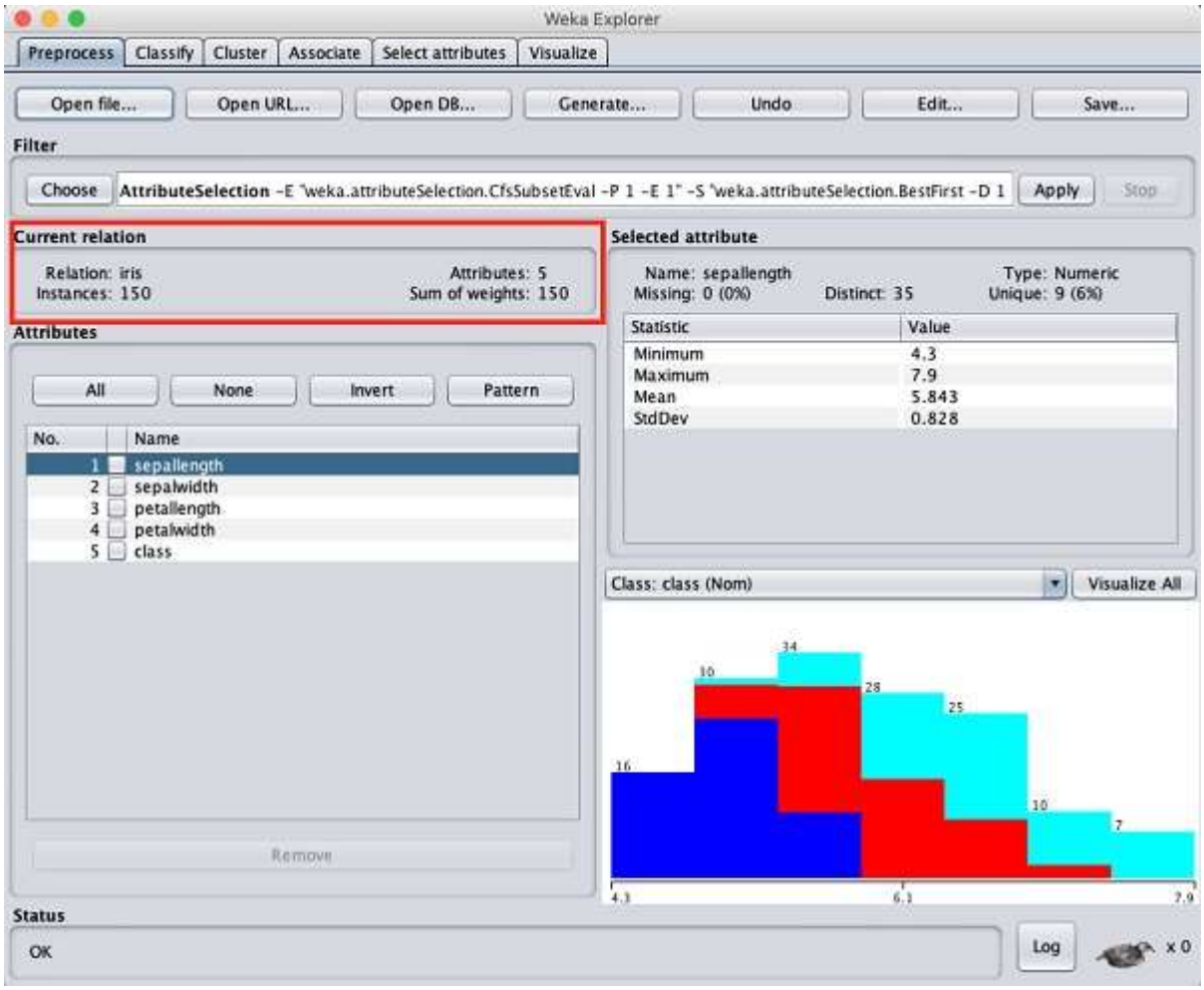
Task 4: Demonstrate performing clustering on data sets

A clustering algorithm finds groups of similar instances in the entire dataset. WEKA supports several clustering algorithms such as EM, FilteredClusterer, HierarchicalClusterer, SimpleKMeans and so on. You should understand these algorithms completely to fully exploit the WEKA capabilities.

As in the case of classification, WEKA allows you to visualize the detected clusters graphically. To demonstrate the clustering, we will use the provided iris database. The data set contains three classes of 50 instances each. Each class refers to a type of iris plant.

Loading Data

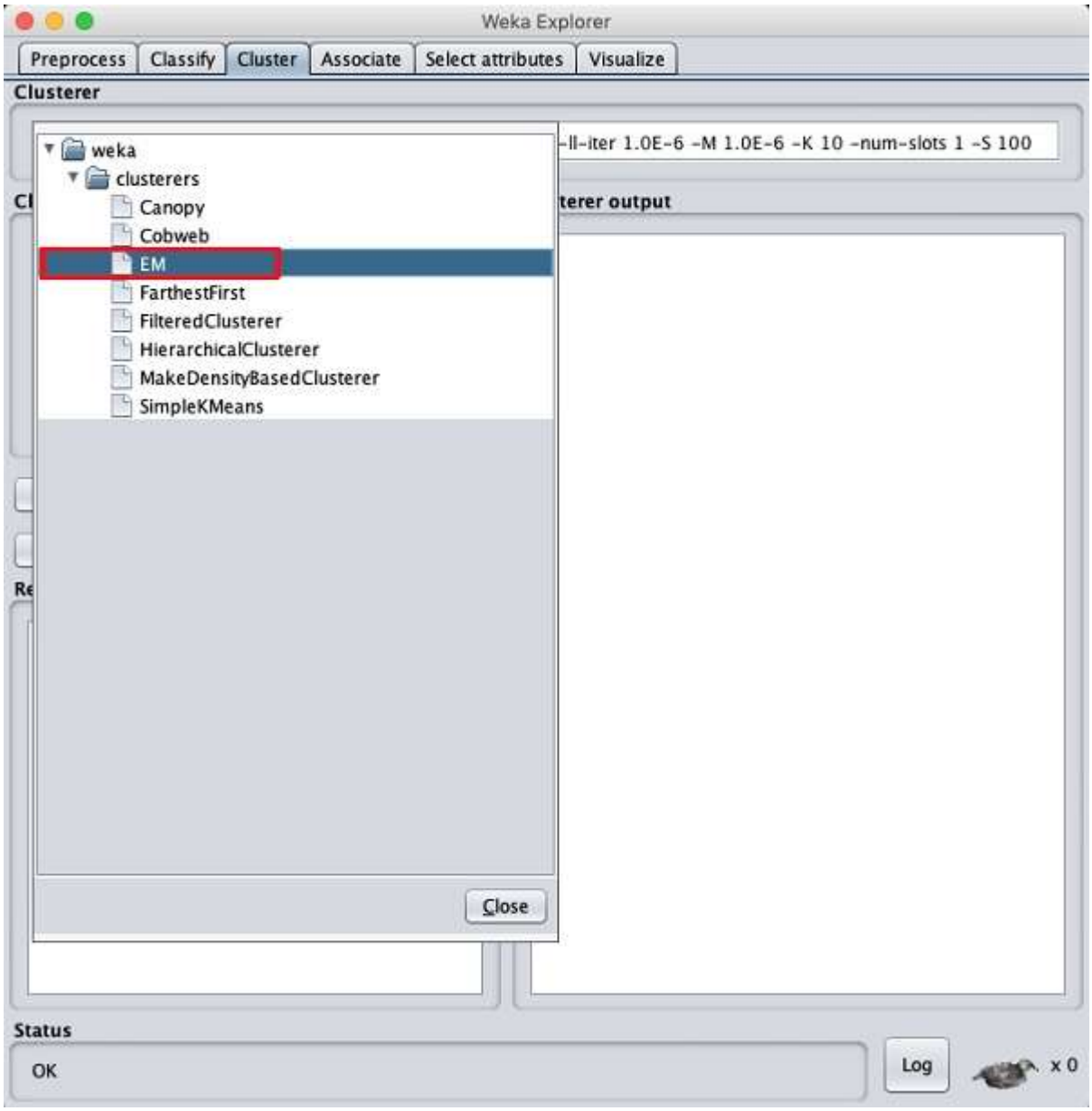
In the WEKA explorer select the **Preprocess** tab. Click on the **Open file** ... option and select the **iris.arff** file in the file selection dialog. When you load the data, the screen looks like as shown below −



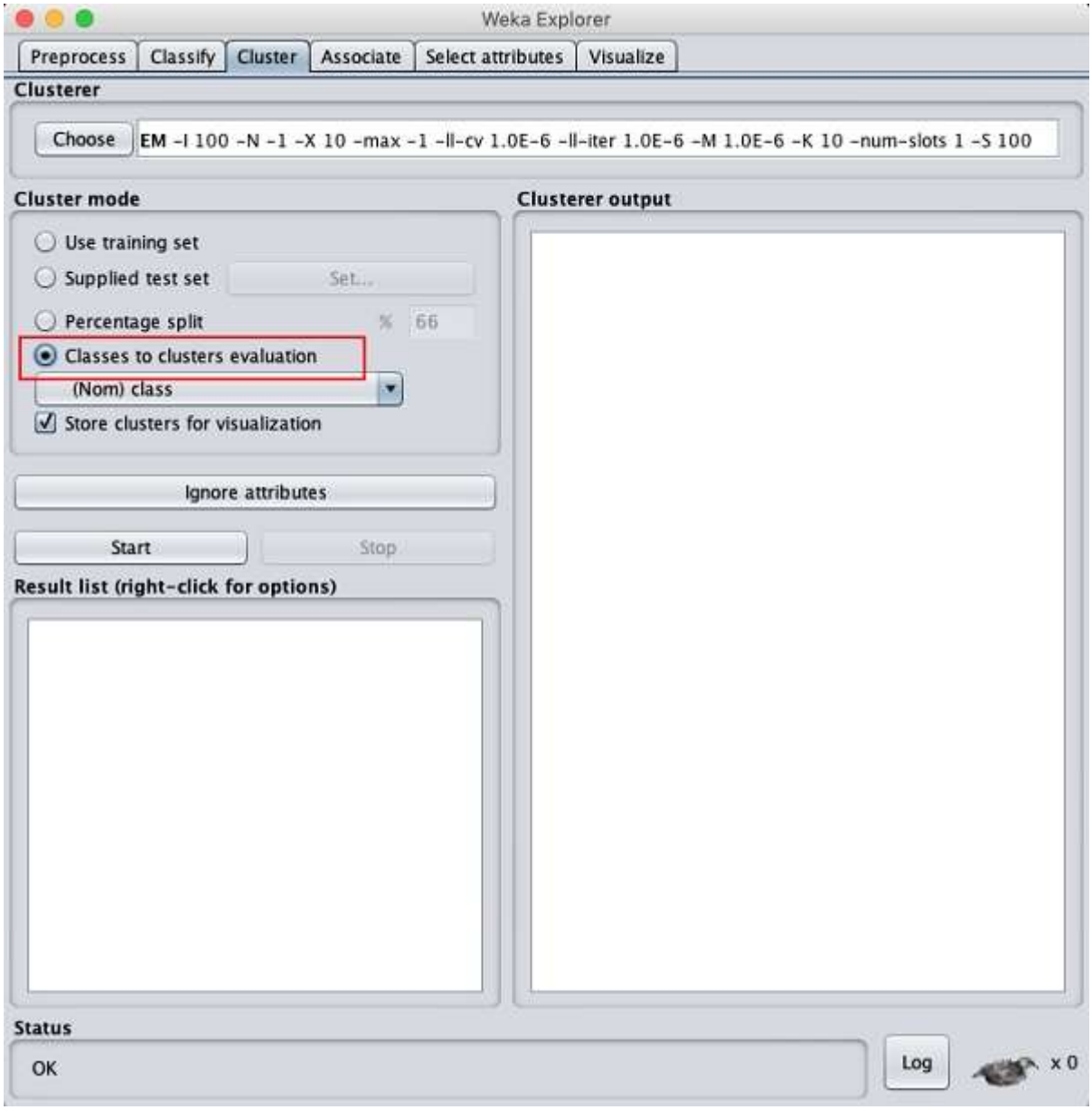
You can observe that there are 150 instances and 5 attributes. The names of attributes are listed as **sepallength**, **sepalwidth**, **petallength**, **petalwidth** and **class**. The first four attributes are of numeric type while the class is a nominal type with 3 distinct values. Examine each attribute to understand the features of the database. We will not do any preprocessing on this data and straight-away proceed to model building.

Clustering

Click on the **Cluster** TAB to apply the clustering algorithms to our loaded data. Click on the **Choose** button. You will see the following screen −



Now, select **EM** as the clustering algorithm. In the **Cluster mode** sub window, select the **Classes to clusters evaluation** option as shown in the screenshot below −

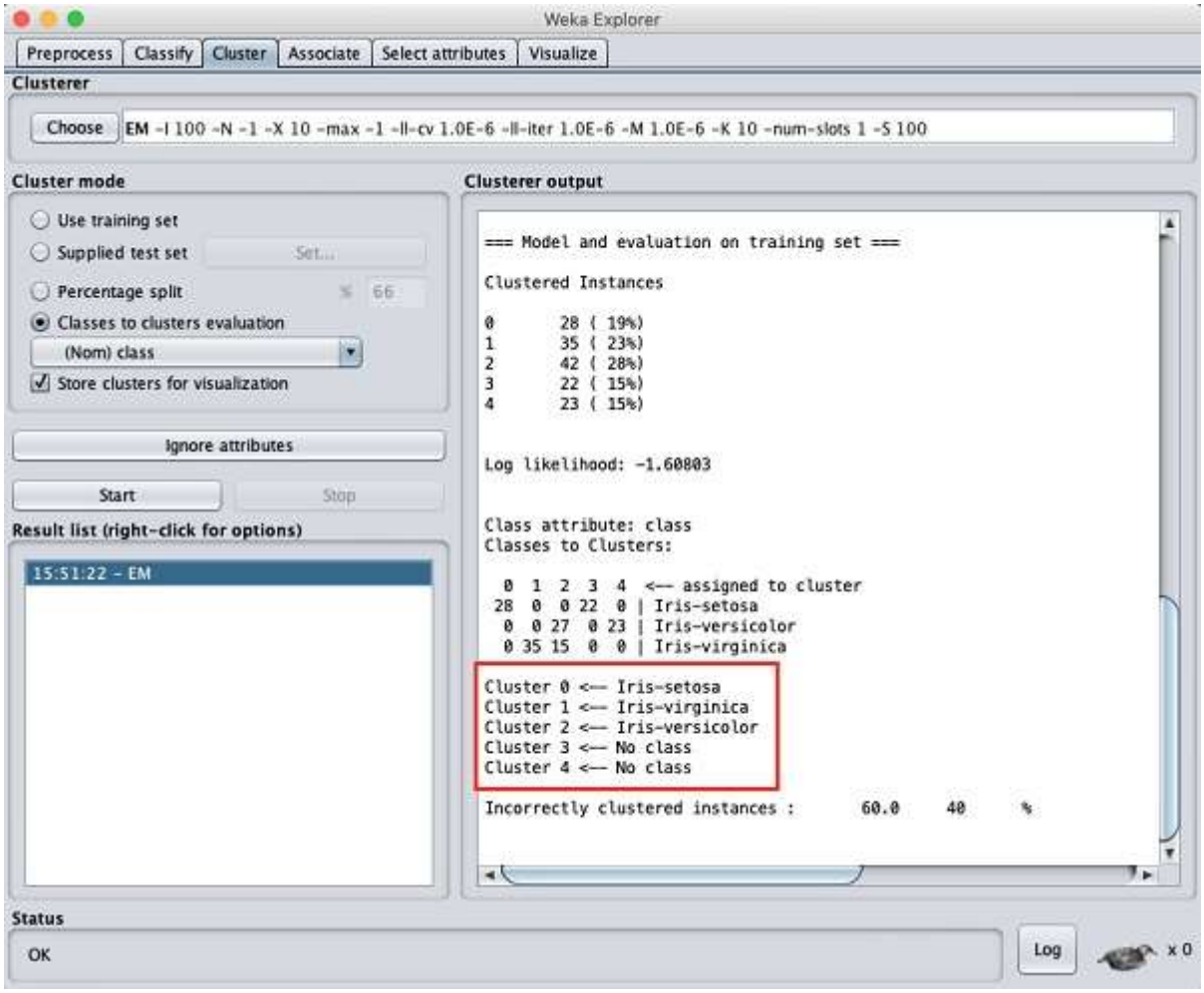


Click on the **Start** button to process the data. After a while, the results will be presented on the screen.

Next, let us study the results.

Examining Output

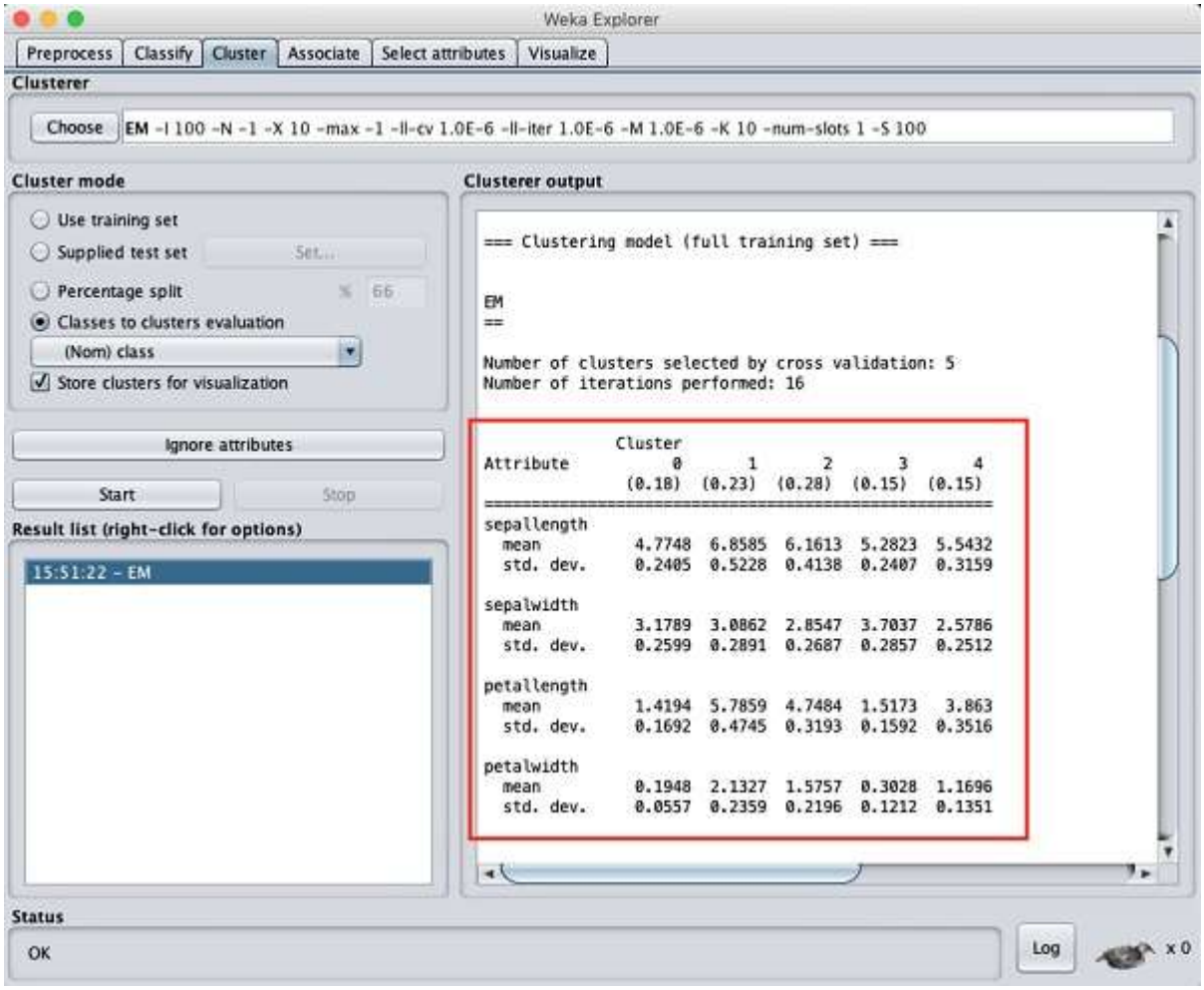
The output of the data processing is shown in the screen below −



From the output screen, you can observe that −

- There are 5 clustered instances detected in the database.
- The **Cluster 0** represents setosa, **Cluster 1** represents virginica, **Cluster 2** represents versicolor, while the last two clusters do not have any class associated with them.

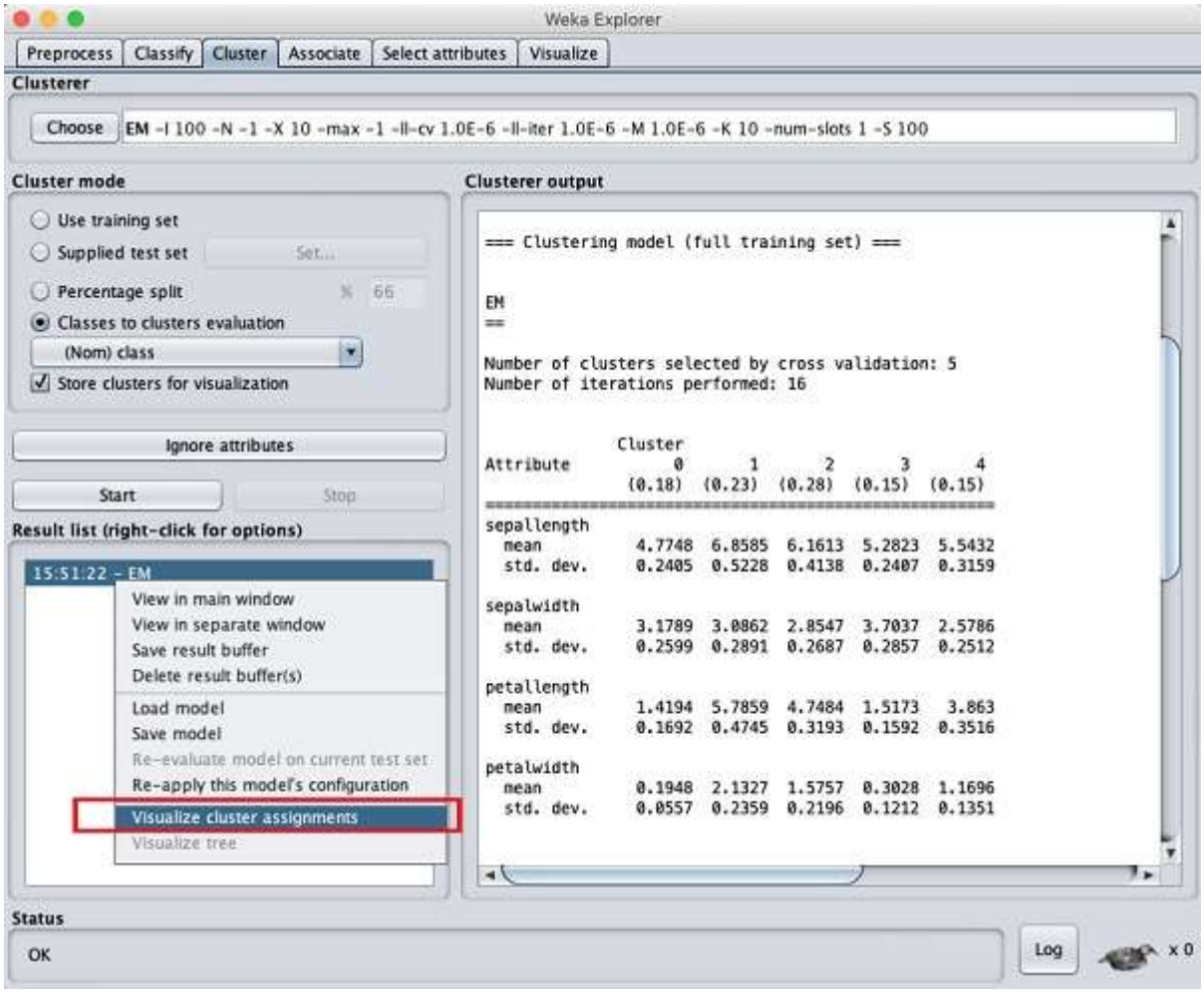
If you scroll up the output window, you will also see some statistics that gives the mean and standard deviation for each of the attributes in the various detected clusters. This is shown in the screenshot given below −



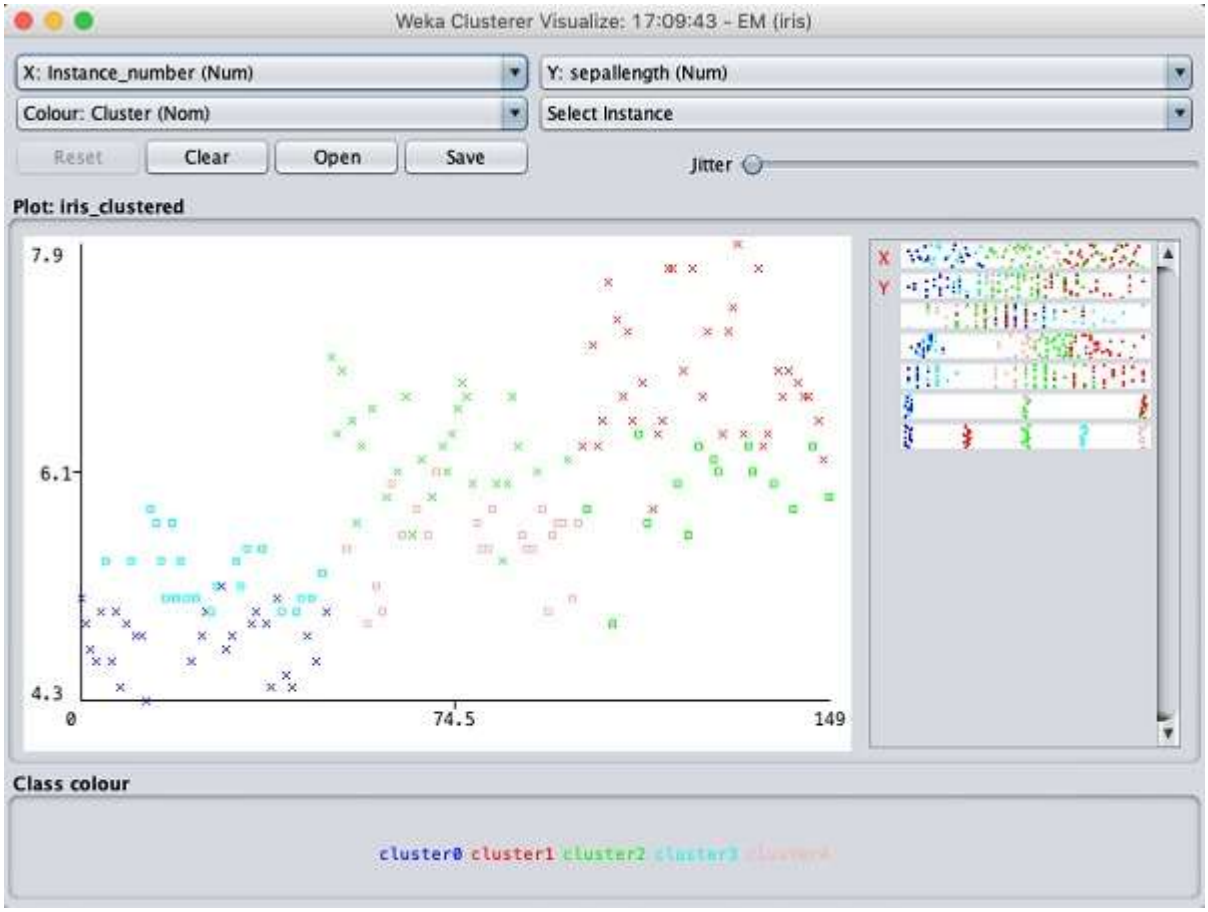
Next, we will look at the visual representation of the clusters.

Visualizing Clusters

To visualize the clusters, right click on the **EM** result in the **Result list**. You will see the following options −



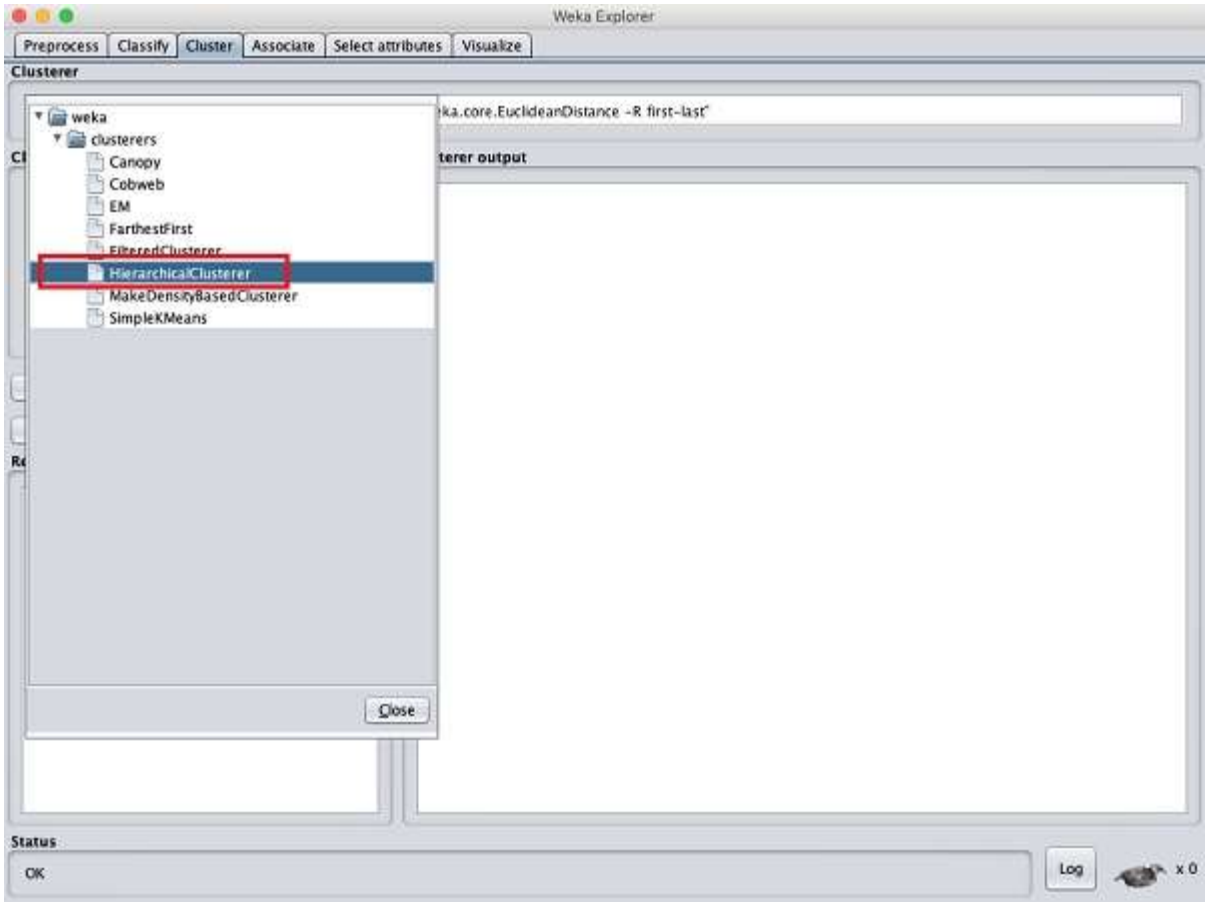
Select **Visualize cluster assignments**. You will see the following output −



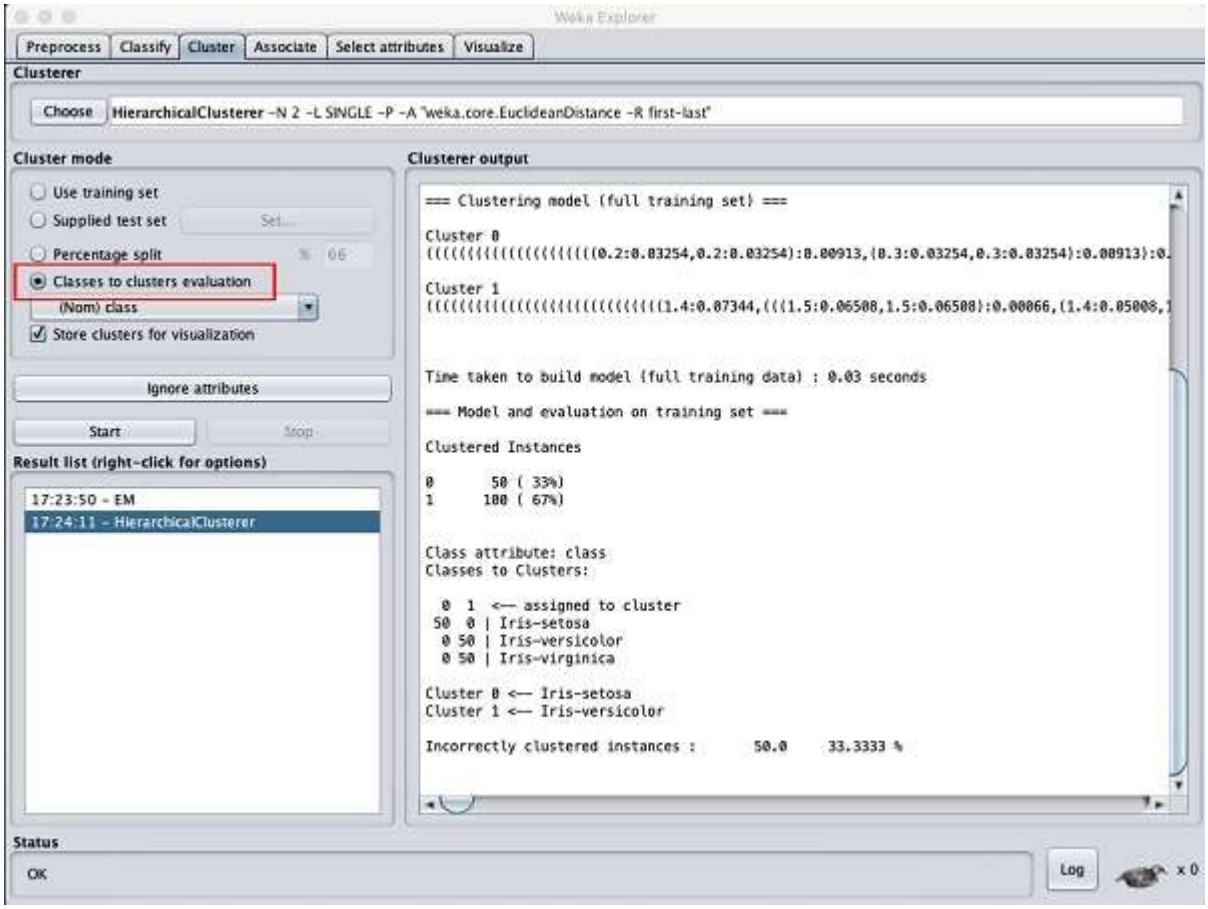
As in the case of classification, you will notice the distinction between the correctly and incorrectly identified instances. You can play around by changing the X and Y axes to analyze the results. You may use jittering as in the case of classification to find out the concentration of correctly identified instances. The operations in visualization plot are similar to the one you studied in the case of classification.

Applying Hierarchical Clusterer

To demonstrate the power of WEKA, let us now look into an application of another clustering algorithm. In the WEKA explorer, select the **HierarchicalClusterer** as your ML algorithm as shown in the screenshot shown below −

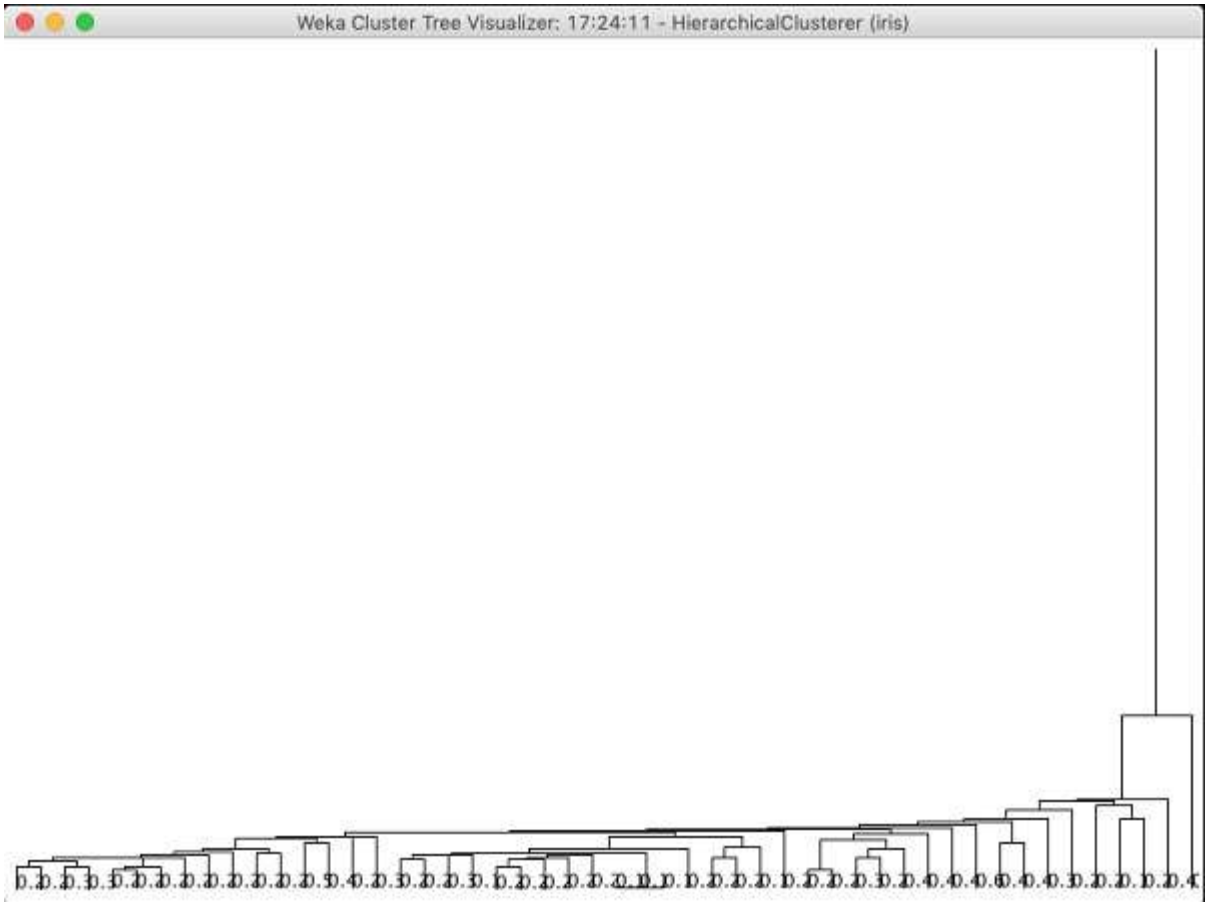


Choose the **Cluster mode** selection to **Classes to cluster evaluation**, and click on the **Start** button. You will see the following output −



Notice that in the **Result list**, there are two results listed: the first one is the EM result and the second one is the current Hierarchical. Likewise, you can apply multiple ML algorithms to the same dataset and quickly compare their results.

If you examine the tree produced by this algorithm, you will see the following output −


Weka Cluster Tree Visualizer: 17:24:11 - HierarchicalClusterer (iris)

Task 5: Demonstrate performing Regression on data sets

We are going to take a tour of 5 top regression algorithms in Weka.

Each algorithm that we cover will be briefly described in terms of how it works, key algorithm parameters will be highlighted and the algorithm will be demonstrated in the Weka Explorer interface.

The 5 algorithms that we will review are:

1. Linear Regression
2. k-Nearest Neighbors
3. Decision Tree
4. Support Vector Machines
5. Multi-Layer Perceptron

These are 5 algorithms that you can try on your regression problem as a starting point.

A standard machine learning regression problem will be used to demonstrate each algorithm.

Specifically, the Boston House Price Dataset. Each instance describes the properties of a Boston suburb and the task is to predict the house prices in thousands of dollars. There are 13 numerical input variables with varying scales describing the properties of suburbs. You can learn more about this dataset on the UCI Machine Learning Repository.

Start the Weka Explorer:

1. Open the Weka GUI Chooser.
2. Click the "Explorer" button to open the Weka Explorer.
3. Load the Boston house price dataset from the *housing.arff* file.
4. Click "Classify" to open the Classify tab.

Let's start things off by looking at the linear regression algorithm.

Linear Regression

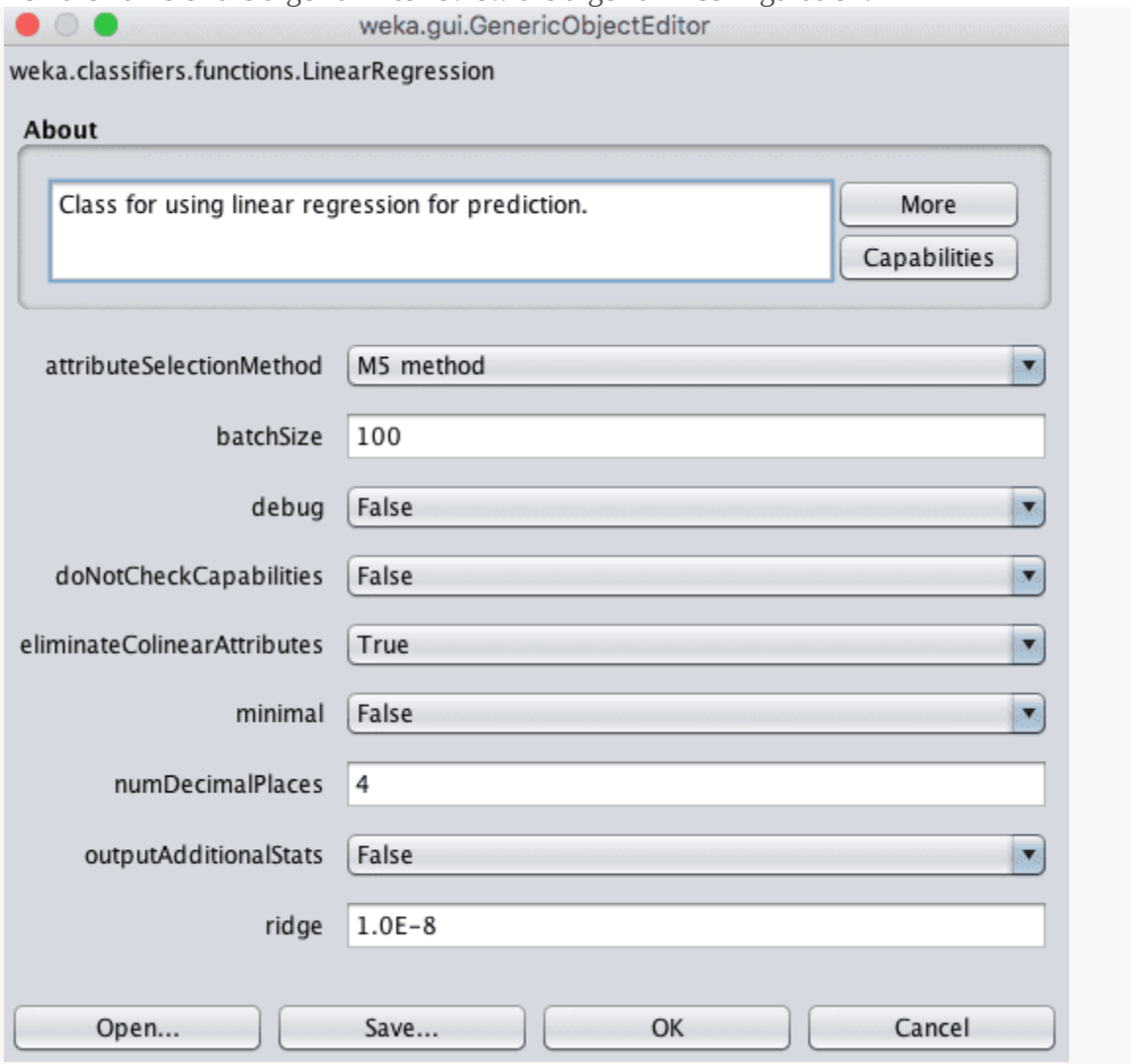
Linear regression only supports regression type problems.

It works by estimating coefficients for a line or hyperplane that best fits the training data. It is a very simple regression algorithm, fast to train and can have great performance if the output variable for your data is a linear combination of your inputs.

It is good idea to evaluate linear regression on your problem before moving onto more complex algorithms in case it performs well.

Choose the linear regression algorithm:

1. Click the “Choose” button and select “LinearRegression” under the “functions” group.
2. Click on the name of the algorithm to review the algorithm configuration.



Weka Configuration of Linear Regression

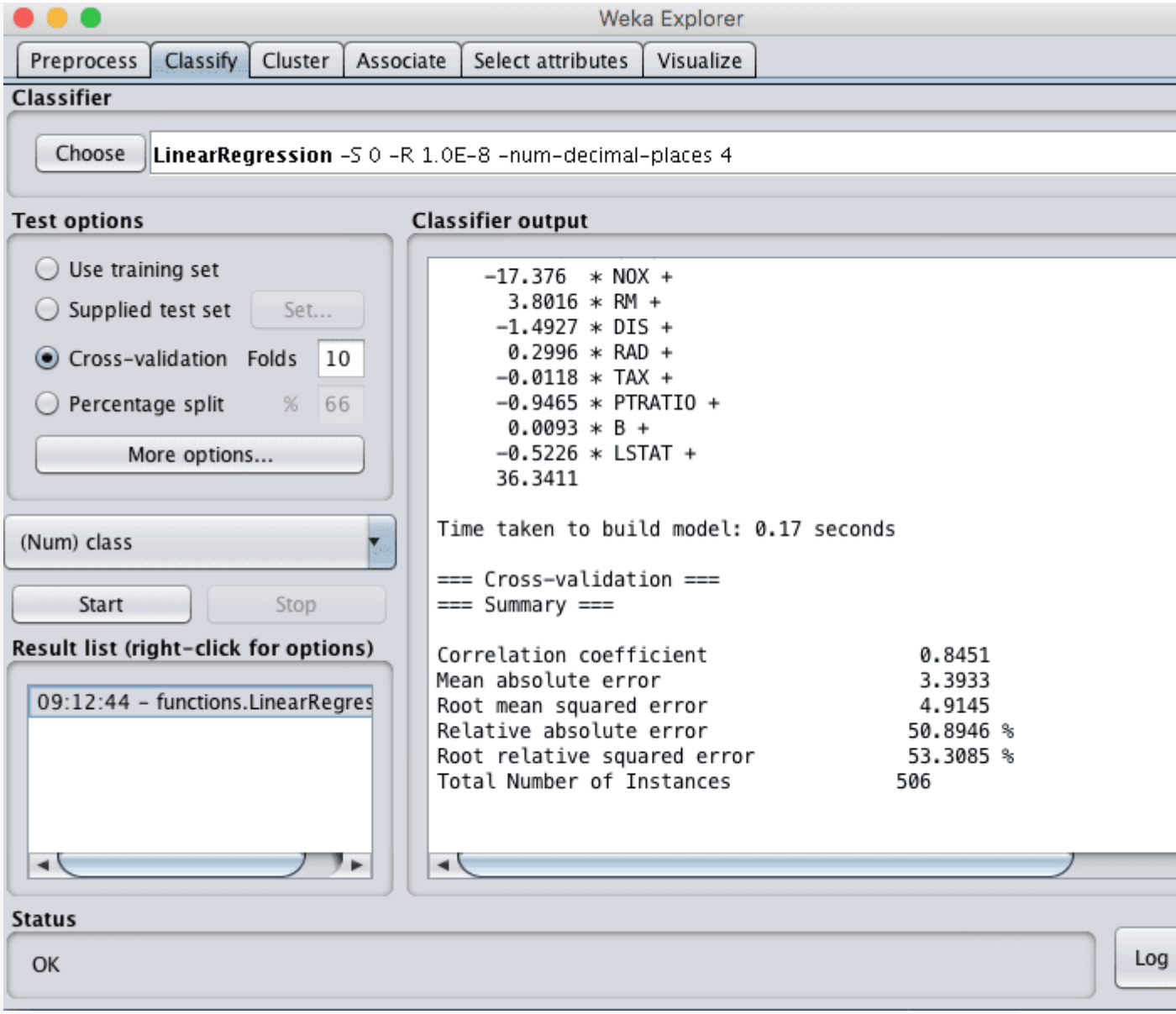
The performance of linear regression can be reduced if your training data has input attributes that are highly correlated. Weka can detect and remove highly correlated input attributes automatically by setting eliminateColinearAttributes to True, which is the default.

Additionally, attributes that are unrelated to the output variable can also negatively impact performance. Weka can automatically perform feature selection to only select those relevant attributes by setting the attributeSelectionMethod. This is enabled by default and can be disabled.

Finally, the Weka implementation uses a ridge regularization technique in order to reduce the complexity of the learned model. It does this by minimizing the square of the absolute sum of the learned coefficients, which will prevent any specific coefficient from becoming too large (a sign of complexity in regression models).

1. Click “OK” to close the algorithm configuration.
2. Click the “Start” button to run the algorithm on the Boston house price dataset.

   You can see that with the default configuration that linear regression achieves an RMSE of 4.9.



Weka Results for Linear Regression

k-Nearest Neighbors

The k-nearest neighbors algorithm supports both classification and regression. It is also called kNN for short. It works by storing the entire training dataset and querying it to locate the k most similar training patterns when making a prediction.

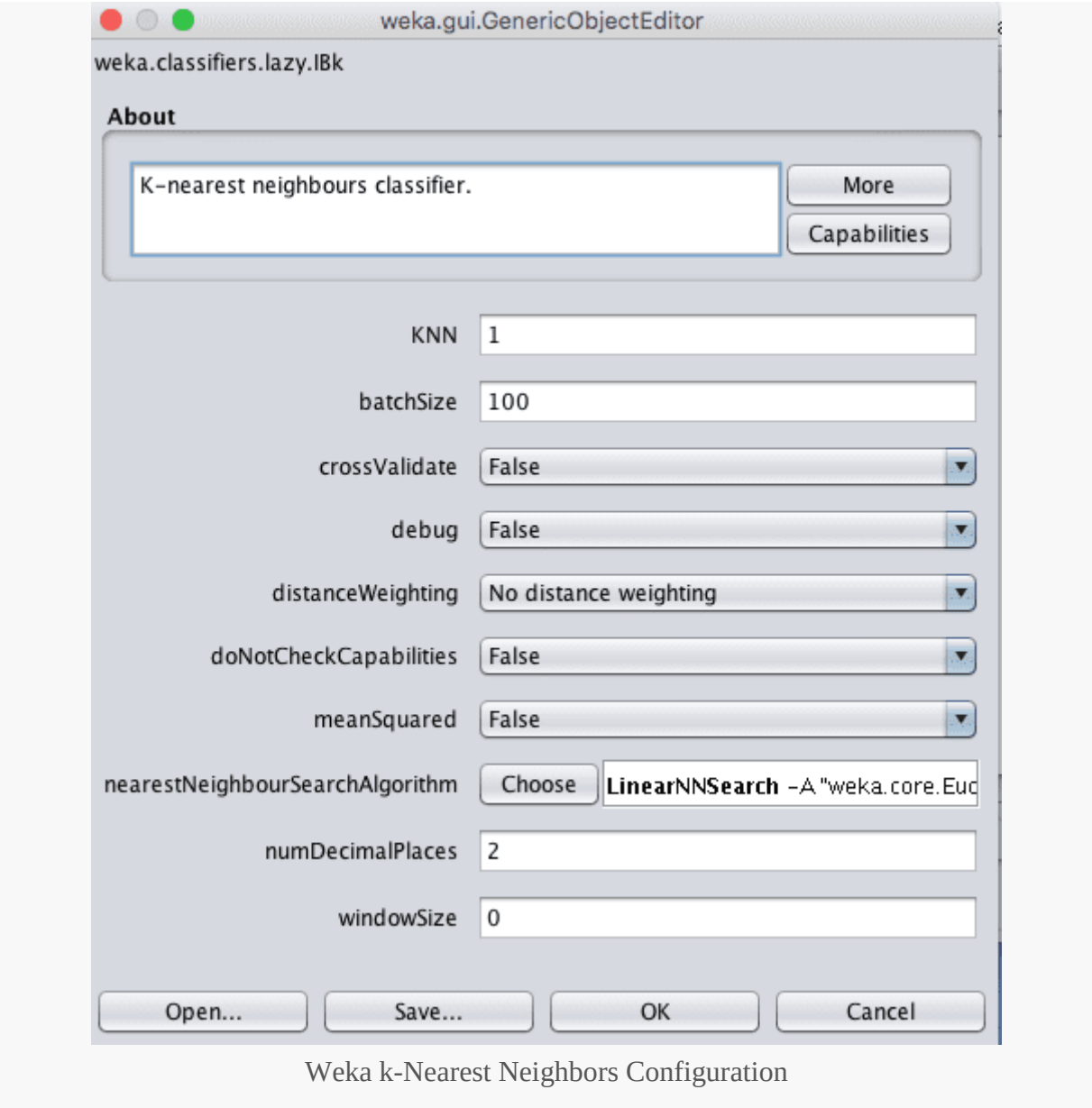
As such, there is no model other than the raw training dataset and the only computation performed is the querying of the training dataset when a prediction is requested.

It is a simple algorithm, but one that does not assume very much about the problem other than that the distance between data instances is meaningful in making predictions. As such, it often achieves very good performance.

When making predictions on regression problems, KNN will take the mean of the k most similar instances in the training dataset. Choose the KNN algorithm:

1. Click the “Choose” button and select “IBk” under the “lazy” group.
2. Click on the name of the algorithm to review the algorithm configuration.

In Weka KNN is called IBk which stands for Instance Based k.



Weka k-Nearest Neighbors Configuration

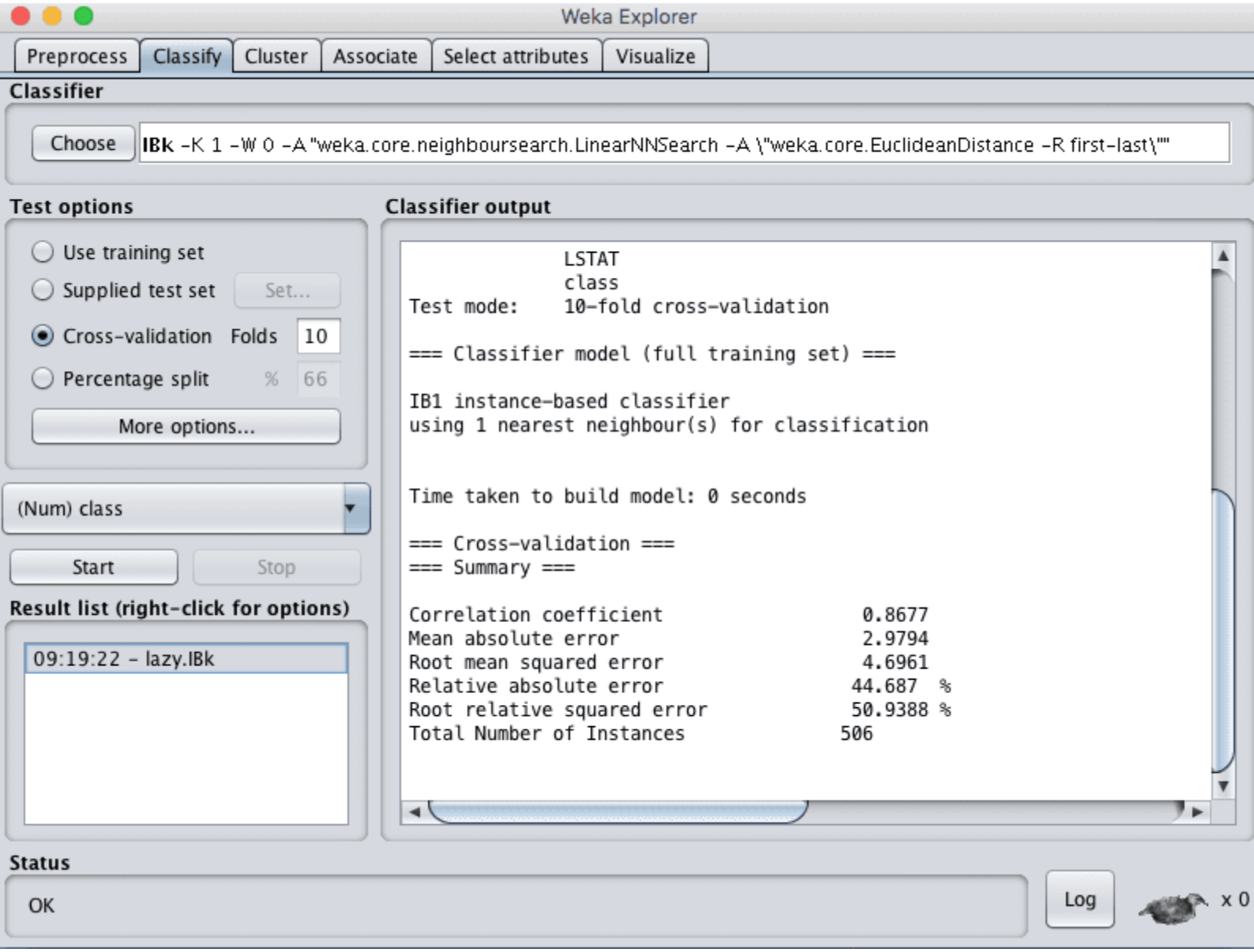
The size of the neighborhood is controlled by the k parameter. For example, if set to 1, then predictions are made using the single most similar training instance to a given new pattern for which a prediction is requested. Common values for k are 3, 7, 11 and 21, larger for larger

dataset sizes. Weka can automatically discover a good value for k using cross validation inside the algorithm by setting the crossValidate parameter to True.

Another important parameter is the distance measure used. This is configured in the nearestNeighbourSearchAlgorithm which controls the way in which the training data is stored and searched. The default is a LinearNNSearch. Clicking the name of this search algorithm will provide another configuration window where you can choose a distanceFunction parameter. By default, Euclidean distance is used to calculate the distance between instances, which is good for numerical data with the same scale. Manhattan distance is good to use if your attributes differ in measures or type.

It is a good idea to try a suite of different k values and distance measures on your problem and see what works best.

1. Click “OK” to close the algorithm configuration.
2. Click the “Start” button to run the algorithm on the Boston house price dataset.
   You can see that with the default configuration that KNN algorithm achieves an RMSE of 4.6.



Weka Regression Results for the k-Nearest Neighbors Algorithm

Decision Tree

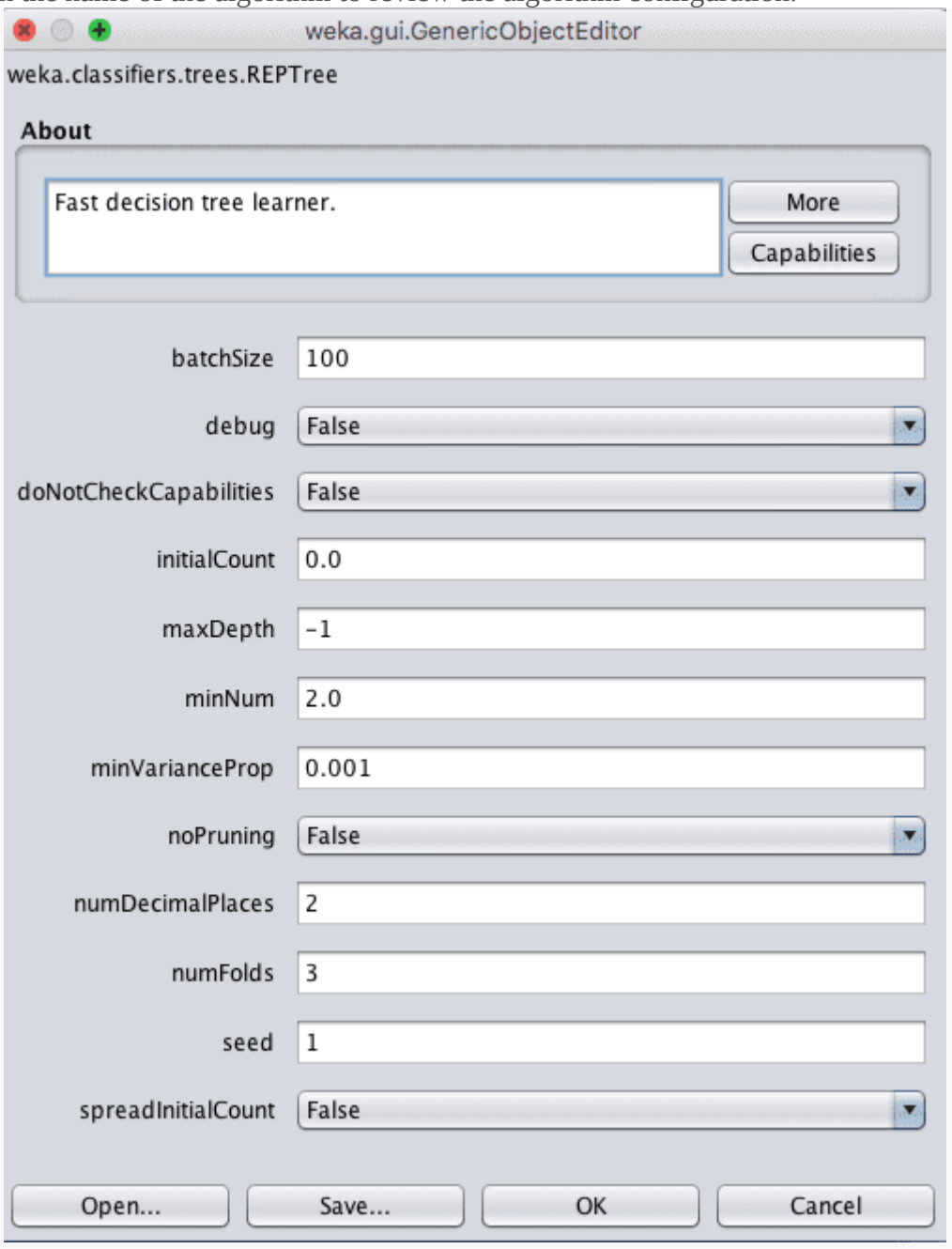
Decision trees can support classification and regression problems.

Decision trees are more recently referred to as Classification And Regression Trees or CART. They work by creating a tree to evaluate an instance of data, start at the root of the tree and moving town to the leaves (roots because the tree is drawn with an inverted prospective) until a prediction can be made. The process of creating a decision tree works by greedily selecting the best split point in order to make predictions and repeating the process until the tree is a fixed depth.

After the tree is construct, it is pruned in order to improve the model's ability to generalize to new data.

Choose the decision tree algorithm:

1. Click the "Choose" button and select "REPTree" under the "trees" group.
2. Click on the name of the algorithm to review the algorithm configuration.



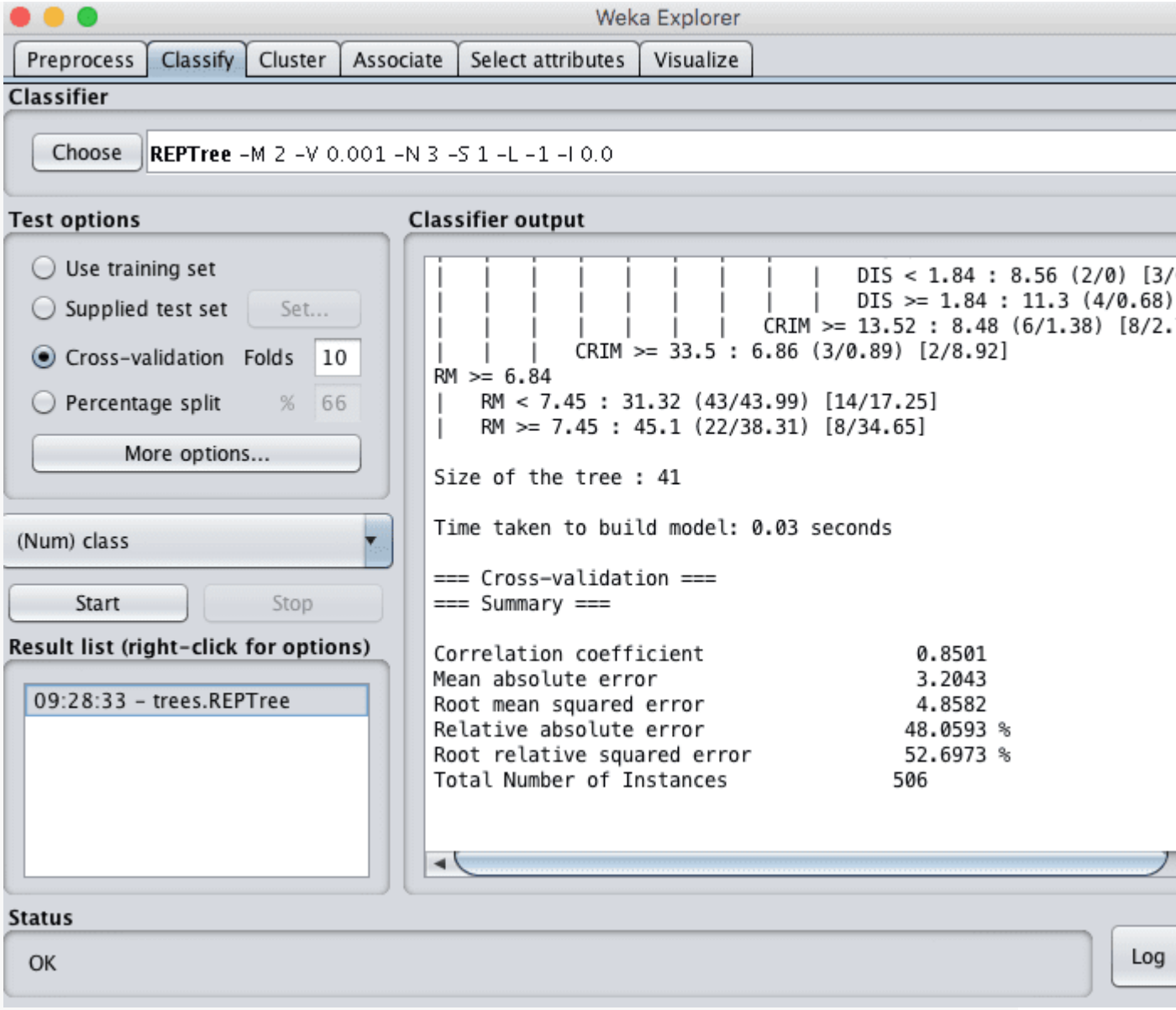
Weka Configuration for Decision Tree Algorithm

The depth of the tree is defined automatically, but can specify a depth in the maxDepth attribute.

You can also choose to turn off pruning by setting the noPruning parameter to True, although this may result in worse performance.

The minNum parameter defines the minimum number of instances supported by the tree in a leaf node when constructing the tree from the training data.

1. Click “OK” to close the algorithm configuration.
2. Click the “Start” button to run the algorithm on the Boston house price dataset.
   You can see that with the default configuration that decision tree algorithm achieves an RMSE of 4.8.



Weka Regression Results for the Decision Tree Algorithm

Support Vector Regression

Support Vector Machines were developed for binary classification problems, although extensions to the technique have been made to support multi-class classification and regression problems. The adaptation of SVM for regression is called Support Vector Regression or SVR for short.

SVM was developed for numerical input variables, although will automatically convert nominal values to numerical values. Input data is also normalized before being used.

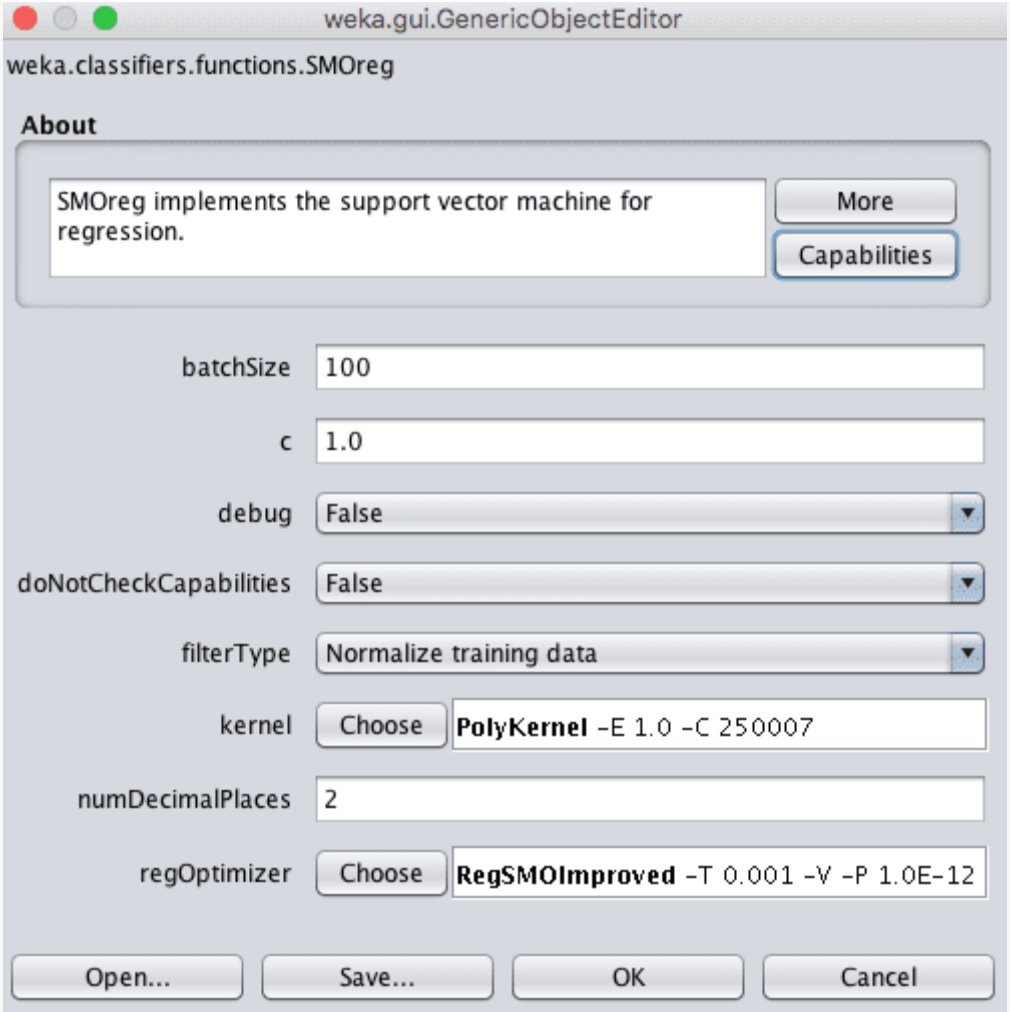
Unlike SVM that finds a line that best separates the training data into classes, SVR works by finding a line of best fit that minimizes the error of a cost function. This is done using an optimization process that only considers those data instances in the training dataset that are closest to the line with the minimum cost. These instances are called support vectors, hence the name of the technique.

In almost all problems of interest, a line cannot be drawn to best fit the data, therefore a margin is added around the line to relax the constraint, allowing some bad predictions to be tolerated but allowing a better result overall.

Finally, few datasets can be fit with just a straight line. Sometimes a line with curves or even polygonal regions need to be marked out. This is achieved by projecting the data into a higher dimensional space in order to draw the lines and make predictions. Different kernels can be used to control the projection and the amount of flexibility.

Choose the SVR algorithm:

1. Click the “Choose” button and select “SMOreg” under the “function” group.
2. Click on the name of the algorithm to review the algorithm configuration.



Weka Configuration for the Support Vector Regression Algorithm

The C parameter, called the complexity parameter in Weka controls how flexible the process for drawing the line to fit the data can be. A value of 0 allows no violations of the margin, whereas the default is 1.

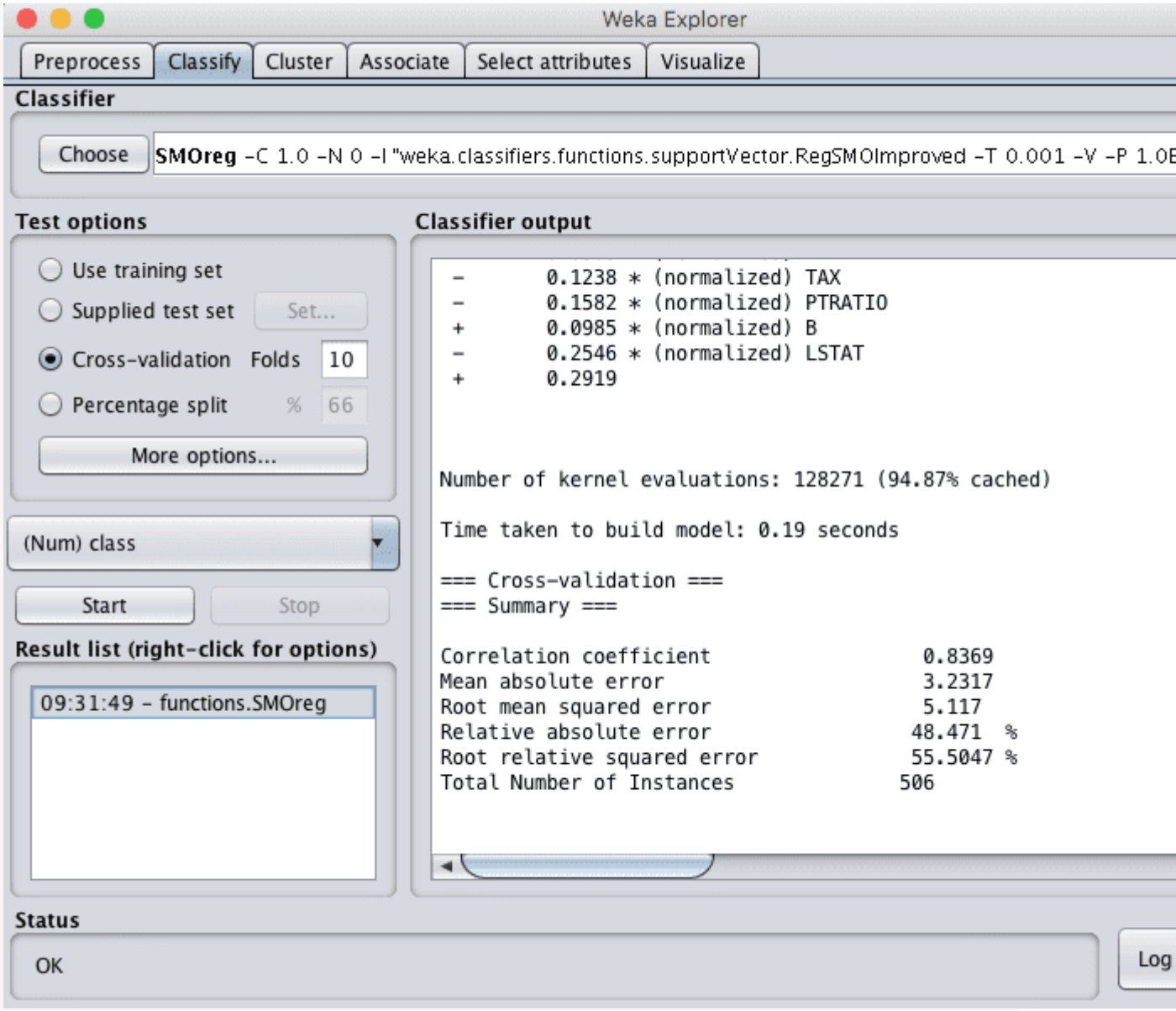
A key parameter in SVM is the type of Kernel to use. The simplest kernel is a Linear kernel that separates data with a straight line or hyperplane. The default in Weka is a Polynomial Kernel that will fit the data using a curved or wiggly line, the higher the polynomial, the more wiggly (the exponent value).

The Polynomial Kernel has a default exponent of 1, which makes it equivalent to a linear kernel. A popular and powerful kernel is the RBF Kernel or Radial Basis Function Kernel that is capable of learning closed polygons and complex shapes to fit the training data.

It is a good idea to try a suite of different kernels and C (complexity) values on your problem and see what works best.

1. Click “OK” to close the algorithm configuration.

2. Click the “Start” button to run the algorithm on the Boston house price dataset.
   You can see that with the default configuration that SVR algorithm achieves an RMSE of 5.1.



Weka Regression Results for the Support Vector Regression Algorithm

Multi-Layer Perceptron

The Multi-Layer Perceptron algorithms supports both regression and classification problems.

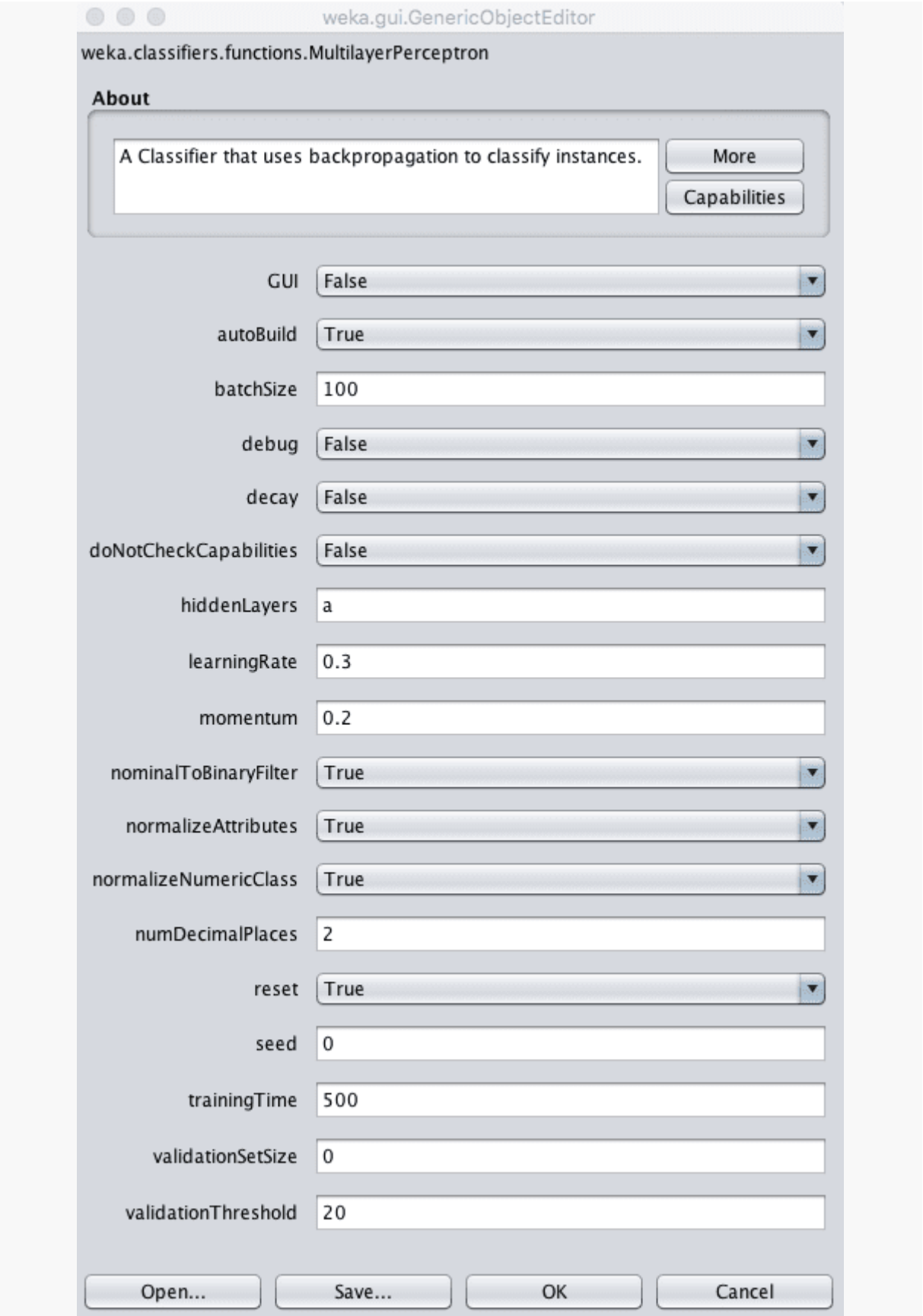
It is also called artificial neural networks or simply neural networks for short.

Neural networks are a complex algorithm to use for predictive modeling because there are so many configuration parameters that can only be tuned effectively through intuition and a lot of trial and error.

It is an algorithm inspired by a model of biological neural networks in the brain where small processing units called neurons are organized into layers that if configured well are capable of approximating any function. In classification we are interested in approximating the underlying function to best discriminate between classes. In regression problems we are interested in approximating a function that best fits the real value output.

Choose the Multi-Layer Perceptron algorithm:

1. Click the “Choose” button and select “MultilayerPerceptron” under the “function” group.
2. Click on the name of the algorithm to review the algorithm configuration.

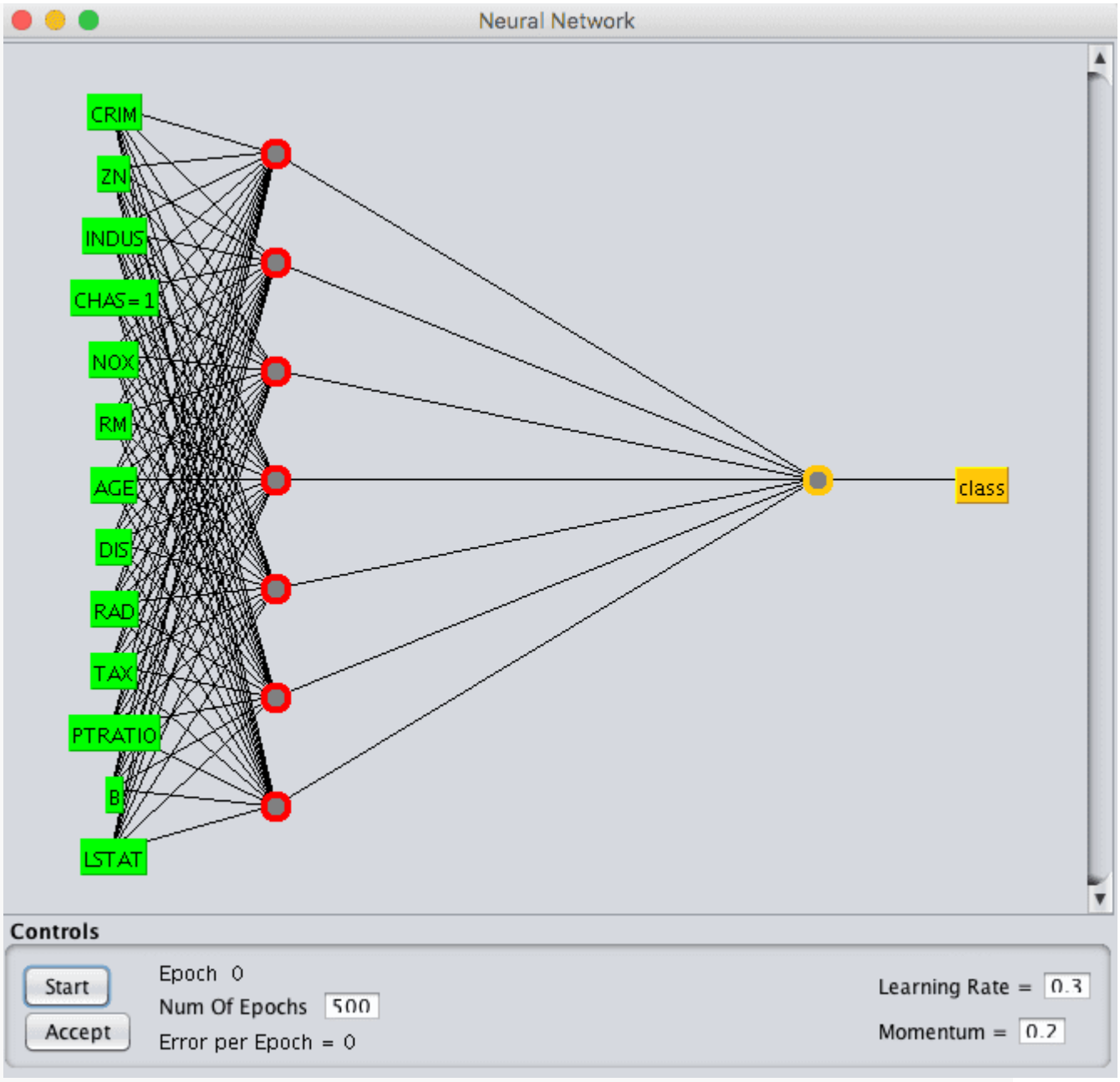


Weka Configuration for the Multi-Layer Perceptron Algorithm

You can manually specify the structure of the neural network that is used by the model, but this is not recommended for beginners.

The default will automatically design the network and train it on your dataset. The default will create a single hidden layer network. You can specify the number of hidden layers in the hiddenLayers parameter, set to automatic “a” by default.

You can also use a GUI to design the network structure. This can be fun, but it is recommended that you use the GUI with a simple train and test split of your training data, otherwise you will be asked to design a network for each of the 10 folds of cross validation.

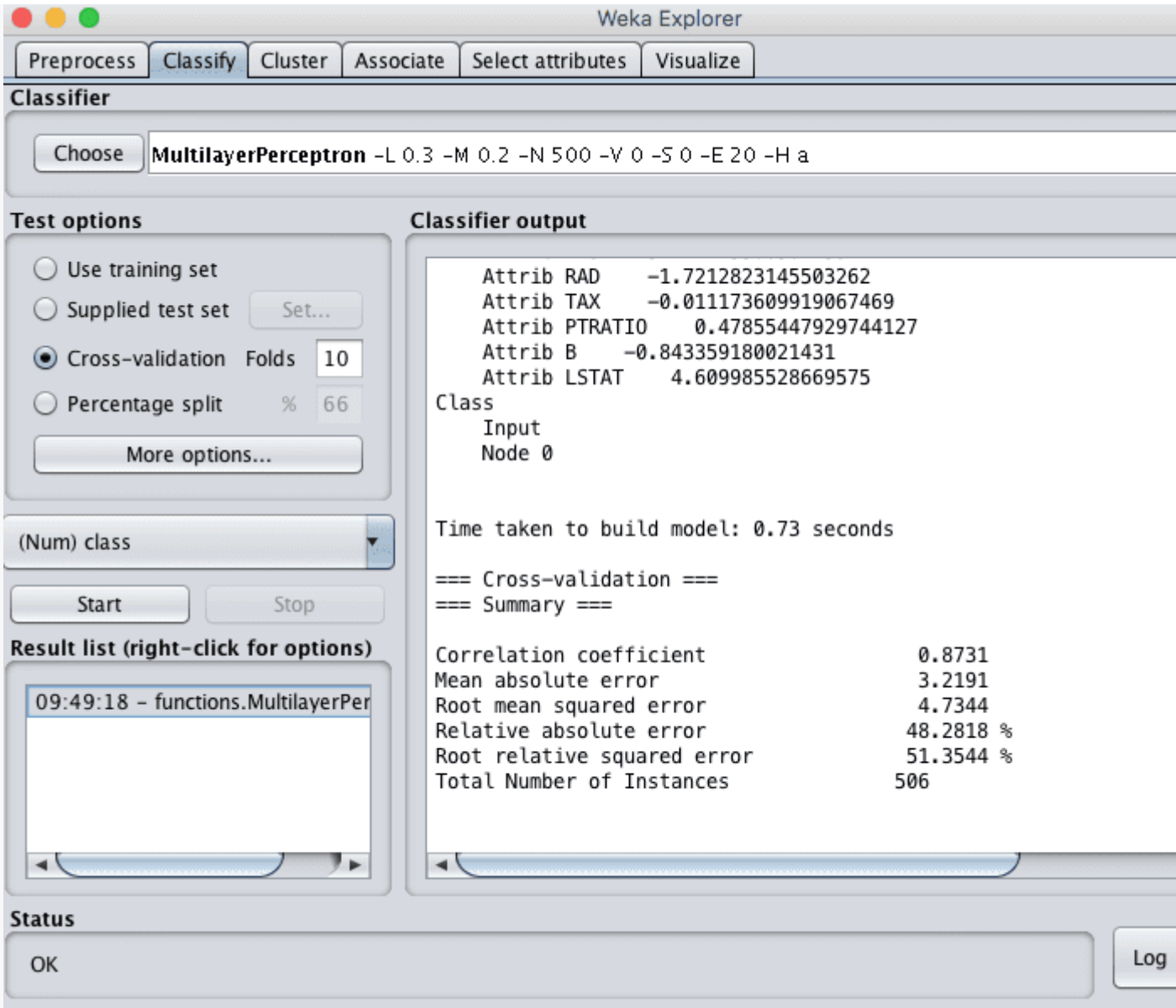


Weka GUI Designer for the Multi-Layer Perceptron Algorithm

You can configure the learning process by specifying how much to update the model each epoch by setting the learning rate. common values are small such as values between 0.3 (the default) and 0.1.

The learning process can be further tuned with a momentum (set to 0.2 by default) to continue updating the weights even when no changes need to be made, and a decay (set decay to True) which will reduce the learning rate over time to perform more learning at the beginning of training and less at the end.

1. Click “OK” to close the algorithm configuration.
2. Click the “Start” button to run the algorithm on the Boston house price dataset.
   You can see that with the default configuration that Multi-Layer Perceptron algorithm achieves an RMSE of 4.7.



Weka Regression Results Multi-Layer Perceptron Algorithm

ಭಾರತೀಯ ವಿಶಿಷ್ಟ ಗುರುತು ಪ್ರಾಧಿಕಾರ

ಭಾರತ ಸರ್ಕಾರ

Unique Identification Authority of India

Government of India

ನೋಂದಾವಣಿ ಕ್ರಮ ಸಂಖ್ಯೆ / Enrollment No.: 0821/89763/20017

To
ದರ್ಶನ್ ಎಚ್ ವಿ
Darshan H V
S/O. Vishwamurthy H B
#705 4th Stage BEML Layout
Rajarajeshwari nagar
Bangalore South
Rajarajeshwarinagar
Bangalore South Bengaluru
Karnataka 560098
9731569955

06/06/2017

1767165677

ME767165677FH

ನಿಮ್ಮ ಆಧಾರ್ ಸಂಖ್ಯೆ / Your Aadhaar No. :

6454 7576 5606

ನನ್ನ ಆಧಾರ್, ನನ್ನ ಗುರುತು

ಭಾರತ ಸರ್ಕಾರ

Government of India

ದರ್ಶನ್ ಎಚ್ ವಿ
Darshan H V
ಜನ್ಮ ದಿನಾಂಕ / DOB : 15/03/1990
ಪುರುಷ / Male

6454 7576 5606

ನನ್ನ ಆಧಾರ್, ನನ್ನ ಗುರುತು

6. Sample Programs using German Credit Data.

Task 1: Credit Risk Assessment

Description: The business of banks is making loans. Assessing the credit worthiness of an applicant is of crucial importance. You have to develop a system to help a loan officer decide **whether the credit of a customer is good. Or bad. A bank's business rules regarding loans must** consider two opposing factors. On th one han, a bank wants to make as many loans as possible.

Interest on these loans is the banks profit source. On the other hand, a bank can not afford to make too many bad loans. Too many bad loans could lead to the collapse of the bank. Th**e bank's** loan policy must involved a compromise. Not too strict and not too lenient.

To do the assignment, you first and foremost need some knowledge about the world of credit. You can acquire such knowledge in a number of ways.

1. Knowledge engineering: Find a loan officer who is willing to talk. Interview her and try to represent her knowledge in a number of ways.

2. Books: Find some training manuals for loan officers or perhaps a suitable textbook on finance. Translate this knowledge from text from to production rule form.

3. Common sense: Imagine yourself as a loan officer and make up reasonable rules which can be used to judge the credit worthiness of a loan applicant.

4. Case histories: Find records of actual cases where competent loan officers correctly judged when and not to. Approve a loan application.

The German Credit Data

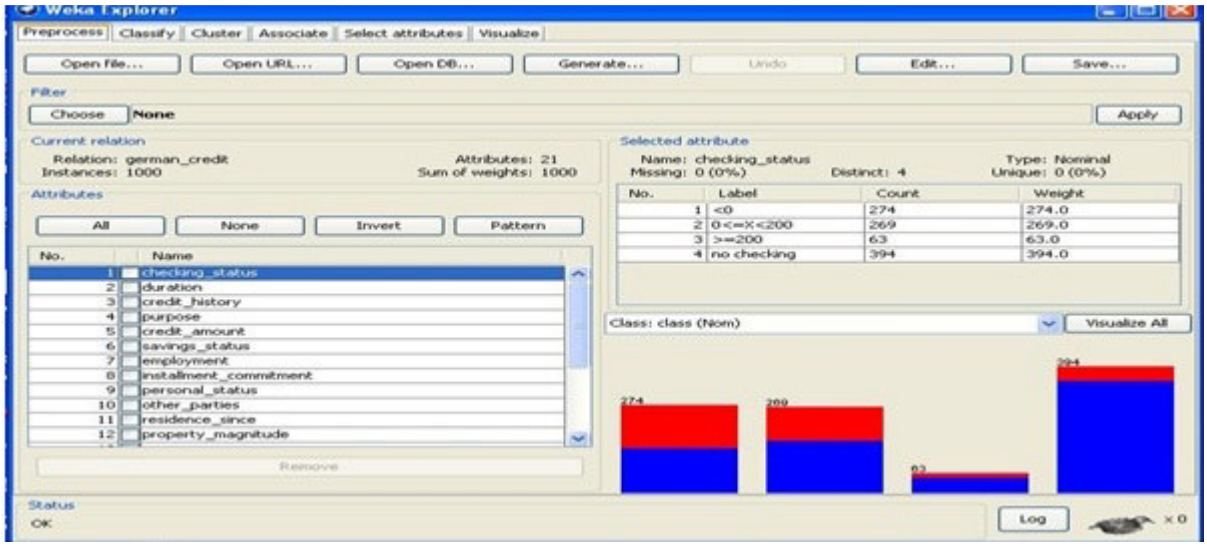
Actual historical credit data is not always easy to come by because of confidentiality rules. Here is one such data set. Consisting of **1000** actual cases collected in Germany.

In spite of the fact that the data is German, you should probably make use of it for this assignment(Unless you really can consult a real loan officer!)

There are 20 attributes used in judging a loan applicant(ie., 7 Numerical attributes and 13 Categoricl or Nominal attributes). The goal is the classify the applicant into one of two categories. Good or Bad.

The total number of attributes present in German credit data are.

1. Checking_Status
2. Duration
3. Credit_history
4. Purpose
5. Credit_amout
6. Savings_status
7. Employment
8. Installment_Commitment
9. Personal_status
10. Other_parties
11. Residence_since
12. Property_Magnitude
13. Age
14. Other_payment_plans
15. Housing
16. Existing_credits
17. Job
18. Num_dependents
19. Own_telephone
20. Foreign_worker
21. Class



Tasks(Turn in your answers to the following taks)

1. **List all the categorical (or nominal) attributes and the real valued attributes separately.**

Ans) Steps for identifying categorical attributes

1. Double click on credit-g.arff file.
2. Select all categorical attributes.
3. Click on invert.
4. Then we get all real valued attributes selected
5. Click on remove
6. Click on visualize all.

Steps for identifying real valued attributes

1. Double click on credit-g.arff file.
2. Select all real valued attributes.
3. Click on invert.
4. Then we get all categorial attributes selected
5. Click on remove
6. Click on visualize all.

The following are the Categorical (or Nominal) attributes)

1. Checking_Status
2. Credit_history
3. Purpose
4. Savings_status
5. Employment
6. Personal_status
7. Other_parties
8. Property_Magnitude
9. Other_payment_plans
10. Housing
11. Job

12. Own_telephone
13. Foreign_worker

The following are the Numerical attributes)

1. Duration
2. Credit_amout
3. Installment_Commitment
4. Residence_since
5. Age
6. Existing_credits
7. Num_dependents

2. What attributes do you think might be crucial in making the credit assessment? Come up with some simple rules in plain English using your selected attributes.

Ans) The following are the attributes may be crucial in making the credit assessment.

1. Credit_amount
2. Age
3. Job
4. Savings_status
5. Existing_credits
6. Installment_commitment
7. Property_magnitude

3. One type of model that you can create is a Decision tree . train a Decision tree using the complete data set as the training data. Report the model obtained after training.

Ans) Steps to model decision tree.

1. Double click on credit-g.arff file.
2. Consider all the 21 attributes for making decision tree.
3. Click on classify tab.
4. Click on choose button.
5. Expand tree folder and select J48
6. Click on use training set in test options.
7. Click on start button.
8. Right click on result list and choose the visualize tree to get decision tree.

We created a decision tree by using J48 Technique for the complete dataset as the training data.

The following model obtained after training.

Output:

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: german_credit
Instances: 1000
Attributes: 21

Checking_status duration credit_history purpose credit_amount savings_status employment installment_commitment personal_status other_parties residence_since property_magnitude age other_payment_plans housing existing_credits job num_dependents own_telephone foreign_worker class

Test mode: evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree

Number of Leaves : 103

Size of the tree : 140

Time taken to build model: 0.08 seconds

=== **Evaluation on training set** ===

=== **Summary** ===

Correctly Classified Instances	855	85.5 %
Incorrectly Classified Instances	145	14.5 %

Kappa statistic 0.6251
Mean absolute error 0.2312
Root mean squared error 0.34
Relative absolute error 55.0377 %
Root relative squared error 74.2015 %
Coverage of cases (0.95 level) 100 %
Mean rel. region size (0.95 level) 93.3 %
Total Number of Instances 1000

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.956	0.38	0.854	0.956	0.902	0.857	good
	0.62	0.044	0.857	0.62	0.72	0.857	bad
WeightedAvg.	0.855	0.279	0.855	0.855	0.847	0.857	

=== Confusion Matrix ===

```
  a b <-- classified as 669
 31 | a = good

 114   186 |  b = bad
```

4. Suppose you use your above model trained on the complete dataset, and classify credit good/bad for each of the examples in the dataset. What % of examples can you classify correctly?(This is also called testing on the training set) why do you think can not get 100% training accuracy?

Ans) Steps followed are:

1. Double click on credit-g.arff file.
2. Click on classify tab.
3. Click on choose button.
4. Expand tree folder and select J48
5. Click on use training set in test options.
6. Click on start button.
7. On right side we find confusion matrix
8. Note the correctly classified instances.

Output:

If we used our above model trained on the complete dataset and classified credit as good/bad for each of the examples in that dataset. We can not get 100% training accuracy only **85.5%** of examples, we can classify correctly.

5. Is testing on the training set as you did above a good idea? Why or why not?

Ans)It is not good idea by using 100% training data set.

6. One approach for solving the problem encountered in the previous question is using cross-validation? Describe what is cross validation briefly. Train a decision tree again using cross validation and report your results. Does accuracy increase/decrease? Why?

Ans) steps followed are:

1. Double click on credit-g.arff file.
2. Click on classify tab.
3. Click on choose button.
4. Expand tree folder and select J48
5. Click on cross validations in test options.
6. Select folds as 10
7. Click on start
8. Change the folds to 5
9. Again click on start
10. Change the folds with 2
11. Click on start.
12. Right click on blue bar under result list and go to visualize tree

Output:

Cross-Validation Definition: The classifier is evaluated by cross validation using the number of folds that are entered in the folds text field.

In Classify Tab, Select cross-validation option and folds size is 2 then Press Start Button, next time change as folds size is 5 then press start, and next time change as folds size is 10 then press start.

i) Fold Size-10

Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	705	70.5 %
Incorrectly Classified Instances	295	29.5 %
Kappa statistic	0.2467	
Mean absolute error	0.3467	
Root mean squared error	0.4796	
Relative absolute error	82.5233 %	
Root relative squared error	104.6565 %	
Coverage of cases (0.95 level)	92.8 %	
Mean rel. region size (0.95 level)	91.7 %	
Total Number of Instances	1000	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.84	0.61	0.763	0.84	0.799	0.639	good
	0.39	0.16	0.511	0.39	0.442	0.639	bad
Weighted Avg.	0.705	0.475	0.687	0.705	0.692	0.639	

=== Confusion Matrix ===

```
  a  b  <-- classified as
 588 112 |  a = good
 183 117 |  b = bad
```

ii) Fold Size-5

Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	733	73.3 %
Incorrectly Classified Instances	267	26.7 %
Kappa statistic	0.3264	
Mean absolute error	0.3293	
Root mean squared error	0.4579	
Relative absolute error	78.3705 %	
Root relative squared error	99.914 %	
Coverage of cases (0.95 level)	94.7 %	
Mean rel. region size (0.95 level)	93 %	
Total Number of Instances	1000	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.851	0.543	0.785	0.851	0.817	0.685	good
	0.457	0.149	0.568	0.457	0.506	0.685	bad
Weighted Avg.	0.733	0.425	0.72	0.733	0.724	0.685	

=== Confusion Matrix ===

a b <-- classified as
596 104 | a = good
163 137 | b = bad

iii) Fold Size-2

Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	721	72.1 %
Incorrectly Classified Instances	279	27.9 %
Kappa statistic	0.2443	
Mean absolute error	0.3407	
Root mean squared error	0.4669	
Relative absolute error	81.0491 %	
Root relative squared error	101.8806 %	
Coverage of cases (0.95 level)	92.8 %	
Mean rel. region size (0.95 level)	91.3 %	
Total Number of Instances	1000	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.891	0.677	0.755	0.891	0.817	0.662	good
	0.323	0.109	0.561	0.323	0.41	0.662	bad
Weighted Avg.	0.721	0.506	0.696	0.721	0.695	0.662	

=== Confusion Matrix ===

a b <-- classified as

624 76 | a = good
203 97 | b = bad

Note: With this observation, we have seen accuracy is increased when we have folds size is 5 and accuracy is decreased when we have 10 folds.

7. Check to see if the data shows a bias against "foreign workers" or "personal-status".

One way to do this is to remove these attributes from the data set and see if the decision tree created in those cases is significantly different from the full dataset case which you have already done. Did removing these attributes have any significantly effect? Discuss.

Ans) steps followed are:

1. Double click on credit-g.arff file.
2. Click on classify tab.
3. Click on choose button.
4. Expand tree folder and select J48
5. Click on cross validations in test options.
6. Select folds as 10
7. Click on start
8. Click on visualization
9. Now click on preprocessor tab
10. Select 9^{th} and 20^{th} attribute
11. Click on remove button
12. Goto classify tab
13. Choose J48 tree
14. Select cross validation with 10 folds
15. Click on start button
16. Right click on blue bar under the result list and go to visualize tree.

Output:

We use the **Preprocess Tab in Weka GUI Explorer to remove an attribute "Foreign-workers" & "Perosnal_status" one by one.** In Classify Tab, Select Use Training set option then

Press Start Button, If these attributes removed from the dataset, we can see change in the accuracy compare to full data set when we removed.

i) If Foreign_worker is removed

Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	859	85.9	%
Incorrectly Classified Instances	141	14.1	%
Kappa statistic	0.6377		
Mean absolute error	0.2233		
Root mean squared error	0.3341		
Relative absolute error	53.1347 %		
Root relative squared error	72.9074 %		
Coverage of cases (0.95 level)	100	%	
Mean rel. region size (0.95 level)	91.9	%	
Total Number of Instances	1000		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.954	0.363	0.86	0.954	0.905	0.867	good
	0.637	0.046	0.857	0.637	0.73	0.867	bad
Weighted Avg	0.859	0.268	0.859	0.859	0.852	0.867	

=== Confusion Matrix ===

a	b	<-- classified as
668	32	a = good
109	191	b = bad

i) If Personal_status is removed

Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	866	86.6	%
Incorrectly Classified Instances	134	13.4	%
Kappa statistic	0.6582		
Mean absolute error	0.2162		
Root mean squared error	0.3288		
Relative absolute error	51.4483 %		
Root relative squared error	71.7411 %		
Coverage of cases (0.95 level)	100	%	
Mean rel. region size (0.95 level)	91.7	%	

Total Number of Instances 1000

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.954	0.34	0.868	0.954	0.909	0.868	good
	0.66	0.046	0.861	0.66	0.747	0.868	bad
Weighted Avg.	0.866	0.252	0.866	0.866	0.86	0.868	

=== Confusion Matrix ===

a	b	<-- classified as
668	32	a = good
102	198	b = bad

Note: With this observation we have seen, **when "Foreign_worker "attribute is removed from the**

Dataset, the accuracy is decreased. So this attribute is important for classification.

8. Another question might be, do you really need to input so many attributes to get good results? May be only a few would do. For example, you could try just having attributes 2,3,5,7,10,17 and 21. Try out some combinations.(You had removed two attributes in problem 7. Remember to reload the arff data file to get all the attributes initially before you start selecting the ones you want.)

Ans) steps followed are:

1. Double click on credit-g.arff file.
2. Select 2,3,5,7,10,17,21 and tick the check boxes.
3. Click on invert
4. Click on remove
5. Click on classify tab
6. Choose trace and then algorithm as J48
7. Select cross validation folds as 2
8. Click on start.

OUTPUT:

We use the **Preprocess Tab** in Weka GUI Explorer to remove 2nd attribute (Duration). In Classify Tab, Select Use Training set option then Press Start Button, If these attributes removed from the dataset, we can see change in the accuracy compare to full data set when we removed.

```
=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances         841               84.1      %
Incorrectly Classified Instances       159               15.9      %
Confusion Matrix ===

  a  b  <-- classified as
647  53 |  a = good
106 194 |  b = bad
```

Remember to reload the previous removed attribute, press Undo option in Preprocess tab. We use the **Preprocess Tab** in Weka GUI Explorer to remove 3rd attribute (Credit_history). In Classify Tab, Select Use Training set option then Press Start Button, If these attributes removed from the dataset, we can see change in the accuracy compare to full data set when we removed.

```
=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances         839               83.9      %
Incorrectly Classified Instances       161               16.1      %

== Confusion Matrix ===

  a  b  <-- classified as
645  55 |  a = good
106 194 |  b = bad
```

Remember to reload the previous removed attribute, press Undo option in Preprocess tab. We use the **Preprocess Tab** in Weka GUI Explorer to remove 5th attribute (Credit_amount). In Classify Tab, Select Use Training set option then Press Start Button, If these attributes removed from the dataset, we can see change in the accuracy compare to full data set when we removed.

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances	864	86.4	%
Incorrectly Classified Instances	136	13.6	%

= Confusion Matrix ===

a	b	<-- classified as
675	25	a = good
111	189	b = bad

Remember to reload the previous removed attribute, press Undo option in Preprocess tab. We use the **Preprocess Tab** in Weka GUI Explorer to remove 7th attribute (Employment). In Classify Tab, Select Use Training set option then Press Start Button, If these attributes removed from the dataset, we can see change in the accuracy compare to full data set when we removed.

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances	858	85.8	%
Incorrectly Classified Instances	142	14.2	%

== Confusion Matrix ===

a	b	<-- classified as
670	30	a = good
112	188	b = bad

Remember to reload the previous removed attribute, press Undo option in Preprocess tab. We use the **Preprocess Tab** in Weka GUI Explorer to remove 10th attribute (Other_parties). In Classify Tab, Select Use Training set option then Press Start Button, If these attributes removed from the dataset, we can see change in the accuracy compare to full data set when we removed.

Time taken to build model: 0.05 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances	845	84.5	%

Incorrectly Classified Instances 155 15.5 %

Confusion Matrix ===

```
  a  b  <-- classified as
663  37 |  a = good
118 182 |  b = bad
```

Remember to reload the previous removed attribute, press Undo option in Preprocess tab. We use the **Preprocess Tab** in Weka GUI Explorer to remove 17th attribute (Job). In Classify Tab, Select Use Training set option then Press Start Button, If these attributes removed from the dataset, we can see change in the accuracy compare to full data set when we removed.

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances 859 85.9 %
Incorrectly Classified Instances 141 14.1 %

=== Confusion Matrix ===

```
  a  b  <-- classified as
675  25 |  a = good
116 184 |  b = bad
```

Remember to reload the previous removed attribute, press Undo option in Preprocess tab. We use the **Preprocess Tab** in Weka GUI Explorer to remove 21st attribute (Class). In Classify Tab, Select Use Training set option then Press Start Button, If these attributes removed from the dataset, we can see change in the accuracy compare to full data set when we removed.

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances 963 96.3 %
Incorrectly Classified Instances 37 3.7 %

=== Confusion Matrix ===

```
  a  b  <-- classified as
```

963 0 | a = yes
37 0 | b = no

Note: With this observation we have seen, when 3rd attribute is removed from the Dataset, the accuracy (83%) is decreased. So this attribute is important for classification. when 2nd and 10th attributes are removed from the Dataset, the accuracy(84%) is same. So we can remove any one among them. when 7th and 17th attributes are removed from the Dataset, the accuracy(85%) is same. So we can remove any one among them. If we remove 5th and 21st attributes the accuracy is increased, so these attributes may not be needed for the classification.

9. Sometimes, The cost of rejecting an applicant who actually has good credit might be higher than accepting an applicant who has bad credit. Instead of counting the misclassification equally in both cases, give a higher cost to the first case (say cost 5) and lower cost to the second case. By using a cost matrix in weak. Train your decision tree and report the Decision Tree and cross validation results. Are they significantly different from results obtained in problem 6.

Ans) steps followed are:

1. Double click on credit-g.arff file.
2. Click on classify tab.
3. Click on choose button.
4. Expand tree folder and select J48
5. Click on start
6. Note down the accuracy values
7. Now click on credit arff file
8. Click on attributes 2,3,5,7,10,17,21
9. Click on invert
10. Click on classify tab
11. Choose J48 algorithm
12. Select Cross validation fold as 2
13. Click on start and note down the accuracy values.
14. Again make cross validation folds as 10 and note down the accuracy values.
15. Again make cross validation folds as 20 and note down the accuracy values.

OUTPUT:

In Weka GUI Explorer, Select Classify Tab, In that Select **Use Training set** option . In Classify Tab then press **Choose** button in that select J48 as Decision Tree Technique. In Classify Tab then press **More options** button then we get classifier evaluation options window in that select cost sensitive evaluation the press set option Button then we get Cost Matrix Editor. In that change classes as 2 then press Resize button. Then we get 2X2 Cost matrix. In Cost Matrix (0,1) location value change as 5, then we get modified cost matrix is as follows.

0.0 5.0
1.0 0.0

Then close the cost matrix editor, then press ok button. Then press start button.

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances	855	85.5	%
Incorrectly Classified Instances	145	14.5	%

=== Confusion Matrix ===

```
   a   b   <-- classified as
 669  31 |  a = good
 114 186 |  b = bad
```

Note: With this observation we have seen that ,total 700 customers in that 669 classified as good customers and 31 misclassified as bad customers. In total 300cusotmers, 186 classified as bad customers and 114 misclassified as good customers.

10. Do you think it is a good idea to prefect simple decision trees instead of having long complex decision tress? How does the complexity of a Decision Tree relate to the bias of the model?

Ans)

steps followed are:-
1)click on credit arff file
2)Select all attributes
3)click on classify tab
4)click on choose and select J48 algorithm
5)select cross validation folds with 2
6)click on start

7)write down the time complexity value

It is Good idea to prefer simple Decision trees, instead of having complex Decision tree.

11. You can make your Decision Trees simpler by pruning the nodes. One approach is to use Reduced Error Pruning. Explain this idea briefly. Try reduced error pruning for training your Decision Trees using cross validation and report the Decision Trees you obtain? Also Report your accuracy using the pruned model Does your Accuracy increase?

Ans)

steps followed are:-
1)click on credit arff file
2)Select all attributes
3)click on classify tab
4)click on choose and select REP algorithm
5)select cross validation 2
6)click on start
7)Note down the results

We can make our decision tree simpler by pruning the nodes. For that In Weka GUI Explorer, Select Classify Tab, In that Select **Use Training set** option . In Classify Tab then press **Choose** button in that select J48 as Decision Tree Technique. Beside Choose Button Press on **J48 –c 0.25 –M2 text** we get Generic Object Editor. In that select **Reduced Error pruning Property** as **True then press ok.** Then press start button.

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances	786	78.6	%
Incorrectly Classified Instances	214	21.4	%

== Confusion Matrix ===

```
  a  b  <-- classified as
662  38 |  a = good
176 124 |  b = bad
```

By using pruned model, the accuracy decreased. Therefore by pruning the nodes we can make our decision tree simpler.

12) How can you convert a Decision Tree into "if-then-else rules". Make up your own small

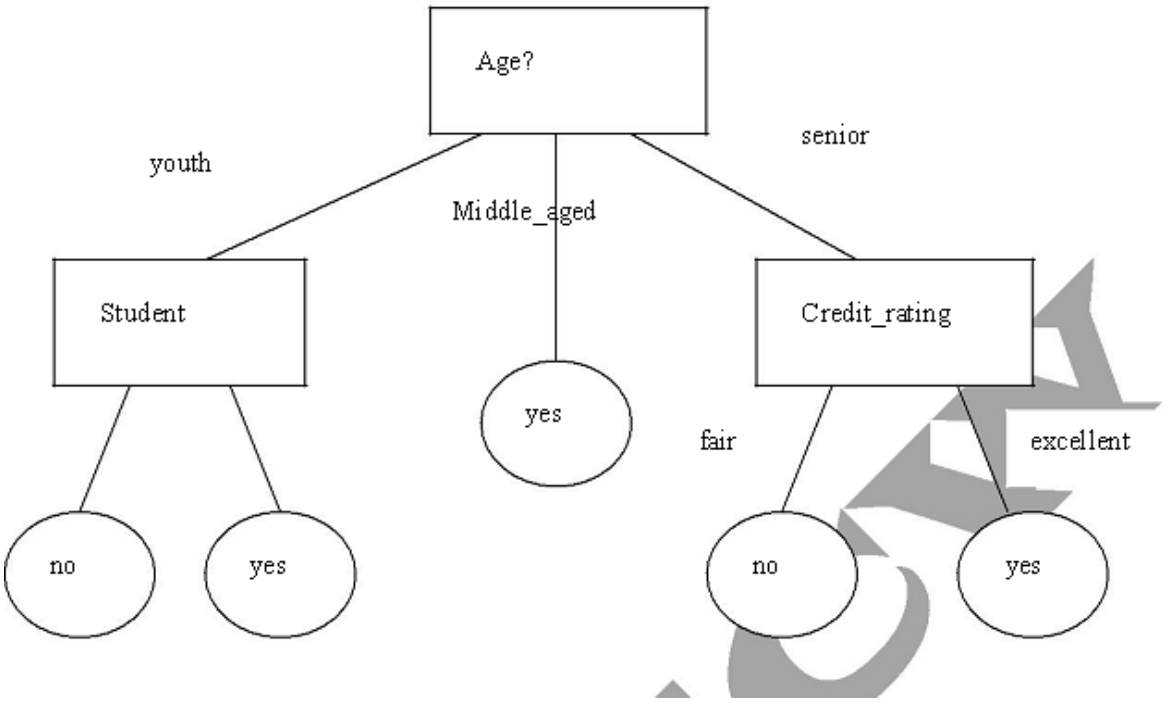
Decision Tree consisting 2-3 levels and convert into a set of rules. There also exist different classifiers that output the model in the form of rules. One such classifier in weka is rules. PART, train this model and report the set of rules obtained. Sometimes just one attribute can be good enough in making the decision, yes, just one ! Can you predict what attribute that might be in this data set? OneR classifier uses a single attribute to make decisions(it chooses the attribute based on minimum error).Report the rule obtained by training a one R classifier. Rank the performance of j48,PART,oneR.

Ans)

Steps For Analyze Decision Tree:

1)click on credit arff file
2)Select all attributes
3)click on classify tab
4)click on choose and select J48 algorithm
5)select cross validation folds with 2
6)click on start
7)note down the accuracy value
8) again goto choose tab and select PART
9)select cross validation folds with 2
10)click on start
11)note down accuracy value
12)again goto choose tab and select One R
13)select cross validation folds with 2
14)click on start
15)note down the accuracy value.

Sample Decision Tree of 2-3 levles.



Converting Decision tree into a set of rules is as follows.

Rule1: If age = youth AND student=yes THEN buys_computer=yes
Rule2: If age = youth AND student=no THEN buys_computer=no
Rule3: If age = middle_aged THEN buys_computer=yes

Rule4: If age = senior AND credit_rating=excellent THEN buys_computer=yes
Rule5: If age = senior AND credit_rating=fair THEN buys_computer=no

In Weka GUI Explorer, Select Classify Tab, In that Select **Use Training set** option .There also exist different classifiers that output the model in the form of Rules. Such classifiers in weka are

"PART" and "OneR" . Then go to Choose and select Rules in that select PART and press start Button.

== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances 897 89.7 %
Incorrectly Classified Instances 103 10.3 %

== Confusion Matrix ===

a b <-- classified as
653 47 | a = good
56 244 | b = bad

Then go to Choose and select Rules in that select OneR and press start Button.

== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances 742 74.2 %
Incorrectly Classified Instances 258 25.8 %
=== Confusion Matrix ===
a b <-- classified as
642 58 | a = good
200 100 | b = bad

Then go to Choose and select Trees in that select J48 and press start Button.

=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances 855 85.5 %
Incorrectly Classified Instances 145 14.5 %
=== Confusion Matrix ===
a b <-- classified as
669 31 | a = good
114 186 | b = bad

Note: With this observation we have seen the performance of classifier and Rank is as follows

1. PART
2. J48 3. OneR

7.Task 2:Hospital Management System

Data warehouse consists dimension table and fact table.

REMEMBER the following

Dimension

The dimension object(dimension);

_name

_attributes(levels),with primary key

_hierarchies

One time dimension is must.

About levels and hierarchies

Dimensions objects(dimension) consists of set of levels and set of hierarchies defined over those levels.the levels represent levels of aggregation.hierarchies describe-child relationships among a set of levels.

For example .a typical calander dimension could contain five levels.two hierarchies can be defined on these levels.

H1: YearL>QuarterL>MonthL>DayL

H2: YearL>WeekL>DayL

The hierarchies are describes from parent to child,so that year is the parent of Quarter,quarter are parent of month,and so forth.

About Unique key constraints

When you create a definition for a hierarchy,warehouse builder creates an identifier key for each level of the hierarchy and unique key constraint on the lowest level (base level)

Design a hospital management system data warehouse(TARGET) consists of dimensions patient,medicine,supplier,time.where measure are ‘ NO UNITS’ ,UNIT PRICE.

Assume the relational database(SOURCE)table schemas as follows TIME(day,month,year)

PATIENT(patient_name,age,address,etc)

MEDICINE(Medicine_brand_name,Drug_name,supplier,no_units,units_price,etc..,)

SUPPLIER:(Supplier_name,medicine_brand_name,address,etc..,)

If each dimension has 6 levels,decide the levels and hierarchies,assumes the level names suitably.

Design the hospital management system data warehousing using all schemas.give the example

4-D cube with assumption names.

8.Simple Project on Data Preprocessing

Data Preprocessing

Objective: Understanding the purpose of unsupervised attribute/instance filters for preprocessing the input data.

Follow the steps mentioned below to configure and apply a filter.

The preprocess section allows filters to be defined that transform the data in various ways. The Filter box is used to set up filters that are required. At the left of the Filter box is a Choose button. By clicking this button it is possible to select one of the filters in Weka. Once a filter has been selected, its name and options are shown in the field next to the Choose button. Clicking on this box brings up a GenericObjectEditor dialog box, which lets you configure a filter. Once you are happy with the settings you have chosen, click OK to return to the main Explorer window.

Now you can apply it to the data by pressing the Apply button at the right end of the Filter panel. The Preprocess panel will then show the transformed data. The change can be undone using the Undo button. Use the Edit button to view your transformed data in the dataset editor.

Try each of the following **Unsupervised Attribute Filters**. (Choose -> weka -> filters -> unsupervised -> attribute)

- Use **ReplaceMissingValues** to replace missing values in the given dataset.
- Use the filter **Add** to add the attribute Average.
- Use the filter **AddExpression** and add an attribute which is the average of attributes M1 and M2. Name this attribute as AVG.
- Understand the purpose of the attribute filter **Copy.**
- Use the attribute filters **Discretize** and **PKIDiscretize** to discretize the M1 and M2 attributes into five bins. (NOTE: Open the file afresh to apply the second filter

since there would be no numeric attribute to dicretize after you have applied the first filter.)

- Perform **Normalize** and **Standardize** on the dataset and identify the difference between these operations.
- Use the attribute filter **FirstOrder** to convert the M1 and M2 attributes into a single attribute representing the first differences between them.
- Add a nominal attribute Grade and use the filter **MakeIndicator** to convert the attribute into a Boolean attribute.
- Try if you can accomplish the task in the previous step using the filter MergeTwoValues.
- Try the following transformation functions and identify the purpose of each
    - NumericTransform
    - NominalToBinary
    - NumericToBinary
    - Remove
    - RemoveType
    - RemoveUseless
    - ReplaceMissingValues
    - SwapValues

Try the following **Unsupervised Instance Filters.**

(Choose -> weka -> filters -> unsupervised -> instance)

- Perform **Randomize** on the given dataset and try to correlate the resultant sequence with the given one.
- Use **RemoveRange** filter to remove the last two instances.
- Use **RemovePercent** to remove 10 percent of the dataset.

- Apply the filter **RemoveWithValues** to a nominal and a numeric attribute

LINUX PROGRAMMING

LABORATORY MANUAL

B.TECH

(IV YEAR – I SEM)

(2016-17)

Department of
Computer Science and Engineering

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous Institution – UGC, Govt. of India)
Recognized under 2(f) and 12 (B) of UGC ACT 1956
Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2008 Certified)
Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

Department of Computer Science and Engineering

Vision

- To acknowledge quality education and instill high patterns of discipline making the students technologically superior and ethically strong which involves the improvement in the quality of life in human race.

Mission

- To achieve and impart holistic technical education using the best of infrastructure, outstanding technical and teaching expertise to establish the students into competent and confident engineers.
- Evolving the center of excellence through creative and innovative teaching learning practices for promoting academic achievement to produce internationally accepted competitive and world class professionals.

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12 (B) of UGC ACT 1956

Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2008 Certified)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

Department of Computer Science and Engineering

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1 – ANALYTICAL SKILLS

1. To facilitate the graduates with the ability to visualize, gather information, articulate, analyze, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problems increasing their productivity.

PEO2 – TECHNICAL SKILLS

2. To facilitate the graduates with the technical skills that prepare them for immediate employment and pursue certification providing a deeper understanding of the technology in advanced areas of computer science and related fields, thus encouraging to pursue higher education and research based on their interest.

PEO3 – SOFT SKILLS

3. To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self-confidence by communicating effectively, having a positive attitude, get involved in team-work, being a leader, managing their career and their life.

PEO4 – PROFESSIONAL ETHICS

To facilitate the graduates with the knowledge of professional and ethical responsibilities by paying attention to grooming, being conservative with style, following dress codes, safety codes,and adapting themselves to technological advancements.

PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B. Tech Computer Science and Engineering, the graduates will have the following Program Specific Outcomes:

1. **Fundamentals and critical knowledge of the Computer System:-** Able to Understand the working principles of the computer System and its components , Apply the knowledge to build, asses, and analyze the software and hardware aspects of it .

2. **The comprehensive and Applicative knowledge of Software Development:** Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.

3. **Applications of Computing Domain & Research:** Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify the research gaps, and provide innovative solutions to them.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design / development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.

12. **Life- long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:
    a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
    b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.
    c. Proper Dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions.Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out ; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

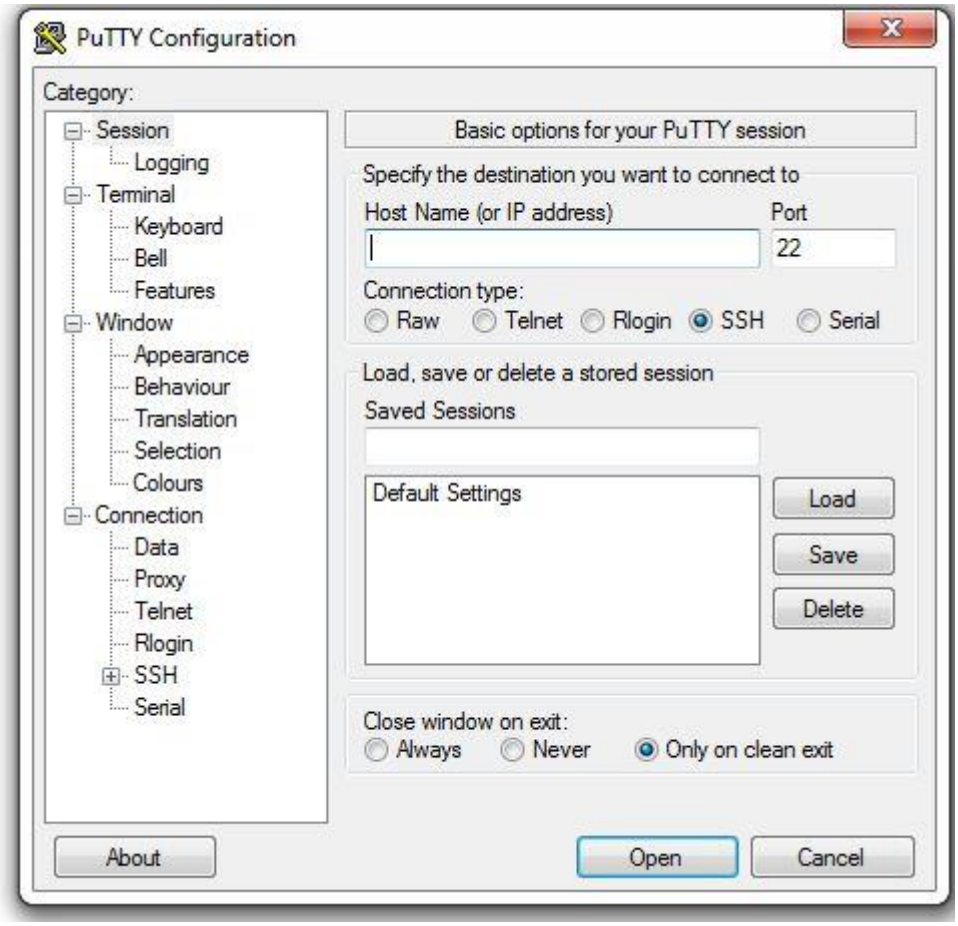
Head of the Department **Principal**

INDEX

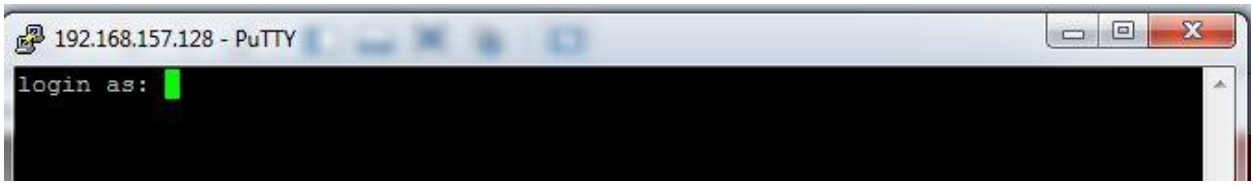
Procedure to connect to LINUX (Steps)

Procedure to connect to LINUX

Step 1:click on putty icon available on desk top. A window is opened



Step 2:fill in ip address of linux server and click open



Step 3: provide login and password (nothing is displayed on screen while typing password)
Step 4:changethe default password at your first login

EXPERIMENT NO: 1 **Date:**

Aim: Write a Shell Script that accepts a file name, starting and ending line numbers as arguments and displays all lines between the given line numbers.

ALGORITHM:

Step 1: Create a file with 5-6 lines of data
File can be created by vi sample.dat or cat sample.dat
Step 2:Now write a shell script with
vi 1.sh
step3:Check the no of arguments for shell script
if 0 arguments then print no arguments
else if 1 argument then print 1 argument
else if 2 arguments then print 2 arguments
else check for file is there or not(if file is not there print file does not exists)
1else sed -ne "$2','$3' p' $1
sed is one of powerful filter(stream editor)
-e default option (script on command line)
-n suppresses automatic output
$2 first line number passed $3 2nd line number passed
p is a print command (prints current content to the pattern space).
$1 is name of file from which we are printing data between the line numbers.
Step 4:top

Script Name: 1sh

```
#!/bin/bash
if [ $# -lt 3 ]
then
        echo "To execute you have to enter 3 arguments in command line in following
order..."
        echo " File Name ,starting line number and ending line number..."
else
        sed -n $2,$3p $1
fi
```

Commands used in the script:

Sed command:

stream editor for filtering and transforming text

1. Replacing or substituting string

Sed command is mostly used to replace the text in a file. The below simple sed command replaces the word "unix" with "linux" in the file.

$sed 's/unix/linux/' file.txt

2. Replacing the nth occurrence of a pattern in a line

$sed 's/unix/linux/2' file.txt

Replaces 2nd occurrence

3.printing pines for a given range

$sed –n 1,5p hello.txt

Prints first 5 lines in the file hello.txt

nl command:

The nl utility in Linux is used to give number lines of a file on console.

Example:

```
$ nl sort.txt
  1    UK
  2    Australia
  3    Newzealand
  4    Brazil
  5    America
```

Execution:

check how many lines of data in the input file

```
root@localhost sh]# cat hello.txt|nl
1 abc
2 def
3 ghi
4 abc
5 abc
6 cccc
```

Executing Shell script:

run1:

```
[root@localhost sh]# sh 1.sh abc1.txt 2 4
def
ghi
abc
```

compare with the data in the file and out put

Viva Questions

1.What is a shell script?
2.How to find current shell name
3.How to switch to another shell
4.How to execute shell Script

EXPERIMENT NO: 2 **Date:**

AIM: Write a shell Script that deletes all lines containing the specified word in one or more files supplied as arguments to it.

ALGORITHM:

Step 1: Create a file with 5-6 lines of data
Create the 2 file f1 and f2 as vi s1and vi s2
Step2: Now write a shell script with
vi 2.sh
step3:Check the no of arguments for shell script
if 0 arguments then print no arguments
else pattern=$1(word will be stored in pattern)
for fname in $*
for every filename in given files
if it is a file if [-f $fname] then
print DELETING $pattern FROM $fname
sed '/'$pattern'/d' $fname
sed acts as filter if word is a file in any line that will be deleted
'/' is used to represent regular expressions
'/d' is a delete command in sed
else print file NOT FOUND

Script name: 2.sh

```
#!/bin/bash
if [ $# -lt 2 ]then
        echo "Enter atlest two files as input in command line"
else
        printf "enter a word to find:"
        read word
        for f in $*
        do
                printf "\n In File $f:\n"
                sed /$word/d $f
        done
fi
```

Execution:
run1:
check data in input files

```
[root@localhost sh]# cat abc1.txt
abc
def
ghi
abc
abc
cccc
[root@localhost sh]# cat abc2.txt
abc
def
ghi
abc
abc
cccc
```

Executing shell script
[root@localhost sh]# sh 2.sh abc1.txt abc2.txt
enter a word to find:abc
In File abc1.txt:
def
ghi
cccc
In File abc2.txt:
def
ghi
cccc

Expected output:
Displays lines from files s1 s2 after deleting the word hi

Viva Questions

1.Explain various loops in shell script
2.Explain grep
3.Explain egep
4.Explain fgep
5. .Explain sed

EXPERIMENT NO: 3 **Date:**

Aim: Write a shell script that displays a list of all files in the current directory to which the user has read, write and execute permissions.

ALGORITHM:

Step1: selects list of files from present working directory
Step 2:check for each file wither its is has read, write and execute permissions if true goto step 3
Step 3: print file
Step 4 :stop

Script name: 3.sh

```
#!/bin/bash
        echo "List of Files which have Read, Write and Execute Permissions in Current Directory
are..."
        for file in *
        do
                if [ -r $file -a -w $file -a -x $file ]
                then
                        echo $file
                fi
        done
```

Execution:

$sh 3.sh

Expected output:
by executing above shell script you will get all files which has read ,write and execute
Permissions in current working directory

sample output

```
[root@localhost sh]# sh 3.sh
List of Files which have Read, Write and Execute Permissions in Current Directory are...
        5.sh
        a.out
```

Viva Questions:

1.Display all files in a directory
2.how to use chmod
3.How to change file permissions

EXPERIMENT NO: 4 **Date:**

Aim:-Write a shell script that receives any number of file names as arguments checks if every argumentsupplied is a file or directory and reports accordingly. whenever the argument is a file it reports no of linespresent in it

ALGORITHM:

step 1: if arguments are less than 1 print Enter atleast one input file name and gotostep 9
Step 2: selects list a file from list of arguments provided in command line
Step 3: check for whether it is directory if yes print is directory and goto step 9
step 4: check for whether it is a regular file if yes goto step 5 else goto step 8
Step 5: print given name is regular file
step 6: print No of lines in file
step 7: goto step
step 8: print not a file or a directory
step 9: stop

Script name: 4.sh

```
#!/bin/bash
if [ $# -lt 1 ]
then
        echo "Enter atleast one input file name"
else
        for i in $*
        do
                if [ -d $i ]
                then
                        echo " given name is directory"
                elif [ -f $i ]
                then
                        echo " given name is file: $i"
                        echo " No of lines in file are : `wc -l $i`"
                else
                        echo "given name is not a file or a directory"
                fi
        done
fi
```

Execution:

provide two file names as input one a regular file and other directory
for example abc1.txt a text file as first argument and japs a directory as second argument

Run1:

```
[root@localhost sh]# sh 4.sh abc1.txt japs
given name is file: abc1.txt
No of lines in file are : 7 abc1.txt
japs is directory
```

run 2:[root@localhost sh]# sh 4.sh abc1.txt abc2.txt
given name is file: abc1.txt
No of lines in file are : 7 abc1.txt
given name is file: abc2.txt
No of lines in file are : 7 abc2.txt

Viva Questions:

1. What is an internal command in Linux?
Internal commands are also called shell built-in commands. Example: cd,fg. Since these are shell built-in, no process is created while executing these commands, and hence are considered to be much faster.

2. x and y are two variables containing numbers? How to add these 2 numbers?
$ expr $x + $y

3. How to add a header record to a file in Linux?
$ sed -i '1i HEADER' file

4. How to find the list of files modified in the last 30 mins in Linux?
$ find . -mmin -30

5. How to find the list of files modified in the last 20 days?
$ find . -mtime -20

EXPERIMENT NO: 5 **Date:**

Aim:-Write a shell script that accepts a list of file names as its arguments, counts and reports the occurrenceof each word that is present in the first argument file on other argument files.

ALGORITHM:

step1: Check the no of arguments for shell script
if 0 arguments then print no arguments
step2:else translate each word in the first file is to be on separate line
which will be stored in temp file
step3: for i in $*
for every filename in given files
step 4: translate each word in the file is to be on separate line
which will be stored in temp1 file
step5: count no of lines in temp file assign it to j
step6: initialize j=1
step 7: while i< j
extract the line that are common in both the file by using
head and tail commands
then apply the filter grep to count and print the lines
which are common to files
increment j
step 8: stop

Script name:5.sh

```
#!/bin/bash
echo "no of arguments $#"
if [ $# -le 2 ]
then
        echo "Error : Invalid number of arguments."
        exit
fi
str=`cat $1 | tr '\n' ' '`
for a in $str
do
        echo "in file $a"
        echo "Word = $a, Count = `grep -c "$a" $2`"
done
```

Execution and output:
check data in abc1.txt file
[root@localhost sh]# cat abc1.txt
abc
def
ghi
abc
abc
cccc
check data in abc1.txt file
[root@localhost sh]# cat abc2.txt

abc
def
ghi
abc
abc
cccc

executing script

[root@localhost sh]# sh 5.sh abc1.txt abc2.txt
Word = abc, Count = 3
Word = def, Count = 1
Word = ghi, Count = 1
Word = abc, Count = 3
Word = abc, Count = 3
Word = cccc, Count = 1

Viva Questions

1. What is Shell Scripting ?
Shell scripting, in Linux or Unix, is programming with the shell using which you can automate your tasks. A shell is the command interpreter which is the interface between the User and the kernel. A shell script allows you to submit a set of commands to the kernel in a batch. In addition, the shell itself is very powerful with many properties on its own, be it for string manipulation or some basic programming stuff.

2. The command "cat file" gives error message "--bash: cat: Command not found". Why?
It is because the PATH variable is corrupt or not set appropriately. And hence the error because the cat command is not available in the directories present PATH variable.

3. How to find the length of a string in Linux?
$ x="welcome" $ echo ${#x} 7

4. What are the different timestamps associated with a file?
Modification time:- Refers to the time when the file is last modified.
Access time :- The time when the file is last accessed.
Changed time :- The time when the attributes of the file are last changed.

5. How to get the list of files alone in a directory in Linux?
$ ls -lrt | grep ^-

EXPERIMENT NO: 6 **Date:**

Aim:-Write a shell script to list all of the directory files in a directory.

Algorithm:

Step1: enter the name of the directory
Read dir
Step2: if it is a directory
Then list the files present in that directory
By using ls command with –p option to list all directory files in a given directory
Step 3: else enter the directory name
Step 4: stop

Script name: 6.sh

```
#!/bin/bash
echo " Enter dir name: "
read dir
if [ -d $dir ]
then
        printf " Files in Directory $dir are...\n`ls $dir`"
else
        echo " Dir does not exist"
fi
```

Execution and output:

```
[root@localhost sh]# sh 6.sh
Enter dir name:
japs
Files in Directory japs are...
abc1.txt
abc2.txt
ls-l.c
prg5
s1
```

Viva Questions

1. How to find the files modified exactly before 30minutes?

```
$ find . -mmin 30
```

2. A string contains a absolute path of a file. How to extract the filename alone from the absolute path in Linux?

```
$ x="/home/guru/temp/f1.txt"
$ echo $x | sed 's^.*/^^'
```

3. How to find all the files created after a pre-defined date time, say after 10th April 10AM?

This can be achieved in 2 steps:

1. Create a dummy file with the time stamp, 10th April 10AM.

2. Find all the files created after this dummy file.

```
$ touch -t 1004101000 file
$ find . -newer file
```

4. How to print the contents of a file line by line in Linux?

```
$ while read line
do
echo $line
done < file
```

5. The word "Unix" is present in many .txt files which is present across many files and also files present in sub directories. How to get the total count of the word "Unix" from all the .txt files?

```
$ find . -name *.txt -exec grep -c Unix '{}' \; | awk '{x+=$0;}END{print x}'
```

EXPERIMENT NO: 7 **Date:**

Aim:-Write a shell script to find factorial of a given number.

ALGORITHM

Step 1: read any number to find factorial
Step 2: initialize fact=1 and i=1
Step 3: while i less than
do
fact=fact* i
i=i+1
done
step 4:print fact
step 5:stop.

Script Name:7.sh

```
#!/bin/bash
echo "Factorial Calculation Script...."
echo "Enter a number: "
read f
fact=1
factorial=1
while [ $fact -le $f ]
do
        factorial=`expr $factorial \* $fact`
        fact=`expr $fact + 1`
done
echo "Factorial of $f = $factorial"
```

Execution and Output:

```
[root@localhost sh]# sh 7.sh
Factorial Calculation Script....
Enter a number: 4
Factorial of 4 = 24
```

EXPERIMENT NO: 8 **Date:**

Aim:-write an awk script to count number of lines in a file that does not contain vowels

ALGORITHM

Step 1: create a file with 5-10 lines of data
Step 2: write an awk script by using grep command to filter the lines
that do not contain vowels
awk ‘ $0 ~/aeiou/ {print $0}’ file1
step3: count=count+1
step4:print count
step5:stop

Awk script name:nm.awk

```
BEGIN{}
{
If($0 !~/[aeiou AEIOU]/)
wordcount+=NF
}
END
{
print "Number of Lines are", wordcount
}
```

input file for awk script:data.dat

```
bcdfghj
abcdfghj
bcdfghj
ebcdfghj
bcdfghj
ibcdfghj
bcdfghj
obcdfghj
bcdfghj
ubcdfghj
```

Executing the script:
[root@localhost awk]# **awk -f nm.awk data.dat**

```
bcdfghj
bcdfghj
bcdfghj
bcdfghj
bcdfghj
```

Number f lines are 5

EXPERIMENT NO: 9 Date:

Aim:-write an awk script to find the no of characters ,words and lines in a file

ALGORITHM

Step 1: create a file with 5 to10 lines of data
Step 2: write an awk script
find the length of file
store it in chrcnt
step3: count the no of fields (NF), store it in wordcount
step4: count the no of records (NR), store it in NR
step5: print chrcnt,NRwordcount
step6: stop

Awk script name:nc.awk

```
BEGIN{}
{
        print len=length($0),"\t",$0
        wordcount+=NF
chrcnt+=len
        }
END {
        print "total characters",chrcnt
        print "Number of Lines are",NR
        print "No of Words count:",wordcount
}
```

input data file name:data.dat

```
bcdfghj
abcdfghj
bcdfghj
ebcdfghj
bcdfghj
ibcdfghj
bcdfghj
obcdfghj
bcdfghj
ubcdfghj
```

Executing the script:

[root@localhost awk]# **awk -f nc.awk data.dat**

```
7 bcdfghj
8 abcdfghj
7 bcdfghj
8 ebcdfghj
7 bcdfghj
8 ibcdfghj
7 bcdfghj
8 obcdfghj
7 bcdfghj
8 ubcdfghj
total characters 75
```

Number of Lines are 10
No of Words count: 10

VIVA QUESTIONS:

1.How to find the last modified file or the newest file in a directory?

$ ls -lrt | grep ^- | awk 'END{print $NF}'

2.How to access the 10th command line argument in a shell script in Linux?

$1 for 1st argument, $2 for 2nd, etc... For 10th argument, ${10}, for 11th, ${11} and so on.

3. How to find the sum of all numbers in a file in Linux?

$ awk '{x+=$0}END{print x}' file

4. How to delete a file which has some hidden characters in the file name?

Since the rm command may not be able to delete it, the easiest way to delete a file with some hidden characters in its name is to delete it with the find command using the inode number of the file.

$ ls –li

total 32

9962571 -rw-r--r-- 1 guru users 0 Apr 23 11:35

$ find . -inum 9962571 -exec rm '{}' \;

5.Using the grep command, how can you display or print the entire file contents?

$ grep '.*' file

6.What is the difference between a local variable and environment variable in Linux?

A local variable is the one in which the scope of the variable is only in the shell in which it is defined. An environment variable has scope in all the shells invoked by the shell in which it is defined.

EXPERIMENT NO: 10 **Date:**

Aim: write a c program that simulates cp command (using system calls)

Description:

cp command is used to move or rename a file

syntax:

cp file1 file2

here data in file 1 is copied to file2

Algorithm:

step 1:read the file from keyboard if the file exists

step 2:write data into the file by means of cat command

step 3:if file not exists, create new file

step 4: copy data of file to another by means of cp command

if target file exists cp command overwrites or replaces with new file

step 5:display contents of copied file by cat command in terminal

Program file name:10.c

```
#include<fcntl.h>
#include<sys/stat.h>
#include<stdio.h>
#define BS 1024
int main(void)
{
int fd1,fd2;
int n;
char sf[12],df[12];
char buf[BS];
        printf("Enter a source file name\n");
        scanf("%s",sf);
        printf("Enter dest file name\n");
        scanf("%s",df);
        fd1=open(sf,O_RDONLY);
        fd2=open(df,O_WRONLY | O_CREAT | O_TRUNC,S_IRUSR | S_IWUSR );
        printf("SOURCE FILE IS:%s\n",sf);
        printf("DESTINATION FILE IS:%s\n",df);
        while((n=read(fd1,buf,BS))>0)
        write(fd2,buf,n);
        close(fd1);
        close(fd2);
        printf("Copied sucessfully..\n");
}
```

Executing the program:

step 1:

create above program by using vi editor

[root@localhost lab]$ vi 10.c

step 2: compiling above program

[root@localhost lab]$ cc -o 10.c sample.c

-o creates an executable file 11a

step 3: use ls command to verify whether the file that is to be copies is existing in present

EXPERIMENT NO: 11 **Date:**

Aim: implement in c language the following Unix commands using system calls

a)cat b)ls c)mv

a)AIM:-Write a c program to implement **cat command** using system calls

Description:

cat COMMAND: cat linux command concatenates files and print it on the standard output.

SYNTAX:

cat [OPTIONS] [FILE]...

OPTIONS:

-A Show all.
-b Omits line numbers for blank space in the output.
-e A $ character will be printed at the end of each line prior to a new line.
-E Displays a $ (dollar sign) at the end of each line.
-n Line numbers for all the output lines.
-s If the output has multiple empty lines it replaces it with one empty line.
-T Displays the tab characters in the output.
-v Non-printing characters (with the exception of tabs, new-lines & form-feeds) are printed visibly.

Operations With cat Command:

1. To Create a new file:

$cat > file1.txt

This command creates a new file file1.txt. After typing into the file press control+d (^d) simultaneously to end the file.

2. To Append data into the file:

$cat >> file1.txt

To append data into the same file use append operator >> to write into the file, else the file will be overwritten (i.e., all of its contents will be erased).

3. To display a file:

$cat file1.txt

This command displays the data in the file.

4. To concatenate several files and display:

$cat file1.txt file2.txt

The above cat command will concatenate the two files (file1.txt and file2.txt) and it will display the output in the screen. Some times the output may not fit the monitor screen. In such situation you can print those files in a new file or display the file using less command.

cat file1.txt file2.txt | less

5. To concatenate several files and to transfer the output to another file.

$cat file1.txt file2.txt > file3.txt

In the above example the output is redirected to new file file3.txt. The cat command will create new file file3.txt and store the concatenated output into file3.txt.

Algorithm:

Step 1:Start

Step 2:read arguments from keyboard at command line
Step 3:if no of arguments are less than two print ENTER CORRECT ARGUMENTS
Else goto step 4
Step4:read the date from specified file and write it to destination file
Step 5 :stop

Program file name:11a.c

```
#include<stdio.h>
        #include<sys/types.h>
        #include<stdlib.h>
        #include<fcntl.h>
        #include<sys/stat.h>
        int main(int argc,char *argv[])
        {
        int fd,n;
        char buff[512];
                if(argc!=2)
                        printf("ENTER CORRECT ARGUMENTS :");
                if((fd=open(argv[1],4))<0)
                {
                        printf("ERROR");
                                return 0;
                }
                while(n=read(fd,buff,sizeof(buff))>0)
                        write(1,buff,n);
        }
```

b)AIM:-Write a c program to implement **ls command** using system calls
Description:

ls command is used to list the files present in a directory

Algorithm:

Step 1. Start.
Step 2. open directory using opendir( ) system call.
Step 3. read the directory using readdir( ) system call.
Step 4. print dp.name and dp.inode .
Step 5. repeat above step until end of directory.
Step 6: Stop.

Program name: 11b.c

```
#include<stdio.h>
#include<dirent.h>
void quit(char*,int);
int main(int argc,char **argv )
{
        DIR *dirop;
        struct dirent *dired;
        if(argc!=2)
        {
                printf("Invalid number of arguments\n");
        }
```

```
        if((dirop=opendir(argv[1]))==NULL)
                printf("Cannot open directory\n");
        while((dired=readdir(dirop))!=NULL)
                printf("%10d %s\n",dired>d_ino,dired>d_name);
        closedir(dirop);
}
```

c)Aim :write a c program that simulates **mv command** (using system calls)

Description:

mv command is used to move or rename a file

synatax:

```
mv file1 file2
```

here file1 is renamed as file2

Algorithm:

Step 1: Start

Step 2: open an existed file and one new open file using open() system call

Step 3: read the contents from existed file using read( ) system call

Step 4:write these contents into new file using write system call using write( ) system call

Step 5: repeat above 2 steps until eof

Step 6: close 2 file using fclose( ) system call

Step 7: delete existed file using using unlink( ) system

Step 8: Stop.

Program File name:11c.c

```
#include<stdio.h>
#include<string.h>
int main(int argc ,char *argv[])
{
int r,i;
char p[20],q[20];
if(argc<3)
        printf("improper arguments\n file names required\n");
else
if( argc==3)
{
        printf("\n%s\n",argv[1],argv[2]);
        r=link(argv[1],argv[2]);
        printf("%d\n",r);
        unlink(argv[1]);
}
else
{
        for(i=1;i<argc-1;i++)
        {
           strcpy(p,argv[argc-1]);
           strcat(p,"/");
           strcat(p,argv[i]);
```

```
            printf("%s%s\n",argv[i],p);
            link(argv[i],p);
            unlink(argv[i]);
          }
}
}
```

EXPERIMENT NO: 12 **Date:**

Aim:Write a C program that takes one or more file/directory names as command line input and reports following information

A)File Type B)Number Of Links

c)Time of last Acces D) Read,write and execute permissions

Algorithm:

Step 1:start
Step 2:Declare struct stat a
Step 3:read arguments at command line
Step 4: set the status of the argument using stat(argv[i],&a);
Step 5:Check whether the given file is Directory file by using S_ISDIR(a.st_mode)
if it is a directory file print Directory file
Else
print is Regular file
Step6: print number of links
Step 7:print last time access
Step 8:Print Read,write and execute permissions
Step 9:stop

Program File name: 13.c

```
#include<stdio.h>
#include<sys/stat.h>
#include<time.h>
int main(int argc,char *argv[])
{
int i,j;
struct stat a;
for (i=1;i<argc;i++)
{
        printf("%s : ",argv[i]);
        stat(argv[i],&a);
        if(S_ISDIR(a.st_mode))
        {
                printf("is a Directory file\n");
        }
        else
        {
                printf("is Regular file\n");
        }
        printf("******File Properties********\n");
        printf("Inode Number:%d\n",a.st_ino);
        printf("UID:%o\n",a.st_uid);
        printf("GID:%o\n",a.st_gid);
        printf("No of Links:%d\n",a.st_nlink);
        printf("Last Access time:%s",asctime(localtime(&a.st_atime)));
        printf("Permission flag:%o\n",a.st_mode%512);
        printf("size in bytes:%d\n",a.st_size);
        printf("Blocks Allocated:%d\n",a.st_blocks);
        printf("Last modification time %s\n",ctime(&a.st_atime)); } }
```

EXPERIMENT NO: 13

Aim:Write a C program to emulate unix ls -l command-line

ALGORITHM :

Step 1: Include necessary header files for manipulating directory.
Step 2: Declare and initialize required objects.
Step 3: Read the directory name form the user.
Step 4: Open the directory using opendir() system call and report error if the directory is
not available.
Step 5: Read the entry available in the directory.
Step 6: Display the directory entry ie., name of the file or sub directory.
Step 7: Repeat the step 6 and 7 until all the entries were read.
Step 8 :stop

Program file name:13.c

```
#include<fcntl.h>
#include<stdio.h>
#include<dirent.h>
#include<unistd.h>
#include<sys/stat.h>main()
{
char dirname[10];
DIR *p;
struct dirent *d;
        printf("Enter directory name ");
        scanf("%s",dirname);
        p=opendir(dirname);
        if(p==NULL)
        {
                perror("Cannot find dir.");
                exit(-1);
        }
while(d=readdir(p))
printf("%s\n",d->d_name);
}
```

EXPERIMENT NO: 14

Aim:Write a C program to list every file in directory,its inode number and file name

Algorithm:

Step 1:Start
Step 2:Read Directory name
Step 3:open the directory
Step 4: print file name and Inode number of each file in the directory
Step 5:Stop

Program file name:14.c

```
#include<fcntl.h>
#include<stdio.h>
#include<dirent.h>
#include<sys/stat.h>
int main(int argc,char*argv[])
{
DIR *dirop;
struct dirent *dired;
if(argc!=2)
{
        printf("Invalid number of arguments\n");
}
else if((dirop=opendir(argv[1]))==NULL)

        printf("Cannot open Directory\n");
else
{
printf("%10s %s \n","Inode","File Name");
        while((dired=readdir(dirop))!=NULL)
         printf("%10d %s\n ",dired->d_ino,dired->d_name);
        closedir(dirop);
}
return 0;
}
```

EXPERIMENT NO: 15 Date:

Aim:Write a C program to demonstrate redirection of standard output to a file ex: ls>f1

Algorithm:

Step 1: create a file pointer
FILE *fd;
Step 2:open the pipe for reading the data of ls –l command
fd=popen("ls -l","r");
step 3: read the data from pipe and store it in line buffer
while((fgets(line,200,fd))!=NULL) print ("%s\n",line);
step 4: create a file
if((f1=creat("xx.txt",0644))<0)
print ERROR IN CREATING
step 5:read the data from line and store it to file
while((n=read(line,buff,sizeof(buff)))>0)
write(f1,line,n);
step 6:stop

Program file name :15.c

```
#include <string.h>            /* String functions (strcat) */
#include <stdio.h>             /* printf, etc. */
#include <stdlib.h>            /* exit, etc. */

/* Define all the error messages */
char * error_msg[] = {
          "\nUsage: ./15redir [<text-file>] [\"<string>\"]\n\n",
          "\nCould not open the file for writing\n\n"
     };
 /* Prototype definition of function “print_error” */
void print_error(int msg_num, int exit_code);

int main(int argc, char * argv[])
{
     FILE * file_to_write;      /* Handle to file to be written */
     char * name_of_file_to_be_written = "redir.txt";
     char * text_to_be_written = "This text was redirected from standard output";

     /* Check if correct number of arguments are supplied */
     if ( argc > 3 )
          print_error(0,2);
        /* Check if first argument was supplied */
     if ( argc == 2 )
     {
          name_of_file_to_be_written = argv[1];
     }

     /* Check if both arguments were supplied */
     if ( argc == 3 )
     {
```

```
        name_of_file_to_be_written = argv[1];
        text_to_be_written = argv[2];
    }

    /* Open the file in append mode */
    if ( (file_to_write = fopen(name_of_file_to_be_written, "a+")) == NULL )
        print_error(1,3);

    /* Redirect standard output to the file */
    dup2(fileno(file_to_write),fileno(stdout));

    /* Write the text to standard output */
    printf("%s\n",text_to_be_written);

    return 1;
}

void print_error(int error_index, int exit_code)
{
    fprintf(stderr, "%s", error_msg[error_index]);
    exit(exit_code);
}
```

EXPERIMENT NO: 16 Date:

Aim:Write a C program to create child process and allow parent process to display “parent” and the child to display “child” on the screen

Algorithm:

Step 1: start
Step2: call the fork() function to create a child process
fork function returns 2 values
step 3: which returns 0 to child process
step 4:which returns process id to the parent process
step 5:stop

Program file name:16.c

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main()
{
int pid,pid1,pid2;
        pid=fork();
        if(pid==-1)
        {
                printf("ERROR IN PROCESS CREATION \n");
                exit(0);
        }
        if(pid!=0)
        {
                pid1=getpid();
                printf("\n the parent process ID is %d", pid1);
        }
        else
        {
                pid2=getpid();
                printf("\n the child process ID is %d\n", pid2);
        }
}
```

Execution:

```
[root@dba ~]# cc -o 16 16.c
[root@dba ~]# ./16

 the child process ID is 4485
 the parent process ID is 4484
```

EXPERIMENT NO: 17 Date:

Aim:Write a C program to create zombie process

Algorithm:

Step 1:call fork function to create a child process
Step 2:if fork()>0
Then creation of Zombie
By applying sleep function for 10 seconds
Step 3: now terminate the child process
Step 4: exit status child process not reported to parent
Step 5: status any process which is zombie can known by
Applying ps(1) command
Step 6: stop

Program file name:17.c

```
#include <stdio.h>
#include<stdlib.h>
int main()
{
int pid;
        pid=fork();
        if(pid == 0)
        {
                printf("Iam child my pid is %d\n",getpid());
                 printf("My parent pid is:%d\n",getppid());
                 exit(0);
        }
         else
        {
                printf("I am parent, my pid is %d\n",getpid());
                sleep(100);
                exit(0);
         }
}
```

Execution:

To see zombie process, after running the program, open a new terminal Give this command $ps -el|grep a.out

First terminal

Compilation:
[root@dba ~]# cc 17.c
Executing binary
[root@dba ~]# ./a.out
Iam child my pid is 4732
My parent pid is:4731

I am parent, my pid is 4731

Checking for zombie process. Z means zombie process

Second terminal

[root@dba ~]# ps -el|grep a.out

0 S 0 4731 4585 0 77 0 - 384 - pts/3 00:00:00 a.out

1 Z 0 4732 4731 0 77 0 - 0 exit pts/3 00:00:00 a.out <defunct>

EXPERIMENT NO: 18 Date:

Aim:-Write a C program to illustrate how an orphan process is created

Algorithm:

Step 1: call the fork function to create the child process
Step 2:if (pid==0)
Then print child id and parent id
else goto step 4
Step 3:Then sleep(10)
Print child id and parent id
Step 4: Print child id and parent id
Step 5:which gives the information of orphan process
Step 6:stop

Program file name:18.c

```
#include <stdio.h>
#include<stdlib.h>
int main()
 {
int pid;
        printf("I am the original process with PID %d and PPID %d\n",getpid(),getppid());
        pid=fork();
        if(pid == 0)
        {
                printf("I am child, my pid is %d ",getpid());
                 printf("My Parent pid is:%d\n",getppid());
                sleep(10);
                 printf("Now my pid is %d  ",getpid());
                printf("My parent pid is:%d\n",getppid());
                exit(0);
        }
        else
        {
                sleep(10);
                printf("I am parent, my pid is %d\n",getpid());
                //printf("I am going to die\n");
        }
printf("PID:%d terminates...\n",getpid());
}
```

Execution:

Compilation :
[root@dba ~]# cc -o 18 18-1.c
Executing Binary:
[root@dba ~]# ./18
I am the original process with PID 5960 and PPID 5778
I am child, my pid is 5961 My Parent pid is:5960
I am parent, my pid is 5960
PID:5960 terminates...
[root@dba ~]# Now my pid is 5961 My parent pid is:1

EXPERIMENT NO: 19 Date:

Aim:-Write a C program that illustrate how to execute two commands concurrently with a command pipe Ex: ls –l|sort

Algorithm:

step 1:Start
step 2:call the fork function to create child process
Step 3 : if process id =0 goto step 4
else goto step 7
close writing option
Step 4:copy the old file descriptor
Step 5:exec the command ls –l(long listing files)
This process will be executed by child
Step 6: goto step 11
step 7:Close reading option
step 8: Copy old file descriptor
step 9:exec the command wc t count no of lines, words, characters
step 10: both commands concurrently executed by pipe
step 11:stop

program File name :19.c

```
#include<stdio.h>
 #include<fcntl.h>
 main()
{
        int pfd[2],p;
        pipe(pfd);
         p=fork();
        if(p==0)
        {
                close(pfd[0]);
                close(1);
                dup(pfd[1]);
                 execlp("ls","ls","-l",(char*)0);
         }
        else
        {
                close(pfd[1]);
                close(0);
                dup(pfd[0]); execlp("wc","wc",(char*)0);
         }
 }
```

EXPERIMENT NO: 20 Date:

Aim:- Write a C program that illustrate communication between two unrelated process using named pipes

Algorithm for server :

step 1:Start
step 2:Create a first named pipe by using mkfifo system call
Pipe1=mkfifo(NP1,0666).
step 3:if mkfifo returns -1 then
print a message that error in creating the pipe.
step 4:Create a second named pipe by using mkfifo system call
Pipe2=mkfifo(NP2,0666).
step 5:if mkfifo returns -1 then
print a message that error in creating the pipe.
step 6:Open the first pipe for reading by open system call by setting
O_RDONLY Fd=open(NP1,O_RDONLY)
step 7: Open the second pipe for writing by open system call by setting
O_WRONLY Fd=open(NP2,O_WRONLY)
step 8:read the data from the first pipe by using read system call
numread=Read(fd,buf,MAX_BUF-SIZE) buf*numread+='\0'
step 9:print the data that we have read from pipe
step 10:convert the data to the upper case.
step 11:write the converted string back to second pipe by write(fd,buf, strlen(buf))
step 12:stop.

Algorithm for client :

Step 1:start
Step 2:check whether the no of arguments specified were correct or not
Step 3:if no of arguments are less then print error message
Step 4:Open the first named pipe for writing by open system call by setting
O_WRONLY Fd=open(NP1,O_WRONLY)
Step 5: .Open the second named pipe for reading by open system call by setting
O_RDONLY Fd=open(NP2,O_RDONLY)
Step 6: write the data to the pipe by using write system call
write(fd,argv[1],strlen(argv[1]))
Step 7: read the data from the first pipe by using read system call
numread=Read(fd,buf,MAX_BUF_SIZE) buf*numread+='\0'
Step 8: print the data that we have read from pipe
Step 9:stop

Program file name:20.c

```
#include<stdio.h>
 #include<stdlib.h>
 #include<sys/types.h>
 #include<sys/stat.h>
#include<string.h>
#include<fcntl.h>
```

```
 void server(int,int);
 void client(int,int);
int main()
{
int p1[2],p2[2],pid;
        pipe(p1);
        pipe(p2);
        pid=fork();
        if(pid==0)
        {
         close(p1[1]);
        close(p2[0]);
        server(p1[0],p2[1]);
         return 0;
 }
        close(p1[0]);
        close(p2[1]);
        client(p1[1],p2[0]);
        wait();
return 0;
}

void client(int wfd,int rfd)
{
int i,j,n;
 char fname[2000];
 char buff[2000];
 printf("ENTER THE FILE NAME :");
 scanf("%s",fname);
 printf("CLIENT SENDING THE REQUEST.....PLEASE WAIT\n");

sleep(10);
 write(wfd,fname,2000);
n=read(rfd,buff,2000);
 buff[n]='\0';
printf("THE RESULTS OF CLIENTS ARE......\n");
write(1,buff,n);
}
void server(int rfd,int wfd)
 {
int i,j,n; char fname[2000];
        char buff[2000];
        n=read(rfd,fname,2000);
        fname[n]='\0';
        int fd=open(fname,O_RDONLY);
        sleep(10);
        if(fd<0)
                write(wfd,"can't open",9);
        else
                 n=read(fd,buff,2000);
        write(wfd,buff,n);
}
```

EXPERIMENT NO: 22 Date:

Aim:Write a C program to create message queue with read and write permissions to write 3 messages to it with different priority numbers

Algorithm:

Step 1.Start
Step 2.Declare a message queue structure
typedef struct msgbuf {
long mtype;
char mtext[MSGSZ];
} message_buf;
Mtype =0 Retrieve the next message on the queue, regardless of its mtype.
Positive Get the next message with an mtype equal to the specified msgtyp.
Negative Retrieve the first message on the queue whose mtype field is less than or equal to the absolute value of the msgtyp argument.

Usually mtype is set to1 and mtext is the data this will be added to the queue.
Step 3.Create message queue and store result in msgflg
msgflg = IPC_CREAT | 0666
msgflg argument must be an octal integer with settings for the queue's permissions and control flags.

Step 4. Get the message queue id for the "name" 1234, which was created by the server
key = 1234

Step 5. if ((msqid = msgget(key, msgflg)) < 0)
Then print error
The msgget() function shall return the message queue identifier associated with the argument key.
Step 6. otherwise
Print msgget succeeded: msqid
Step 7. Now we will send message of type 1 through message queue
message_buf sbuf sbuf.mtype=1
Step 8.mtext contains data to be send through message queue By using strcpy function we we
will copy some message Into mtext
strcpy(sbuf.mtext, "Did you get this?")
Step 9. then find the length of text by using strlen function buf_length = strlen(sbuf.mtext) + 1
Step 10.now we will send message to message to the message queue By using msgsnd function
#include < sys/msg.h>
int msgsnd(int msqid, const void *msgp, size_t msgsz, int msgflg);
msgsnd(msqid, &sbuf, buf_length, IPC_NOWAIT)msqid: message queue id
&sbuf: pointer to user defined structure Buf_length :length of data
IPC_NOWAIT: returns error on wait
Step 11.if msgsnd <0 return error
Step 12.otherwise print message sent is sbuf.mext
13.stop

Programs

Header file: msg1.h

```
#define MAX_MSG_LEN 200              /* Maximum message length */
#define MSGQ_KEY 100                 /* An arbirary key (name) for message queue */
#define BLOCKED 0                    /* Flag value of 0 in 'msgsnd' indicates 'block possible */
#define NUM_MSGS 3                   /* Number of messages in the queue */

struct struct_msg_datagram           /* Structure to hold messages in the queue */
{
      long mtype;                    /* Message type */
      char mtext[MAX_MSG_LEN];       /* Message text */
};

void print_qstat(int qd, struct msqid_ds * qstat)
{
      printf("PID = %d\n", getpid());
      printf("Queue %d Status:\n----------------------\n", (int)qd);
      printf("# of messages in queue = %d\n", (int)qstat->msg_qnum);
      printf("Queue owner's effective user ID = %d\n", (int)qstat->msg_perm.uid);
      printf("Queue owner's effective group ID = %d\n", (int)qstat->msg_perm.gid);
      printf("Queue creator's effective user ID = %d\n", (int)qstat->msg_perm.cuid);
      printf("Queue creator's effective group ID = %d\n", (int)qstat->msg_perm.cgid);
      printf("Permissions = %o\n", (int)qstat->msg_perm.mode);
      printf("Last sender's PID = %d\n", (int)qstat->msg_lspid);
      printf("Last receiver's PID = %d\n", (int)qstat->msg_lrpid);
      printf("Time of last send = %s", ctime(&qstat->msg_stime));
      printf("Time of last receive = %s", ctime(&qstat->msg_rtime));
      printf("Time of last change = %s\n", ctime(&qstat->msg_ctime));
}
```

Writer:writer21.c

```
#include <stdio.h>                   /* printf, etc. */
#include <stdlib.h>                  /* exit, etc. */
#include <string.h>                  /* strcpy, strlen, etc. */
#include <time.h>                    /* ctime, etc. */
#include <sys/msg.h>                 /* msgget, msgsnd, msgrcv, MSGMAX, etc. */
#include <sys/types.h>               /* Data type 'key_t' for 1st arg of 'msgget' */
#include <sys/ipc.h>                 /* 'struct ipc_perm' in 'struct msgid_ds' in 'msgctl' */
#include "21msgq1.h"                 /* User defined header file for message queues */

/* Define all the error messages */
char * error_msg[] = {
            "\nUsage: ./21msgqwrite1\n\n",
            "\nError while creating message queue with 'msgget'\n\n",
            "\nError while sending message to queue with 'msgsnd'\n\n",
            "\nError while retrieving status of queue with 'msgctl'\n\n"
};
```

```
/* Declare the prototype for function 'print_error' */
void print_error(int msg_num, int exit_code);

/* Declare the prototype for function 'print_qstat' */
void print_qstat(int qd, struct msqid_ds * qstat);

int main(int argc, char * argv[])
{
	int i;			/* A local counter */
	int qd;			/* Queue descriptor */
	struct struct_msg_datagram msg_datagram;	/* Holds msg type and msg text */
	struct msqid_ds qstat;		/* To retrieve queue status */
	char * message[] = {		/* Messages for the queue */
		"Message 1, Linux is powerful",
		"Message 2, These are System V standards of message queues",
		"Message 3, Message queues is just 1 mechanism for IPC",
		"Message 4, Other IPC mechanisms include semaphores",
		"Message 5, Still other mechanism for IPC is shared memory",
		"Message 6, Further more memory mapped I/O also allows IPC"
	};

	/* Check if server is run using correct syntax */
	if ( argc != 1 )
		print_error(0,2);

	/* Try to get the queue descriptor */
	if ( ( qd = msgget(MSGQ_KEY, IPC_CREAT | 0666) ) == -1 )
		print_error(1,3);

	/* Retrieve queue status */
	if ( (msgctl(qd, IPC_STAT, &qstat)) == -1 )
		print_error(3,5);

	/* Display the initial status of queue */
	printf("\nMessage queue created with ID %d\n\n", qd);
	printf("Initial status of queue is as follows:\n");
	print_qstat(qd, &qstat);

	/* Write down the messages to the queue */
	for ( i = 0; i < NUM_MSGS; i++ )
	{
		/* Prepare the message datagram */
		msg_datagram.mtype = i+1;
		sprintf(msg_datagram.mtext,"%s",message[i]);

		printf("\nSending to queue %d the message:\n\t'%s'\n\n", qd, message[i]);

		/* Send the message to queue */
		if ( (msgsnd(qd, &msg_datagram, strlen(message[i])+1, BLOCKED)) == -1 )
			print_error(2,4);

		/* Retrieve queue status */
```

```
        if ( (msgctl(qd, IPC_STAT, &qstat)) == -1 )
            print_error(3,5);

        /* Display the current status of queue */
        printf("Current status of queue is as follows:\n");
        print_qstat(qd, &qstat);
    }

    return 0;
}
void print_error(int error_index, int exit_code)
{
    fprintf(stderr, "%s", error_msg[error_index]);
    exit(exit_code);
}
```

Reader:reader21.c

```
#include <stdio.h>              /* printf, etc. */
#include <stdlib.h>             /* exit, etc. */
#include <string.h>             /* strcpy, strlen, etc. */
#include <time.h>               /* ctime, etc. */
#include <sys/msg.h>            /* msgget, msgsnd, msgrcv, MSGMAX, etc. */
#include <sys/types.h>          /* Data type 'key_t' for 1st arg of 'msgget' */
#include <sys/ipc.h>            /* 'struct ipc_perm' in 'struct msgid_ds' in 'msgctl' */
#include "21msgq1.h"            /* User defined header file for message queues */

/* Define all the error messages */
char * error_msg[] = {
   "\nUsage: ./22msgqread1\n\n",
   "\nError while acquiring message queue with 'msgget'\n\n",
   "\nError while receiving message from queue with 'msgsnd'\n\n",
   "\nError while retrieving status of queue with 'msgctl'\n\n"
};

/* Declare the prototype for function 'print_error' */
void print_error(int msg_num, int exit_code);

/* Declare the prototype for function 'print_qstat' */
void print_qstat(int qd, struct msqid_ds * qstat);

int main(int argc, char * argv[])
{
   int i;                       /* A local counter */
   int qd;                      /* Queue descriptor */
   struct struct_msg_datagram msg_datagram;    /* Holds msg type and msg text */
   struct msqid_ds qstat;           /* To retrieve queue status */
   char message[NUM_MSGS][MAX_MSG_LEN];       /* Messages for the queue */
   int num_bytes_read;              /* Actual num of bytes read from msg queue */

   /* Check if server is run using correct syntax */
   if ( argc != 1 )
```

```
        print_error(0,2);

    /* Try to get the queue descriptor */
    if ( ( qd = msgget(MSGQ_KEY, 0) ) == -1 )
        print_error(1,3);

    /* Retrieve queue status */
    if ( (msgctl(qd, IPC_STAT, &qstat)) == -1 )
        print_error(3,5);

    /* Display the initial status of queue */
    printf("\nMessage queue acquired with ID %d\n\n", qd);
    printf("Initial status of queue is as follows:\n");
    print_qstat(qd, &qstat);

    /* Read the messages from the queue */
    for ( i = 0; i < NUM_MSGS; i++ )
    {
        /* Retrieve a message datagram from queue */
        if    (    (num_bytes_read =
                    msgrcv(qd, &msg_datagram, MAX_MSG_LEN, i+1, IPC_NOWAIT |
MSG_NOERROR)
                )
                == -1
            )
            print_error(2,4);

        /* Extract the message from datagram */
        sprintf(message[i],"%s",msg_datagram.mtext);

        printf("\nRetrieved from queue %d the message (%d bytes read):\n\t'%s'\n\n",
                qd, num_bytes_read, message[i]);

        /* Retrieve queue status */
        if ( (msgctl(qd, IPC_STAT, &qstat)) == -1 )
            print_error(3,5);
        /* Display the current status of queue */
        printf("Current status of queue is as follows:\n");
        print_qstat(qd, &qstat);
    }
    return 0;
}
void print_error(int error_index, int exit_code)
{
    fprintf(stderr, "%s", error_msg[error_index]);
    exit(exit_code);
}
```

EXPERIMENT NO: 23 Date:

Aim:-Write a C program that receives a message from message queue and display them

Algorithm:

Step 1:Start
Step 2:Declare a message queue structure
typedef struct msgbuf {
long mtype;
char mtext[MSGSZ];
} message_buf;
Mtype =0 Retrieve the next message on the queue, regardless of its mtype.
PositiveGet the next message with an mtype equal to the specified
msgtyp.
Negative Retrieve the first message on the queue whose mtype field is
less than or equal to the absolute value of the msgtyp argument.
Usually mtype is set to1
mtext is the data this will be added to the queue.
Step 3:Get the message queue id for the "name" 1234, which was created by the server
key = 1234
Step 4 : if ((msqid = msgget(key, 0666< 0) Then print error
The msgget() function shall return the message queue identifier associated with the argument key.
Step 5: Receive message from message queue by using msgrcv function
int msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp, int msgflg);
#include < sys/msg.h>
(msgrcv(msqid, &rbuf, MSGSZ, 1, 0)
msqid: message queue id
&sbuf: pointer to user defined structure MSGSZ: message size
Message type: 1
Message flag:The msgflg argument is a bit mask constructed by ORing together zero or more of the following flags: IPC_NOWAIT or MSG_EXCEPT or MSG_NOERROR
Step 6:if msgrcv <0 return error
Step 7:otherwise print message sent is sbuf.mext
Step 8:stop

EXPERIMENT NO: 24 Date:

Aim:-Write a C program that illustrate the suspending and resuming process using signal

Algorithm:

Step 1: call the signal function to generate the signal
Step 2:execution of process will be started
Step 3:call alarm function to suspend the execution of current process
Step 4:then it will execute the signal function
Step 5:again the process will be resumed
Step 6:stop

Program

```
#include<stdio.h>
int main()
{
int n;
        if(signal(SIGALRM,sig_alarm)==SIG_ERR)
        printf(,Signal error');
        alarm(5);
        for(n=0;n<=15;n++)
                printf(,from for loop n=%d',n);
        printf(,main program terminated');
}

void sig_alarm(int signo)
{
        printf(,from sigalarm function');
}
```

EXPERIMENT NO: 25 Date:

Aim:-Write client server programs using c for interaction between server and client process using Unix Domain sockets

Algorithm:-

Sample UNIX server
Step 1:define NAME "socket"
Step 2: sock = socket(AF_UNIX, SOCK_STREAM, 0);
Step 3:if (sock < 0) perror("opening stream socket"); exit(1);
step4: server.sun_family = AF_UNIX;
strcpy(server.sun_path, NAME);
if (bind(sock, (struct sockaddr *) &server, sizeof(struct sockaddr_un)))
{
perror("binding stream socket"); exit(1);
}
step 5: print ("Socket has name %s\n", server.sun_path);
listen(sock, 5);
step 6: for (;;)
{
msgsock = accept(sock, 0, 0);
if (msgsock == -1)
perror("accept");
else
do { bzero(buf, sizeof(buf));
if ((rval = read(msgsock, buf, 1024)) < 0)

perror("reading stream message");
else if (rval == 0)
else print ("-->%s\n", buf);
} while (rval > 0);
close(msgsock);
}
close(sock);
unlink(NAME);
}

Step 7:stop

Programs:

Server.c

```
#include <stdio.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
```

```
int connection_handler(int connection_fd)
{
 int nbytes;
 char buffer[256];
 nbytes = read(connection_fd, buffer, 256);
 buffer[nbytes] = 0;
 printf("MESSAGE FROM CLIENT: %s\n", buffer);
 nbytes = snprintf(buffer, 256, "hello from the server");
 write(connection_fd, buffer, nbytes);
 close(connection_fd);
 return 0;
}

int main(void)
{
 struct sockaddr_un address;
 int socket_fd, connection_fd;
 socklen_t address_length;
 pid_t child;
 socket_fd = socket(PF_UNIX, SOCK_STREAM, 0);
 if(socket_fd < 0)
 {
  printf("socket() failed\n");
  return 1;
 }

 unlink("./demo_socket");

 /* start with a clean address structure */
 memset(&address, 0, sizeof(struct sockaddr_un));

 address.sun_family = AF_UNIX;
 snprintf(address.sun_path, UNIX_PATH_MAX, "./demo_socket");

 if(bind(socket_fd,
         (struct sockaddr *) &address,
         sizeof(struct sockaddr_un)) != 0)
 {
  printf("bind() failed\n");
  return 1;
 }

 if(listen(socket_fd, 5) != 0)
 {
  printf("listen() failed\n");
```

```
 return 1;
}

while((connection_fd = accept(socket_fd,
                        (struct sockaddr *) &address,
&address_length)) > -1)
{
 child = fork();
 if(child == 0)
 {
  /* now inside newly created connection handling process */
  return connection_handler(connection_fd);
 }
 /* still inside server process */
 close(connection_fd);
}
close(socket_fd);
unlink("./demo_socket");
return 0;
}
```

Client.c

```
#include <stdio.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#include <string.h>

int main(void)
{
 struct sockaddr_un address;
 int  socket_fd, nbytes;
 char buffer[256];

 socket_fd = socket(PF_UNIX, SOCK_STREAM, 0);
 if(socket_fd < 0)
 {
  printf("socket() failed\n");
  return 1;
 }
 /* start with a clean address structure */
 memset(&address, 0, sizeof(struct sockaddr_un));

 address.sun_family = AF_UNIX;
 snprintf(address.sun_path, UNIX_PATH_MAX, "./demo_socket");
```

```
if(connect(socket_fd,
        (struct sockaddr *) &address,
        sizeof(struct sockaddr_un)) != 0)
{
 printf("connect() failed\n");
 return 1;
}
nbytes = snprintf(buffer, 256, "hello from a client");
write(socket_fd, buffer, nbytes);

nbytes = read(socket_fd, buffer, 256);
buffer[nbytes] = 0;

printf("MESSAGE FROM SERVER: %s\n", buffer);

close(socket_fd);
return 0;
}
```

EXPERIMENT NO: 26 Date:

Aim:-Write client server programs using c for interaction between server and client process using Internet Domain sockets

Programs

Server.c

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>

int main(int argc, char *argv[])
{
   int listenfd = 0, connfd = 0;
   struct sockaddr_in serv_addr;

   char sendBuff[1025];
   time_t ticks;

   listenfd = socket(AF_INET, SOCK_STREAM, 0);
   memset(&serv_addr, '0', sizeof(serv_addr));
   memset(sendBuff, '0', sizeof(sendBuff));

   serv_addr.sin_family = AF_INET;
   serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
   serv_addr.sin_port = htons(5000);

   bind(listenfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr));

   listen(listenfd, 10);

   while(1)
   {
      connfd = accept(listenfd, (struct sockaddr*)NULL, NULL);

      ticks = time(NULL);
      snprintf(sendBuff, sizeof(sendBuff), "%.24s\r\n", ctime(&ticks));
```

```
        write(connfd, sendBuff, strlen(sendBuff));

        close(connfd);
        sleep(1);
     }
}
```

Client.c

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>

int main(int argc, char *argv[])
{
    int sockfd = 0, n = 0;
    char recvBuff[1024];
    struct sockaddr_in serv_addr;

    if(argc != 2)
    {
        printf("\n Usage: %s <ip of server> \n",argv[0]);
        return 1;
    }

    memset(recvBuff, '0',sizeof(recvBuff));
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Error : Could not create socket \n");
        return 1;
    }

    memset(&serv_addr, '0', sizeof(serv_addr));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(5000);

    if(inet_pton(AF_INET, argv[1], &serv_addr.sin_addr)<=0)
    {
```

```
        printf("\n inet_pton error occured\n");
        return 1;
    }

    if( connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
      printf("\n Error : Connect Failed \n");
      return 1;
    }

    while ( (n = read(sockfd, recvBuff, sizeof(recvBuff)-1)) > 0)
    {
        recvBuff[n] = 0;
        if(fputs(recvBuff, stdout) == EOF)
        {
            printf("\n Error : Fputs error\n");
        }
    }
    if(n < 0)
    {
        printf("\n Read error \n");
    }

    return 0;
}
```

EXPERIMENT NO: 27 Date:

Aim:-Write a C program that illustrates two processes communicating using Shared memory

Algorithm:-

step1.Start
step 2.Include header files required for the program are
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
step 3.Declare the variable which are required as
pid_t pid
int *shared /* pointer to the shm */
int shmid
step 4.Use shmget function to create shared memory
#include <sys/shm.h>
int shmget(key_t key, size_t size, int shmflg)
The shmget() function shall return the shared memory identifier associated with key The argument key is equal to IPC_PRIVATE. so that the operating system selects the next available key for a newly created shared block of memory. Size represents size of shared memory block Shmflg shared memory permissions which are represented by octal integer
shmid = shmget (IPC_PRIVATE, sizeof(int), IPC_CREAT | 0666);
print the shared memory id
step 5.if fork()==0 Then
begin
shared = shmat(shmid, (void *) 0, 0)
print the shared variable(shared) *shared=2
print *shared sleep(2)
print *shared
end
step 6.else
begin
shared = shmat(shmid, (void *) 0, 0)
print the shared variable(shared)
print *shared sleep(1) *shared=30
printf("Parent value=%d\n", *shared);
sleep(5)
shmctl(shmid, IPC_RMID, 0)
end
step 7.stop.

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
```

```
int main(void) {
pid_t pid;
int *shared; /* pointer to the shm */ int shmid;
shmid = shmget(IPC_PRIVATE, sizeof(int), IPC_CREAT | 0666); printf("Shared Memory
ID=%u",shmid);
if (fork() == 0) { /* Child */

/* Attach to shared memory and print the pointer */ shared = shmat(shmid, (void *) 0, 0);
printf("Child pointer %u\n", shared); *shared=1;
printf("Child value=%d\n", *shared); sleep(2);
printf("Child value=%d\n", *shared); } else { /* Parent */
/* Attach to shared memory and print the pointer */ shared = shmat(shmid, (void *) 0, 0);
printf("Parent pointer %u\n", shared); printf("Parent value=%d\n", *shared); sleep(1);
*shared=42;
printf("Parent value=%d\n", *shared); sleep(5);
shmctl(shmid, IPC_RMID, 0);
}

}
```

```
sampath@localhost ipc]$cc shared_mem.c
 [sampath@localhost ipc]$ ./a.out
Shared Memory ID=65537Child pointer 3086680064 Child value=1
Shared Memory ID=65537Parent pointer 3086680064 Parent value=1
Parent value=42 Child value=42
```

1.define shared memory

2.what are file locking functions. 3.what are shared memory system calls.
4.define internet domain sockets
5.Difference between internet and unix domain sockets.