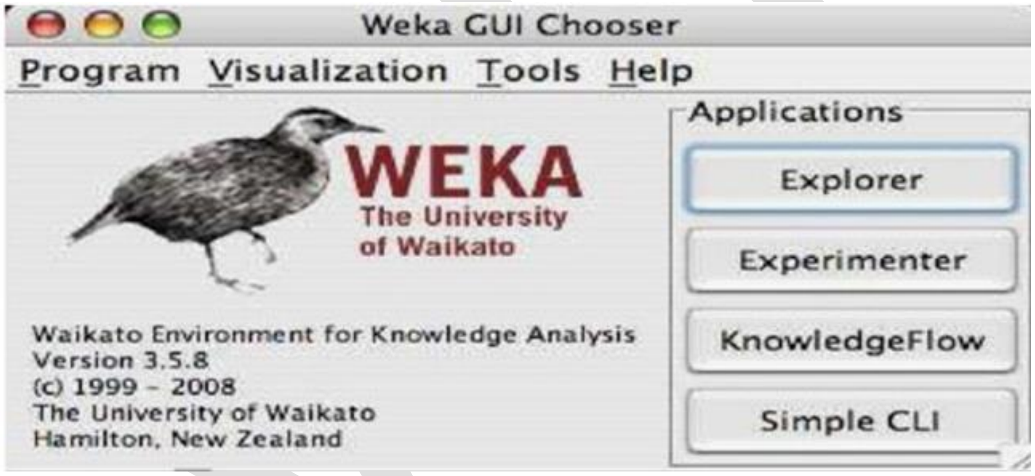
WEKA

Weka is created by researchers at the university WIKATO in New Zealand. University of Waikato, Hamilton, New Zealand Alex Seewald (original Command-line primer) David Scuse (original Experimenter tutorial)

- It is java based application.
- It is collection often source, Machine Learning Algorithm.
- The routines (functions) are implemented as classes and logically arranged in packages.
- It comes with an extensive GUI Interface.
- Weka routines can be used standalone via the command line interface.

The Graphical User Interface;-



Waikato Environment for Knowledge Analysis
Version 3.5.8
(c) 1999 - 2008
The University of Waikato
Hamilton, New Zealand

The buttons can be used to start the following applications:

- **Explorer An environment** for exploring data with WEKA (the rest of this Documentation deals with this application in more detail).

 Experimenter: An environment for performing experiments and conducting statistical tests between learning schemes.

- **Knowledge Flow** This environment supports essentially the same functions as the Explorer but with a drag-and-drop interface. One advantage is that it supports incremental learning.

- **SimpleCLI Provides** a simple command-line interface that allows direct execution of WEKA commands for operating systems that do not provide their own command line interface.

1. Explorer

The Graphical user interface

Section Tabs

At the very top of the window, just below the title bar, is a row of tabs. When the Explorer is first started only the first tab is active; the others are grayed out. This is because it is necessary to open (and potentially pre-process) a data set before starting to explore the data.

The tabs are as follows:

1. **Preprocess.** Choose and modify the data being acted on.

2. **Classify.** Train & test learning schemes that classify or perform regression

3. **Cluster.** Learn clusters for the data.

4. **Associate.** Learn association rules for the data.

5. **Select attributes.** Select the most relevant attributes in the data.

6. **Visualize.** View an interactive 2D plot of the data.

Once the tabs are active, clicking on them flicks between different screens, on which the respective actions can be performed. The bottom area of the window (including the status box, the log button, and the Weka bird) stays visible regardless of which section you are in. The Explorer can be easily extended with custom tabs.

Study the ARFF file format

ARFF File Format

An ARFF (= Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes.

ARFF files are not the only format one can load, but all files that can be converted with

Week 1: Demonstrate Apriori based Association Rule Mining

Steps for run Aprior algorithm in WEKA

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Choose WEKA folder in C drive.
6. Select and Click on data option button.
7. Choose Weather data set and open file.
8. Click on Associate tab and Choose Aprior algorithm
9. Click on start button.

Output :
```
=== Run information ===
Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -
1
Relation:     weather.symbolic
Instances:    14
Attributes:   5
outlook
temperature
humidity
windy play
=== Associator model (full training set) ===
Apriori
=======

Minimum support: 0.15 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17
```

Generated sets of large itemsets:

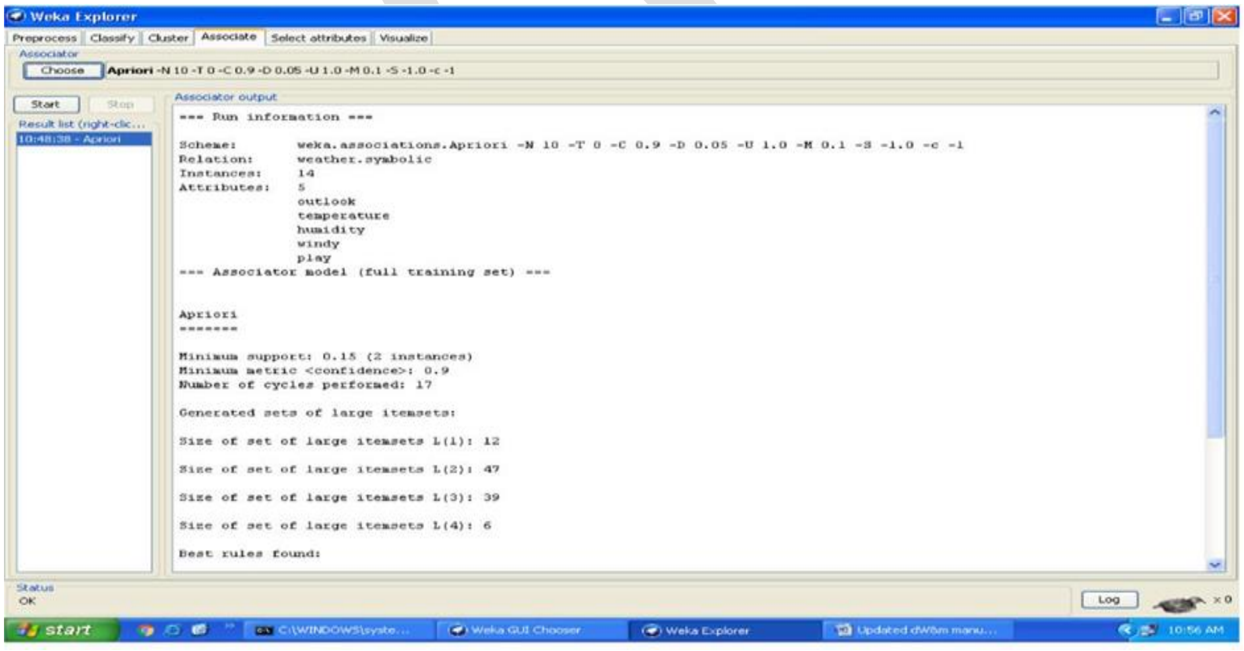
Size of set of large itemsets L(1): 12

Size of set of large itemsets L(2): 47
Size of set of large itemsets L(3): 39

Size of set of large itemsets L(4): 6

Best rules found:

1. outlook=overcast 4 ==> play=yes 4 conf:(1)
2. temperature=cool 4 ==> humidity=normal 4 conf:(1)
3. humidity=normal windy=FALSE 4 ==> play=yes 4 conf:(1)
4. outlook=sunny play=no 3 ==> humidity=high 3 conf:(1)
5. outlook=sunny humidity=high 3 ==> play=no 3 conf:(1)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3 conf:(1)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3 conf:(1)
8. temperature=cool play=yes 3 ==> humidity=normal 3 conf:(1)
9. outlook=sunny temperature=hot 2 ==> humidity=high 2 conf:(1)
10. temperature=hot play=no 2 ==> outlook=sunny 2 conf:(1)



Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation:     weather.symbolic
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.15 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12

Size of set of large itemsets L(2): 47

Size of set of large itemsets L(3): 39

Size of set of large itemsets L(4): 6

Best rules found:
```

Association Rule:

An association rule has two parts, an antecedent (if) and a consequent (then). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent.

Association rules are created by analyzing data for frequent if/then patterns and using the criteria support and confidence to identify the most important relationships. Support is an indication of how frequently the items appear in the database. Confidence indicates the number of times the if/then statements have been found to be true.

In data mining, association rules are useful for analyzing and predicting customer behavior. They play an important part in shopping basket data analysis, product clustering, catalog design and store layout.

Support and Confidence values:

- Support count: The support count of an itemset *X*, denoted by *X.count*, in a data set *T* is the number of transactions in *T* that contain *X*. Assume *T* has *n* transactions.
- Then,

Support = support({*A* U *C*})

Confidence = support({*A* U *C*})/support({*A*})

Week 2: Demonstrate FP – growth based Association Rule Mining

Steps for run FP-growth algorithm in WEKA

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Choose WEKA folder in C drive.
6. Select and Click on data option button.
7. Choose Weather data set and open file.
8. Click on Associate tab and Choose fp-growth algorithm
9. Click on start button.

Output : === Run information ===

```
Scheme:       weka.associations.FP-TREE -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -
1
Relation:     weather.symbolic
Instances:    14
Attributes:   5
outlook
temperature
humidity
windy play
=== Associator model (full training set) ===FP-growth=======
Minimum support: 0.15 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17
```

```
Generated sets of large itemsets:

Size of set of large itemsets L(1): 12

Size of set of large itemsets L(2): 47
Size of set of large itemsets L(3): 39
Size of set of large itemsets L(4): 6

Best rules found:

1. outlook=overcast 4 ==> play=yes 4  conf:(1)
2. temperature=cool 4 ==> humidity=normal 4  conf:(1)
3. humidity=normal windy=FALSE 4 ==> play=yes 4    conf:(1)
4. outlook=sunny play=no 3 ==> humidity=high 3    conf:(1)
5. outlook=sunny humidity=high 3 ==> play=no 3    conf:(1)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3    conf:(1)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3    conf:(1)
8. temperature=cool play=yes 3 ==> humidity=normal 3   conf:(1)
9. outlook=sunny temperature=hot 2 ==> humidity=high 2   conf:(1)
10. temperature=hot play=no 2 ==> outlook=sunny 2   conf:(1)
```

Week 3: Analyze a sample dataset using classification technique in WEKA Tool

Test Options

The result of applying the chosen classifier will be tested according to the options that are set by clicking in the Test options box. There are four test modes:

1. Use training set. The classifier is evaluated on how well it predicts the class of the instances it was trained on.

2. Supplied test set. The classifier is evaluated on how well it predicts the class of a set of instances loaded from a file. Clicking the Set... button brings up a dialog allowing you to choose the file to test on.

3. Cross-validation. The classifier is evaluated by cross-validation, using the number of folds that are entered in the Folds text field.

4. Percentage split. The classifier is evaluated on how well it predicts a certain percentage of the data which is held out for testing. The amount of data held out depends on the value entered in the % field.

1. Load each dataset into Weka and run id3, j48 classification algorithm, study the classifier output. .

Ans:

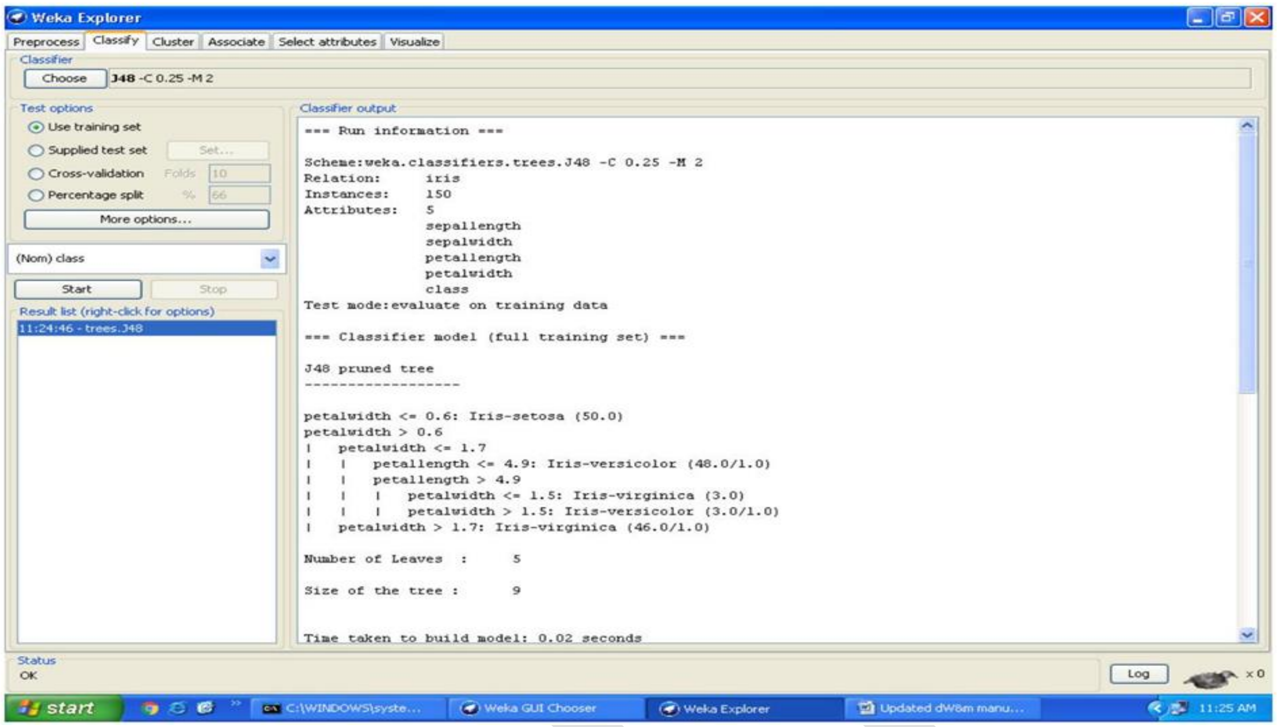
teps for run ID3 and J48 Classification algorithms in WEKA

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Choose WEKA folder in C drive.
6. Select and Click on data option button.
7. Choose iris data set and open file.
8. Click on classify tab and Choose J48 algorithm and select use training set test option.
9. Click on start button.

10. Click on classify tab and Choose ID3 algorithm and select use training set test option.
11. Click on start button.

Output:

```
=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth class

Test mode: evaluate on training data

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

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

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
|  petalwidth <= 1.7
|  |  petallength <= 4.9: Iris-versicolor (48.0/1.0)
|  |  petallength > 4.9
|  |  |  petalwidth <= 1.5: Iris-virginica (3.0)
|  |  |  petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|  petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves :      5

Size of the tree :      9


Time taken to build model: 0 seconds

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

Correctly Classified Instances          147               98        %
Incorrectly Classified Instances          3                2        %
Kappa statistic                        0.97
K&B Relative Info Score               14376.1925 %
K&B Information Score                   227.8573 bits     1.519   bits/instance
Class complexity | order 0            237.7444 bits       1.585 bits/instance
Class complexity | scheme               16.7179 bits      0.1115 bits/instance
                                                            1.4735
Complexity improvement   (Sf)            221.0265 bits      bits/instance
Mean absolute error                    0.0233
Root mean squared
error                                    0.108
Relative absolute error                 5.2482 %
Root relative squared error           22.9089 %
Total Number of Instances                150

=== Detailed Accuracy By Class ===

TP Rate         FP Rate  Precision  Recall F-Measure   ROC Area Class
1         0          1         1         1        1       Iris-setosa
0.98        0.02       0.961      0.98    0.97        0.99   Iris-versicolor
0.96        0.01       0.98       0.96    0.97       0.99   Iris-virginica
Weighted Avg.       0.98      0.01      0.98        0.98       0.98      0.993

=== Confusion Matrix ===

a b c <-- classified as 50 0 0
| a = Iris-setosa
0 49 1 | b = Iris-versicolor
0 2 48 | c = Iris-virginica
```



Classifier: J48 -C 0.25 -M 2

Test options: [✓ Use training set] Supplied test set, Cross-validation (Folds 10), Percentage split (% 66)

(Nom) class

Result list (right-click for options): 11:24:46 - trees.J48

Classifier output

```
=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:evaluate on training data

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

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

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
|   petalwidth <= 1.7
|   |   petallength <= 4.9: Iris-versicolor (48.0/1.0)
|   |   petallength > 4.9
|   |   |   petalwidth <= 1.5: Iris-virginica (3.0)
|   |   |   petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|   petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves  :     5

Size of the tree :      9


Time taken to build model: 0.02 seconds
```

Status: OK

2. Load each dataset into Weka and perform Naïve-bayes classification and k-Nearest Neighbor classification, Interpret the results obtained.

Ans:

- Steps for run Naïve-bayes and k-nearest neighbor Classification algorithms in WEKA

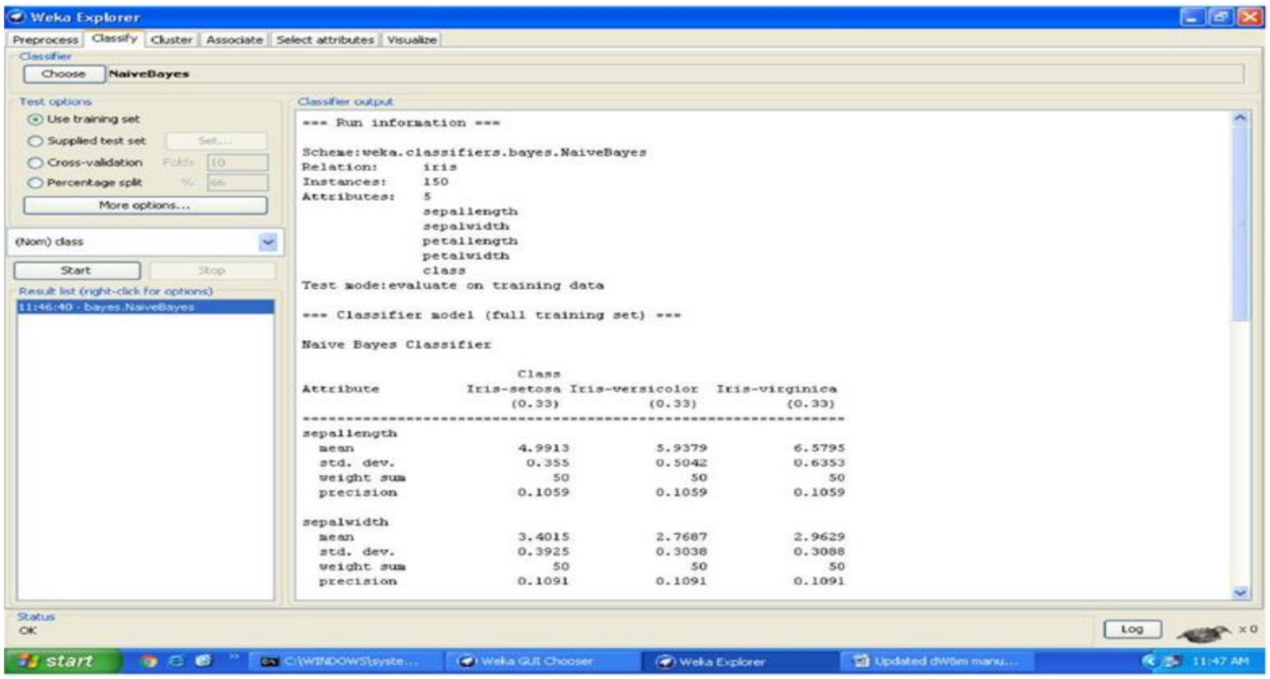
1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Choose WEKA folder in C drive.
6. Select and Click on data option button.
7. Choose iris data set and open file.
8. Click on classify tab and Choose Naïve-bayes algorithm and select use training set test option.
9. Click on start button.
10. Click on classify tab and Choose k-nearest neighbor and select use training set test option.
11. Click on start button.

Output: Naïve Bayes

```
=== Run information ===

Scheme:weka.classifiers.bayes.NaiveBayes
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth class

Test mode:evaluate on training data


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

Naive Bayes Classifier

Class
Attribute           Iris-setosa Iris-versicolor Iris-virginica
(0.33)          (0.33)          (0.33)
===============================================================
sepallength
mean                  4.9913        5.9379        6.5795
std. dev.             0.355         0.5042        0.6353
weight sum                50            50            50
precision             0.1059      0.1059        0.1059

sepalwidth
mean                  3.4015        2.7687        2.9629
std. dev.            0.3925        0.3038        0.3088
weight sum                50            50            50
precision             0.1091      0.1091        0.1091

petallength
```

```
mean               1.4694      4.2452      5.5516
std. dev.          0.1782      0.4712      0.5529
weight sum             50          50          50
precision          0.1405      0.1405      0.1405

petalwidth
mean               0.2743      1.3097      2.0343
std. dev.          0.1096      0.1915      0.2646
weight sum             50          50          50
precision          0.1143      0.1143      0.1143



Time taken to build model: 0 seconds

===  Evaluation on training set ===


===  Summary ===
Correctly Classified Instances          144            96      %
Incorrectly Classified Instances          6             4      %
Kappa statistic                       0.94
Mean absolute error                       0.0324
Root mean squared
error                                          0.1495
Relative absolute error                     7.2883 %
Root relative squared error                  31.7089 %
Total Number of Instances               150

=== Detailed Accuracy By Class ===

TP Rate       FP Rate  Precision  Recall F-Measure     ROC Area Class
1       0         1         1        1         1       Iris-setosa
0.96       0.04      0.923   0.96      0.941      0.993  Iris-versicolor
0.92       0.02      0.958   0.92      0.939      0.993  Iris-virginica
Weighted Avg.     0.96    0.02       0.96     0.96     0.96      0.995

=== Confusion Matrix ===
a b c <-- classified as
50 0 0 | a = Iris-setosa
0 48 2 | b = Iris-versicolor
```

0 4 46 | c = Iris-virginica



```
=== Run information ===

Scheme:weka.classifiers.bayes.NaiveBayes
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:evaluate on training data

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

Naive Bayes Classifier

                          Class
Attribute           Iris-setosa Iris-versicolor  Iris-virginica
                         (0.33)           (0.33)          (0.33)
===============================================================
sepallength
  mean                   4.9913           5.9379          6.5795
  std. dev.               0.355           0.5042          0.6353
  weight sum                 50               50              50
  precision              0.1059           0.1059          0.1059

sepalwidth
  mean                   3.4015           2.7687          2.9629
  std. dev.              0.3925           0.3038          0.3088
  weight sum                 50               50              50
  precision              0.1091           0.1091          0.1091
```

Output: KNN (IBK)

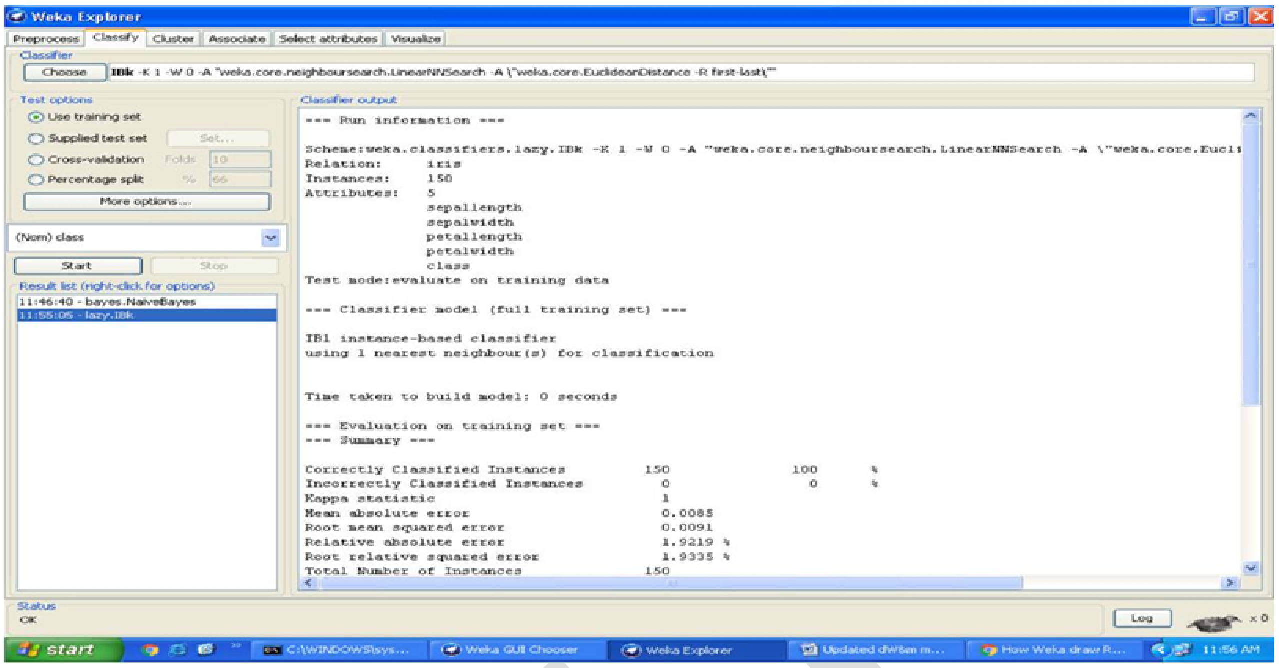
=== Run information ===

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
class
Test mode:evaluate on training data

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

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


Time taken to build model: 0 seconds

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

Correctly Classified Instances          150              100        %
Incorrectly Classified Instances          0                0      %
Kappa statistic                       1
Mean absolute error                    0.0085
Root mean squared error                  0.0091
Relative absolute error                1.9219 %
Root relative squared
error                                    1.9335 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

TP Rate      FP Rate    Precision  Recall  F-Measure  ROC Area  Class
1          0          1        1        1        1       Iris-setosa
1          0          1        1        1        1       Iris-versicolor
1          0          1        1        1        1       Iris-virginica
Weighted Avg.  1             0        1        1         1          1

=== Confusion Matrix ===

a b c <-- classified as 50 0 0
| a = Iris-setosa
0 50 0 | b = Iris-versicolor
0 0 50 | c = Iris-virginica
```



Classifier: IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Test options: (● Use training set)

Result list:
11:46:40 - bayes.NaiveBayes
11:55:05 - lazy.IBk

```
=== Run information ===

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.Eucli
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:evaluate on training data

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

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


Time taken to build model: 0 seconds

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

Correctly Classified Instances         150              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0.0085
Root mean squared error                  0.0091
Relative absolute error                  1.9219 %
Root relative squared error              1.9335 %
Total Number of Instances              150
```

Week 4: Demonstrate K-means Clustering algorithm

Selecting a Cluster

By now you will be familiar with the process of selecting and configuring objects. Clicking on the clustering scheme listed in the Cluster box at the top of the window brings up a GenericObjectEditor dialog with which to choose a new clustering scheme.

1. Load each dataset into Weka and run simple k-means clustering algorithm with different values of k(number of desired clusters). Study the clusters formed. Observe the sum of squared errors and centroids, and derive insights.

Ans:

Steps for run K-mean Clustering algorithms in WEKA

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Choose WEKA folder in C drive.
6. Select and Click on data option button.
7. Choose iris data set and open file.
8. Click on cluster tab and Choose k-mean and select use training set test option.
9. Click on start button.

Output:

```
=== Run information ===

Scheme: weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500
-S 10
Relation:     iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth class
```

```
Test mode:evaluate on training data

=== Model and evaluation on training set ===

kMeans
======
Number of iterations: 7
Within cluster sum of squared errors: 62.1436882815797
Missing values globally replaced with mean/mode

Cluster centroids:
Cluster#
Attribute            Full Data          0          1
(150)        (100)         (50)
==================================================================
sepallength          5.8433         6.262        5.006
sepalwidth            3.054         2.872        3.418
petallength          3.7587         4.906        1.464
petalwidth           1.1987         1.676        0.244
class            Iris-setosa Iris-versicolor  Iris-setosa

Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      100 ( 67%)
1       50 ( 33%)
```

Week 5: Demonstrate PAM Based Clustering

Steps for run K-medoids Clustering algorithms in WEKA

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Choose WEKA folder in C drive.
6. Select and Click on data option button.
7. Choose iris data set and open file.
8. Click on cluster tab and Choose K-medoids(Farthest first) and select use training set test option.
9. Click on start button.

Output:

```
=== Run information ===

Scheme:      weka.clusterers.FarthestFirst -N 2 -S 1
Relation:    iris
Instances:   150
Attributes:  5
             sepallength
             sepalwidth
             petallength
             petalwidth
             class
Test mode:   evaluate on training data


=== Clustering model (full training set) ===


FarthestFirst
==============

Cluster centroids:

Cluster 0
             7.7 3.0 6.1 2.3 Iris-virginica
Cluster 1
             4.3 3.0 1.1 0.1 Iris-setosa

Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      84 ( 56%)
1      66 ( 44%)
```

Week 6: Analyze German Credit Data Assessment using various Data mining Techniques

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.

Interest on these loans is the banks profit source. On the other hand, a bank cannot 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 involve 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.

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

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.

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.

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 Categorical or Nominal attributes). The goal is to classify the applicant into one of two categories. Good or Bad.

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

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 tasks)

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 cannot 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

                 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 9th and 20th 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         0.8590.852              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          %


ConfusionMatrix==
          a b   <-- classified as

          3     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.

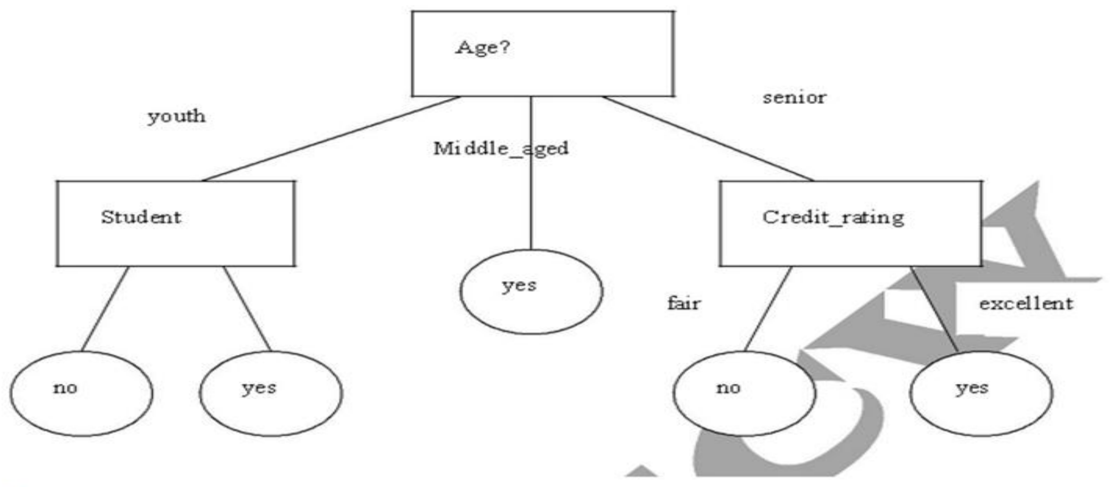
12) How can you convert a Decision Tree into “if-then-else rules”. Make up your own small

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