COMP 309 Assignment 1

Vincent Yu
300390526

Dataset using : Spect

Result：（by using weka and keel）

Naive Bayes

Tool: Weka(10 folds cross-validation)

=== Summary ===

Correctly Classified Instances	211	79.0262 %
Incorrectly Classified Instances	56	20.9738 %
Kappa statistic	0.4666	
Mean absolute error	0.2299	
Root mean squared error	0.4217	
Relative absolute error	69.9562 %	
Root relative squared error	104.2572 %	
Total Number of Instances	267	

== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.764	0.203	0.494	0.764	0.600	0.487	0.845	0.597	0
	0.797	0.236	0.929	0.797	0.858	0.487	0.845	0.950	1
Weighted Avg.	0.790	0.229	0.839	0.790	0.805	0.487	0.845	0.878	

=== Confusion Matrix ===

```
  a   b   <-- classified as
 42  13 |   a = 0
 43 169 |   b = 1
```

Multilayer Perceptron(Neural Network)

Tool: Weka(10 folds cross-validation)

=== Summary ===

Correctly Classified Instances	210	78.6517 %
Incorrectly Classified Instances	57	21.3483 %
Kappa statistic	0.343	
Mean absolute error	0.2153	
Root mean squared error	0.3977	
Relative absolute error	65.5212 %	
Root relative squared error	98.3225 %	
Total Number of Instances	267	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.473	0.132	0.481	0.473	0.477	0.343	0.799	0.444	0

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.868	0.527	0.864	0.868	0.866	0.343	0.799	0.937	1
Weighted Avg.0.787	0.446	0.785	0.787	0.786	0.343	0.799	0.835	

=== Confusion Matrix ===

a b <-- classified as

26 29 | a = 0

28 184 | b = 1

K-Nearest Neighbour

Tool: Weka(IBK--10 folds cross-validation)

=== Summary ===

Correctly Classified Instances	201	75.2809 %
Incorrectly Classified Instances	66	24.7191 %
Kappa statistic	0.3094	
Mean absolute error	0.2452	
Root mean squared error	0.4154	
Relative absolute error	74.6216 %	
Root relative squared error	102.6943 %	
Total Number of Instances	267	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.527	0.189	0.420	0.527	0.468	0.313	0.733	0.388	0
	0.811	0.473	0.869	0.811	0.839	0.313	0.733	0.893	1
Weighted Avg.	0.753	0.414	0.776	0.753	0.763	0.313	0.733	0.789	

=== Confusion Matrix ===

a b <-- classified as

29 26 | a = 0

40 172 | b = 1

Decision tree (J48)

Tool: Weka(10 folds cross-validation)

=== Summary ===

Correctly Classified Instances	216	80.8989 %
Incorrectly Classified Instances	51	19.1011 %
Kappa statistic	0.3957	
Mean absolute error	0.2422	
Root mean squared error	0.3724	
Relative absolute error	73.6951 %	
Root relative squared error	92.0706 %	
Total Number of Instances	267	

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

0.491 0.108 0.540 0.491 0.514 0.396 0.781 0.477 0
0.892 0.509 0.871 0.892 0.881 0.396 0.781 0.910 1
Weighted Avg.0.809 0.427 0.803 0.809 0.806 0.396 0.781 0.821

=== Confusion Matrix ===
a b <-- classified as
27 28 | a = 0
23 189 | b = 1

Evolutionaries GP-C (Genetic programming)

TEST RESULTS
============
Classifier= .a/comp309fixedaset/comp309fixedaset
Fold 0 : CORRECT=0.6716417910447761 N/C=0.0
Fold 1 : CORRECT=0.7293233082706767 N/C=0.0
Global Classification Error + N/C:
0.2995174503422736
stddev Global Classification Error + N/C:
0.028840758612950042
Correctly classified:
0.7004825496577264
Global N/C:
0.0

	Naive Bayes	**Multilayer Perceptron**	**KNN**	**Decision Tree**	**Genetic Programming**
Correct	**79.0262 %**	**78.6517 %**	**75.2809 %**	**80.8989%**	**70.04%**
Incorrect	**20.9738 %**	**21.3483 %**	**24.7191 %**	**19.1011 %**	**29.96%**

Methodology & Result

For part1 I used both Weka and Keel to analyzed the data. From the table above, we can easily observe Decision Tree provided the highest correctly classified instances is 80.8989% And Genetic Programming has the poorest performance among the five techniques. But I changed to 2-folds cross-validation instead of the default 10 folds. Because when I run keel with GP under ten folds it takes a very long time and it doesn’t give me the result properly.

KNN (K-Nearest Neighbors:

Description, Representation & evaluation method:

KNN is in the supervised learning family of algorithms. It is represented by support vectors. The KNN algorithm is a robust and versatile classifier that is often used as a benchmark for more complex classifiers such as Artificial Neural Networks (ANN) and Support Vector Machines (SVM). It is one of the supervised learning technique. It belongs to Analogizers because it is instance-based learning, the algorithm doesn't learn a model but it chooses to memorize the instances in the training set. And uses this memory for the prediction phase.
Here is one of the popular choice-- Euclidean distance is given by :

$$d(x, x') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \ldots + (x_n - x'_n)^2}$$

Optimization:

For KNN, it is important to pick a suitable value for K. When K is small, says 1 we are restraining, and our classifier cannot consider the overall distribution. On the other hand, it will provide the most flexible fit, which will provide low bias but a very large variance.

Relate to the dataset:

First time I run with KNN I chose K=1 and I got accuracy about 75%, which I think is not good enough. And I tried several K values in order to find the highest accuracy. Then it turned out K=3,5 can provide the best performance at 80.1498%. This makes sense when we using higher K, it averages more voters in each prediction and hence is more resilient to outliers. And this can slightly improve my performance for the data I was using. But it didn't provide a significant improvement, I think it is due to the dataset I was using is nominal(Binary) and there is only a small amount of outliers.

More thinking on KNN:

I think this technique is more useful for data with a lot of outliers comparing to other techniques because it can effectively limit a variance range which can significantly improve the performance for a dataset with lots of outliers.

Naive Bayes :

Description, Representation :

Naive Bayes is one of the techniques from the Bayesian tribe. It is a very simple probability-based technique. When we import the data and analyze with naive Bayes it will compute P(class| instance data) for each class, and it will choose the class with the highest probability. And the Bayes rule is showing below.

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

One important thing for naive bayes is it requires features are independent given class**.**
Bayesians can be represented by Graphical Models.

Evaluation method & Optimization:

Bayesian can be evaluated by posterior probability, the higher the posterior probability we get the better performance it is. Because the function is unknown, for bayesian it will generate a random function. As we import the training set it will take the evaluations, which are treated as data, the initial function is updated to form the posterior distribution over the objective distribution. Then the posterior distribution will be used to find the next query point.

Decision Tree:

Description, Representation:

Decision tree is one of the techniques widely used in data mining. It is a classifier belongs to Symbolists. And it is easy to interpret and it has good performance on categorical features. Decision tree is constructed with a root node and decision nodes, at the bottom is leaf nodes. And the leaf nodes are the class nodes. Decision tree contains Classification and Regression Trees, but for the dataset, I was using, it was using Regression Tree.

Evaluation & optimization :

For evaluation in decision tree, it is usually based on Gini Impurity. It separates the subtree by choosing the minimum impurity and to get less impurity in the subtrees. The formula of Gini impurity is shown below.

- Gini impurity: $2P(A)P(B) = 2\frac{m}{m+n} \times \frac{n}{m+n} = \frac{2mn}{(m+n)^2}$

It aims to average the impurity of child nodes. In order to have better performance, it adds weighting the impurities by the probability of nodes.

It is using Pruning to optimize the performance. Pruning is the inverse of splitting. Let's say using DT, it separates too many branches which end with a huge tree. It becomes too large and complex and it may be overfitting. It will slow down the process, and in the real world, it will cost more time to analyze.Overfitting may lead to having a bad accuracy when we processing the test sets. So we need to a tradeoff between with tree complexity and accuracy. As a result, we need to use Pruning.

Relate to data

As I mentioned before, DT has a good performance on categorical features. Fortunately, the dataset I was using is a categorical feature. As a result, it performs very well.

Multilayer Perceptron:

Description & Representation:

A multilayer perceptron is a feedforward artificial neural network which generates a set of outputs from a set of input. It can generate multiple layers in a directed graph. As we mentioned in class it is using backpropagation for construction of the network.

Evaluation & Optimization

Stopping criteria is using in MLP when a certain number of epochs is reached or the error on the training set is smaller than a threshold. And the proportion of correctly classified accuracy is larger than a threshold. It uses validation control to avoid overfitting.

Relate to data:

As shown above, the accuracy for MLP is around 78%, and in the next part after preprocessing the data, the accuracy slightly increased.

Genetic Programming:

Description & Representation

This technique belongs to the tribe of EVOLUTIONARIES. It is a model of programming uses the ideas of biological evolution to handle a complex problem. For each generation, it chooses the trials which have the lowest error to the output and use these as the models of the next generation. In the next generation, it will evolve again by making small changes to the one are chosen from the previous generation. A function set consists of a set of functions or operators. It is represented by tree structures.

Evaluation & Optimization

It has three operators in GP.

Reproduction :

-- Simply copy a selected program from the current generation to the new generation

-- allow good programs to survive

--Elitism

1. Mutation:

-- Operate on a single selected program.

-- Remove a random subtree of the program,

-- Put a new subtree in the same place

-- Use a program generation method to generate the new subtree

3. Crossover:

-- Swap a subtree of one parent with a subtree of the other

-- Put the two newly formed programs into next generation (From 307 Slide)

Reason of technique are different in relation to the dataset:

The dataset spect I was using is a binary dataset (Categorical)

1. DT :

Among these five techniques, DT has the best accuracy. The binary dataset is very suitable for DT.

2.KNN:

Although it showed a high accuracy with my data, I think it is not suitable. Comparing to binary sets, numerical suits KNN much better.

3.GP
The data I was using is not very suitable for GP, as a result, the accuracy is not very good.

4.Naive Bayes:
As mentioned before, Naive Bayes only evaluate based on probability. And the features are independent to the class, as a result, it provides a medium good accuracy. After some preprocessing the accuracy reach a higher level.

5.MLP:
The accuracy of using MLP is around 78.6%. I think this technique is also suited to my dataset.

Optional:
The first reason of my dataset can achieve high accuracy is that my dataset has a good quality.
We always try to find the best algorithm, but for now there is no such algorithm can handle all the data and achieve a high performance. But for different dataset we can use different techniques that suits to the using dataset. For my dataset, it has a good quality and relevantly easy.
As mentioned before, binary dataset can be well performanced by several techniques. Different techniques using different model, and it will generate different algorithm or different way to classify. These results after well training may all can well handle the problems.
Therefore, I think in order to have a accurate classifier, we should first well collect good data. And we should choose a suitable technique. After that it needs to be well trained.

Part2 :

Cross Validation:

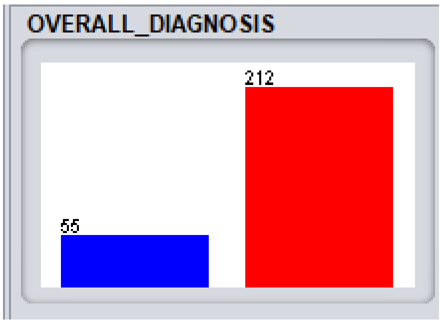
As mentioned before i used 10-fold cross validation. It is a useful technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it. 10-fold cross validation can randomly partitioned, and use ten data as test set and rest as training set. And it will repeat 10 times. The advantage of using this technique is all the observations are used both for training and test. And all the observation can only be used once as test.

1 understanding of the dataset:

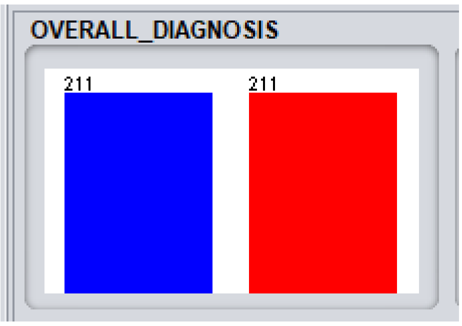
The dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the patients is classified into two categories: normal and abnormal. The database of 267 SPECT image sets (patients) was processed to extract features that summarize the original SPECT images. As a result, 44 continuous feature pattern was created for each patient.The pattern was further processed to obtain 22 binary feature patterns.
The dataset I was using has good quality, there has no missing value and no outliers. But the only problem is the dataset is not well balanced.

2.Preprocessing needed:
In order to have a better performance, I did preprocessing on the dataset with the assistance of filters in Weka. I used “resample” filter which is in the supervised filter class.

Before Resampling :



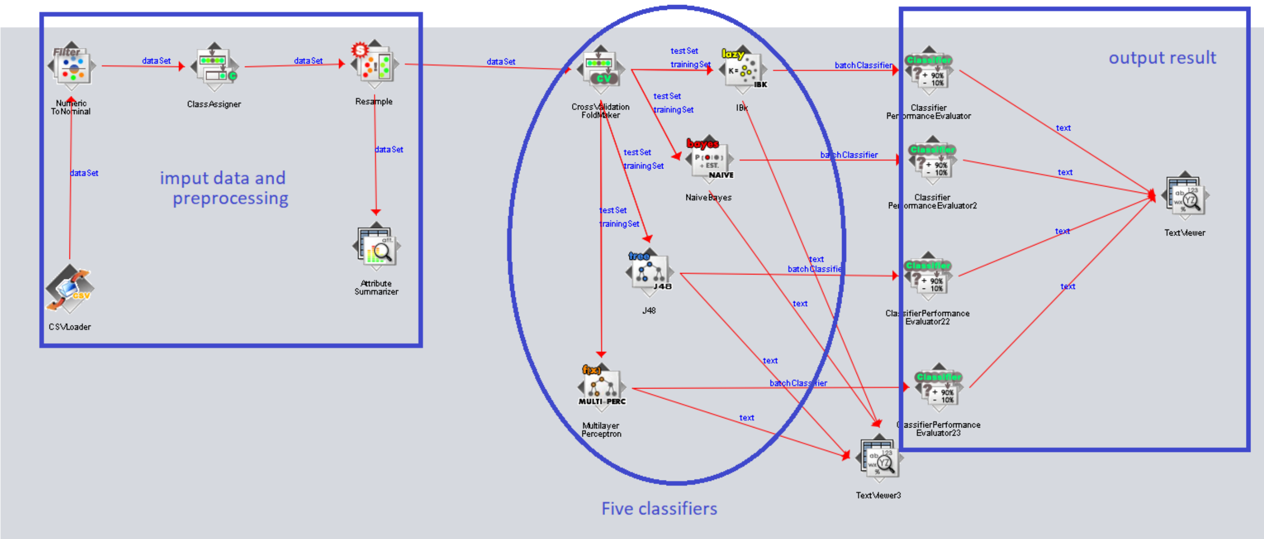
After resampling:



After resampling, the number of instances of both classes are the same. In weka I changed the default setting for the filter. I changed “biasTouniformclass” from zero to 1., which can ensure the class distribution is uniform in the output data. Also changed the “samplesizepercent” from 100 to 158.8%. It sets the size of the subsample, as a percentage of the original size. 158.8% is worked out by (212-55)/267 + 1=158.8%.

3:
The pipeline I used is not suitable for all the five techniques I used. Because two of accuracies decreased after I applied the dataset to it. As mentioned before there has no outliers and no missing value so I think there is no need for additional effort. Like for some datasets which have missing values may would need laplace regression etc.

Part3:



Result:

MLP:

```
=== Evaluation result ===

Scheme: MultilayerPerceptron
Options: -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H 8
Relation: Comp309fixed_dataset-weka.filters.unsupervised.attribute.NumericToNom


Correctly Classified Instances         391               92.654  %
Incorrectly Classified Instances        31                7.346  %
Kappa statistic                          0.8531
Mean absolute error                      0.0914
Root mean squared error                  0.2451
Relative absolute error                 18.2717 %
Root relative squared error             49.0237 %
Total Number of Instances              422

=== Detailed Accuracy By Class ===
```

J48:

Text

```
=== Evaluation result ===

Scheme: J48
Options: -C 0.25 -M 2
Relation: Comp309fixed_dataset-weka.filters.unsupervised.attribute.NumericToNom


Correctly Classified Instances         377               89.3365 %
Incorrectly Classified Instances        45               10.6635 %
Kappa statistic                          0.7867
Mean absolute error                      0.1527
Root mean squared error                  0.3018
Relative absolute error                 30.5423 %
Root relative squared error             60.3639 %
Total Number of Instances              422

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC
                 0.957    0.171    0.849      0.957    0.900      0.793    0.91
```

IBK:

```
=== Evaluation result ===

Scheme: IBk
Options: -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core
Relation: Comp309fixed_dataset-weka.filters.unsupervised.attribute.NumericToNo


Correctly Classified Instances         386               91.4692 %
Incorrectly Classified Instances        36                8.5308 %
Kappa statistic                          0.8294
Mean absolute error                      0.1008
Root mean squared error                  0.2642
Relative absolute error                 20.1545 %
Root relative squared error             52.8394 %
Total Number of Instances              422

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC
```

Naive Bayes:

```
=== Evaluation result ===

Scheme: NaiveBayes
Relation: Comp309fixed_dataset-weka.filters.unsupervised.attribute.NumericToNom


Correctly Classified Instances         337               79.8578 %
Incorrectly Classified Instances        85               20.1422 %
Kappa statistic                          0.5972
Mean absolute error                      0.2106
Root mean squared error                  0.4079
Relative absolute error                 42.1272 %
Root relative squared error             81.582  %
Total Number of Instances              422

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC
                 0.848    0.251    0.772      0.848    0.808      0.600    0.88
```

Classifier	MLP	J48	IBK	Naive Bayes

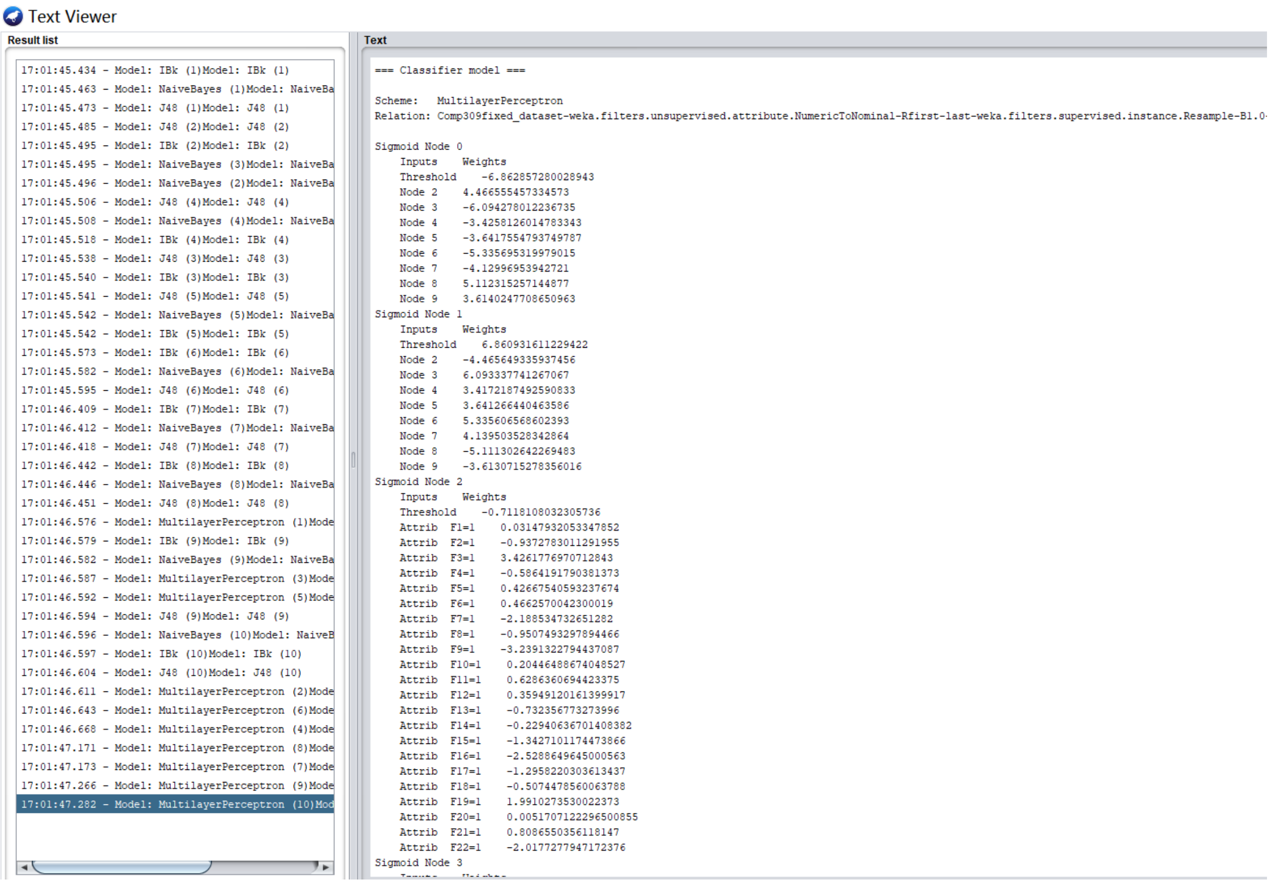
Correctly Accuracy	92.654%	89.336%	91.4692%	79.8587%

Comparing :

From the table above, the highest accuracy is provided by MLP. And before pipelining it was 78%. This means preprocessing make the dataset much better. And all the rest three techniques achieved a higher accuracy. Because the dataset became well balanced after preprocessing.

Learned classifier:

MLP:



J48:

Text

```
=== Classifier model ===

Scheme:   J48
Relation: Comp309fixed_dataset-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last-weka.filters.supervised.instance.Resample-B1.0-S1-Z158.8

J48 pruned tree
------------------

 F17 = 0
|   F13 = 0
|   |   F18 = 0
|   |   |   F20 = 0
|   |   |   |   F4 = 0
|   |   |   |   |   F6 = 0: 0 (149.0/14.0)
|   |   |   |   |   F6 = 1
|   |   |   |   |   |   F5 = 0: 0 (14.0)
|   |   |   |   |   |   F5 = 1: 1 (10.0)
|   |   |   |   F4 = 1
|   |   |   |   |   F3 = 0: 1 (9.0/1.0)
|   |   |   |   |   F3 = 1: 0 (3.0/1.0)
|   |   |   F20 = 1
|   |   |   |   F16 = 0
|   |   |   |   |   F11 = 0: 0 (21.0/6.0)
|   |   |   |   |   F11 = 1: 1 (6.0)
|   |   |   |   F16 = 1: 1 (12.0)
|   |   F18 = 1: 1 (5.0)
|   F13 = 1
|   |   F22 = 0
|   |   |   F14 = 0
|   |   |   |   F10 = 0
|   |   |   |   |   F8 = 0: 0 (11.0/1.0)
|   |   |   |   |   F8 = 1
|   |   |   |   |   |   F1 = 0: 1 (5.0)
|   |   |   |   |   |   F1 = 1: 0 (4.0)
|   |   |   |   F10 = 1
|   |   |   |   |   F5 = 0: 0 (6.0/1.0)
|   |   |   |   |   F5 = 1: 1 (5.0)
|   |   |   F14 = 1: 1 (18.0/2.0)
|   |   F22 = 1: 1 (63.0/2.0)
 F17 = 1: 1 (39.0)

Number of Leaves  :     17

Size of the tree :      33
```

IBK:

```
=== Classifier model ===

Scheme:   IBk
Relation: Comp309fixed_dataset-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last-weka.filters.supervised.instance.Resample-B1.0-S1-Z158.8

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification
```

Naive Bayes:

```
=== Classifier model ===

Scheme:   NaiveBayes
Relation: Comp309fixed_dataset-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last-weka.filters.supervised.instance.Resample-B1.0-S1-Z158.8

Naive Bayes Classifier

               Class
Attribute          0     1
               (0.5) (0.5)
===========================
 F1
  0             139.0 104.0
  1              53.0  88.0
  [total]       192.0 192.0

 F2
  0             171.0 131.0
  1              21.0  61.0
  [total]       192.0 192.0

 F3
  0             159.0 105.0
  1              33.0  87.0
  [total]       192.0 192.0

 F4
  0             177.0 131.0
  1              15.0  61.0
  [total]       192.0 192.0

 F5
  0             148.0 112.0
  1              44.0  80.0
  [total]       192.0 192.0

 F6
  0             175.0 131.0
  1              17.0  61.0
  [total]       192.0 192.0

 F7
  0             176.0 129.0
  1              16.0  63.0
  [total]       192.0 192.0

 F8
  0             163.0  86.0
  1              29.0 106.0
  [total]       192.0 192.0
```

```
 [total]        192.0 192.0

F13
 0              168.0  70.0
 1               24.0 122.0
 [total]        192.0 192.0

F14
 0              167.0 132.0
 1               25.0  60.0
 [total]        192.0 192.0

F15
 0              183.0 148.0
 1                9.0  44.0
 [total]        192.0 192.0

F16
 0              173.0 121.0
 1               19.0  71.0
 [total]        192.0 192.0

F17
 0              191.0 152.0
 1                1.0  40.0
 [total]        192.0 192.0

F18
 0              191.0 158.0
 1                1.0  34.0
 [total]        192.0 192.0

F19
 0              172.0 142.0
 1               20.0  50.0
 [total]        192.0 192.0

F20
 0              171.0 121.0
 1               21.0  71.0
 [total]        192.0 192.0

F21
 0              174.0  97.0
 1               18.0  95.0
 [total]        192.0 192.0

F22
 0              160.0  94.0
 1               32.0  98.0
 [total]        192.0 192.0
```