# Project 1: CS3348

**-DE Ankolika**

**55350200**

1.
   a.

| Subset Size (No. of Attributes) | Attributes in "Best" Subset | Classification Accuracy |
|---|---|---|
| 9 | 1,2,3,4,5,6,7,8,9,10 | 70.56% |
| 8 | 1,2,3,4,5,6,7,8,10 | 77.10% |
| 7 | 1,2,3,4,6,7,8,10 | 77.57% |
| 6 | 1,2,3,6,7,8,10 | 78.97% |
| 5 | 1,2,3,6,7,10 | 78.04% |
| 4 | 1,2,6,7,10 or 1,3,6,7,10 | 77.10% |
| 3 | 1,6,7,10 or 1,3,7,10 | 73.83% |
| 2 | 1,3,10 | 65.89% |
| 1 | 1,10 | 47.20% |
| 0 | 10 | 35.51% |

The following were the steps as to how I came to the above result:

- The first step was the longest one, wherein I tried to remove each attribute and check, which one gave me the highest accuracy.
- This was done via the trial and error method.
- I found out that the removal of the attribute with index 9, resulted in the accuracy being the highest, which was 77.1028%.
- Similarly, I needed to repeat the step and check, for the next removal.
- Now I remove another attribute from this reduced dataset, and it can be seen that removing attribute with indices 9 and 5 gives me a higher accuracy of 77.5701%.
- These steps have to be repeated till, there are no more attributes that can be removed. (Till Null Set)
- Here, one of the main points is that ten is the classifying attribute and necessarily has to be a part of all the cases.

- The above process is basically an explanation of the backward elimination algorithm, done manually.

b.

No, the best accuracy is not the unbiased estimate of accuracy on future data. The following are the reasons for the same:

- This way of assumption is only looking at the local extrema and this doesn't consider the best global accuracy because every single step is based only on its previous step, and a different series of steps may lead to a better accuracy at a different step.
- The test data and the training data should be kept different and separated at all times.
- This is because, the fundamental concept of model selection is in fact a part of the training system.
- This estimate thus becomes biased to be used for prediction of accuracy on future data.
- This would result in the problem of over-fitting, as in it would be too similar to the current data and may fail to fit into any additional data.
- The problem with cross validation to be used for selecting a model is that the training data is also used for testing, which fails to separate the two.
- This estimate then cannot be used to see how the model performs in the future.
- The training of the classifier should not in any way include the test data, or in other words, the test data cannot be involved in training the classifier.
- To have an unbiased estimate of accuracy, the test data and training data should never be overlapped and must be kept separated at all times.
- Moreover, if we have to select the best choice of features, we need to learn and analyse by observing the performance.
- This concludes to the fact, that the model that has the best performance catering to several conditions will be selected.
- This means that the selection process will be somewhat dependant on the test data because that is where the accuracy value will be derived from.
- Hence, if the test data also makes the performance depend on it, it gives rise to a clear bias.
- This can be due to several factors, for instance the possibility of the test data containing noise which can result in the accuracy being high.
- This can be understood from a simple experiment of tossing two coins and your desirable outcome being heads.
- You might end up choosing the coin, that would give a higher experimental probability of having heads, but this probability can be greater than the theoretical chance of heads being 50%.

2.

According to me, the problem can be solved in the following way, so as to get a better performance.

- Instead of doing test and training data, the dataset can be divided into three parts which are training set, test set and validation set.
- For this, a similar approach can be used, wherein backward search algorithm can be implemented, by using the principle of "best first".
- So, a model needs to be built for each subset of attributes. This is built on the training set.
- This model can be compared to the validation set, and so every model will have to be validated with the validation set.
- Now, we can get the best attribute set by selecting the subset that performs the best with the validation set.
- We now train more models, with the training and validation sets.
- We then test the model, using the test set.
- This is our final test set, that can finally be used for testing this model.
- All in all, this is a good manner in which the estimation of the classifiers performance on future data can be assessed.
  The accuracy might then be as follows:

| Subset | Accuracy |
|---|---|
| Rl, Na, Mg, Al, Si, K, Ca, Ba | 76.1% |
| Rl, Na, Mg, Al, Si, K, Ca | 78.5% |
| Rl, Na, Mg, Al, K, Ca | 71.4% |
| Na, Mg, Al, K, Ca | 69.0% |
| Mg, Al, K, Ca | 73.8% |
| Mg, Al, K | 50.0% |
| Mg, K | 47.0% |
| Mg | 35.0% |
| | |

So the best deployment would be to the subset: Mg, Na, Rl, Si, Ca, K, Al, as it gives the best accuracy for future performance.

3.

a.

```
Attribute Evaluator (supervised, Class (nominal): 17 class):
        Information Gain Ranking Filter

Ranked attributes:
 0.2948   2 wage-increase-first-year
 0.1893   3 wage-increase-second-year
 0.1624  11 statutory-holidays
 0.1341  14 contribution-to-dental-plan
 0.1164  16 contribution-to-health-plan
 0.1091  12 vacation
 0.0855  13 longterm-disability-assistance
 0.0717   9 shift-differential
 0.0548   7 pension
 0.0484   5 cost-of-living-adjustment
 0.0333  15 bereavement-assistance
 0.0307   4 wage-increase-third-year
 0.024   10 education-allowance
 0.0195   8 standby-pay
 0        6 working-hours
 0        1 duration

Selected attributes: 2,3,11,14,16,12,13,9,7,5,15,4,10,8,6,1 : 16
```

According to the above screenshot, the top four attributes selected are

- 2 wage-increase-first-year
- 3 wage-increase-second-year
- 11 statutory-holidays
- 14 contribution-to-dental-plan.

Therefore, the series goes like the following indices:

1. (2): 0.2948
2. (3): 0.1893
3. (11): 0.1624
4. (14): 0.1341

b.

On running the CfsSubsetEval using the BestFirst search and then running the wrapper method with J48 as the base learner, some interesting conclusions could be made.

The attribute subset that was received from the CfsSubsetEval and BestFirst as the search algorithm contained the attributes with indices: 2,3,5,11,12,13, 14: 7.

They were the following:

- wage-increase-first-year
- wage-increase-second-year
- cost-of-living-adjustment
- statutory-holidays
- vacation
- longterm-disability-assistance
- contribution-to-dental-plan

The attribute subset that was received from the Wrapper method, with J48 as the base learner and BestFirst as the search algorithm contained the attributes with indices: 1,2,4,6,11,12: 6

The following were the attributes:

- duration
- wage-increase-first-year
- wage-increase-third-year
- working-hours
- statutory-holidays
- vacation

The attributes that are selected by both the methods are the attributes with indices: 2,11 and 12.

They are:

(2) wage-increase-first-year,

(11) statutory-holidays and

(12) vacation.

The top 4 attributes in the output generated by information gain and the ones generated by the above methods, have a few common overlapping attributes. The second and eleventh attribute both come in the top four attributes of the information gain ranking method. Yet, there is a major difference when it comes to the other attributes, as they are ranked at much lower positions in InfoGain. This comes from the fact that the evaluation done by InfoGain is individualistic. Hence, it only does the evaluation individually whereas wrapper and CfsSubsetEval take the correlation between attributes into consideration during the proess of evaluation.

c.
After loading the diabetes dataset and adding a single copy of the first attribute, and attempting to measure the performance of NaiveBayes using Supervised Discretization, we can easily establish that adding copies of the attributes, decreases the accuracy and hence the performance of the classifier.
When there are no copies, the correctly classified instances or the accuracy percentage remain to be 74.349%, whereas after adding copies, the accuracies decrease as follows:

- 1 copy of preg ->74.0885%
- 2 copies of preg->73.6979%
- 3 copies of preg->71.6146%
- 4 copies of preg->71.224%
- 5 copies of preg->70.8333%

This is perhaps because of the redundancy being a cause of introducing misleading data, as there is irrelevant data in the set. The redundancy causes the main issue of there not being enough unique data to estimate the true value of the accuracy of a classifier. The other problem being that any duplicated data will make the classifier skewed towards the redundant data, as they are more common to be found.

d.

- Using CfsSubsetEval, the redundancy could be removed, as attributes with indices 2, 6, 7, 8 were chosen to be in the subset. Here, the accuracy with or without copies remain to be 75.52 percent, when 6 copies are made.

- Using infoGainAttributeEval, with ranker, the redundancy could not be removed as attributes chosen to be in the selected subset, included the copies of the first attribute (2, 6, 8, 5, 4, 12, 1, 3). Here, the accuracy with the duplicated values seemed to be 72.01% while the accuracy without copies was 74.35%.

- By using the wrapperSubsetval method, the redundancy could again be removed, as the selected attributes in the subset were 1, 2, 3, 6, 7. The accuracy with and without copies remained to be 65.19%.

- Here the following are the main observations:

  If we simply take the evaluators and compare, without the duplicate data values, we reach the conclusion that Wrapper method is a poor performer.

InfoGain performs in the middle, whereas Cfs is the best.
Naïve Bayes, works better than all of these, when classifying without attribute selection and thus supercedes them all.

However, the trick comes when redundancy is brought in. The hierarchy remains the same, but Cfs performs the best, after which InfoGainAttributeEval comes and is followed by WrapperSubsetEval. Yet, we notice lowered accuracy values in Naïve Bayes and InfoGain.

- The last factor comes to the fact that infoGainAttributeEval is not able to remove the redundancy, whereas WrapperEval as well as CfsSubsetEval are able to do so. This is because infoGain does the comparism and evaluation individually, thus the ranker search method that it uses does the ranking in an independent and individual manner.
- Hence, all of the duplicate values or the copies are chosen according to their respective rankings. i.e. if they are one of the 8 attributes on the top.

```
Ranked attributes:
 0.1901    2 plas
 0.0749    6 mass
 0.0725    8 age
 0.0595    5 insu
 0.0443    4 skin
 0.0392   14 Copy of Copy of Copy of Copy of Copy of preg
 0.0392   15 Copy of Copy of Copy of Copy of Copy of Copy of preg
 0.0392    1 preg
```

e.
In the case of using IBk settings with the parameter tuned version, an accuracy of 73.9583% was found, whereas with the normal default settings the accuracy was found to be 70.18%. The 7 nearest neighbours were chosen for the purpose of classification.

- It is evident that the paramenter tuned version gives a higher accuracy and thus a better performance, when compared to the untuned version.
- The reason behind this is that the K value which varies from 1 to 10, has to be such that it will give the highest accuracy, and then this K value is selected for evaluation process in the version that has the parameter tuned.

Thus, the value taken by the parameter in the tuned part, is 7, which translates to the best result being possible when the IBK algorithm uses 7 nearest neighbours for the evalutation process.

f.

- Using J48 with default parameters gives an accuracy of 73.8281%, with the size of the tree being 39 and number of leaves being 20.

- On the other hand, with CVParameter selection, the accuracy stands to be 74.349%, with a size of the tree being 29 and the number of leaves being 15.
- The tuned version works much better than the normal default version, and thus gives much higher accuracy. (0.5%)
- Yet, even though the difference in performance accuracies is nothing extraordinary, the number of leaves and size of the tree have increased by a huge number, when changed from parameter tuned version to the default version.
- The valued that work as selected parameters are: C=0.2 and M=10.

4.

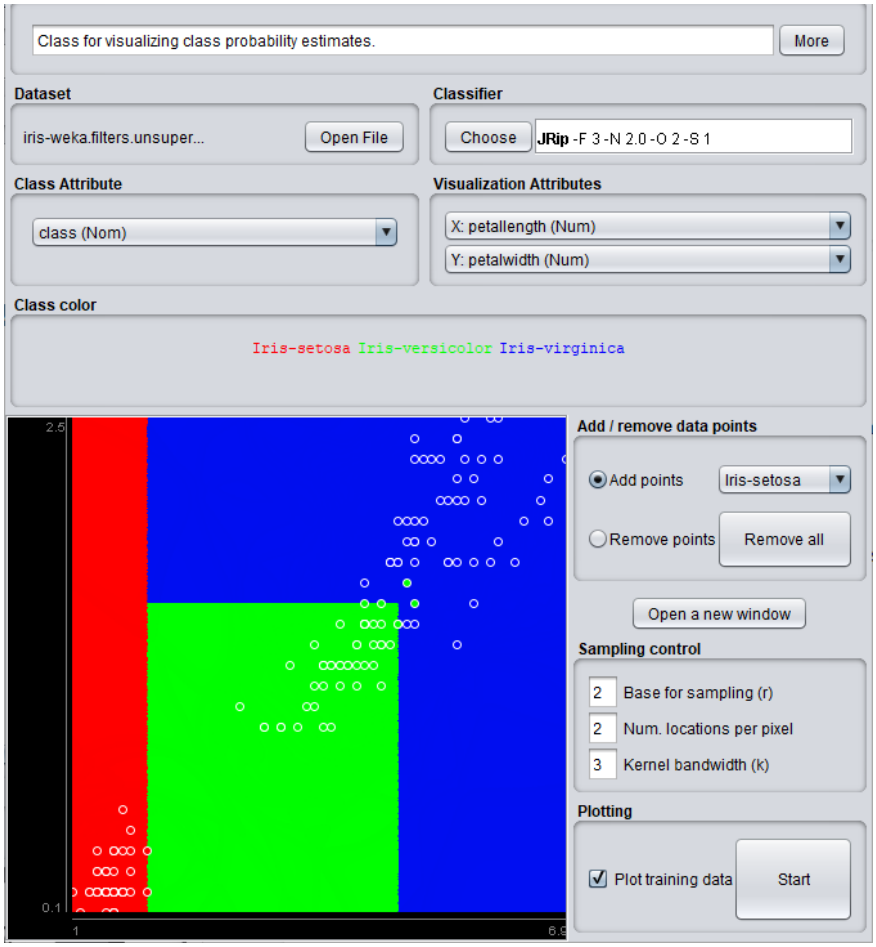
a.

Here we observe that the rules that are outputted on the explorer and the output in the plot result are similar and have the same values. However, some boundary cases don't follow the rules and have differently labelled values. These cases can be seen in the second figure, wherein different coloured points may be on a different coloured zone.

The rules from the explorer are as follows.

- (petallength <= 1.9) => class=Iris-setosa (50.0/0.0)
- (petalwidth <= 1.6) and (petallength <= 4.9) => class=Iris-versicolor (47.0/0.0)
- => class=Iris-virginica (53.0/3.0)

The plot result is as follows.




b.

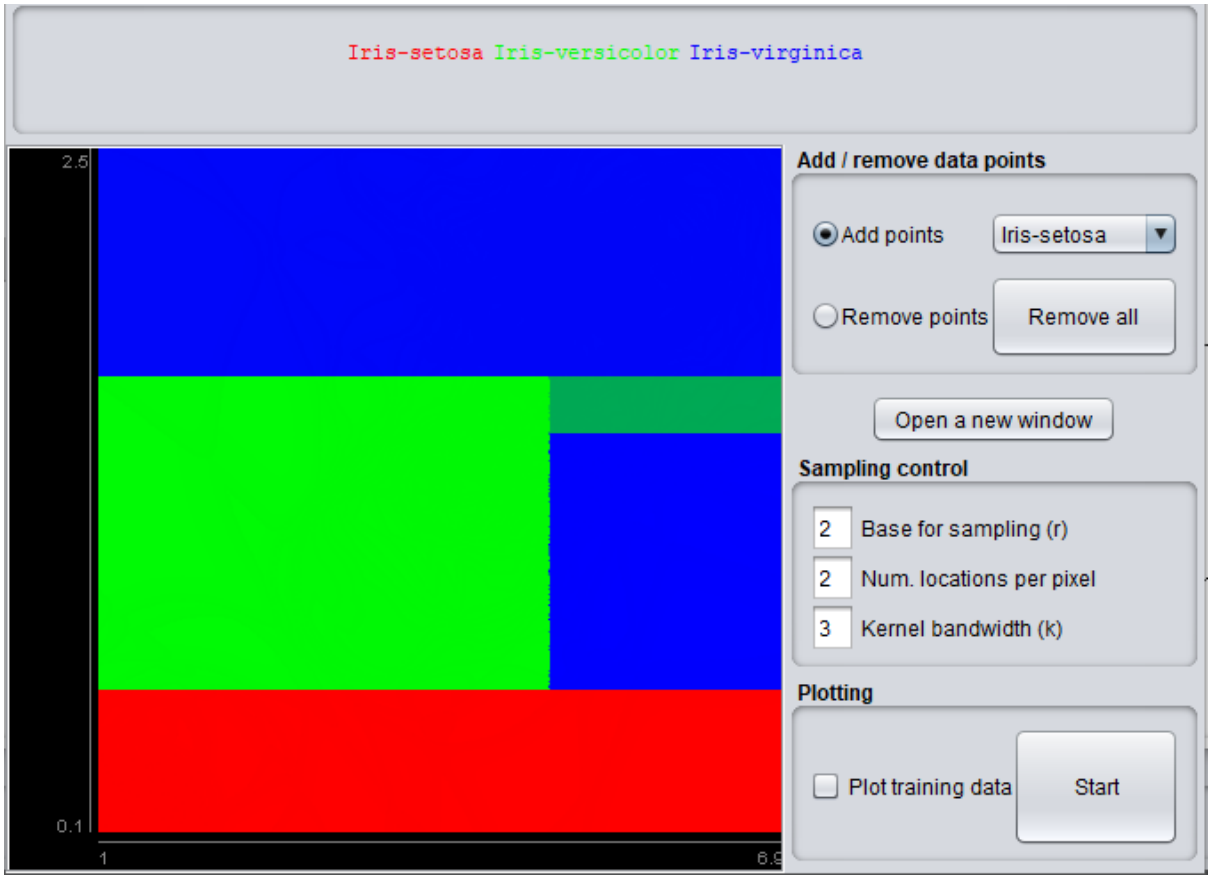
The independent rules for the question are the following:

- petallength <= 1.9) => class=Iris-setosa (50.0/0.0)
- (petalwidth <= 1.6) and (petallength > 1.9 and petallength<= 4.9) => class=Iris-versicolor (47.0/0.0)
- (petallength > 4.9) and (petalwidth > 1.6) => class=Iris-virginica (53.0/3.0)

These rules will give the equivalent result regardless of the order in which they are executed, because they are independent of one another and hold no dependency. They all have individual commands which can run, without any other assistance.

c.

The data boundaries that have been visualized, are consistent with the conditions of the tree that has formed in the explorer.

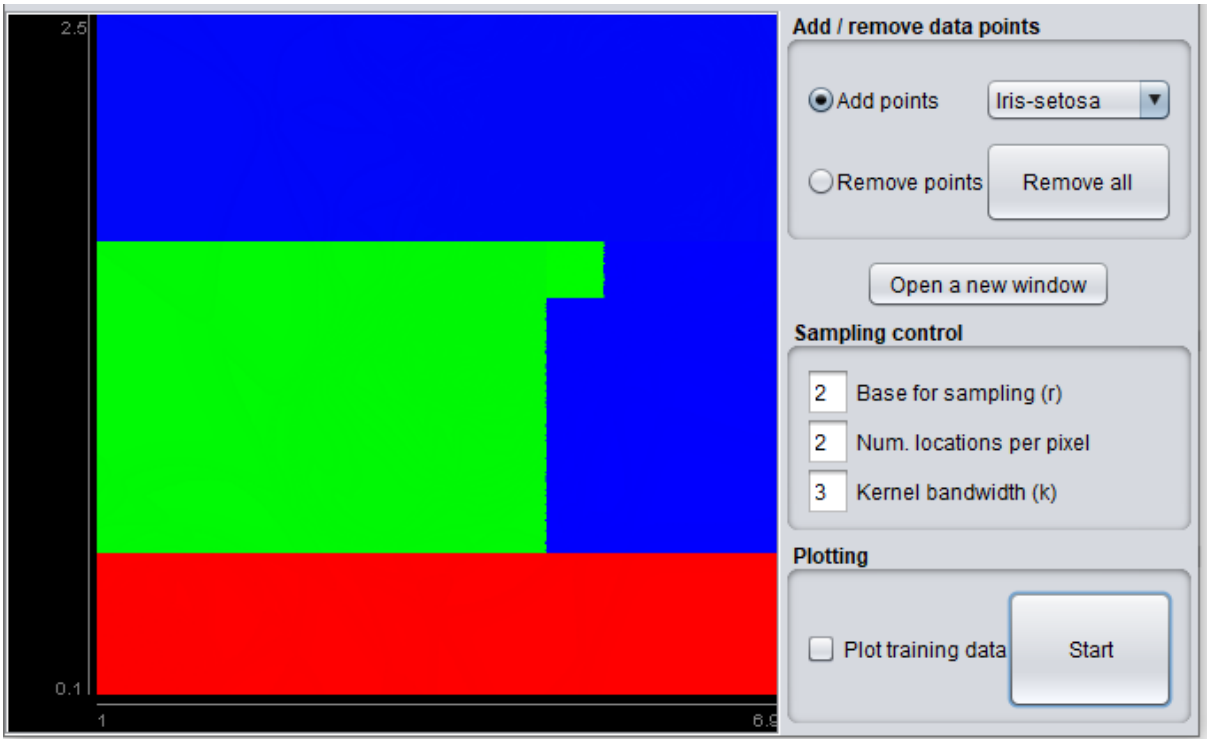



The above is the plot for the data. The tree has been drawn below, with 5 leaves and the size being 9.

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
| petalwidth <= 1.7
| | petallength <= 4.9: Iris-versicolor (48.0/1.0)
| | petallength > 4.9
| | | petalwidth <= 1.5: Iris-virginica (3.0)
| | | petalwidth > 1.5: Iris-versicolor (3.0/1.0)
| petalwidth > 1.7: Iris-virginica (46.0/1.0)

However, there is some conflict in certain areas: For example,

- x>4.9, 1.5<y<1.7: This area has ambiguity, as the results appear to be mixed, due to the existence of more points belonging to the green area(versicolor), thus it is more green than blue(verginica).
- As the number of values belonging to versicolor is higher, the plot region is greener.
- Adjusting the minimum number of instances required in a leaf, will result in the adjustment of pruning in J48.
- Thus, if we change the value of minNumObj from 2 to 1, we will have achieved our goal of reducing the number of instances in a leaf node. This will also remove part of the conflicting area. The tree formed so will have 6 leaves and a size of 11.




The tree is as follows:

```
petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
|   petalwidth <= 1.7
|   |   petallength <= 4.9: Iris-versicolor (48.0/1.0)
|   |   petallength > 4.9
|   |   |   petalwidth <= 1.5: Iris-virginica (3.0)
|   |   |   petalwidth > 1.5
|   |   |   |   petallength <= 5.4: Iris-versicolor (2.0)
|   |   |   |   petallength > 5.4: Iris-virginica (1.0)
|   petalwidth > 1.7: Iris-virginica (46.0/1.0)
```

- This is because we have managed to lessen the number of instances that are required in a leaf node.
- If we change minNumObj to 1, our accuracy becomes maximum, as boundaries become sharper.
- On making this value bigger, our accuracy reduces.

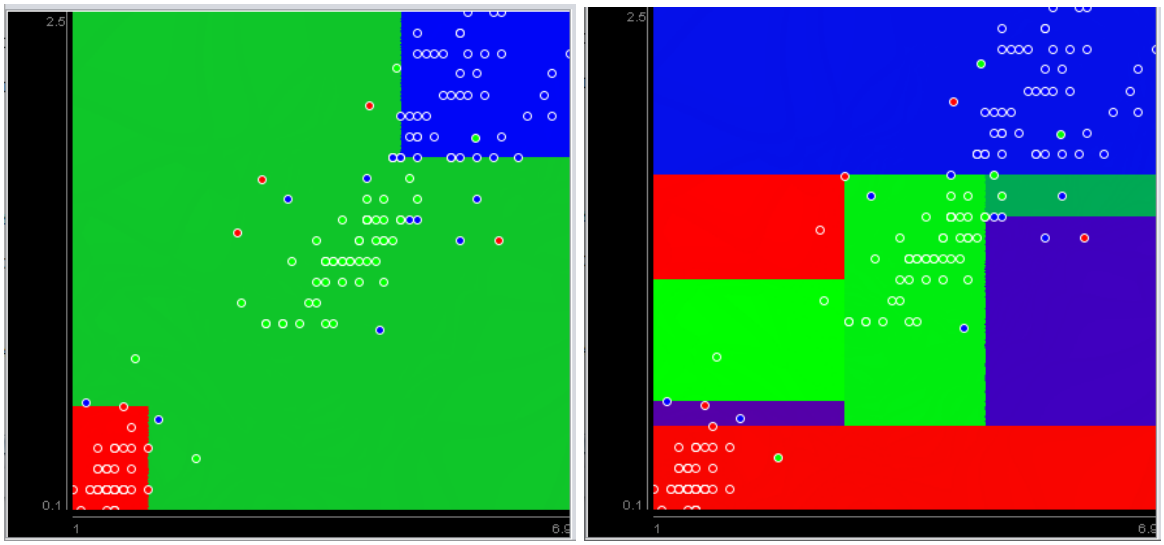
d.

To generate trees with 3,2 and 1 leaf node, the exact range for minNumObj are as follows:

- For a tree with 3 leaf nodes the range is from 7 to 49.
- For a tree with 2 leaf nodes, the range is from 50 to 75.
- For a tree with 1 leaf node the range will be all values above 75(greater than or equal to 76).

e.

The noise impacts the performance of the two attributes in different ways.




The graph on the left is JRip with noise, while the graph on the right is J48 with noise.
The following are the observations from these graphs:

- JRip is more stark as compared to J48, that is, it is willing to sacrifice its accuracy so that it can be sharper at the boundary regions, and give lesser areas with mixtures of colours. J48 does not have sharp boundaries, the boundaries are rather disconnected.
- There is a distinct horizontality in J48, whereas a verticality in JRip. The connections also follow the respective horizontality and verticality.