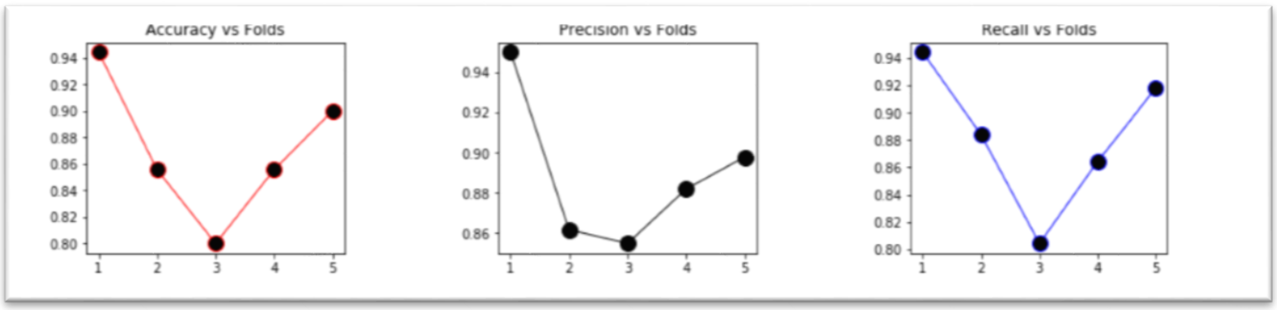
- Steps taken are the ones already mentioned under 1b section for k_fold testing.

Parameters Considered:

- Linear SVM is built taking the parameters: **Kernal = Linear, C=2.8**, **random_state=200 in SVC** and the **Shuffle = True in Kfold** function.
- Linear SVM only depend on the parameter C (Penalty parameter) in defining the boundary which trades correct classification of training examples against maximization of the decision function's margin.
- Though the default value is 1.0, the value of C is increased so that a smaller margin which is able to classify the training points better is chosen. In short, it is observed that as the C value increased from 0.0 to 2.8 the boundary shifted from extreme left to the point of better classification decreasing the margin width.

Observations:

- Accuracy, Precision and recall for the five SVMs look like below:



- Hence 1st model is chosen as the best model for the next steps.
- Tested the best model on the 450 data points and the following are obtained:
    - **Confusion Matrix**: `Predicted vs Actual`

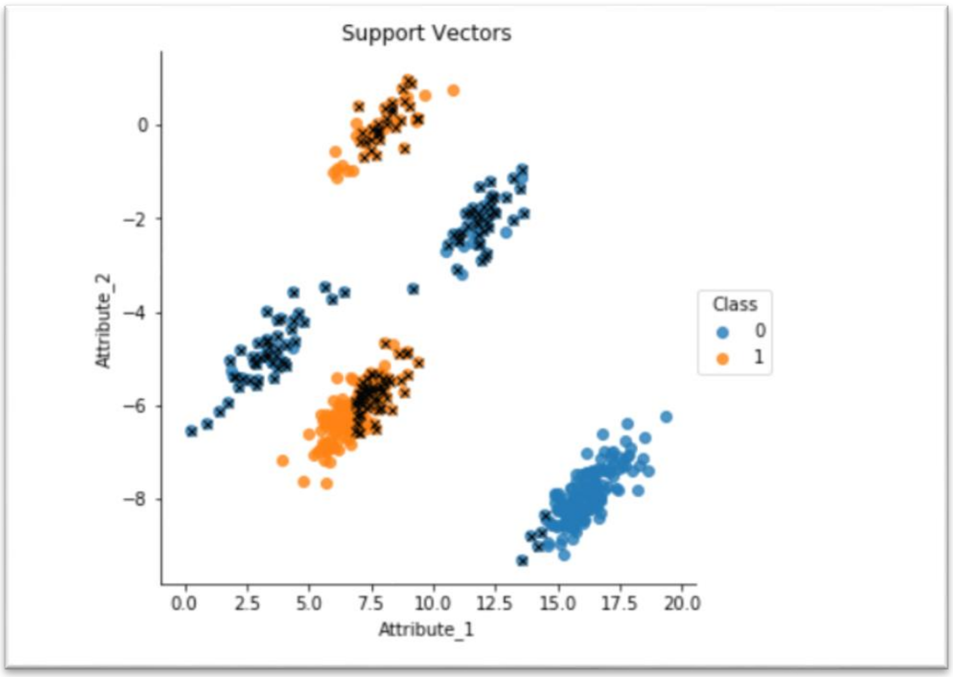
197	53
0	200

Making TN = 197, TP = 200, FP =53, FN =0

- **Precision**: Precision is the fraction of retrieved documents that are relevant to the query. As it is the true positives among the true positives plus false positives, the precision for this hypothesis is :
  Class 0: 197/197+0 = 1 or 100%
  Class 1: 200+53/200 =0.79 or 79%
- **Recall**: Recall is the fraction of the relevant documents that are successfully retrieved. As this is fraction between true positives and true positives plus false negatives, the recall for this hypothesis is:
  Class 0: 197/197+53 = 0.788 or 78.8%
  Class 1: 200/200+0 = 1 or 100%
- **Accuracy**: The number of correct predictions made divided by the total number of predictions made :
  Accuracy = TP+TN/TP+TN+FP+FN
  =200+197/200+197+0+53
  =0.882 or 88.2%

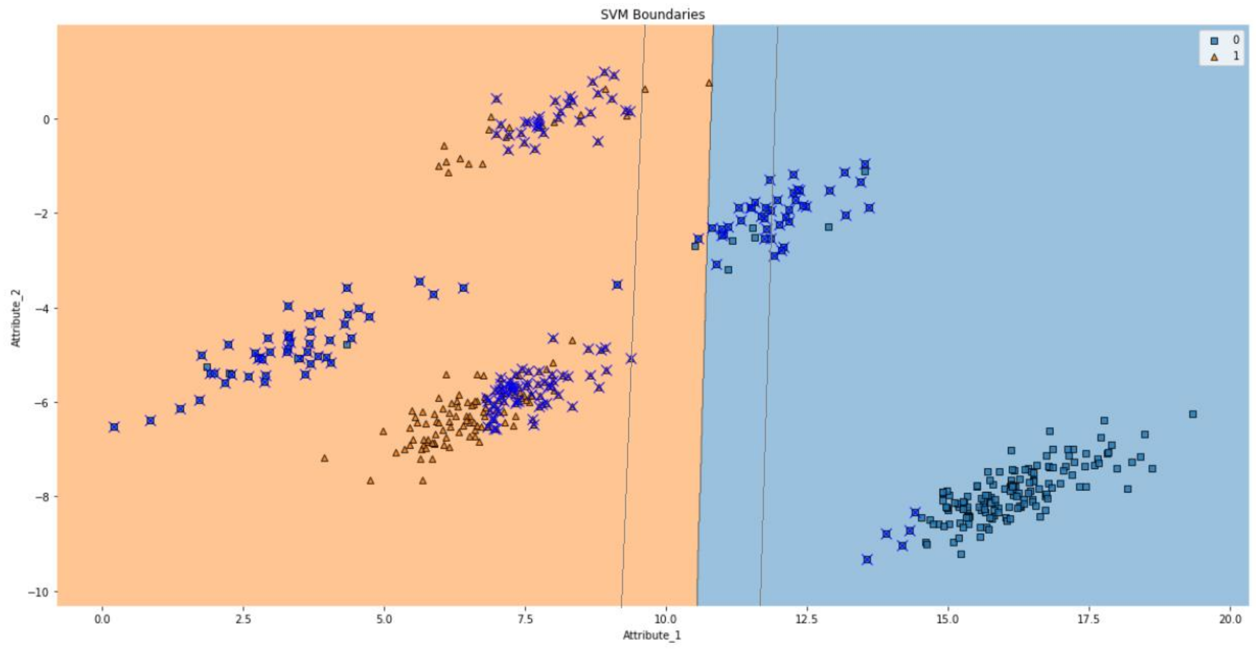
Support Vectors used:

- Support vectors are found using the SVC attribute: Support_vectors_. These are plotted on the 2_D grid of 1a.



- 187 Support vectors are used to decide on the decision boundary represented using marker –'x'

Boundaries drawn by the Linear SVM:



The markers denoted by “x” are the support vectors based on which the decision boundary is defined.

Average performance of Linear SVM model on the dataset:

- K fold is used to find the estimate how well a model can fit the data. This is done analyzing or summarizing the metrics globally.
- **Average Accuracy**: Averaged over all the folds: **0.871 or 87.1%**
- **Average Precision**: Averaged over all the folds (both classes combined): **0.889 or 88.9%**
- **Average Precision**: Averaged over all the folds (both classes combined): **0.883 or 88.3%**
- The above summarized metrics suggest that Linear SVM would be a decent model to be considered for the data set.

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

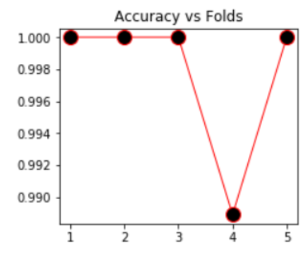
- Steps taken are the ones already mentioned under 1b section for k_fold testing

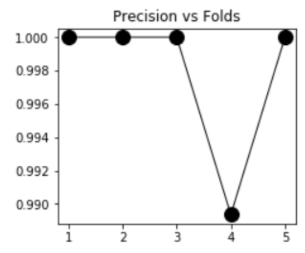
Parameters Considered:

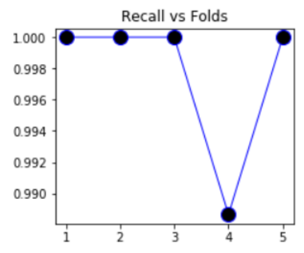
- RBF SVM is built taking the parameters: **Kernal = RBF, C=1.0**, **gamma= 1.0 random_state=200 in SVC** and the **Shuffle = True in Kfold** function.
- RBF SVM only depends on the parameters C (Penalty parameter) and gamma in defining the boundary. C trades correct classification of training examples against maximization of the decision function’s margin. While gamma is the inverse of the radius of influence of samples selected by the model as support vectors.
- As gamma increases the model leads to overfitting and no value of regularization provided by C wouldn’t help.
- The values c=1.0 and gamma =1.0 seems to be the optimal values for the model to give good accuracy taking into consideration the final test set is the 450 data points.

Observations:

- Accuracy, Precision and recall for the five SVMs look like below:







- Hence 1st model (2,3,5 folds also give the same performance) is chosen as the best model for the next steps.
- Tested the best model on the 450 data points and the following are obtained:

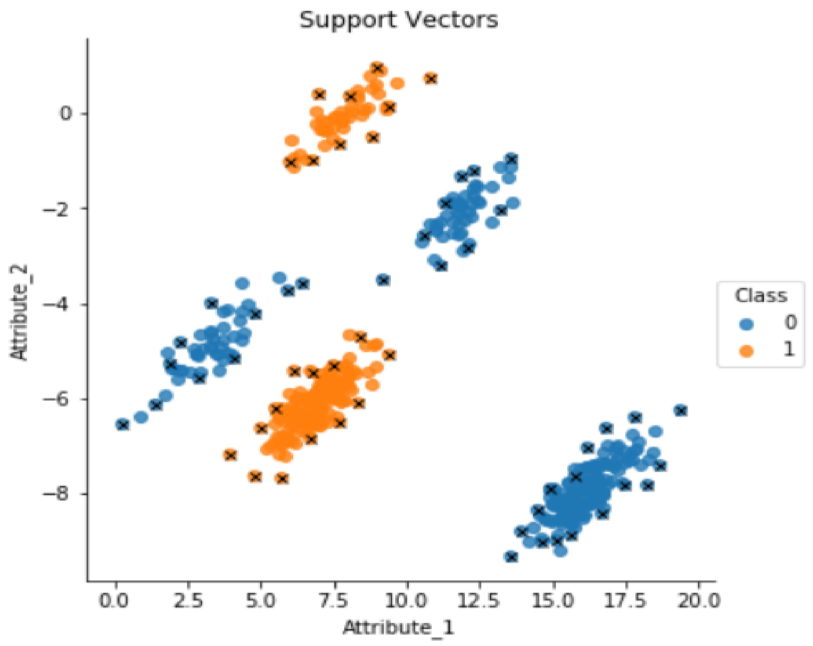
- **Confusion Matrix**: `Predicted vs Actual`

250	0
0	200

  Making TN = 250, TP = 200, FP =0, FN =0
- **Precision**: Precision is the fraction of retrieved documents that are relevant to the query. As it is the true positives among the true positives plus false positives, the precision for this hypothesis is:
  Class 0: 250/250+0 = 1 or 100%
  Class 1: 200/200+0 = 1 or 100%
- **Recall**: Recall is the fraction of the relevant documents that are successfully retrieved. As this is fraction between true positives and true positives plus false negatives, the recall for this hypothesis is:
  Class 0: 250/250+0 = 1 or 100%
  Class 1: 200/200+0 = 1 or 100%
- **Accuracy**: The number of correct predictions made divided by the total number of predictions made:
  Accuracy = TP+TN/TP+TN+FP+FN
  =200+250/200+200+250
  =1 or 100%

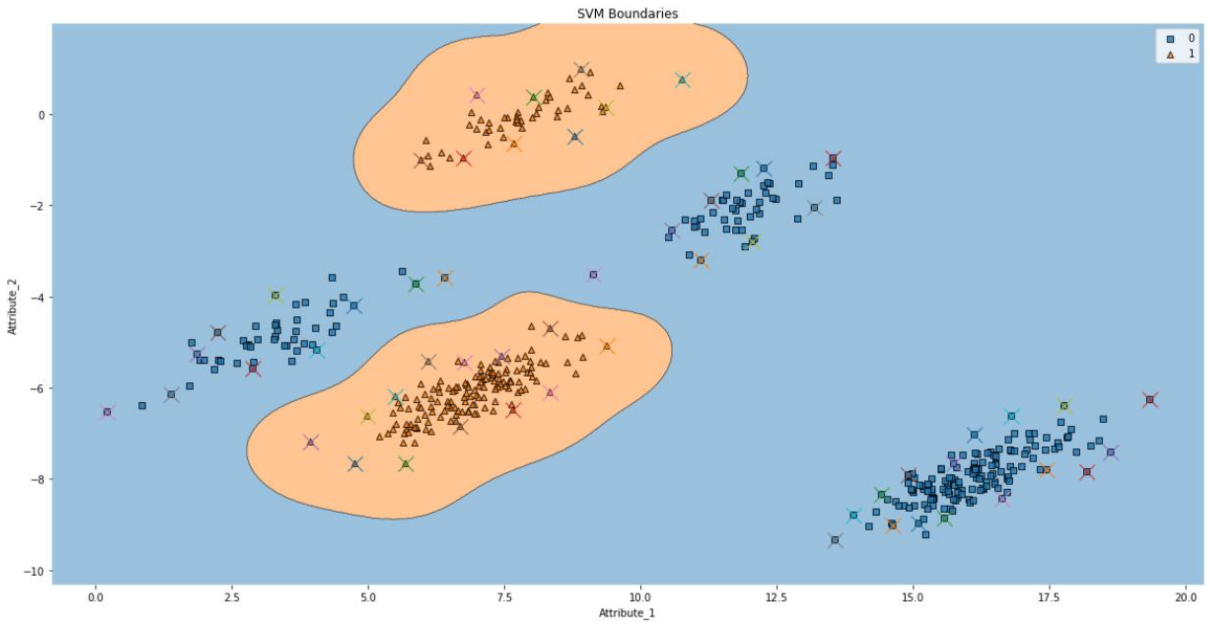
Support Vectors used:

- Support vectors are found using the SVC attribute: Support_vectors_. These are plotted on the 2_D grid of 1a.



- 57 Support vectors(represented by marker 'x') are used to decide on the decision boundary which are way lesser than the number of support vectors used for Linear SVM.

Boundaries drawn by the RBF SVM:



The markers 'x' represents the support vectors in the model.

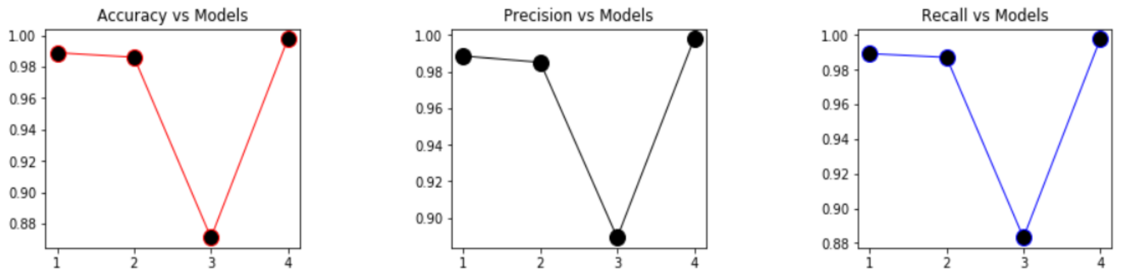
Average performance of RBF SVM model on the dataset:

- K fold is used to find the estimate how well a model can fit the data. This is done analyzing or summarizing the metrics globally.
- **Average Accuracy**: Averaged over all the folds: **0.997 or 99.7%**
- **Average Precision**: Averaged over all the folds (both classes combined): **0.997 or 99.7%**
- **Average Precision**: Averaged over all the folds (both classes combined): **0.997 99.7%.**
- The above summarized metrics suggest that RBF SVM would be a good model to be considered for the data set.

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

The problem here we are dealing with is a **linearly non- separable case**, hence though the performance of the decision trees seems promising, it may really turn out to be bad when the unseen examples not very similar to the training examples used. Linear SVM are not very popular with the linearly non-separable data because of their linear boundary. Nonlinear svm, rbf kernel used here is a best fit to this kind of data however keeping in mind that the dataset is small. As the data grows, the rbf SVM may be very expensive.

Evaluating the Performances:



On X axis it represents model type : 1- Tree(c), 2- Ideal Tree(d),3- Linear SVM(e),4-RBF SVM(f)

On Y axis, average performance of each model on the dataset.

As the plots suggests, RBF SVM has the higher performance metrics among all of them.

Evaluating the Decision Boundaries:

Decision Tree : Performed very well as the training set is very closely representing the testing set, else the boundaries adopted by the algorithm may lead to overfitting for the unseen examples as explained in the 1d part.

Ideal tree: The decision tree boundaries proposed in the 1d part are taken keeping in view of the overfitting scenario that needs to be avoided in the decision trees case. As the boundaries are again only axis parallel lines as in the case of 1c, they cannot perfectly classify all the datapoints properly.

Linear SVM: The decision boundary by Linear SVM is not very optimum as the linearly non-separable data makes it difficult for the margin width and the cost function trade off pretty difficult.

RBF SVM: The RBF SVM with its non - linear functionality can classify the classes appropriately. However the parameters C and gamma have to be chosen with caution to avoid overfitting.

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.**

   - Attributes – X vector = {x1,x2,x3,x4} already given. Adding an unit dimension in order to execute the dot product which is x0 making the final X vector to be: {x0,x1,x2,x3,x4}

- True class – {C0,C1}
- Initial weight vector ={3,4,5,6,7}
- Used the perceptron learning algorithm as the following sequence of steps.
    - Calculated the dot product between X vector and the weights .
    - Checked if the error exists or not:
        - If dot product>0 and true_class= C0 then Error = Y.
        - If dot product <0 and true_class= C1 then Error = Y.
        - In the other cases Error = N
    - Updated the weights :
        - If Error = Y and true class = C0 then W'= W-X
        - If Error = Y and true class = C1 then W' = W+X
        - In other cases W' = W
- Implemented all the above steps using Excel and obtained the following results :

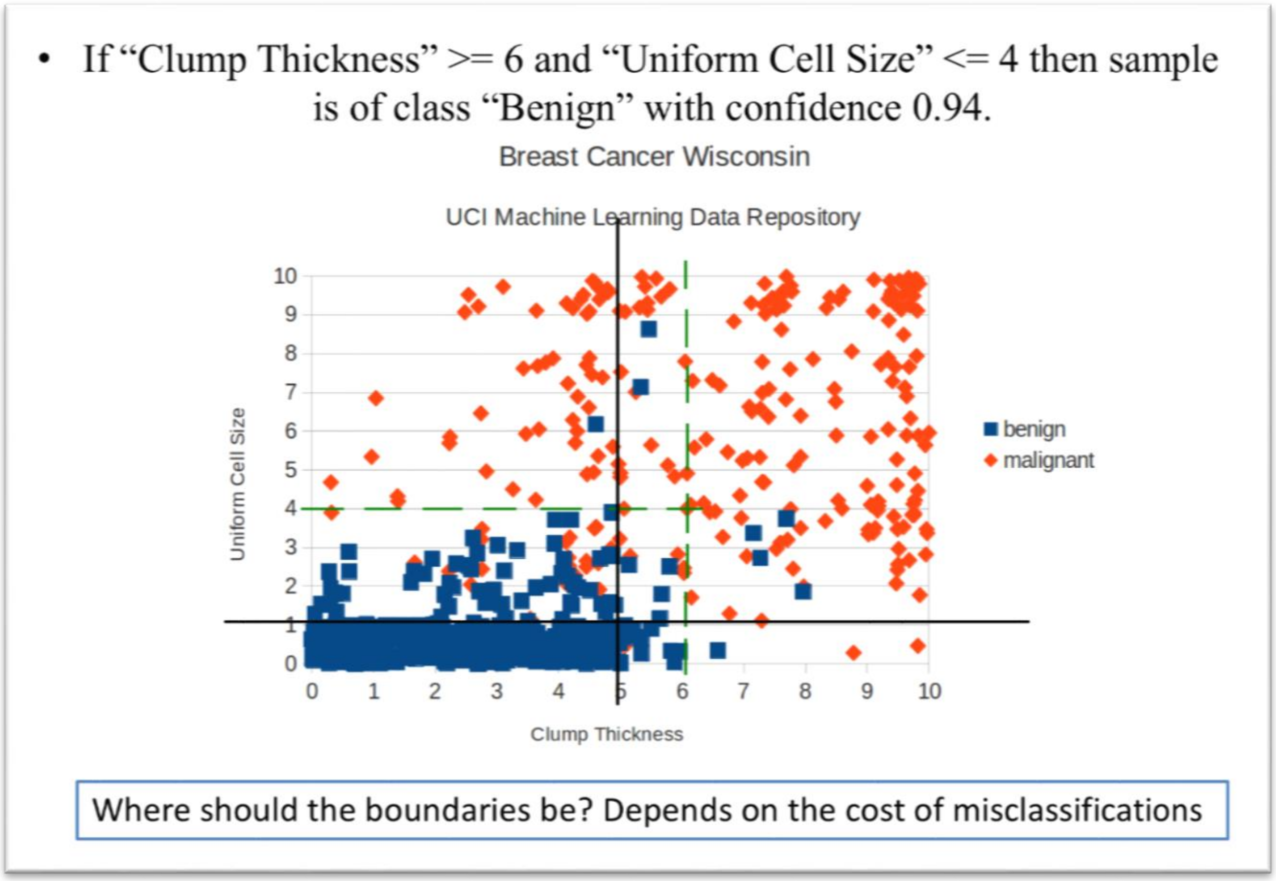
Input Vector					**Weight Vector**					**Dot Product**	**True class**		**Updated Weight Vectors(w+x)**				
X1	X2	X3	X4	X0	W0	W1	W2	W3	W4	W.X		Error	W0'	W1'	W2'	W3'	W4'
3	4	2	5	1	3	4	5	6	7	72	1	N	3	4	5	6	7
5	9	3	10	1	3	4	5	6	7	133	0	Y	-2	-5	2	-4	6
1	8	11	6	1	-2	-5	2	-4	6	-38	1	Y	-1	3	13	2	7
6	1	6	9	1	-1	3	13	2	7	100	0	Y	-7	2	7	-7	6
12	-2	1	8	1	-7	2	7	-7	6	-131	0	N	-7	2	7	-7	6
5	6	0	2	1	-7	2	7	-7	6	-31	1	Y	-2	8	7	-5	7
3	4	2	5	1	-2	8	7	-5	7	22	1	N	-2	8	7	-5	7
5	9	3	10	1	-2	8	7	-5	7	40	0	Y	-7	-1	4	-15	6
1	8	11	6	1	-7	-1	4	-15	6	-55	1	Y	-6	7	15	-9	7
6	1	6	9	1	-6	7	15	-9	7	-13	0	N	-6	7	15	-9	7
12	-2	1	8	1	-6	7	15	-9	7	-136	0	N	-6	7	15	-9	7
5	6	0	2	1	-6	7	15	-9	7	1	1	N	-6	7	15	-9	7
3	4	2	5	1	-6	7	15	-9	7	2	1	N	-6	7	15	-9	7
5	9	3	10	1	-6	7	15	-9	7	-5	0	N	-6	7	15	-9	7
1	8	11	6	1	-6	7	15	-9	7	168	1	N	-6	7	15	-9	7
6	1	6	9	1	-6	7	15	-9	7	-13	0	N	-6	7	15	-9	7
12	-2	1	8	1	-6	7	15	-9	7	-136	0	N	-6	7	15	-9	7
5	6	0	2	1	-6	7	15	-9	7	1	1	N	-6	7	15	-9	7

3.a) 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.

Given: The tree minimizes the number of misclassifications. – Objective would be to maximize the accuracy.
Constraint: Cost of announcing a malignant case as benign is fifty times greater then the cost of announcing a benign case as malignant.
Current rule: Clump thickness >=6 and Uniform Cell size <=4.

Current Misclassified Malignant cases: There are around 24 malignant cases misclassified as benign cases.
Current Misclassified Benign Cases: There are 8 benign cases misclassified as Malignant.
Current Cost: Based on the present boundaries we have 24*50 + 8 ~ 1208(approximately)

Updating the boundaries to reduce the cost of the model:



- If "Clump Thickness" >= 6 and "Uniform Cell Size" <= 4 then sample is of class "Benign" with confidence 0.94.

Where should the boundaries be? Depends on the cost of misclassifications

New rule for Benign Class: Clump thickness >=5 and Uniform Cell size <=1.1

Calculating the cost now: There are zero malignant cases misclassified and around 60 benign cases misclassifications**. Cost ~ 60** (approximately)

3.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?

Constraint: Precision of the benign class has to be improved.
Precision in general: It is the true positives(benign) ratio among all the samples classified as benign or positive class.

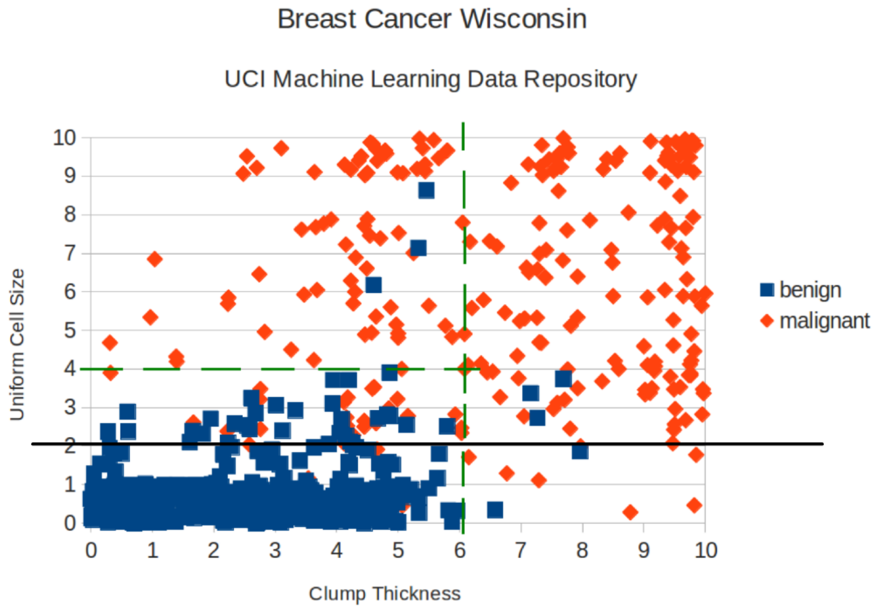
We can just choose a very small region with only the benign cases. We would achieve the 100% precision for benign class. However, the given behavior of tree suggests that the tree always maximizes the accuracy. Therefore, the trade off between the accuracy and precision needs to be taken care of.

Present Precision: b(benign inside the region as per the old rule)/b+24(misclassified malignant cases)
Present Accuracy: b+r/all

Updating the rule: Clump thickness >=6 and Uniform Cell size <=2.
This helps in achieving b'/b'+3(only 3 misclassifications) which is very much higher than the precision of the benign class earlier and the accuracy also is not shattered much.

(Please scroll down for the new boundary)

- If “Clump Thickness” >= 6 and “Uniform Cell Size” <= 4 then sample is of class “Benign” with confidence 0.94.



Where should the boundaries be? Depends on the cost of misclassifications