SENG 474: Assignment 1

Name: Subah Mehrotra

Introduction

This document contains analysis of two different datasets: **Cleveland** and **Students' Academic Performance.** The three machine learning algorithms that have been analyzed on the above-mentioned datasets are decision tree, random forest, and neural networks. Weka, a machine learning software was used to perform all the tests and the analysis.

Section 1

This section includes analysis of decision trees, random forests, and neural networks with **Cleveland dataset.**

Decision Tree

Decision tree is one of the predictive modelling approaches used in statistics, data mining and machine learning. The split criterion used was information gain. Reduced error pruning was used to reduce the complexity and improve predictive accuracy. The decision tree formed consisted of 22 leaves and its size was 31. Figure 1 shows the decision tree formed with Cleveland dataset.

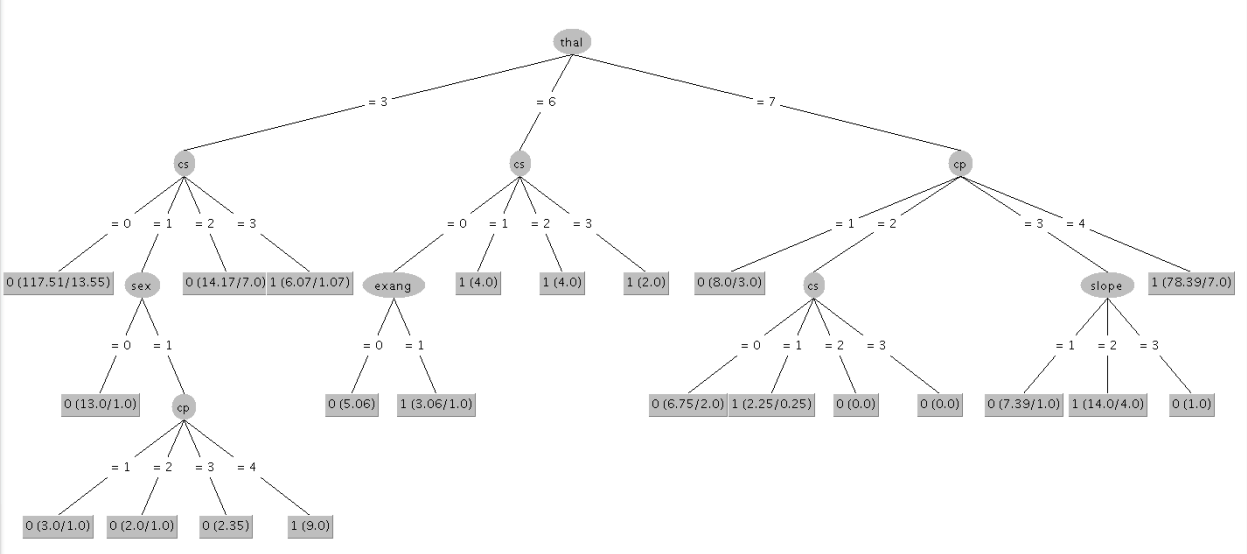


Figure 1: Decision Tree for Cleveland Dataset

The analysis was performed using the following test options:

- Cross Validation
- Percentage Split

Cross Validation

Cross validation is a technique for evaluating machine learning models by training several machine learning models on subsets of the available input data and evaluating them on the complementary subset of the data. It trains a model on all of the partitions except one that is held out as the test set, then repeat this process creating k-different models and give each fold a chance of being held out as the test set. Then calculate the average performance of all k models. First it was tested on **10 folds** with the following results (see fig 2):

- Correctly Classified Instances: 72.2772%
- Incorrectly Classified Instances: 27.7228%
- Mean Absolute Error: 0.3508
- Total Number of Instances: 303

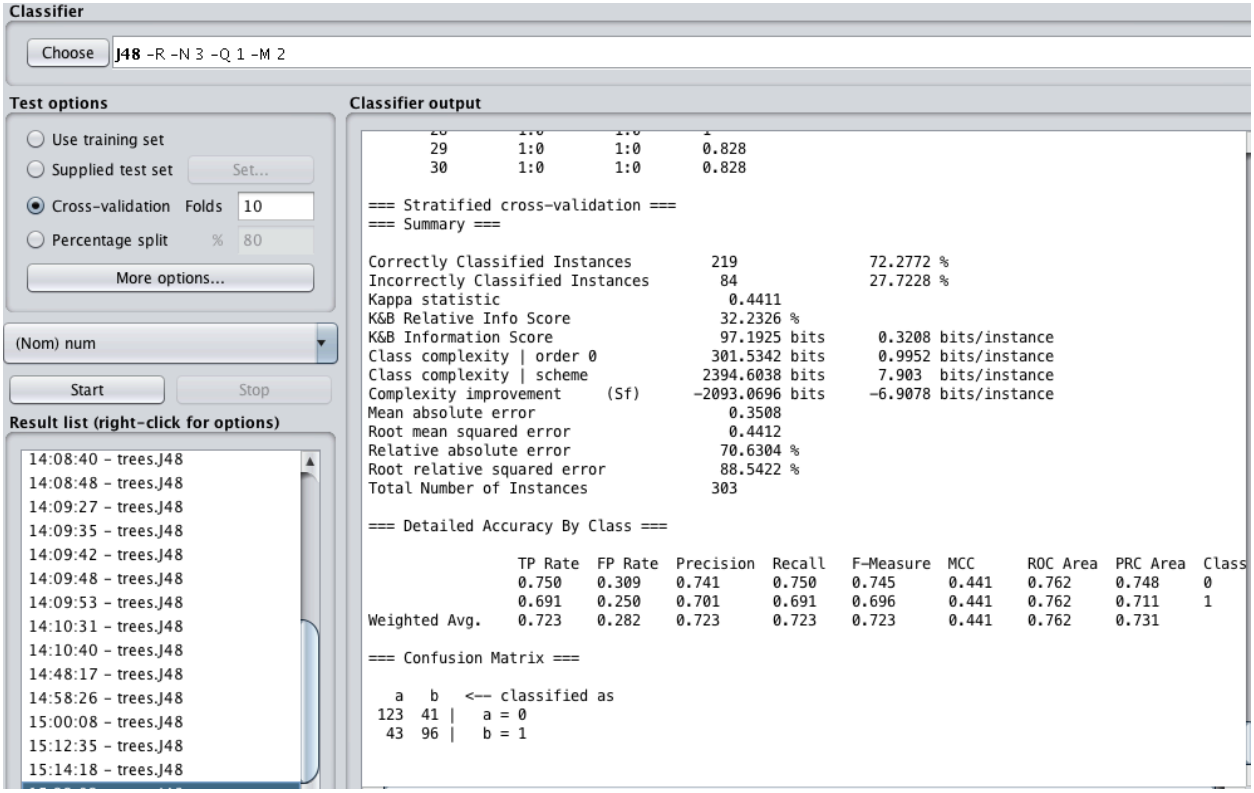


Figure 2: Cross validation result with 10 folds

It was then tested on **5 folds** and the result are as follows (see fig 3):

- Correctly Classified Instances: 74.2574%
- Incorrectly Classified Instances: 25.7426%
- Mean Absolute Error: 0.3203
- Total Number of Instances: 303

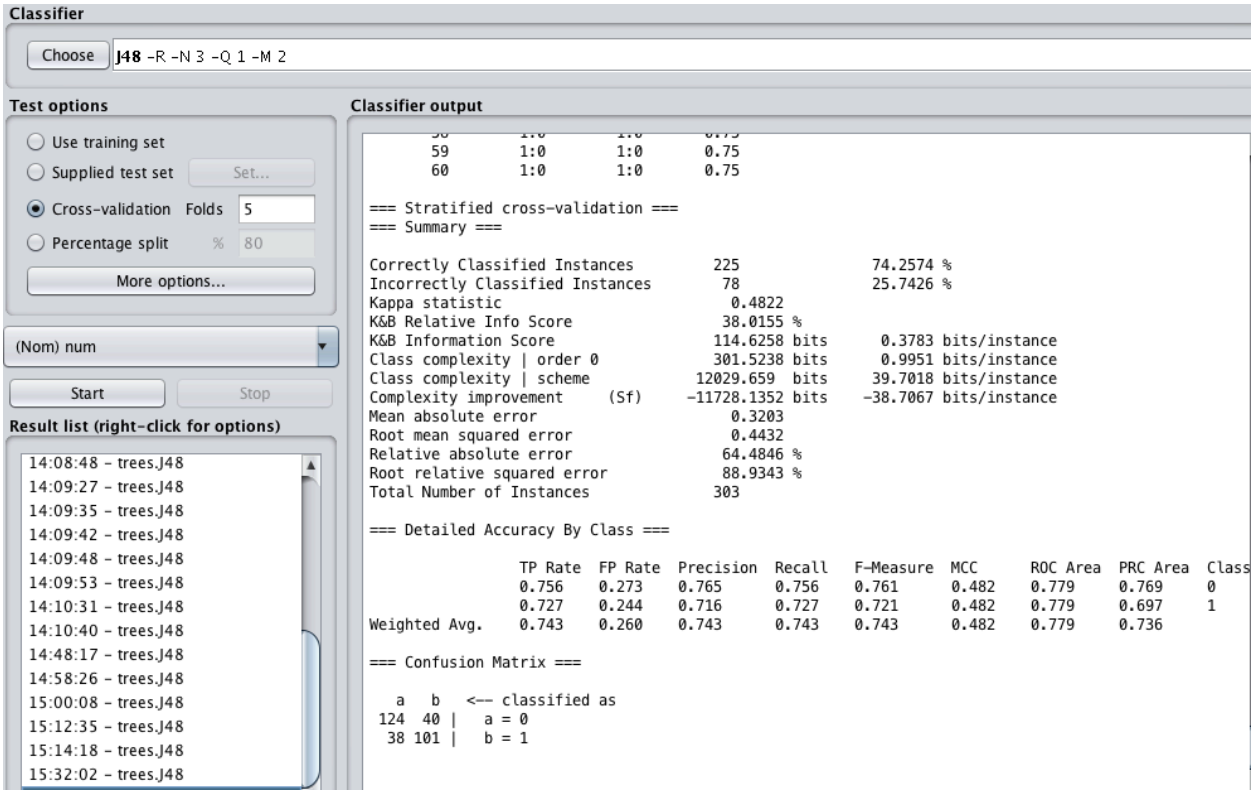


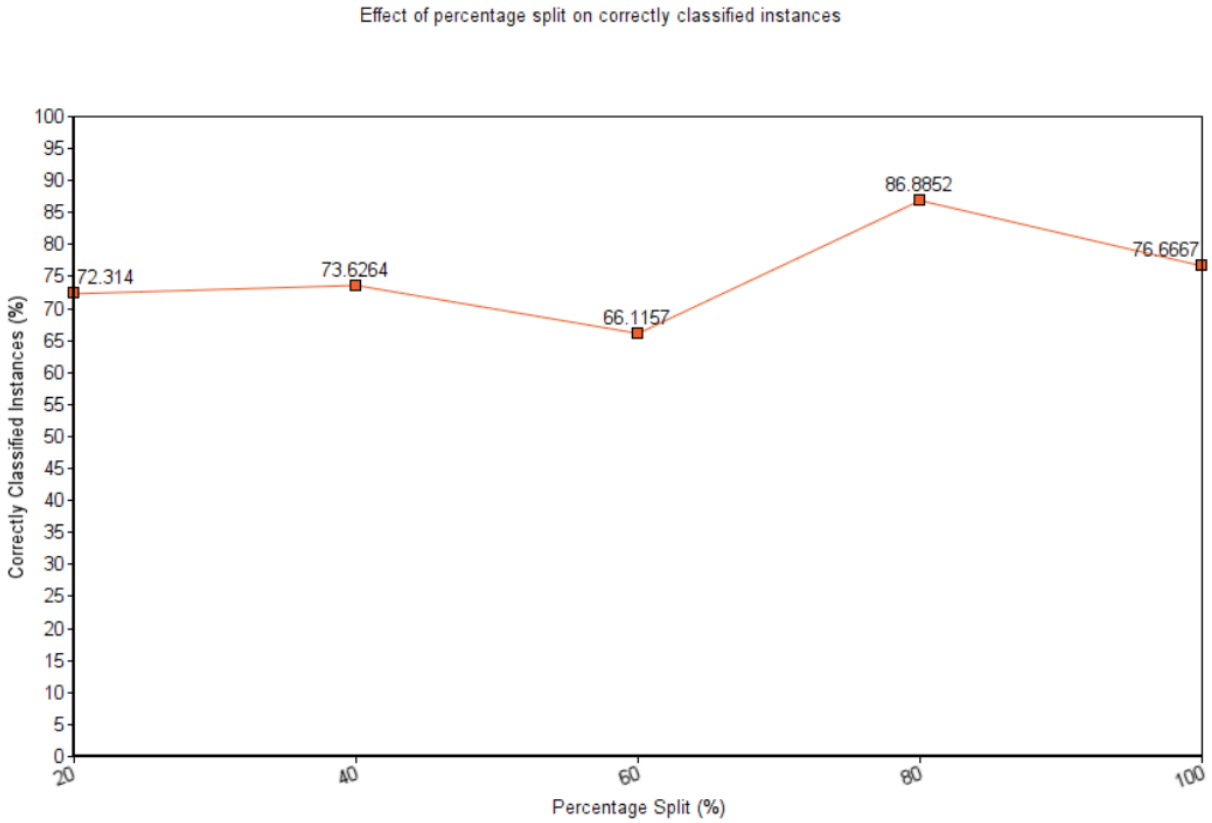
Figure 3: Cross validation result with 5 folds

The correctly classified instances were seen to be a little better with **5 folds**. This is because the test results are seen to be best with 80% training set and 20% validation set. This provides enough data for the algorithm to be trained quite well. With 5 folds, the algorithm is trained on 4/5 = 80% of the data and is tested on the remaining 1/5 = 20% of the data. With 10 folds the algorithm is provided with 9/10 = 90% of the training set and 1/10 = 10% of the validation set. This does not provide the algorithm with enough validation data to calculate an accurate enough result.

Percentage Split

Percentage Split randomly splits the dataset into a training and a testing partitions each time model is evaluated. This can give a very quick estimate of the performance. Following are the results with different percentage split results:

Percentage Split	Correctly Classified Instances	Mean Absolute Error	Total Number of Instances
20%	72.314%	0.3281	242
40%	73.6264%	0.3032	182
60%	66.1157%	0.346	121
80%	86.8852%	0.3078	61
90%	76.6667%	0.3409	30



Graph 1: Effect of percentage split on correctly classified instances

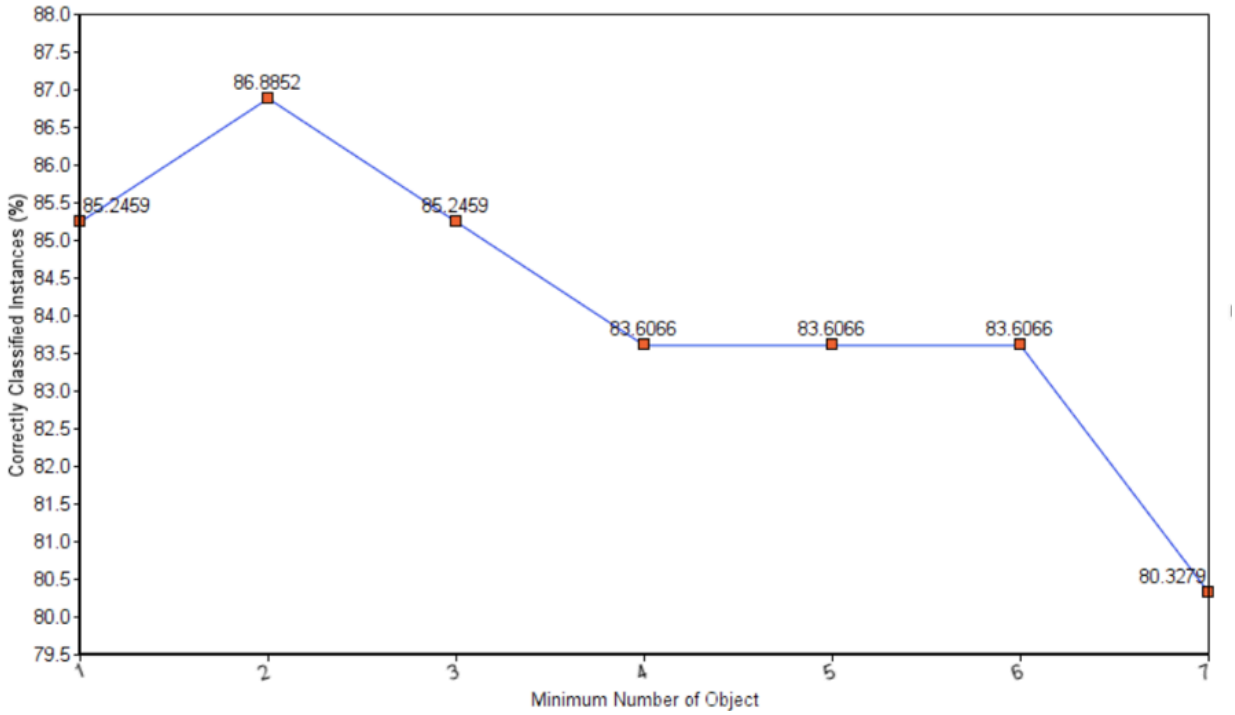
The best results were seen when the percentage split was set to 80%. The correctly classified instances were 86.8852%. As explained above, this was because the algorithm enough data for both training and evaluation. 80% of the dataset was used for training and the rest 20% for evaluation. The lowest correctly classified instances appeared when the percentage split was set to 60% (see graph 1). This could be because the algorithm didn't get enough data for training or evaluation when it was split almost halfway.

The algorithm was also tested by updating the following parameters:

- **Confidence Factor:** The confidence factor represents a threshold of allowed inherent error in data while pruning the decision tree. It was seen that lowering the confidence factor resulted in more pruning.

- **minNumObj:** The minNumObj parameter defines the minimum number of instances supported by the tree in a leaf node when constructing the tree from the training data. minNumObj should be higher for noisy data. The best correctly classified instances were seen to be with value **2** for minNumObj (see graph 2). This was because the Cleveland dataset is not very noisy. Table below shows the change in correctly classified instances with minNumObj.

minNumObj	Number of leaves	Size of the tree	Correctly Classified Instances
1	96	101	85.2459%
2	97	102	86.8852%
3	13	19	85.2459%
4	13	19	83.6066%
5	10	14	83.6066%
6	7	10	83.6066%
7	10	15	80.3279%



Graph 2: Effect of minNumObj on correctly classified instances

Random Forest

The random forest is a model made up of many decision trees. This model uses two key concepts:

- Random sampling of training data points when building trees
- Random subsets of features considered when splitting nodes

Figure 4 shows the result running random forest with default values (bagSizePercent = 100 and batchSize = 100) with the following results:

- Correctly Classified Instances: 70.297%
- Incorrectly Classified Instances: 29.703%

- Mean Absolute Error: 0.4264
- Total Number of Instances: 303

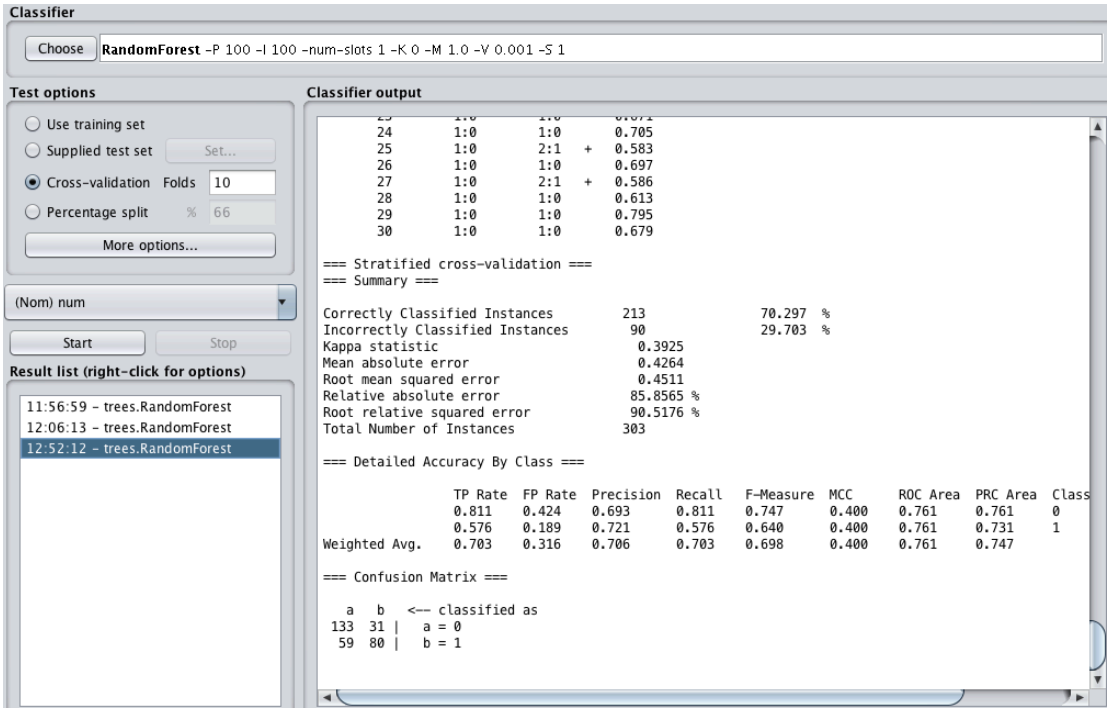
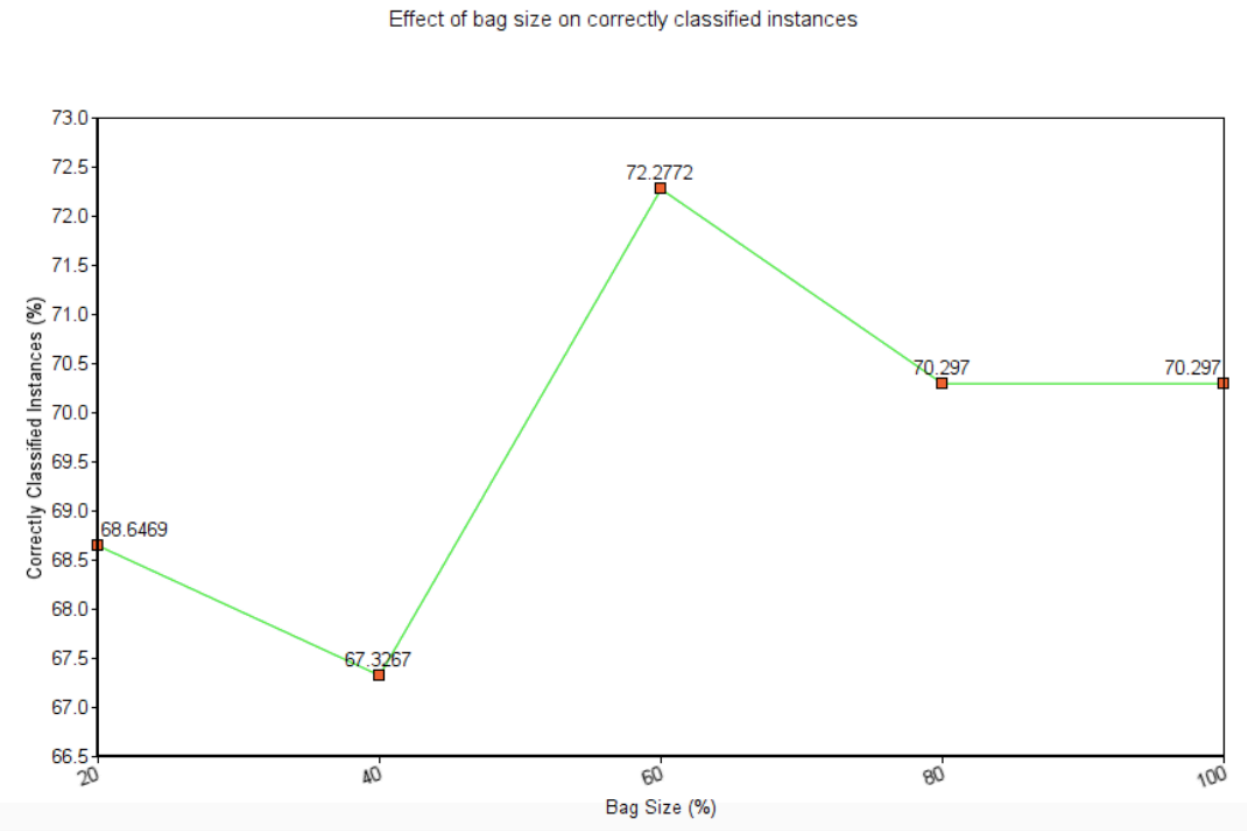


Figure 4: Results with default bagSizePercent and batchSize

The random forest results were then tested with different values of:

- **bagSizePercent:** used for bagging a classifier to reduce variance. It is the size of each bag as a percentage of the training set size. The out of bag data is used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance. The best results were seen when the bag size was set to 60% (see graph 3).

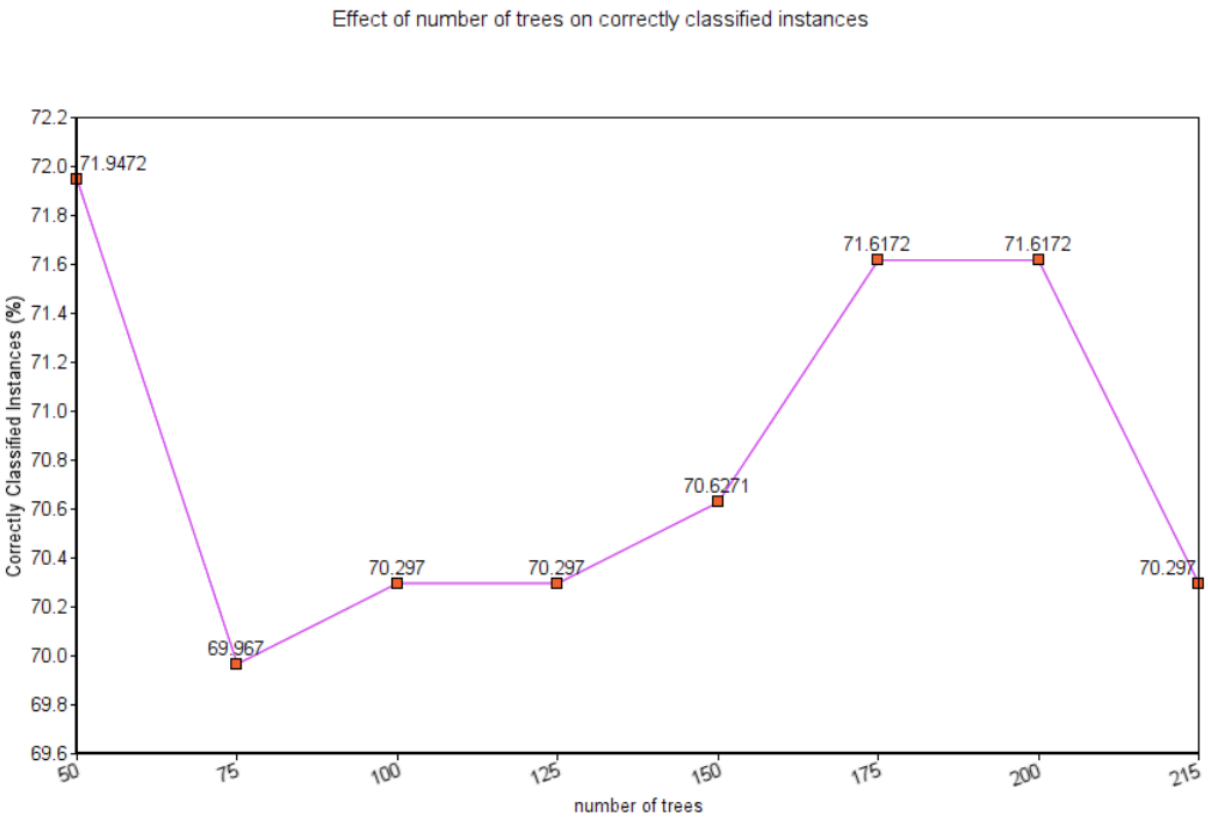
bagSizePercent	Correctly Classified Instances
20	68.6469%
40	67.3267%
60	72.2772%
80	70.297%
100	70.297%



Graph 3: Effect of bagSizePercent on correctly classified instances

- **numIterations:** The **number of trees** in the random forest. Updating number of trees did not have a huge change in the percentage of correctly classified instances. This could be because the dataset used was not that big. It was seen that increasing the number of trees resulted in a better accuracy. The best results were seen with **50** trees (see graph 4). It was also seen that the number of correctly classified instances were reduced when the number of trees were **225**. The prediction performance went down with learning 225 trees.

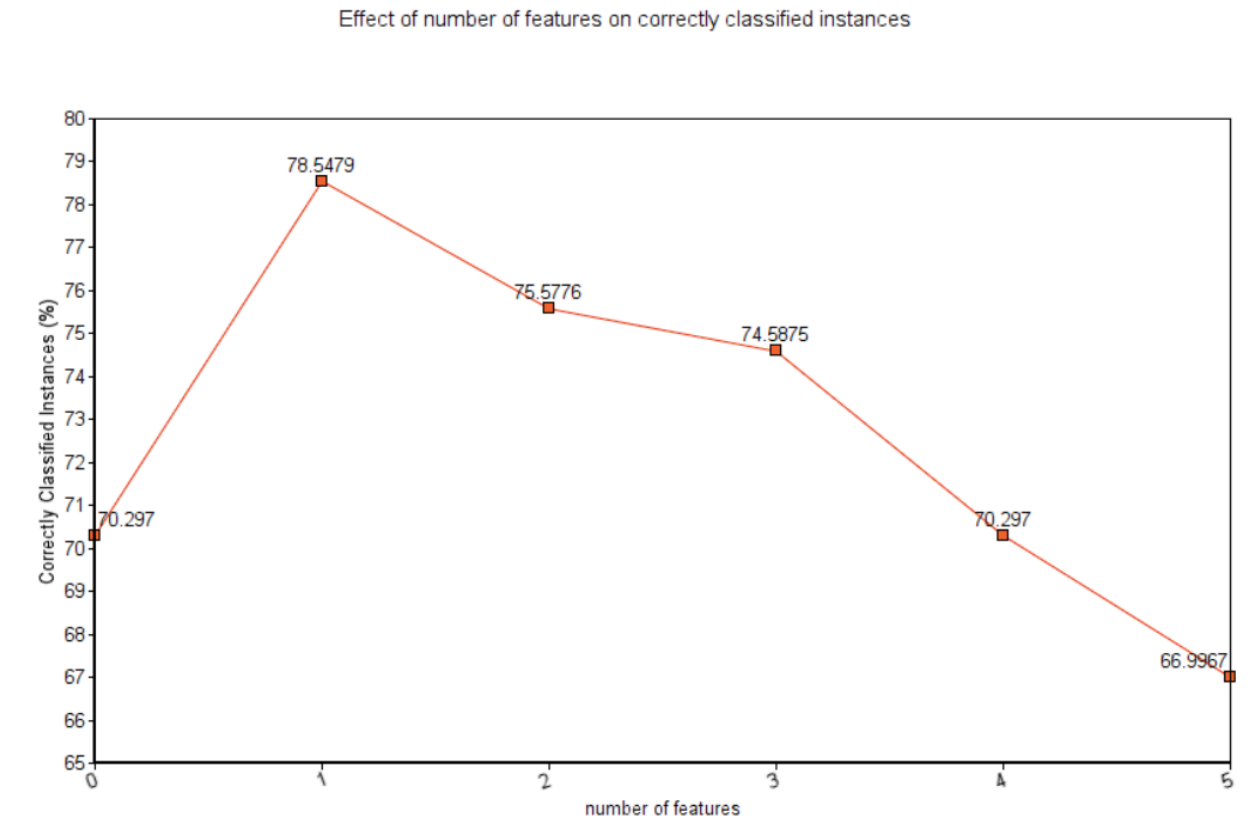
numIterations	Correctly Classified Instances
50	71.9472%
75	69.967%
100	70.297%
125	70.297%
150	70.6271%
175	71.6172%
200	71.2871%
225	70.297%



Graph 3: Effect of number of trees on correctly classified instances

- **numFeatures:** numFeatures represents the number of features to consider in random feature selection. If it is less than 1, it will use int(logM+1)). The best result was seen to be when the numFeature was set to value **1**, which was 78.5479% of correctly classified instances (see graph 4).

numFeatures	Int[$\log_2$(numFeature)+1]	Correctly Classified Instances
0	0.0	70.297%
1	1.0	78.5479%
2	1.58	75.5776%
3	2.0	74.5875%
4	2.32	70.297%
5	2.58	66.9967%



Graph 4: Effect of number of features on correctly classified instances

Neural Network

Neural networks are a set of algorithms designed to recognize patterns by using clustering and classification techniques in multiple layers.

Hidden Layers with Number of Neurons	Error per Epoch	Correctly Classified Instances
2, 4, 6, 8	0.2498893	49.1803 %
2, 10, 20, 30	0.249893	49.1803 %
20, 40, 60, 80	0.0537698	86.8852 %
10, 50, 100, 150	0.249895	49.1803 %
0, 50, 100, 150	0.4999998	50.8197 %

The number of hidden layers used for this analysis were 4. Most of the results were quite similar with different number of neurons in the hidden layers. The correctly classified instances were mostly seen to be around 49.1803%, but the best result was 86.8852% when the number of neurons were set to 20, 40, 60, 80 in 4 hidden layers respectively. It was seen that having either too little or too much neurons resulted a lower percentage of correctly classified instances. In figure 5 the green labels on the left is the input layer, the red cells in the middle shows the neurons in the hidden layer and the yellow cell on the tight is the output layer.

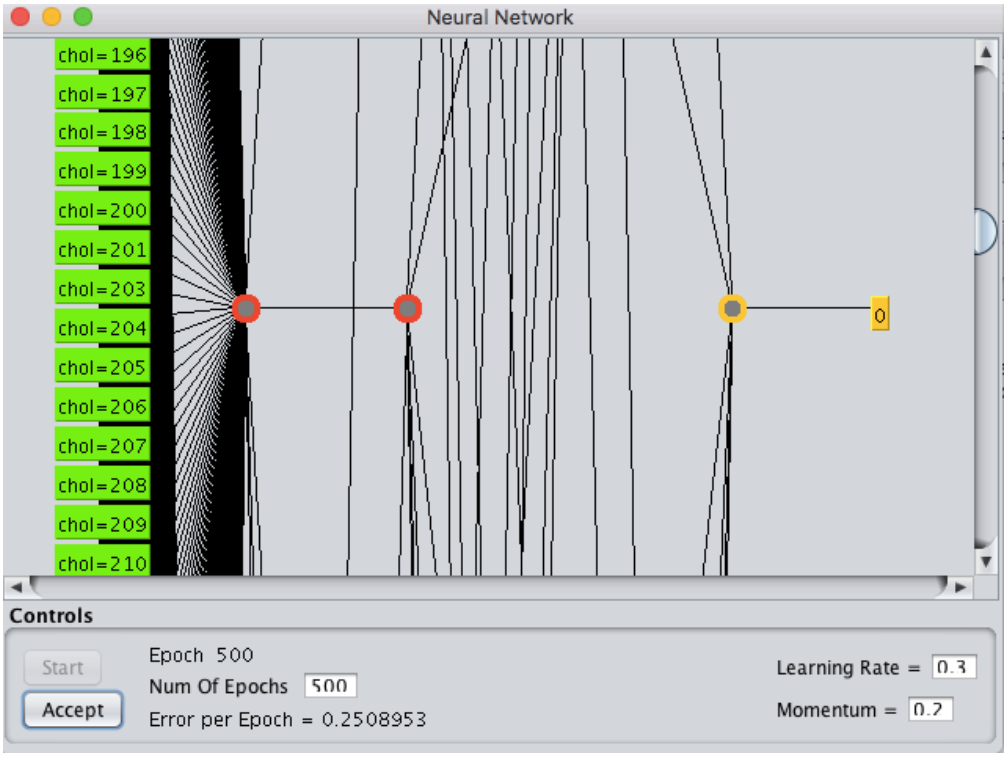
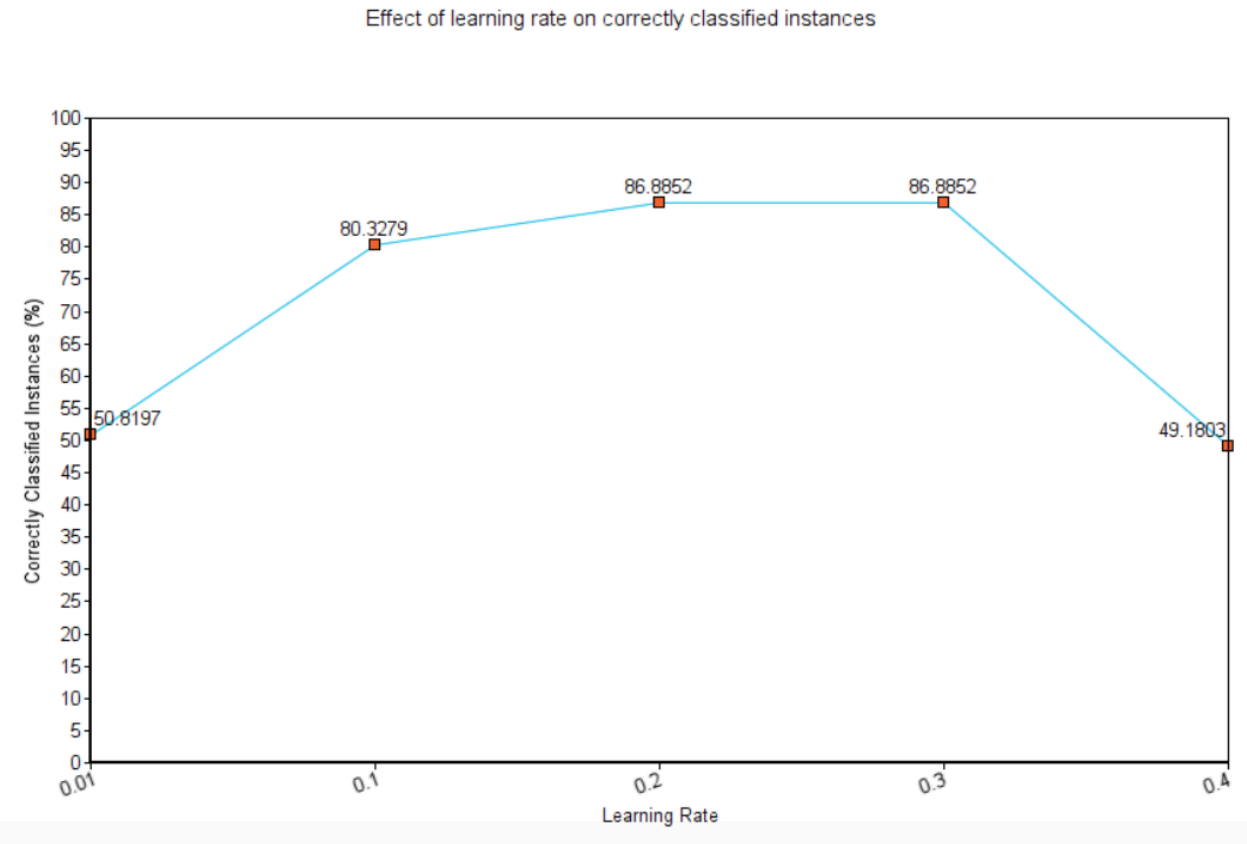


Figure 5: Neural Network Result with layers 2, 5, 6

The analysis was also done by updating the learning rate of the algorithm. Learning rate is the amount that the weights are updated during training. The learning rate was updated with the same hidden layers and the number of neurons that gave the best results (20, 40, 60, 80). The best result was seen to be with either 0.2 or 0.3 learning rate with correctly classified instances at 86.8852%. After 0.3, the values were seen to be constant (see graph 5).

Learning Rate	Error per Epoch	Correctly Classified Instances
0.01	0.2491133	50.8197 %
0.1	0.0084251	80.3279 %
0.2	0.0366191	86.8852 %
0.3	0.0537698	86.8852 %
0.4	0.252667	49.1803 %



Graph 5: Effect of learning rate on correctly classified instances

Conclusion

After doing a thorough analysis of the Cleveland dataset, the best parameters to get the best results for various methods were as follows:

- Decision Tree
  - Cross Validation – 5 folds
  - Percentage Split – 80% training and 20% validation
- Random Forest
  - Bag Size – 60%
  - Number of Trees – 50
- Neural Network
  - 4 Hidden Layers – 20, 40, 60, 80
  - Learning Rate – 0.2 or 0.3

Section 2

This section includes analysis of decision trees, random forests, and neural networks with **Student Academic Performance.** The dataset describes students' academic performance and was retrieved from https://www.kaggle.com/aljarah/xAPI-Edu-Data/data#. This dataset contains labels like gender, nationality, grade, topic, student absence, etc. Such data can help figuring out what factors affect students' academic performance the most. This could help in understanding and implementing rules or make changes in the education system to get a better result from students.

Decision Tree

The split criterion used was information gain. Reduced error pruning was used to reduce the complexity and improve predictive accuracy. The decision tree formed consisted of 24 leaves and its size was 35. Figure 6 shows the decision tree formed with Student Academic Performance dataset.

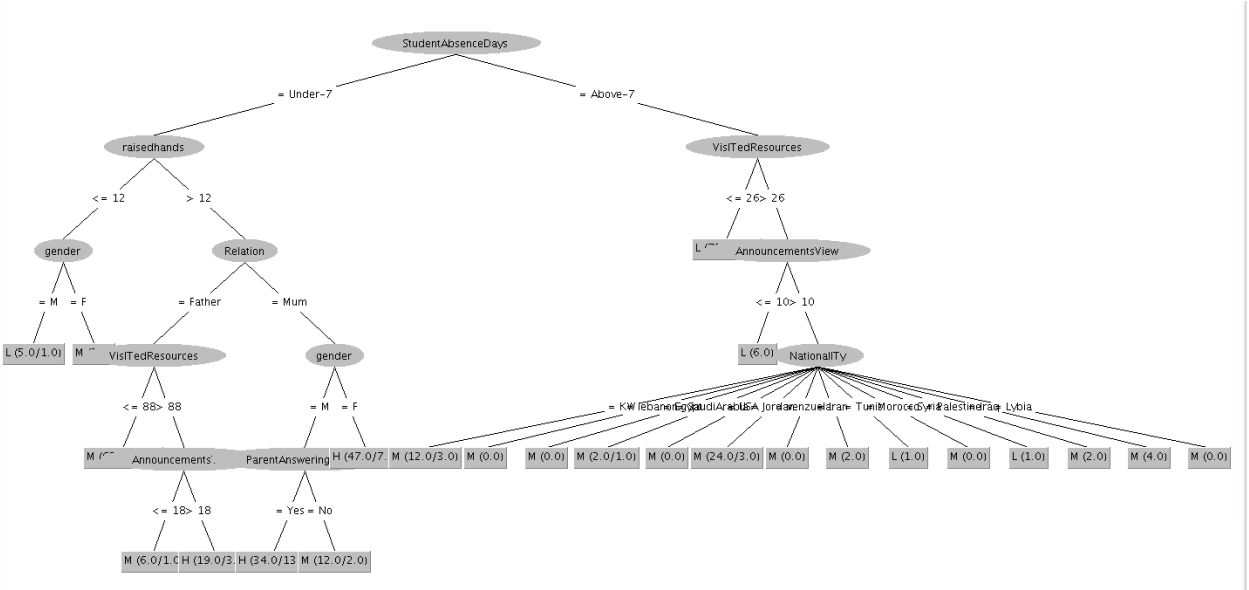


Figure 6: Decision Tree for Academic Dataset

The analysis was performed using the following test options:

- Cross Validation
- Percentage Split

Cross Validation

First it was tested on **10 folds** with the following results (see fig 7):

- Correctly Classified Instances: 70.2083%
- Incorrectly Classified Instances: 29.7917%
- Mean Absolute Error: 0.2407
- Total Number of Instances: 480

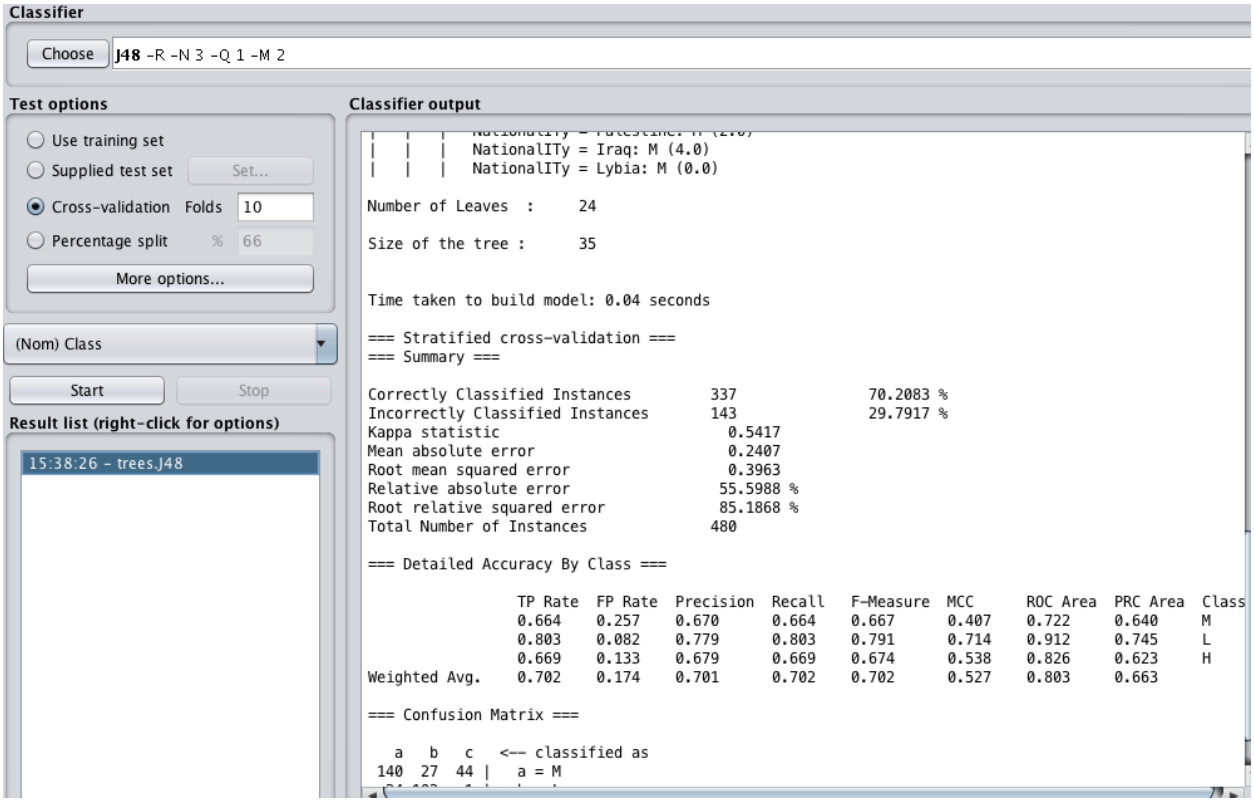


Figure 7: Cross validation result with 10 folds

It was then tested on **5 folds** and the result are as follows (see fig 8):

- Correctly Classified Instances: 71.4583 %
- Incorrectly Classified Instances: 28.5417 %
- Mean Absolute Error: 0.2412
- Total Number of Instances: 480

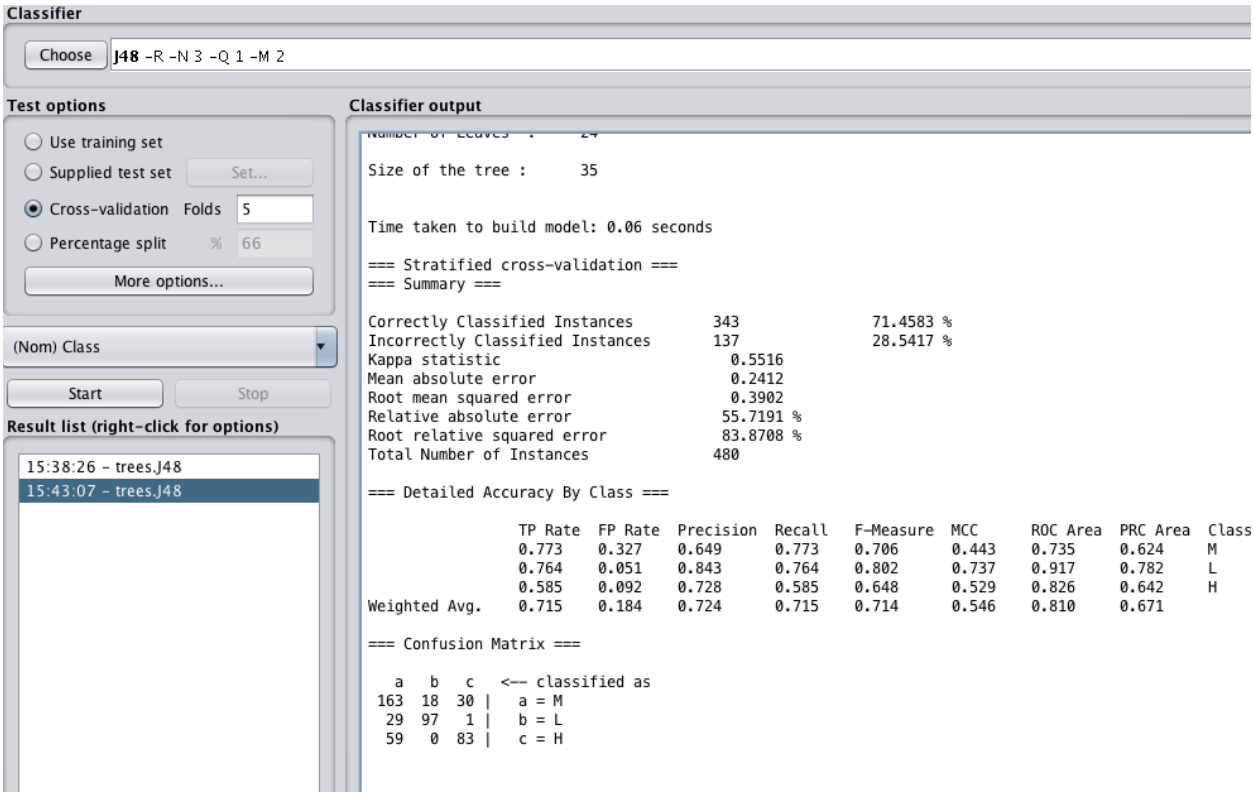


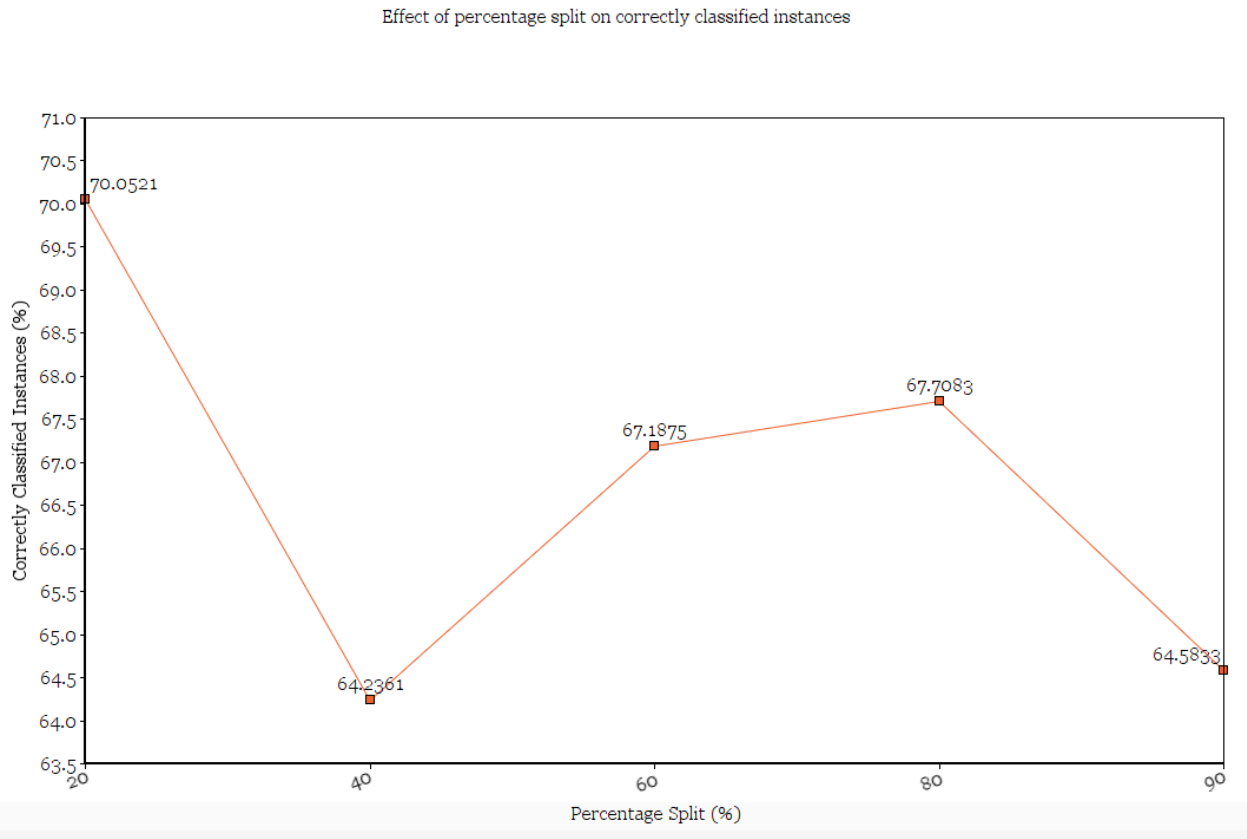
Figure 8: Cross validation result with 5 folds

The correctly classified instances were seen to be a little better with **5 folds**. This is because the test results are seen to be best with 80% training set and 20% validation set. This provides enough data for the algorithm to be trained quite well. With 5 folds, the algorithm is trained on 4/5 = 80% of the data and is tested on the remaining 1/5 = 20% of the data. With 10 folds the algorithm is provided with 9/10 = 90% of the training set and 1/10 = 10% of the validation set. This does not provide the algorithm with enough validation data to calculate an accurate enough result.

Percentage Split

Following are the results with different percentage split results:

Percentage Split	Correctly Classified Instances	Mean Absolute Error	Total Number of Instances
20%	70.0521 %	0.26	384
40%	64.2361 %	0.2812	288
60%	67.1875 %	0.2535	192
80%	67.7083 %	0.2526	96
90%	64.5833 %	0.2779	48



Graph 6: Effect of percentage split on correctly classified instances

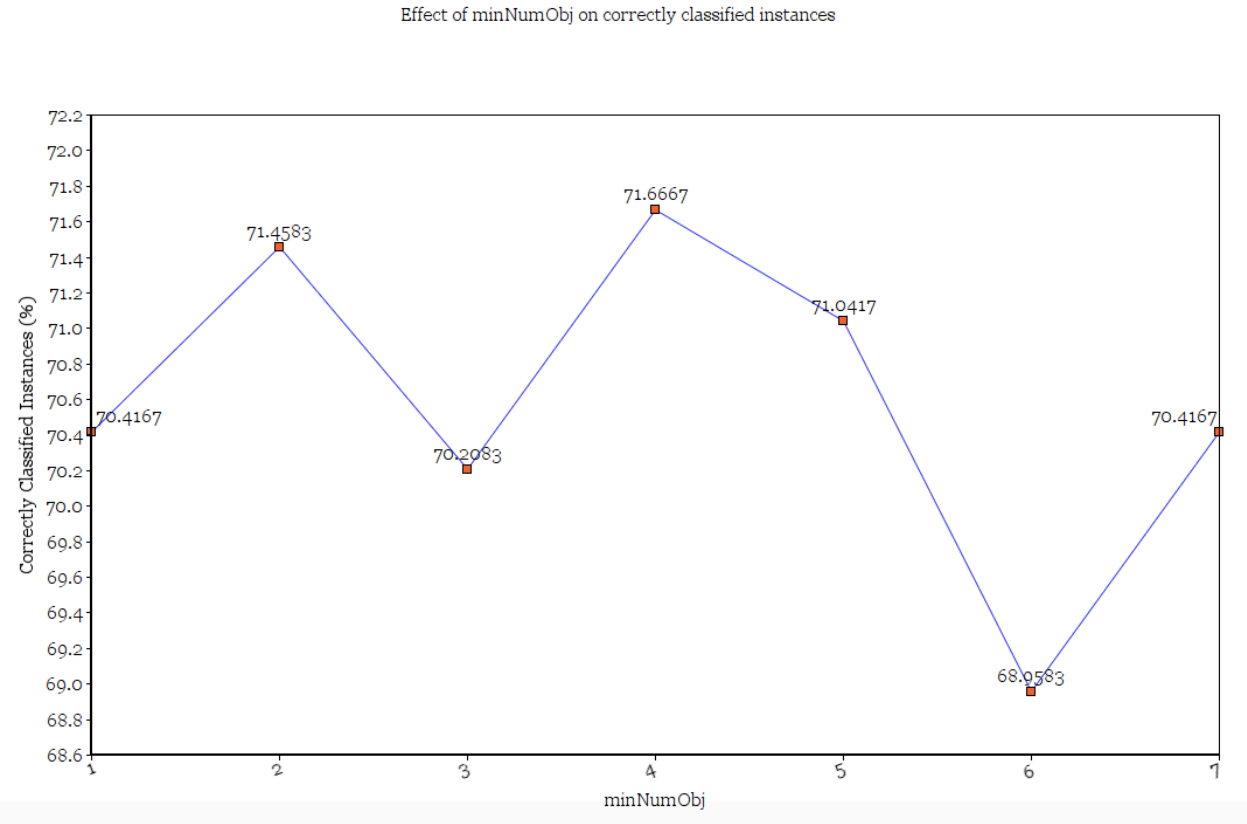
The best results were seen when the percentage split was set to 20%. The correctly classified instances were 70.0521%. This was because the algorithm enough data for both training and evaluation. 20% of the dataset was used for training and the rest 80% for evaluation. The lowest correctly classified instances appeared when the percentage split was set to 90% (see graph 6). This could be because the algorithm didn't get enough data for training or evaluation when it was split almost halfway.

The algorithm was also tested by updating the following parameters:

- **Confidence Factor:** The confidence factor represents a threshold of allowed inherent error in data while pruning the decision tree. It was seen that lowering the confidence factor resulted in more pruning.

- **minNumObj:** minNumObj should be higher for noisy data. The best correctly classified instances were seen to be with value **4** for minNumObj (see graph 7). This was because the Student Academic dataset is noisier than the Cleveland dataset for which the best results were seen with value 2. Table below shows the change in correctly classified instances with minNumObj.

minNumObj	Number of leaves	Size of the tree	Correctly Classified Instances
1	24	35	70.4167 %
2	24	35	71.4583 %
3	24	35	70.2083 %
4	24	35	71.6667 %
5	24	35	71.0417 %
6	23	33	68.9583 %
7	11	21	70.4167 %



Graph 7: Effect of minNumObj on correctly classified instances

Random Forest

Figure 9 shows the result running random forest with default values (bagSizePercent = 100 and batchSize = 100) with the following results:

- Correctly Classified Instances: 75%
- Incorrectly Classified Instances: 25%
- Mean Absolute Error: 0.2515
- Total Number of Instances: 480

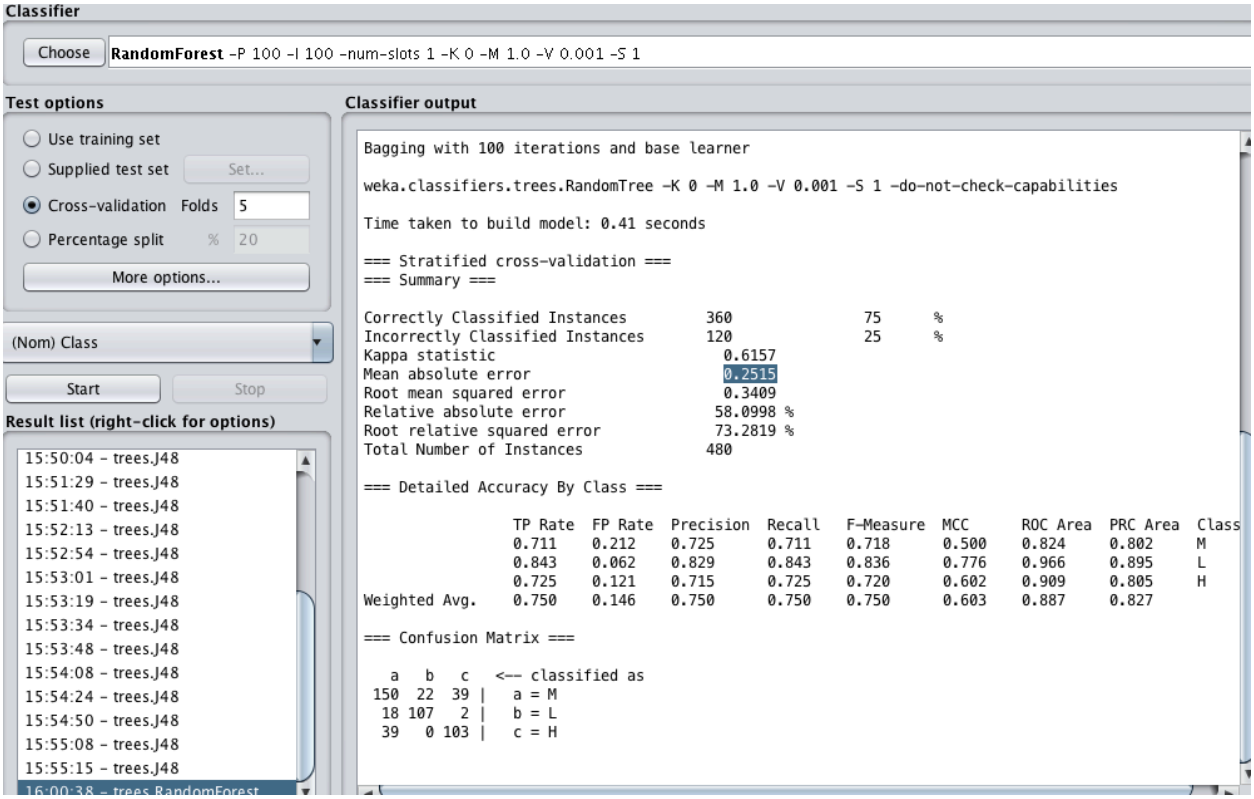
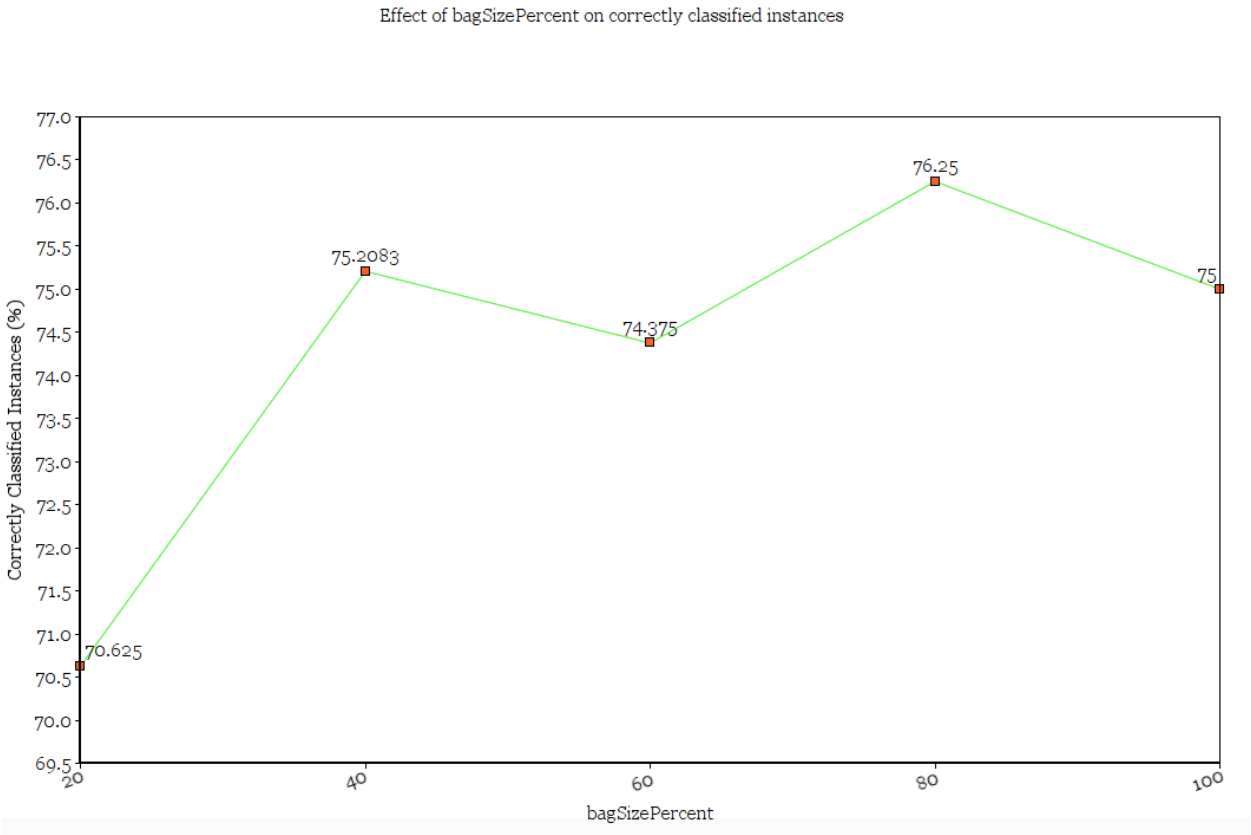


Figure 9: Results with default bagSizePercent and batchSize

The random forest results were then tested with different values of:

- **bagSizePercent:** It is the size of each bag as a percentage of the training set size. The out of bag data is used to get a running unbiased estimate of the classification error as trees are added to the forest. The best results were seen when the bag size was set to **80%** (see graph 8).

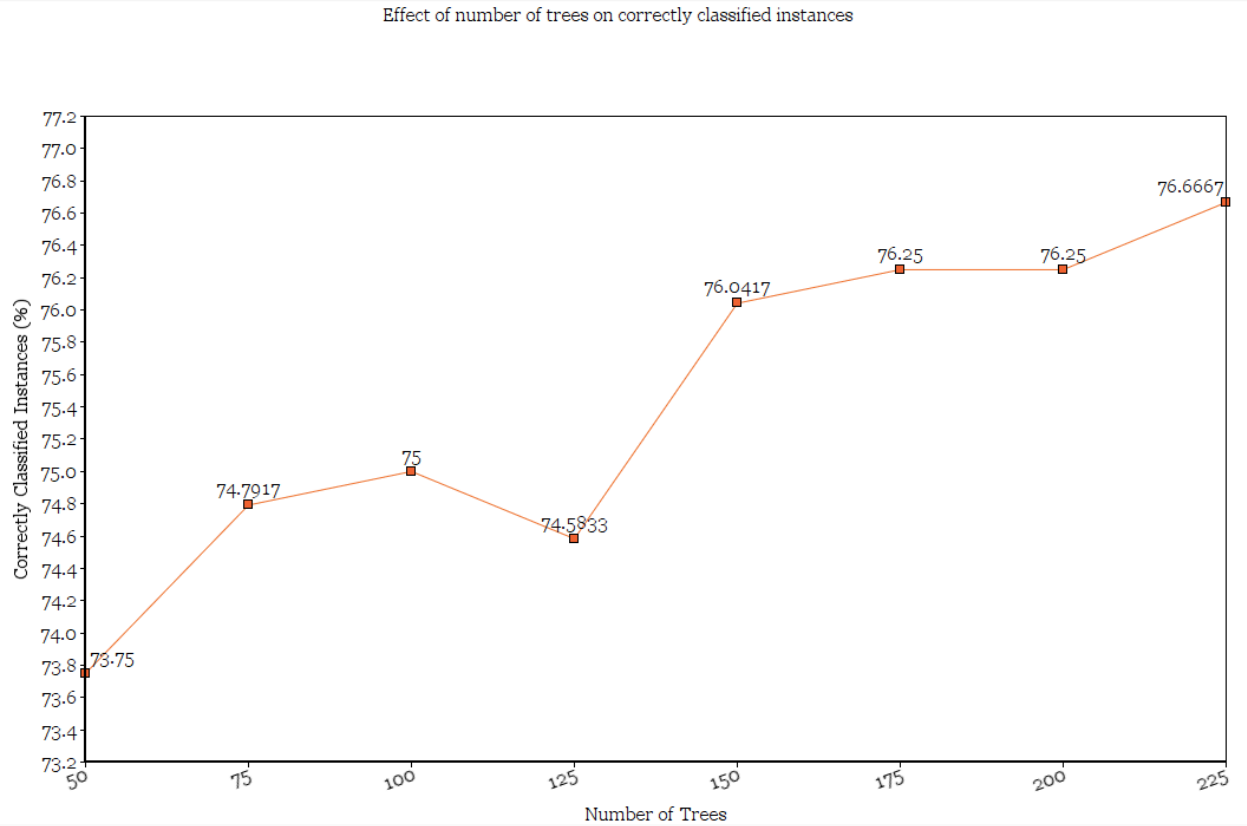
bagSizePercent	Correctly Classified Instances
20	70.625%
40	75.2083%
60	74.375%
80	76.25%
100	75%



Graph 8: Effect of bagSizePercent on correctly classified instances

- **numIterations:** The **number of trees** in the random forest. Updating number of trees did not have a huge change in the percentage of correctly classified instances. This could be because the dataset used was not that big. It was seen that increasing the number of trees resulted in a better accuracy. The best results were seen with **225** trees (see graph 9).

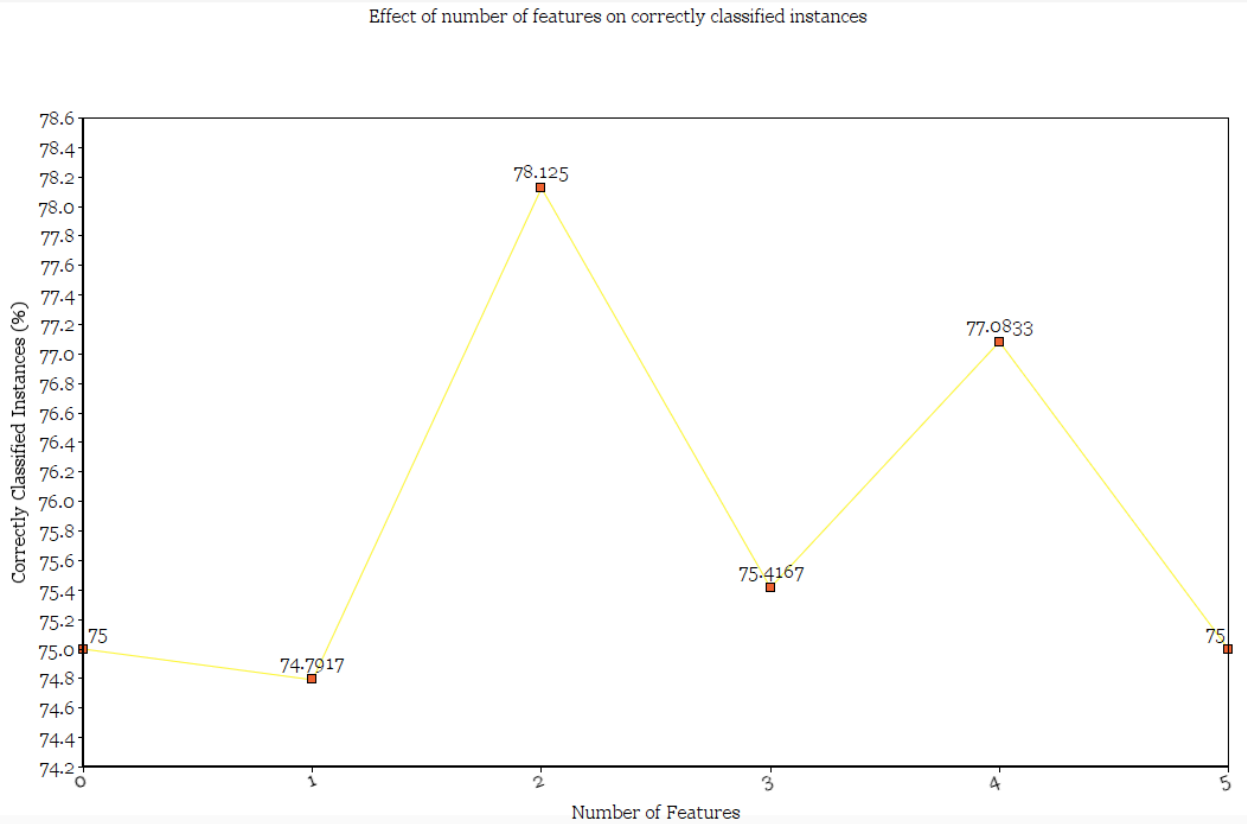
numIterations	Correctly Classified Instances
50	73.75%
75	74.7917%
100	75%
125	74.5833%
150	76.0417%
175	76.25%
200	76.25%
225	76.6667%



Graph 9: Effect of number of trees on correctly classified instances

- **numFeatures:** numFeatures represents the number of features to consider in random feature selection. If it is less than 1, it will use int(logM+1)). The best result was seen to be when the numFeature was set to value **2**, which was 78.125% of correctly classified instances (see graph 10).

numFeatures	Int[$\log_2$(numFeature)+1]	Correctly Classified Instances
0	0.0	75%
1	1.0	74.7917%
2	1.58	78.125%
3	2.0	75.4167%
4	2.32	77.0833%
5	2.58	75%



Graph 10: Effect of number of features on correctly classified instances

Neural Networks

The number of hidden layers used in this analysis were 3. For every run, the number of neurons were changed in the hidden layers and then the results were analyzed. The best results were seen with neurons equal to 5, 5, 5 or 10, 20, 30 in the three hidden layers respectively. The best correctly classified instances were 77.9167%. In figure 10 the green labels on the left is the input layer, the red cells in the middle shows the neurons in the hidden layer and the yellow cell on the tight is the output layer.

Hidden Layers with Number of Neurons	Error per Epoch	Correctly Classified Instances
5, 5, 5	0.0307521	77.9167%
5, 7, 10	0.027085	75.2083%
5, 10, 15	0.0326936	75.8333 %
10, 20, 30	0.0122451	77.9167%
20, 40, 60	0.0163038	74.1667%

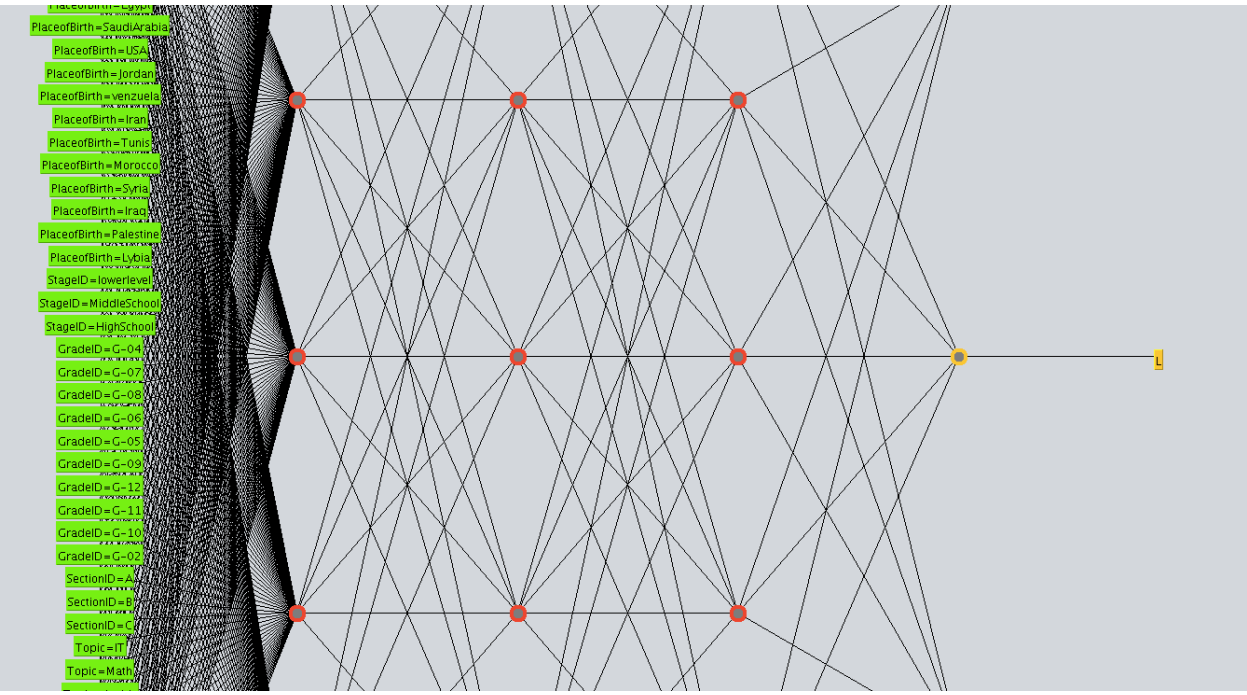
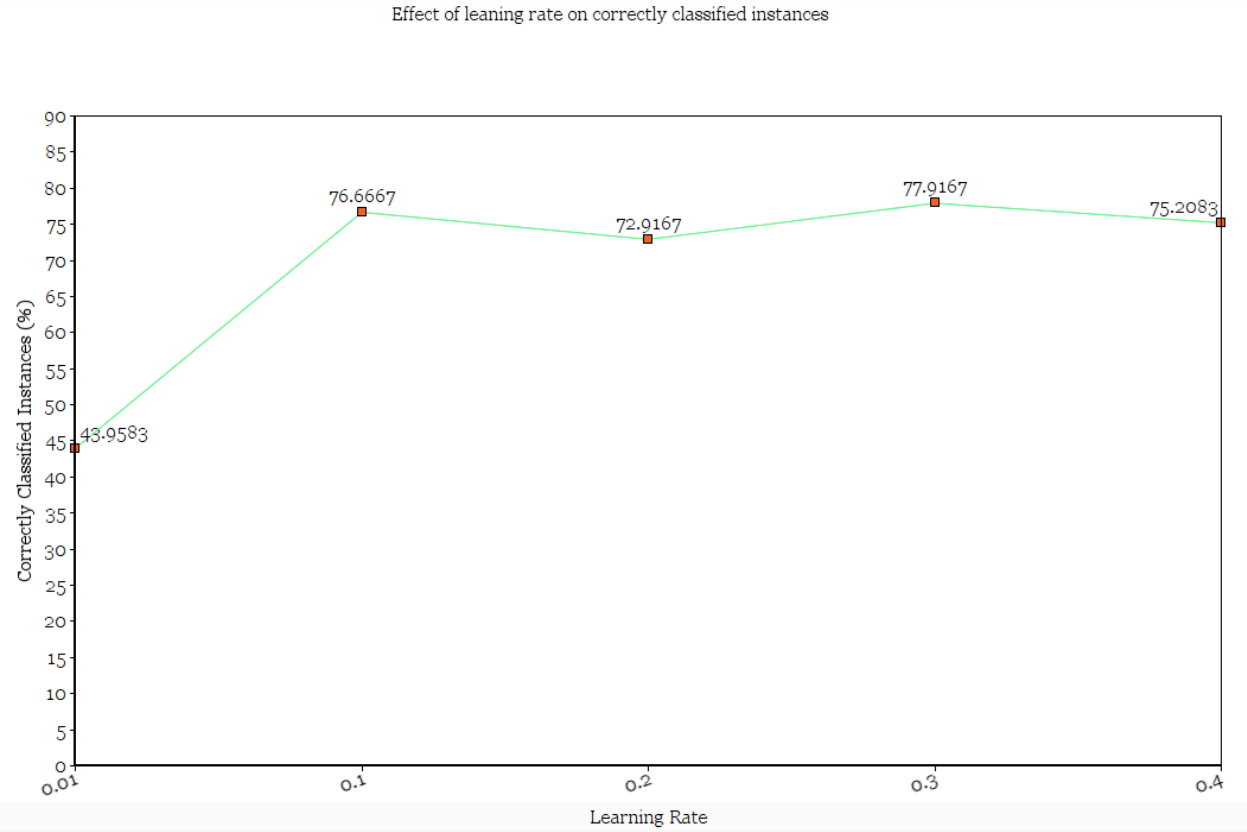


Figure 10: Neural Network Result with 3 hidden layers

The analysis was also done by updating the learning rate of the algorithm. Learning rate is the amount that the weights are updated during training. The learning rate was updated with the same hidden layers and the number of neurons that gave the best results (5, 5, 5). The best result was seen to be with 0.3 learning rate with correctly classified instances at 77.9167%. After 0.4, the values were seen to be decreasing (see graph 11).

Learning Rate	Error per Epoch	Correctly Classified Instances
0.01	0.2169045	43.9583 %
0.1	0.0253349	76.6667 %
0.2	0.0255594	72.9167 %
0.3	0.0307521	77.9167 %
0.4	0.0305524	75.2083 %



Graph 11: Effect of learning rate on correctly classified instances

Conclusion

After doing a thorough analysis of the Students' Academic Performance dataset, the best parameters to get the best results for various methods were as follows:

- Decision Tree
  - Cross Validation – 5 folds
  - Percentage Split – 20% training and 80% validation
- Random Forest
  - Bag Size – 80%
  - Number of Trees – 225
- Neural Network
  - 3 Hidden Layers – 5, 5, 5 or 10, 20, 30
  - Learning Rate – 0.1