Assignment 1

Yan zichu 300476924
dataset:SPECT

Part 1:(2.1)

Naive bayes: Bayesians

(without cross-validation)

```
=== Summary ===

Correctly Classified Instances         213               79.7753 %
Incorrectly Classified Instances        54               20.2247 %
Kappa statistic                          0.4798
Mean absolute error                      0.2206
Root mean squared error                  0.4116
Relative absolute error                 67.1653 %
Root relative squared error            101.7664 %
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    0.193    0.506      0.764    0.609      0.498    0.865     0.622     0
                 0.807    0.236    0.929      0.807    0.864      0.498    0.865     0.958     1
Weighted Avg.    0.798    0.228    0.842      0.798    0.811      0.498    0.865     0.889

=== Confusion Matrix ===

   a   b   <-- classified as
  42  13 |   a = 0
  41 171 |   b = 1
```

Multilayer Perceptron: Connectionists

(without cross-validation)

```
=== Summary ===

Correctly Classified Instances         249               93.2584 %
Incorrectly Classified Instances        18                6.7416 %
Kappa statistic                          0.7911
Mean absolute error                      0.0926
Root mean squared error                  0.2191
Relative absolute error                 28.2075 %
Root relative squared error             54.1702 %
Total Number of Instances              267

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.818    0.038    0.849      0.818    0.833      0.791    0.958     0.890     0
                 0.962    0.182    0.953      0.962    0.958      0.791    0.958     0.979     1
Weighted Avg.    0.933    0.152    0.932      0.933    0.932      0.791    0.958     0.961

=== Confusion Matrix ===

   a   b   <-- classified as
  45  10 |   a = 0
   8 204 |   b = 1
```

J48 : Symbolists
(without cross-validation)

```
Correctly Classified Instances         235               88.015  %
Incorrectly Classified Instances        32               11.985  %
Kappa statistic                          0.5894
Mean absolute error                      0.1937
Root mean squared error                  0.3112
Relative absolute error                 58.9873 %
Root relative squared error             76.9528 %
Total Number of Instances              267

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.564    0.038    0.795      0.564    0.660      0.602    0.861     0.651     0
                 0.962    0.436    0.895      0.962    0.927      0.602    0.861     0.940     1
Weighted Avg.    0.880    0.354    0.874      0.880    0.872      0.602    0.861     0.881

=== Confusion Matrix ===

   a   b   <-- classified as
  31  24 |   a = 0
   8 204 |   b = 1
```

IBK : Analogizers(KNN)
(without cross-validation)

```
=== Summary ===

Correctly Classified Instances         251               94.0075 %
Incorrectly Classified Instances        16                5.9925 %
Kappa statistic                          0.8216
Mean absolute error                      0.0799
Root mean squared error                  0.1962
Relative absolute error                 24.3279 %
Root relative squared error             48.5105 %
Total Number of Instances              267

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.891    0.047    0.831      0.891    0.860      0.822    0.985     0.921     0
                 0.953    0.109    0.971      0.953    0.962      0.822    0.985     0.995     1
Weighted Avg.    0.940    0.096    0.942      0.940    0.941      0.822    0.985     0.980

=== Confusion Matrix ===

   a   b   <-- classified as
  49   6 |   a = 0
  10 202 |   b = 1
```

Techniques / Results :

	Naive bayes	Multilayer perceptron	KNN(IBK)	Decision tree
correct	79.7753%	93.2584%	94.0075%	88.051%
incorrect	20.2247%	6.7416%	5.9925%	11.985%

In conclusion, KNN have the highest correctly classified instances is 97.3262%.

Bayesians:(**Naive bayes**) classifier :bayes

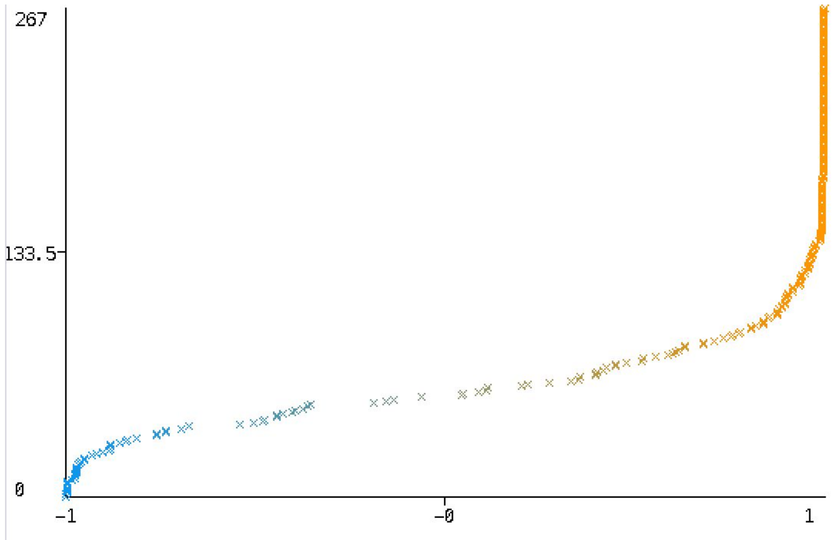
1.General description:

My personal understanding of this :There is a basic tool in statistics called Bayesian formula, also known as Bayesian rule. Although it is a mathematical formula, its principle requires no more figures. If you see that a person is always doing something good, that person will probably be a good person. That is to say, when you can't accurately understand the nature of a thing, you can rely on the number of events related to the specific nature of the thing to judge the probability of its essential attributes. Expressed in a mathematical language: the more events that support an attribute occur, the greater the likelihood that the attribute will be established.

P(A|B)= $\frac{P(B|A)P(A)}{P(B)}$ 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.

2. Representaion:

Graphical models.



Non-normal distribution

3.Evaluation method:

Posterior Probability.

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.

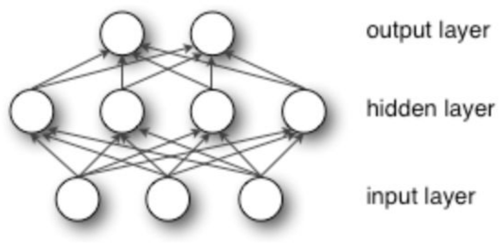
4.Optimization:

Probabilitic Inference.

Connectionists:(**Multilayer Perceptron**) classifier :function

1.General description:

Multilayer Perceptron, In addition to the input layer and the output layer, there can be multiple hidden layers in between. The simplest MLP has only one hidden layer, that is, a three-layer structure, as shown below:



As can be seen from the above figure, the multilayer perceptron layer is fully connected to the other layers.
There is nothing to say about the input layer. For example, if the input is an n-dimensional vector, there are n neurons.

The whole model of MLP is like this. The three-layer MLP mentioned above is summarized by the formula. The function G is softmax.

$$f(x) = G(b^{(2)} + W^{(2)}(s(b^{(1)} + W^{(1)}x))),$$

Therefore, all parameters of the MLP are the connection weights and offsets between the layers, including W1, b1, W2, and b2.

2. Representaion:

Neural network

3.Evaluation method:

squared error

4.Optimization:

Solving the optimization problem, the simplest is the gradient descent method: first randomly initialize all parameters, then iteratively train, continuously calculate the gradient and update the parameters until a certain condition is met. (For example, when the error is small enough and the number of iterations is enough).

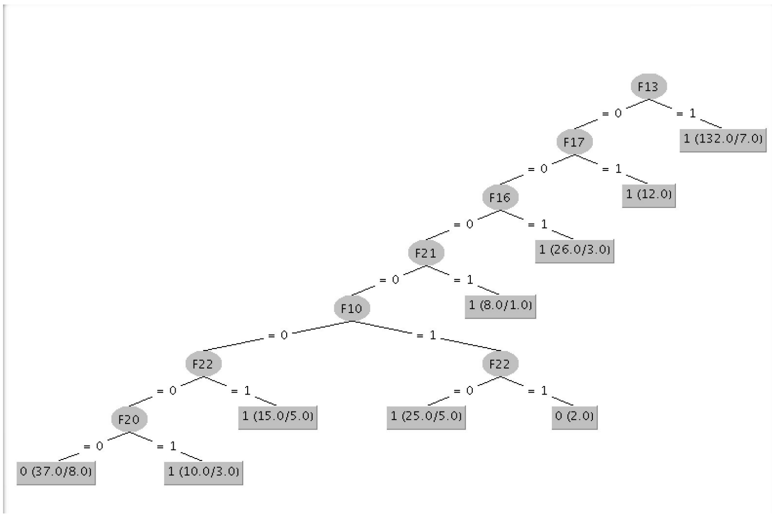
Symbolists:(J48) classifer :**decision tree**

1.General description:

J48 is a top-down, recursive strategy, selecting an attribute to be placed at the root node, generating a branch for each possible attribute value, dividing the instance into multiple subsets, each subset corresponding to a branch of the root node, and then This process is repeated recursively on each branch. Stop when all instances have the same classification.

2. Representaion:

logic:



3.Evaluation method:

Accuracy.

What we want to get is pure splitting, that is, splitting into pure nodes, hoping to find a property, one of its nodes is yes, and one node is all no. This is the best case, because if it is a hybrid node, it needs to split again.(My dataset is the best case.)

Quantization is used to determine the attributes that produce the purest child nodes , calculating purity (the goal is to get the smallest decision tree). The top-down tree induction method uses some heuristic methods--the heuristic method that produces pure nodes is based on information theory, that is, information entropy, which measures information in bits.

Information gain = information entropy of pre-split distribution - information entropy of post-split distribution, selecting the attribute with the largest information gain.

4.Optimization:

Inverse deduction.

<u>Advantages</u>: Simple to understand and interpret.
Able to handle both numerical and categorical data.
Requires little data preparation.
Uses a white box model.
Possible to validate a model using statistical tests.
Mirrors human decision making more closely than other approaches.

<u>In addition, as I mentioned before, Decision Tree has a good performance on categorical features. And, the dataset I was using is a categorical feature. As the results show, it performs very well.</u>

Analogizers:(IBK) classifier :**KNN**

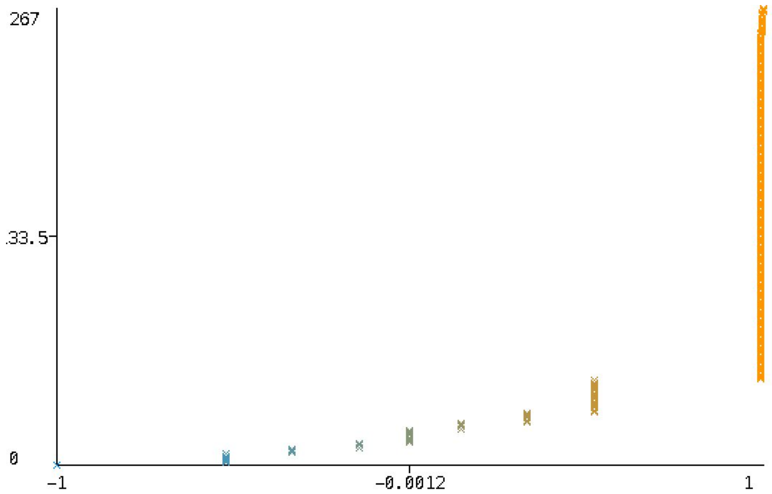
1.General description:

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

A supervised machine learning algorithm (as opposed to an unsupervised machine learning algorithm) is one that relies on labeled input data to learn a function that produces an appropriate output when given new unlabeled data.

2. Representaion:

support vector



3.Evaluation method:

Margin.

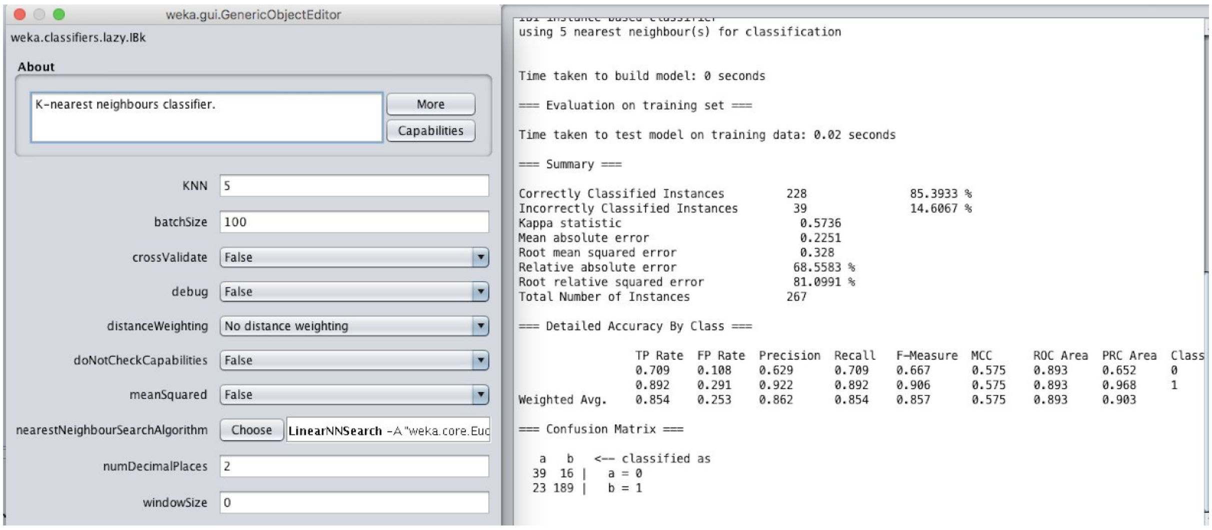
The algorithm doesn’t learn a model but it chooses to memorize the instances in the training set. And uses this memory for the prediction phase.Here is one of the popular choice, Euclidean distance is given by :

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

4.Optimization:

Constrained 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. What's more , I tried K value from 2 to 7. and When I change the K value to 5 I get the best performance at 85.3933%.



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.

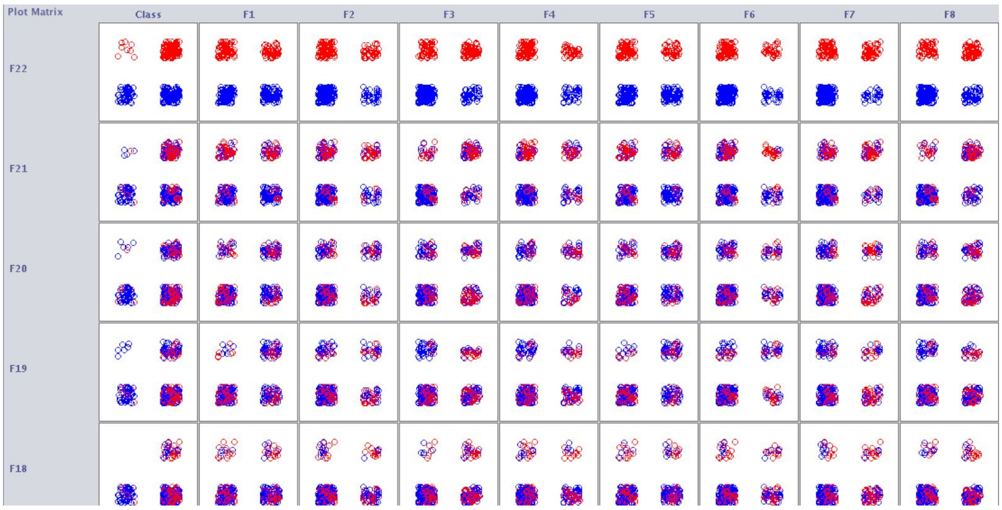
The difference between each technique:

The dataset spect I was using is a binary dataset.

	Naive bayes	Multilayer perceptron	KNN(IBK)	Decision tree
correct	82.8877%	96.7914%	97.3262%(85.3933%.with K value = 5)	91.9786%
incorrect	17.1123%	3.2086%	2.6738%	8.0214%

Conclusion:

We always try to find the best algorithm, but now there is no such algorithm that can handle all the data and achieve high performance. But for different datasets, we can use different techniques that are suitable for using datasets. For my dataset, it has good quality.



1.KNN: The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. As the graph shown above, it can easily be distinguished ,so it means that if there is a unknown data want to be classify ,it's easy to find the nearest value and classify it. Although it showed a high accuracy with my data, I think it is not suitable(when I change K value to 3,5,7, the accuracy goes down). Comparing to binary sets, numerical suits KNN much better.

2.J48:For decision tree, J48,the dataset I was using is a categorical feature. So 91.9786% is also a good performance.

3.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.

4.Multilayer perceptron:the accuracy of MP is 96.7914%,so this algorithm is also suit to my dataset.And this one actually is the best one without cross validation.

Part 2:(2.2)

1.Business understanding:

Background:

Single-photon emission computed tomography (SPECT, or less commonly, SPET) is a nuclear medicinetomographic imaging technique using gamma rays.[1] 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.

<u>And I think main reason of collecting the data is that train the machine to determine whether a person has heart disease through a series of data.</u>

2.Data understanding:

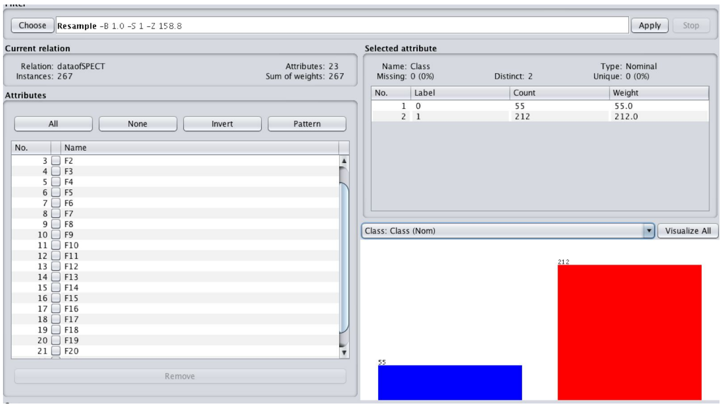
The dataset I was using has good quality, there has no missing value and no outliers. But the only problem is the dataset is not well balanced.

3.Data preparation:

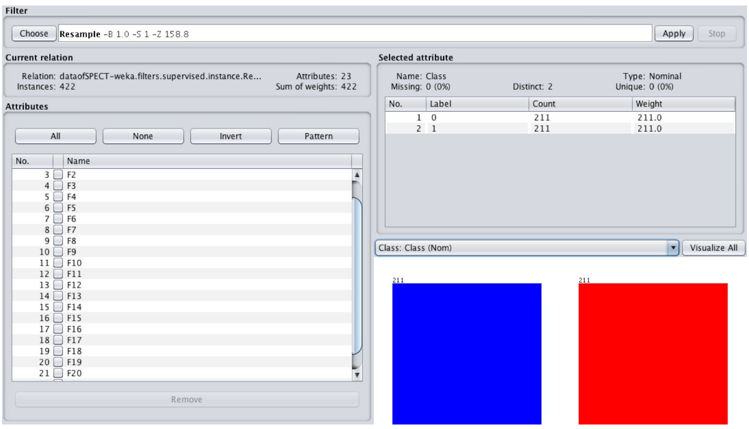
Resample:

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

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)/ + 1=158.8%.

Numeric to nominal:

Although my data is composed of 1 and 0, they represent yes or no.So they are nominal actually.A filter for turning numeric attributes into nominal ones. Unlike discretization, it just takes all numeric values and adds them to the list of nominal values of that attribute.

Cross-validation:

In a given modeling sample, take most of the samples to build the model, leaving a small number of samples to be tested with the model just created.

As mentioned before i used 10-fold cross validation. 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.

4. Modelling

Yes.But The pipeline I used is not suitable for all the five techniques I used. It just contains Analogizers,Symbolists,Bayesians,Connectionists.

5. Evaluation

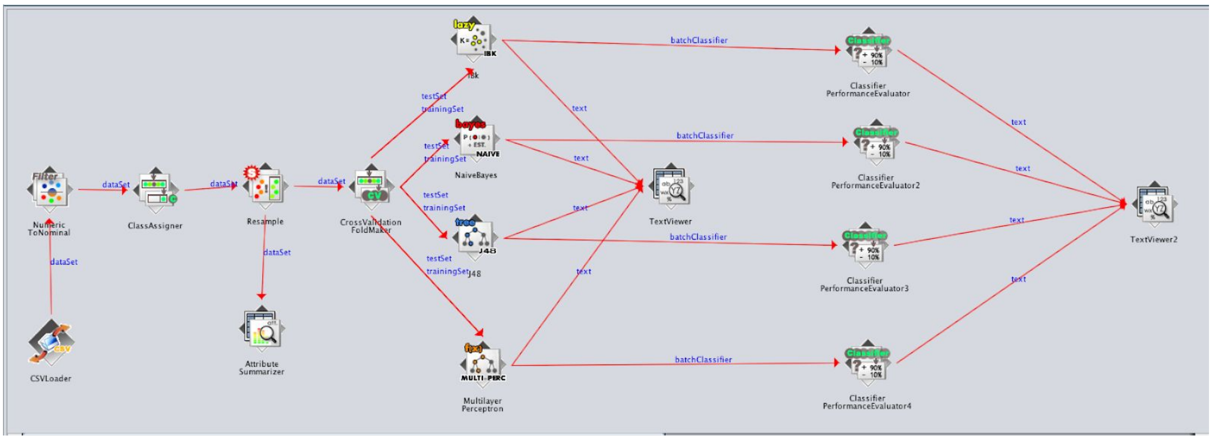
Yes.I used four methods :J48, Multilayer Perceptron , KNN ,decision tree.

6.Deployment

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.

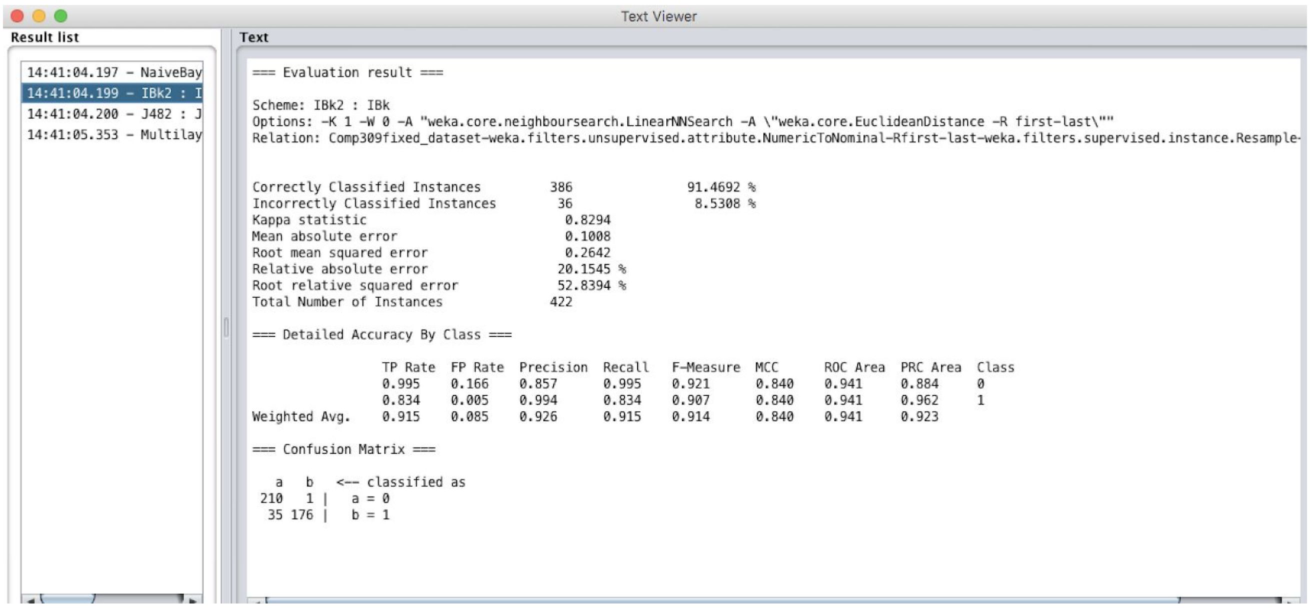
Part 3:(2.3)

The pipeline I made :



Output:

IBK:




```
=== Evaluation result ===

Scheme: IBk2 : IBk
Options: -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Relation: Comp309fixed_dataset-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last-weka.filters.supervised.instance.Resample-

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 Area  PRC Area  Class
                 0.995    0.166    0.857      0.995    0.921      0.840    0.941     0.884     0
                 0.834    0.005    0.994      0.834    0.907      0.840    0.941     0.962     1
Weighted Avg.    0.915    0.085    0.926      0.915    0.914      0.840    0.941     0.923

=== Confusion Matrix ===

   a   b   <-- classified as
 210   1 |   a = 0
  35 176 |   b = 1
```

NaiveBayes:

Text Viewer

Result list

```
14:41:04.197 - NaiveBay
14:41:04.199 - IBk2 : I
14:41:04.200 - J482 : J
14:41:05.353 - Multilay
```

Text

```
=== Evaluation result ===

Scheme: NaiveBayes2 : NaiveBayes
Relation: Comp309fixed_dataset-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last-weka.filters.supervised.instance.Resample-

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 Area  PRC Area  Class
                 0.848    0.251    0.772      0.848    0.808      0.600    0.889     0.874     0
                 0.749    0.152    0.832      0.749    0.788      0.600    0.889     0.897     1
Weighted Avg.    0.799    0.201    0.802      0.799    0.798      0.600    0.889     0.886

=== Confusion Matrix ===

   a   b   <-- classified as
 179  32 |   a = 0
  53 158 |   b = 1
```

J48:

Text Viewer

Result list

```
14:41:04.197 - NaiveBay
14:41:04.199 - IBk2 : I
14:41:04.200 - J482 : J
14:41:05.353 - Multilay
```

Text

```
=== Evaluation result ===

Scheme: J482 : J48
Options: -C 0.25 -M 2
Relation: Comp309fixed_dataset-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last-weka.filters.supervised.instance.Resample-

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 Area  PRC Area  Class
                 0.957    0.171    0.849      0.957    0.900      0.793    0.912     0.853     0
                 0.829    0.043    0.951      0.829    0.886      0.793    0.912     0.931     1
Weighted Avg.    0.893    0.107    0.900      0.893    0.893      0.793    0.912     0.892

=== Confusion Matrix ===

   a   b   <-- classified as
 202   9 |   a = 0
  36 175 |   b = 1
```

Multilayer perceptron:

Text Viewer

Result list

```
14:41:04.197 - NaiveBay
14:41:04.199 - IBk2 : I
14:41:04.200 - J482 : J
14:41:05.353 - Multilay
```

Text

```
=== Evaluation result ===

Scheme: MultilayerPerceptron2 : MultilayerPerceptron
Options: -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
Relation: Comp309fixed_dataset-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last-weka.filters.supervised.instance.Resample-

Correctly Classified Instances         388               91.9431 %
Incorrectly Classified Instances        34                8.0569 %
Kappa statistic                          0.8389
Mean absolute error                      0.1005
Root mean squared error                  0.2623
Relative absolute error                 20.0928 %
Root relative squared error             52.4632 %
Total Number of Instances              422

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.957    0.118    0.890      0.957    0.922      0.841    0.934     0.911     0
                 0.882    0.043    0.954      0.882    0.916      0.841    0.934     0.932     1
Weighted Avg.    0.919    0.081    0.922      0.919    0.919      0.841    0.934     0.922

=== Confusion Matrix ===

   a   b   <-- classified as
 202   9 |   a = 0
  25 186 |   b = 1
```

This is the table of result without using cross-validation and resampling:

	Naive bayes	Multilayer perceptron	KNN(IBK)	Decision tree
correct	82.8877%	96.7914%	97.3262%(85.3933%.with K value = 5)	91.9786%
incorrect	17.1123%	3.2086%	2.6738%	8.0214%

This is the table of result with using cross-validation and resampling:

	Naive bayes	Multilayer perceptron	KNN(IBK)	Decision tree
correct	79.8578%	91.9431%	91.4692%(82.9384 %with K value = 5)	89.3365%
incorrect	20.1422%	8.0569%	8.5308%	10.6635%

The accuracy decrease because cross-validation is more reasonable way to check the correctness than only using validation(using training set).

The reason why I use cross validation is it can divide my original data into several files and leave one file as test dataset and others be the training set which can check the accuracy more reasonable.It's better than only using training set.

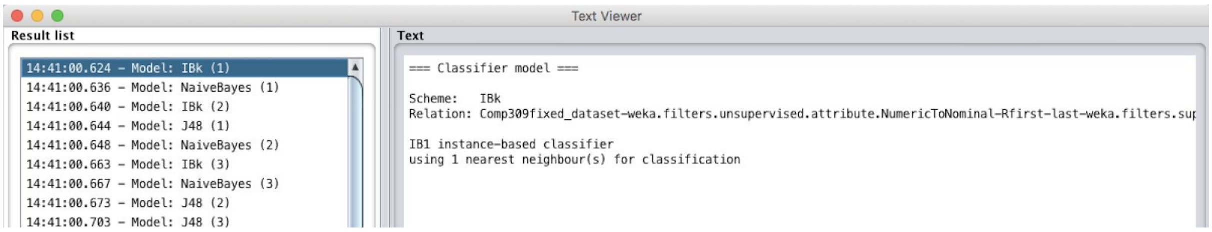
Cross-validation : In a given modeling sample, take most of the samples to build the model, leaving a small number of samples to be tested with the model just created.

As mentioned before i used 10-fold cross validation. 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.

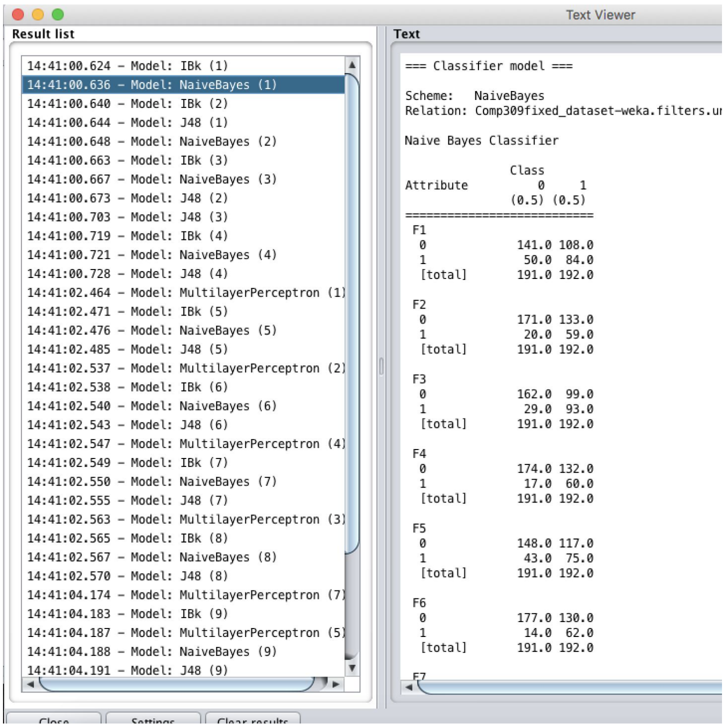
==

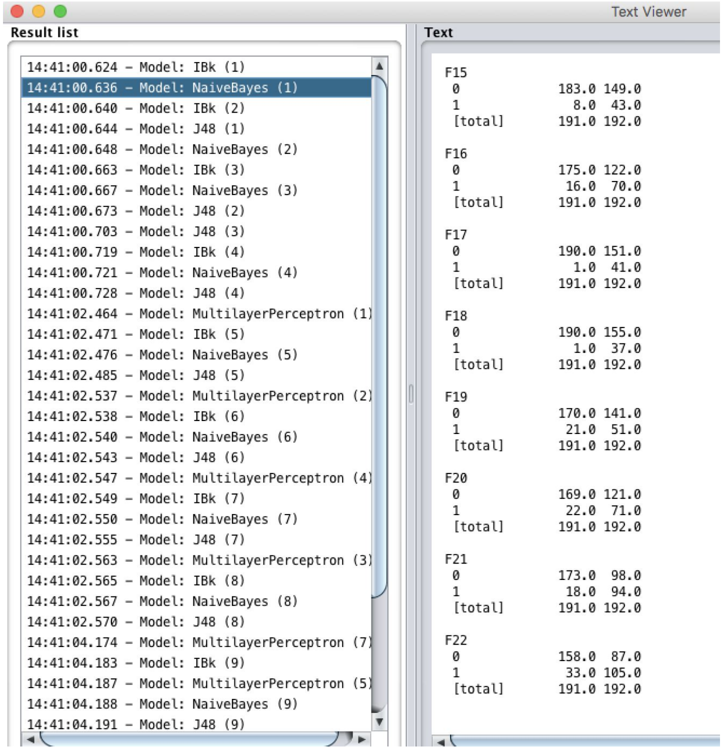
Model of them:

IBK:



Naive Bayes:

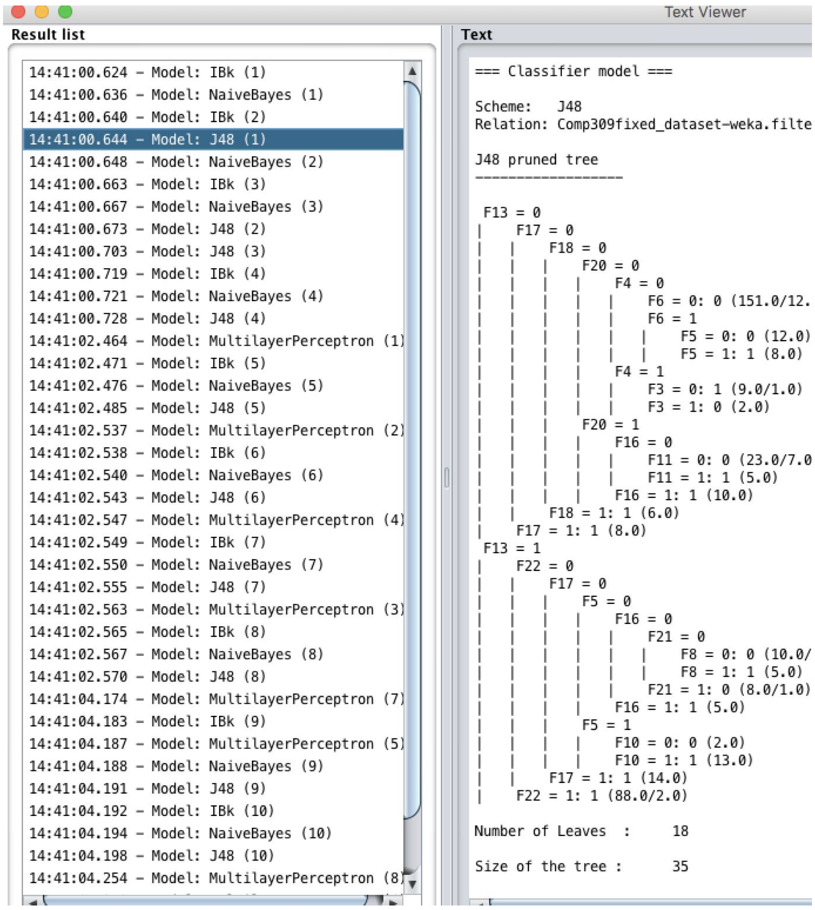




```
Result list
14:41:00.624 - Model: IBk (1)
14:41:00.636 - Model: NaiveBayes (1)
14:41:00.640 - Model: IBk (2)
14:41:00.644 - Model: J48 (1)
14:41:00.648 - Model: NaiveBayes (2)
14:41:00.663 - Model: IBk (3)
14:41:00.667 - Model: NaiveBayes (3)
14:41:00.673 - Model: J48 (2)
14:41:00.703 - Model: J48 (3)
14:41:00.719 - Model: IBk (4)
14:41:00.721 - Model: NaiveBayes (4)
14:41:00.728 - Model: J48 (4)
14:41:02.464 - Model: MultilayerPerceptron (1)
14:41:02.471 - Model: IBk (5)
14:41:02.476 - Model: NaiveBayes (5)
14:41:02.485 - Model: J48 (5)
14:41:02.537 - Model: MultilayerPerceptron (2)
14:41:02.538 - Model: IBk (6)
14:41:02.540 - Model: NaiveBayes (6)
14:41:02.543 - Model: J48 (6)
14:41:02.547 - Model: MultilayerPerceptron (4)
14:41:02.549 - Model: IBk (7)
14:41:02.550 - Model: NaiveBayes (7)
14:41:02.555 - Model: J48 (7)
14:41:02.563 - Model: MultilayerPerceptron (3)
14:41:02.565 - Model: IBk (8)
14:41:02.567 - Model: NaiveBayes (8)
14:41:02.570 - Model: J48 (8)
14:41:04.174 - Model: MultilayerPerceptron (7)
14:41:04.183 - Model: IBk (9)
14:41:04.187 - Model: MultilayerPerceptron (5)
14:41:04.188 - Model: NaiveBayes (9)
14:41:04.191 - Model: J48 (9)
```

```
Text Viewer
Text

F15
 0              183.0 149.0
 1                8.0  43.0
 [total]        191.0 192.0

F16
 0              175.0 122.0
 1               16.0  70.0
 [total]        191.0 192.0

F17
 0              190.0 151.0
 1                1.0  41.0
 [total]        191.0 192.0

F18
 0              190.0 155.0
 1                1.0  37.0
 [total]        191.0 192.0

F19
 0              170.0 141.0
 1               21.0  51.0
 [total]        191.0 192.0

F20
 0              169.0 121.0
 1               22.0  71.0
 [total]        191.0 192.0

F21
 0              173.0  98.0
 1               18.0  94.0
 [total]        191.0 192.0

F22
 0              158.0  87.0
 1               33.0 105.0
 [total]        191.0 192.0
```

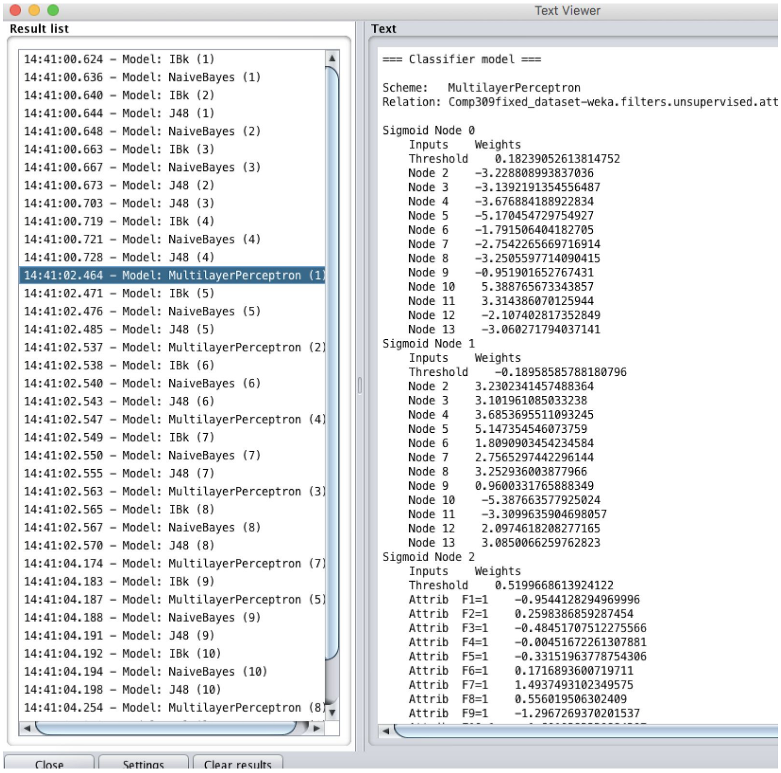
J48:



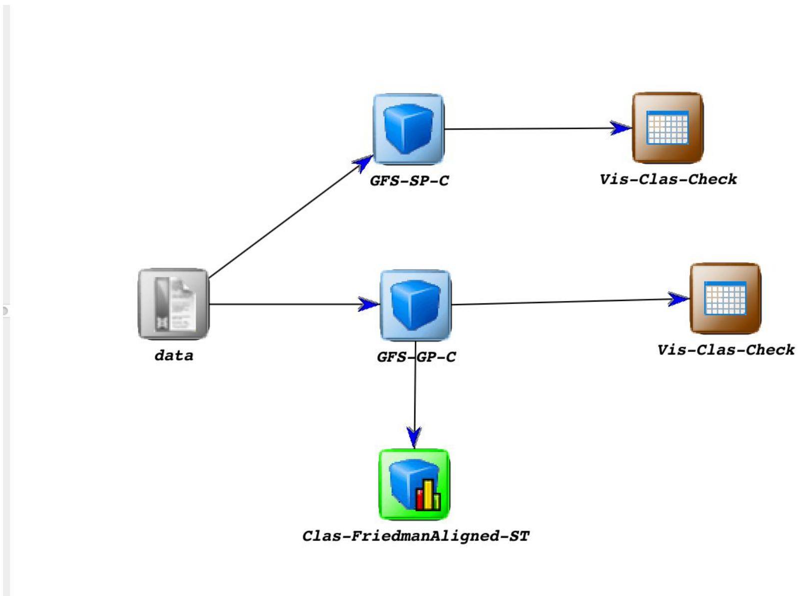
```
Result list
14:41:00.624 - Model: IBk (1)
14:41:00.636 - Model: NaiveBayes (1)
14:41:00.640 - Model: IBk (2)
14:41:00.644 - Model: J48 (1)
14:41:00.648 - Model: NaiveBayes (2)
14:41:00.663 - Model: IBk (3)
14:41:00.667 - Model: NaiveBayes (3)
14:41:00.673 - Model: J48 (2)
14:41:00.703 - Model: J48 (3)
14:41:00.719 - Model: IBk (4)
14:41:00.721 - Model: NaiveBayes (4)
14:41:00.728 - Model: J48 (4)
14:41:02.464 - Model: MultilayerPerceptron (1)
14:41:02.471 - Model: IBk (5)
14:41:02.476 - Model: NaiveBayes (5)
14:41:02.485 - Model: J48 (5)
14:41:02.537 - Model: MultilayerPerceptron (2)
14:41:02.538 - Model: IBk (6)
14:41:02.540 - Model: NaiveBayes (6)
14:41:02.543 - Model: J48 (6)
14:41:02.547 - Model: MultilayerPerceptron (4)
14:41:02.549 - Model: IBk (7)
14:41:02.550 - Model: NaiveBayes (7)
14:41:02.555 - Model: J48 (7)
14:41:02.563 - Model: MultilayerPerceptron (3)
14:41:02.565 - Model: IBk (8)
14:41:02.567 - Model: NaiveBayes (8)
14:41:02.570 - Model: J48 (8)
14:41:04.174 - Model: MultilayerPerceptron (7)
14:41:04.183 - Model: IBk (9)
14:41:04.187 - Model: MultilayerPerceptron (5)
14:41:04.188 - Model: NaiveBayes (9)
14:41:04.191 - Model: J48 (9)
14:41:04.192 - Model: IBk (10)
14:41:04.194 - Model: NaiveBayes (10)
14:41:04.198 - Model: J48 (10)
14:41:04.254 - Model: MultilayerPerceptron (8)
```

```
Text Viewer
Text

=== Classifier model ===

Scheme:    J48
Relation: Comp309fixed_dataset-weka.filte

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

 F13 = 0
|   F17 = 0
|   |   F18 = 0
|   |   |   F20 = 0
|   |   |   |   F4 = 0
|   |   |   |   |   F6 = 0: 0 (151.0/12.
|   |   |   |   |   F6 = 1
|   |   |   |   |   |   F5 = 0: 0 (12.0)
|   |   |   |   |   |   F5 = 1: 1 (8.0)
|   |   |   |   F4 = 1
|   |   |   |   |   F3 = 0: 1 (9.0/1.0)
|   |   |   |   |   F3 = 1: 0 (2.0)
|   |   |   F20 = 1
|   |   |   |   F16 = 0
|   |   |   |   |   F11 = 0: 0 (23.0/7.0
|   |   |   |   |   F11 = 1: 1 (5.0)
|   |   |   |   F16 = 1: 1 (10.0)
|   |   F18 = 1: 1 (6.0)
|   F17 = 1: 1 (8.0)
 F13 = 1
|   F22 = 0
|   |   F17 = 0
|   |   |   F5 = 0
|   |   |   |   F16 = 0
|   |   |   |   |   F21 = 0
|   |   |   |   |   |   F8 = 0: 0 (10.0/
|   |   |   |   |   |   F8 = 1: 1 (5.0)
|   |   |   |   |   F21 = 1: 0 (8.0/1.0)
|   |   |   |   F16 = 1: 1 (5.0)
|   |   |   F5 = 1
|   |   |   |   F10 = 0: 0 (2.0)
|   |   |   |   F10 = 1: 1 (13.0)
|   |   F17 = 1: 1 (14.0)
|   F22 = 1: 1 (88.0/2.0)

Number of Leaves  :     18

Size of the tree :      35
```

MP:


Text Viewer
Result list
14:41:00.624 - Model: IBk (1)
14:41:00.636 - Model: NaiveBayes (1)
14:41:00.640 - Model: IBk (2)
14:41:00.644 - Model: J48 (1)
14:41:00.648 - Model: NaiveBayes (2)
14:41:00.663 - Model: IBk (3)
14:41:00.667 - Model: NaiveBayes (3)
14:41:00.673 - Model: J48 (2)
14:41:00.703 - Model: J48 (3)
14:41:00.719 - Model: IBk (4)
14:41:00.721 - Model: NaiveBayes (4)
14:41:00.728 - Model: J48 (4)
14:41:02.464 - Model: MultilayerPerceptron (1)
14:41:02.471 - Model: IBk (5)
14:41:02.476 - Model: NaiveBayes (5)
14:41:02.485 - Model: J48 (5)
14:41:02.537 - Model: MultilayerPerceptron (2)
14:41:02.538 - Model: IBk (6)
14:41:02.540 - Model: NaiveBayes (6)
14:41:02.543 - Model: J48 (6)
14:41:02.547 - Model: MultilayerPerceptron (4)
14:41:02.549 - Model: IBk (7)
14:41:02.550 - Model: NaiveBayes (7)
14:41:02.555 - Model: J48 (7)
14:41:02.563 - Model: MultilayerPerceptron (3)
14:41:02.565 - Model: IBk (8)
14:41:02.567 - Model: NaiveBayes (8)
14:41:02.570 - Model: J48 (8)
14:41:04.174 - Model: MultilayerPerceptron (7)
14:41:04.183 - Model: IBk (9)
14:41:04.187 - Model: MultilayerPerceptron (5)
14:41:04.188 - Model: NaiveBayes (9)
14:41:04.191 - Model: J48 (9)
14:41:04.192 - Model: IBk (10)
14:41:04.194 - Model: NaiveBayes (10)
14:41:04.198 - Model: J48 (10)
14:41:04.254 - Model: MultilayerPerceptron (8)
Text
=== Classifier model ===
Scheme: MultilayerPerceptron
Relation: Comp309fixed_dataset-weka.filters.unsupervised.att
Sigmoid Node 0
Inputs Weights
Threshold 0.18239052613814752
Node 2 -3.228808993837036
Node 3 -3.1392191354556487
Node 4 -3.676884188922834
Node 5 -5.170454729754927
Node 6 -1.791506404182705
Node 7 -2.7542265669716914
Node 8 -3.2505597714090415
Node 9 -0.951901652767431
Node 10 5.388765673343857
Node 11 3.314386070125944
Node 12 -2.107402817352849
Node 13 -3.060271794037141
Sigmoid Node 1
Inputs Weights
Threshold -0.18958585788180796
Node 2 3.2302341457488364
Node 3 3.101961085033238
Node 4 3.6853695511093245
Node 5 5.147354546073759
Node 6 1.8090903454234584
Node 7 2.7565297442296144
Node 8 3.252936003877966
Node 9 0.9600331765888349
Node 10 -5.387663577925024
Node 11 -3.3099635904698057
Node 12 2.0974618208277165
Node 13 3.0850066259762823
Sigmoid Node 2
Inputs Weights
Threshold 0.5199668613924122
Attrib F1=1 -0.9544128294969996
Attrib F2=1 0.2598386859287454
Attrib F3=1 -0.48451707512275566
Attrib F4=1 -0.00451672261307881
Attrib F5=1 -0.33151963778754306
Attrib F6=1 0.1716893600719711
Attrib F7=1 1.4937493102349575
Attrib F8=1 0.556019506302409
Attrib F9=1 -1.2967269370201537
Close
Settings
Clear results

Part 4:(2.4)


GFS-SP-C
Vis-Clas-Check
data
GFS-GP-C
Vis-Clas-Check
Clas-FriedmanAligned-ST

GFS-GP-C:

```
TEST RESULTS
============
Classifier= .a/comp309fixedaset-7.26/comp309fixedaset-7.26
Fold 0 : CORRECT=0.7037037037037037 N/C=0.0
Fold 1 : CORRECT=0.6666666666666667 N/C=0.0
Fold 2 : CORRECT=0.7407407407407407 N/C=0.0
Fold 3 : CORRECT=0.5925925925925926 N/C=0.0
Fold 4 : CORRECT=0.6666666666666667 N/C=0.0
Fold 5 : CORRECT=0.7777777777777778 N/C=0.0
Fold 6 : CORRECT=0.6296296296296297 N/C=0.0
Fold 7 : CORRECT=0.7692307692307692 N/C=0.0
Fold 8 : CORRECT=0.3076923076923077 N/C=0.0
Fold 9 : CORRECT=0.5769230769230769 N/C=0.0
Global Classification Error + N/C:
0.3568376068376068
stddev Global Classification Error + N/C:
0.12960497245223082
Correctly classified:
0.6431623931623932
Global N/C:
0.0
```

```
TRAIN RESULTS
============
Classifier= .a/comp309fixedaset-7.26/comp309fixedaset-7.26
Summary of data, Classifiers: .a/comp309fixedaset-7.26/comp309fixedaset-7.26
Fold 0 : CORRECT=0.7333333333333334 N/C=0.0
Fold 1 : CORRECT=0.7625 N/C=0.0
Fold 2 : CORRECT=0.7708333333333334 N/C=0.0
Fold 3 : CORRECT=0.7458333333333333 N/C=0.0
Fold 4 : CORRECT=0.7625 N/C=0.0
Fold 5 : CORRECT=0.75 N/C=0.0
Fold 6 : CORRECT=0.7625 N/C=0.0
Fold 7 : CORRECT=0.7136929460580913 N/C=0.0
Fold 8 : CORRECT=0.7053941908713692 N/C=0.0
Fold 9 : CORRECT=0.7800829875518672 N/C=0.0
Global Classification Error + N/C:
0.25133298755186717
stddev Global Classification Error + N/C:
0.02322286507418669
Correctly classified:
0.7486670124481328
Global N/C:
0.0
```

GFS-SP-C:

result0s0.stat — TSTGFS-GP-C | result0s0.stat — TSTGFS-SP-C

```
TEST RESULTS
============
Classifier= .a/comp309fixedaset-7.26/comp309fixedaset-7.26
Fold 0 : CORRECT=0.8148148148148149 N/C=0.0
Fold 1 : CORRECT=0.7777777777777778 N/C=0.0
Fold 2 : CORRECT=0.8148148148148149 N/C=0.0
Fold 3 : CORRECT=0.7777777777777778 N/C=0.0
Fold 4 : CORRECT=0.6666666666666667 N/C=0.0
Fold 5 : CORRECT=0.7777777777777778 N/C=0.0
Fold 6 : CORRECT=0.7407407407407407 N/C=0.0
Fold 7 : CORRECT=0.6538461538461539 N/C=0.0
Fold 8 : CORRECT=0.7692307692307692 N/C=0.0
Fold 9 : CORRECT=0.5769230769230769 N/C=0.0
Global Classification Error + N/C:
0.26296296296296295
stddev Global Classification Error + N/C:
0.07459057273517566
Correctly classified:
0.737037037037037
Global N/C:
0.0
```

```
TRAIN RESULTS
============
Classifier= .a/comp309fixedaset-7.26/comp309fixedaset-7.26
Summary of data, Classifiers: .a/comp309fixedaset-7.26/comp309fixedaset-7.26
Fold 0 : CORRECT=0.7458333333333333 N/C=0.0
Fold 1 : CORRECT=0.75 N/C=0.0
Fold 2 : CORRECT=0.7416666666666667 N/C=0.0
Fold 3 : CORRECT=0.7458333333333333 N/C=0.0
Fold 4 : CORRECT=0.7583333333333333 N/C=0.0
Fold 5 : CORRECT=0.7458333333333333 N/C=0.0
Fold 6 : CORRECT=0.75 N/C=0.0
Fold 7 : CORRECT=0.7593360995850622 N/C=0.0
Fold 8 : CORRECT=0.7427385892116183 N/C=0.0
Fold 9 : CORRECT=0.7676348547717842 N/C=0.0
Global Classification Error + N/C:
0.2492790456431535
stddev Global Classification Error + N/C:
0.00797905643104204
Correctly classified:
0.7507209543568465
Global N/C:
0.0
```

▼ results	今天 下午12:45	--	文件夹
▶ Clas-FriedmanAligned-ST	今天 下午12:45	--	文件夹
▶ GFS-GP-C.comp309fixedaset-7.26	今天 下午12:38	--	文件夹
▶ GFS-SP-C.comp309fixedaset-7.26	今天 下午12:43	--	文件夹
▼ Vis-Clas-Check	今天 下午12:56	--	文件夹
▼ TSTGFS-GP-C	今天 下午12:38	--	文件夹
result0s0.stat	今天 下午12:38	1 KB	文稿
▼ TSTGFS-SP-C	今天 下午12:43	--	文件夹
result0s0.stat	今天 下午12:43	1 KB	文稿

1.Symbolists use decision trees, production rule systems, and inductive logic programming.
2.Connectionists rely on deep learning technologies, including RNN, CNN, and deep reinforcement learning.
3.Bayesians use Hidden Markov Models, graphical models, and causal inference.
4.Evolutionaries use genetic algorithms, evolutionary programming, and evolutionary game theory.
5.Analogizers use k-nearest neighbor, and support vector machines.

Weka and Keel:

Personally, I think weka is easier to use and the visualization is very powerful and supports multiple file formats. Easy to run. These are all that keel does not have. The algorithm available in keel is much more than weka, and contains the evaluation tribe not included in weka. The only deficiency, I think the operation is too cumbersome.