Aayush Shrestha

Experiment 1 :

* What are the test options provided in Weka?
  + The test options for Weka are 'Use training set', 'Supplied test set', 'Cross-validation folds' and 'Percentage split'.
* What is the default test option? Briefly explain the default test option
  + Default test option is 'Cross-validation folds = 10'. With 10-fold cross-validation, Weka invokes the learning algorithm 11 times, one for each fold of the cross-validation and then a final time on the entire dataset.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Precision  (Exp1a) | Precision  (Exp1b) | Precision  (Exp1c) | Recall  (Exp1a) | Recall  (Exp1b) | Recall  (Exp1c) |
| k-NN classifier, k = 1 | 0.980 | 0.950 | 0.765 | 0.995 | 0.950 | 0.650 |
| k-NN classifier, k = 10 | 0.889 | 0.875 | 1.000 | 1.000 | 0.980 | 0.650 |
| Naïve Bayes classifier | 0.954 | 1.000 | ? | 0.930 | 0.610 | 0.000 |
| Decision Tree | 0.995 | 1.000 | 0.875 | 0.985 | 0.930 | 0.700 |

[1.1] For k-NN classifier, the distance weighting method allows neighbors closer to the data point to have more influence on the predicted value.

[1.2] For k-NN classifier the performance, k = 1, in terms of performance in respect to precision and recall, Exp1a > Exp1b > Exp1c

[1.3] For k-NN classifier the performance, k = 10, in terms of performance in respect to precision and recall, Exp1a > Exp1b > Exp1c. However, Exp1c being last when it came to recall, it was first in terms of precision. To get performance, I added up precision and recall then divided by 2.

[1.4] For Decision Tree classifier the performance, in terms of performance in respect to precision and recall, Exp1a > Exp1b > Exp1c. However, Exp1b had the best precision at 1.000. To get performance, I added up precision and recall then divided by 2.

[1.5] For Naïve Bayes classifier the performance, in terms of performance in respect to precision and recall, Exp1a > Exp1b. The performance of Exp1c is missing as recall is 0 and didn't output any precision values. As Exp1c is sparse as shown in figure 1c in the rectangular region. The precision for Exp1b > Exp1a.

[1.6] The observed differences are due to difference in method of classification. All 4 algorithm classify the data differently, hence we are seeing difference in performance.

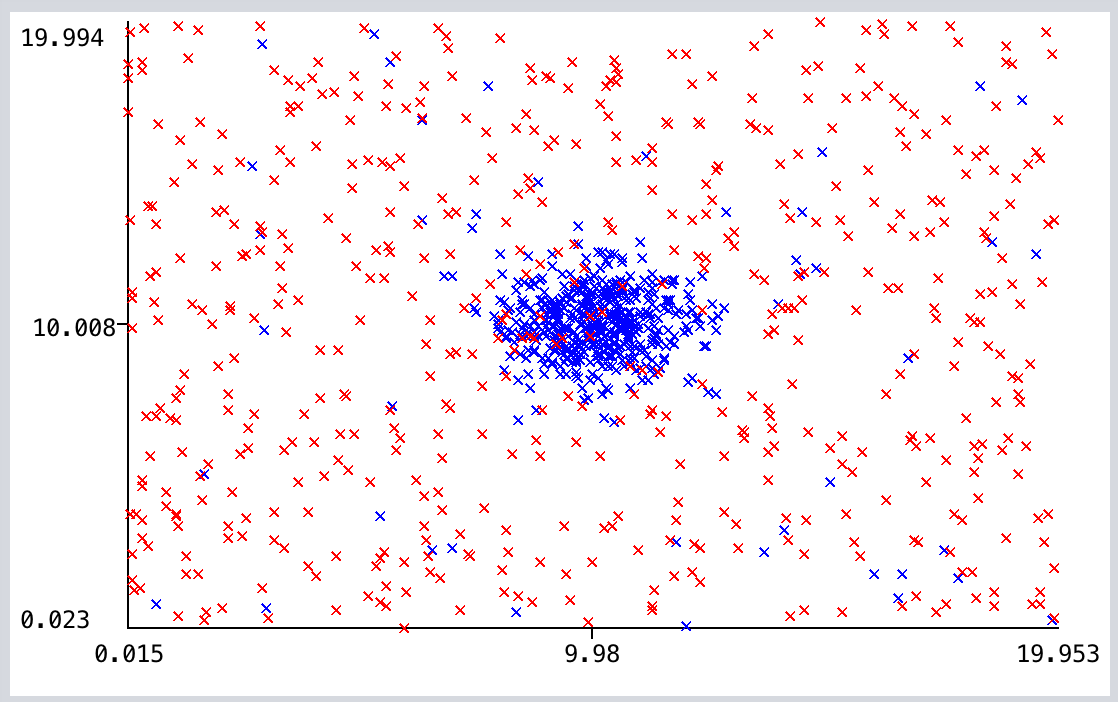
Experiment 2:

[2.1]

From the results, we observe the pattern that as minNumObj increased then the leaf nodes decreased. We also observed that as minNumObj increased so did training error %. However, test error % initially was high but it went down up until 16 and 32 of the minNumObj. After that it stated to increase. Initially the variability between test error and training error was large, but it got smaller as minNumObj increased. There was no change in training error and test error in 32 and 16 minNumObj.

|  |  |  |  |
| --- | --- | --- | --- |
| Minimum Number of Instances per leaf (**minNumObj**) | Number of leaves (leaf nodes) | Training error (%) | Test error (%) |
| 1 | 146 | 0 | 13.6523 |
| 2 | 116 | 3.9815 | 14.7428 |
| 4 | 90 | 6.7593 | 13.8477 |
| 8 | 62 | 8.3333 | 7.9115 |
| 16 | 38 | 9.0741 | 8.0658 |
| 32 | 24 | 9.0741 | 8.0658 |
| 64 | 15 | 10.8333 | 9.2798 |
| 128 | 6 | 15.7407 | 16.1728 |
| 256 | 4 | 16.3889 | 15.4218 |
| 512 | 2 | 45 | 47.4486 |

Plot Of error rates with respect to the numbers Of leaf nodes 
• «ror (96) • Test error 

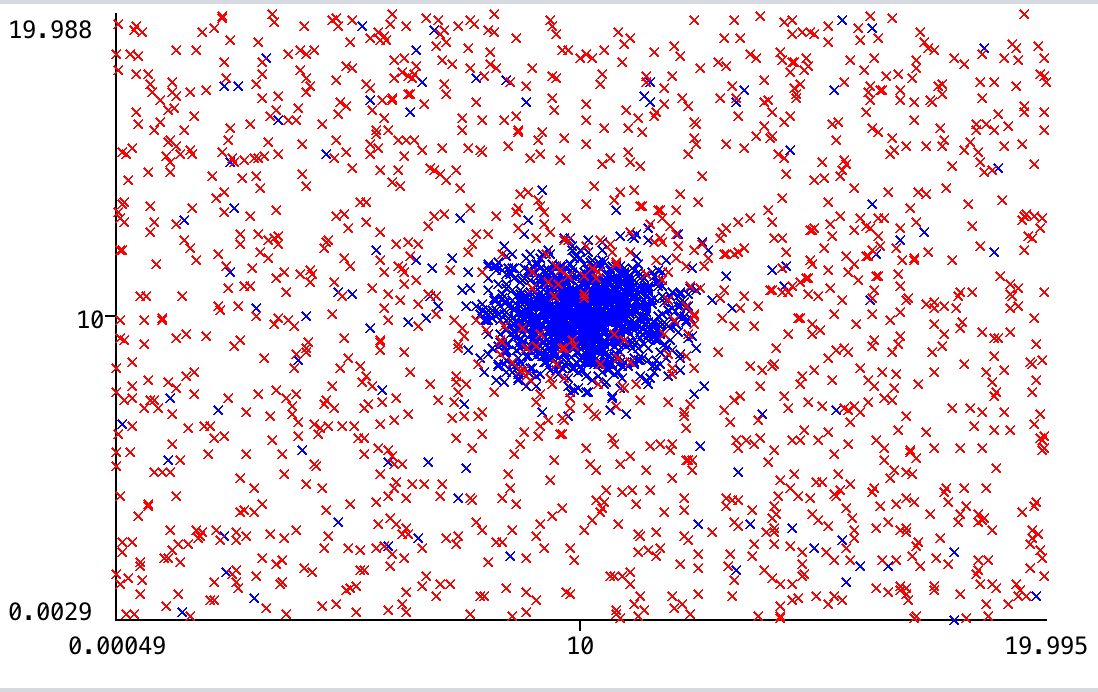


[2.2]

From the result, we observe that there isn't much variability between training error and test error for given minNumObj. Here, we observed that between 8 to 128, there weren't much fluctuation between training error and test error. However, we saw a bump in 256 and then another bump in 1080 minNumObj.

|  |  |  |  |
| --- | --- | --- | --- |
| Minimum Number of Instances per leaf (**minNumObj**) | Number of leaves (leaf nodes) | Training error (%) | Test error (%) |
| 8 | 141 | 7.037 | 8.2222 |
| 16 | 97 | 7.4074 | 8.2716 |
| 32 | 61 | 7.8148 | 8.1852 |
| 64 | 32 | 7.8148 | 8.4198 |
| 128 | 18 | 8 | 8.4074 |
| 256 | 9 | 14.2593 | 14.1481 |
| 512 | 5 | 15.0741 | 16.963 |
| 1080 | 2 | 40.6667 | 42.4938 |

Plot of error rates with respect to the numbers of leaf nodes 
• Tr error (S) • Test error (X) 



[2.3]

Between the two plots, 2.1 and 2.2, we see the difference that variation between test and train errors are less in 2.2. Also, 2.2 has less training error % and test error % corresponding to the respective minNumObj.

The plot in 2.b is more densely populated than that of 2.a. The increase of approximately in 40 range happens to 2.1 in 512 minNumObj, whereas for 2.2 it happens in 1080.

Experiment 3:

[3.1] K-NN classifier with k = 3 has the best performance with precision of 1.0 and recall of 1.0. Then it is decision tree with high precision of test of 0.991 precision and 0.957 test.

Collective attributes as shown in the graph is clustered in 4 corners of the graph, with similar attributes.

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). That's why it has higher performance.

K-NN object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors. Since attributes are clustered in 4 corners of the graph, this gives us high performance.

Bayes classifier works based on probability, as it calculates probability from 4 corners, it gets average in the middle which wouldn't be on either of the quadrant of the attribute which is like guessing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision  (Training) | Precision  (Test) | Recall  (Training) | Recall  (Test) |
| k-NN classifier, k = 3 | 1.000 | 1.000 | 1.000 | 1.000 |
| Naïve Bayes classifier | 0.500 | 0.496 | 0.487 | 0.480 |
| Decision Tree | 0.993 | 0.991 | 0.987 | 0.957 |

[3.2] Here we observe that with decrease in node, the training error and test error goes up. We also observe that with minNumObj increase there is decrease in number of leaf nodes.

Table 3.

|  |  |  |  |
| --- | --- | --- | --- |
| Minimum Number of Instances per leaf (**minNumObj**) | Number of leaves (leaf nodes) | Training error (%) | Test error (%) |
| 1 | 10 | 0 | 1.4286 |
| 2 | 9 | 0.3333 | 1.4286 |
| 32 | 7 | 13 | 15.5714 |
| 50 | 5 | 20.6667 | 24.4286 |
| 70 | 3 | 33.3333 | 40.1429 |

[3.3] When it came to training model Decision tree had the best performance but when testing k-NN has the best performance. Decision tree in terms of test has close to random performance as it is closest to .5 in respect of precision and recall of test set.

Decision tree and k-NN both showed big drop of performance compared to data 1. It’s because the data is different, and it’s not clustered like data 1 in 4 quadrants.

Naïve Bayes classifier performs same as it did in data 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision  (Training) | Precision  (Test) | Recall  (Training) | Recall  (Test) |
| k-NN classifier, k = 3 | 0.778 | 0.554 | 0.840 | 0.631 |
| Naïve Bayes classifier | 0.763 | 0.474 | 0.793 | 0.443 |
| Decision Tree | 0.862 | 0.485 | 0.793 | 0.449 |

[3.4] Here, as number of leaf nodes decreased the training error increased as well. However, the test error fluctuated between 47 to 52, without any distinct pattern of increasing or decreasing.

|  |  |  |  |
| --- | --- | --- | --- |
| Minimum Number of Instances per leaf (**minNumObj**) | Number of leaves (leaf nodes) | Training error (%) | Test error (%) |
| 1 | 48 | 0 | 47.2857 |
| 2 | 46 | 2.6667 | 47.2857 |
| 4 | 37 | 10.6667 | 51.1429 |
| 8 | 27 | 11.3333 | 48.8571 |
| 16 | 15 | 21 | 51.5714 |
| 32 | 7 | 29.3333 | 52 |
| 50 | 5 | 32.3333 | 51 |
| 70 | 4 | 38.3333 | 47.5714 |