

---

# DATA WAREHOUSING & DATA MINING

## LAB MANUAL



**Department of Information Technology**  
**St.Martin's Engineering College**

**Dhulapally,Secunderabad-014**

---

# St.Martin's Engineering College

Dhulapally,Secunderabad-014

**INFORMATION TECHNOLOGY**

<b>Program Outcomes</b>	
P01	<b>Engineering knowledge:</b> Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
P02	<b>Problem analysis:</b> Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
P03	<b>Design/development of solutions:</b> 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.
P04	<b>Conduct investigations of complex problems:</b> 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.
P05	<b>Modern tool usage:</b> 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.
P06	<b>The engineer and society:</b> 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.
P07	<b>Environment and sustainability:</b> Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
P08	<b>Ethics:</b> Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
P09	<b>Individual and team work:</b> Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
P010	<b>Communication:</b> 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.
P011	<b>Project management and finance:</b> 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 multidisciplinary environments.
P012	<b>Life-long learning:</b> 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.
<b>Program Specific Outcomes</b>	
PSO1	<b>Professional Skills:</b> The ability to research, understand and implement computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient analysis and design of computer-based systems of varying complexity.
PSO2	<b>Problem-Solving Skills:</b> The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.
PSO3	<b>Successful Career and Entrepreneurship:</b> The ability to employ modern computer languages, environments, and platforms in creating innovative career paths, to be an entrepreneur, and a zest for higher studies.

## DATA WAREHOUSING AND DATA MINING LAB SYLLABUS

S. No.	List of Experiments	Page No.
<b>DATA MINING</b>		
1	List all the categorical (or nominal) attributes and the real-valued attributes separately.	1
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.	2
3	*What attributes do you think might be crucial in making the bank assessment?	5
4	One type of model that you can create is a Decision Tree -train a Decision Tree using the complete dataset as the training data. Report the model obtained after training.	6
5	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 you cannot get 100 % training accuracy?	9
6	*Find out the correctly classified instances, root mean squared error, kappa statistics, and mean absolute error for weather data set?	13
7	Is testing on the training set as you did above a good idea? Why or Why not?	17
8	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 your accuracy increase/decrease? Why?	18
9	Check to see if the data shows a bias against "foreign workers" (attribute 20), or "personal -status" (attribute 9). One way to do this (perhaps rather simple minded) is to remove these attributes from the dataset and see if the decision tree created in those cases is significantly different from the full dataset case which you have already done. To remove an attribute you can use the preprocess tab in Weka's GUI Explorer. Did removing these attributes have any significant effect? Discuss.	21
10	*Load the „weather.arff“ dataset in Weka and run the ID3 classification algorithm. What problem do you have and what is the solution?	23
11	Another question might be, do you really need to input so many attributes to get good results? Maybe only a few would do. For example, you could try just having attributes 2, 3, 5, 7, 10, 17 (and 21, the class attribute (naturally)). 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.)	24
12	Sometimes, the cost of rejecting an applicant who actually has a good credit (case 1) might be higher than accepting an applicant who has bad credit (case 2). Instead of counting the misclassifications equally in both cases, give a higher cost to the first case (say cost 5) and lower cost to the second case. You can do this by using a cost matrix in Weka. Train your Decision Tree again and report the Decision Tree and cross -validation results. Are they significantly different from results obtained in problem 6 (using equal cost)?	26
13	Do you think it is a good idea to prefer simple decision trees instead of having long complex decision trees? How does the complexity of a Decision Tree relate to the bias of the model?	28
14	*Run the J48 and 1Bk classifiers using-the cross-validation strategy with various fold levels. Compare the accuracy results.	29

15	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 (you can do this in Weka) and report the Decision Tree you obtain? Also, report your accuracy using the pruned model. Does your accuracy increase?	31
16	(Extra Credit): How can you convert a Decision Trees into "if -then -else rules". Make up your own small Decision Tree consisting of 2 - 3 levels and convert it 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 dataset? 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 and oneR.	33
17	*Run J48 and Naïve Bayes classifiers on the following datasets and determine the accuracy: 1.vehicle.arff 2.kr-vs-kp.arff 3.glass.arff 4.wave-form-5000.arff On which datasets does the Naïve Bayes perform better?	37

\*Content beyond the university prescribed syllabi

## DATA WAREHOUSING AND DATA MINING LABORATORY

### OBJECTIVE:

Data mining tools allow predicting future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. The data mining laboratory course is designed to exercise the data mining techniques such as classification, clustering, pattern mining etc with varied datasets and dynamic parameters. Weka data mining tool is used for the purpose of acquainting the students with the basic environment of the data mining tools.

### OUTCOMES:

Upon the completion of Data Mining practical course, the student will be able to:

1. **Learn** to build a data warehouse and query it using open source tools.
2. **Learn** to execute data mining tasks using a data mining toolkit (such as WEKA) and visualize the results.
3. **Demonstrate** the working of algorithms for data mining tasks such association rule mining, classification, clustering and regression.

**DATA WAREHOUSING  
AND  
DATA MINING LAB**

## **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 the one hand, a bank wants to make as many loans as possible. Interest on these loans is the bank's 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. The 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.

1. Knowledge Engineering. Find a loan officer who is willing to talk. Interview her and try to represent her knowledge in the form of production rules.
2. Books. Find some training manuals for loan officers or perhaps a suitable textbook on finance. Translate this knowledge from text form 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 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 dataset (original) Excel spreadsheet version of the German credit data (download from web).

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 !)

A few notes on the German dataset :

- DM stands for Deutsche Mark, the unit of currency, worth about 90 cents Canadian (but looks and acts like a quarter).
- Owns\_telephone. German phone rates are much higher than in Canada so fewer people own telephones.
- Foreign\_worker. There are millions of these in Germany (many from Turkey). It is very hard to get German citizenship if you were not born of German parents.
- There are 20 attributes used in judging a loan applicant. The goal is to classify the applicant into one of two categories, good or bad.

Subtasks : (Turn in your answers to the following tasks)

# EXPERIMENT-1

## 1.1 OBJECTIVE:

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

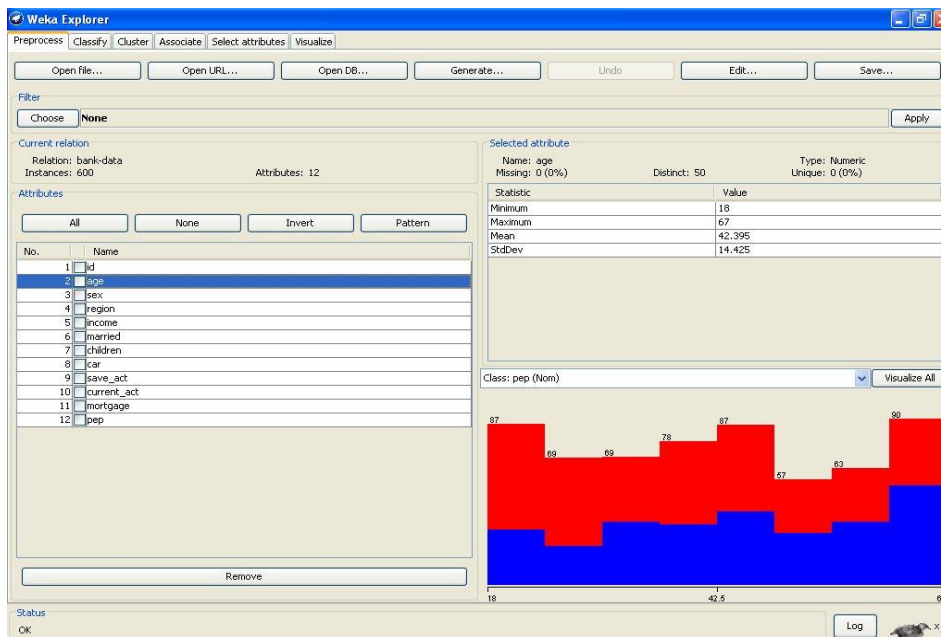
## 1.2 RESOURCES:

Weka mining tool

## 1.3 PROCEDURE:

- 1) Open the Weka GUI Chooser.
- 2) Select EXPLORER present in Applications.
- 3) Select Preprocess Tab.
- 4) Go to OPEN file and browse the file that is already stored in the system “bank.csv”.
- 5) Clicking on any attribute in the left panel will show the basic statistics on that selected attribute.

## 1.4 OUTPUT:





## EXPERIMENT-2

### 2.1 OBJECTIVE:

Which 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.

### 2.2 RESOURCES:

Weka mining tool

### 2.3 THEORY:

Association rule mining is defined as: Let  $I$  be a set of  $n$  binary attributes called items. Let  $D$  be a set of transactions called the database. Each transaction in  $D$  has a unique transaction ID and contains a subset of the items in  $I$ . A rule is defined as an implication of the form  $X \Rightarrow Y$  where  $X, Y \subseteq I$  and  $X \cap Y = \emptyset$ . The sets of items (for short itemsets)  $X$  and  $Y$  are called antecedent (left hand side or LHS) and consequent (right hand side or RHS) of the rule respectively.

To illustrate the concepts, we use a small example from the supermarket domain.

The set of items is  $I = \{\text{milk, bread, butter, beer}\}$  and a small database containing the items (1 codes presence and 0 absence of an item in a transaction) is shown in the table to the right. An example rule for the supermarket could be meaning that if milk and bread is bought, customers also buy butter.

Note: this example is extremely small. In practical applications, a rule needs a support of several hundred transactions before it can be considered statistically significant, and datasets often contain thousands or millions of transactions.

To select interesting rules from the set of all possible rules, constraints on various measures of significance and interest can be used. The best known constraints are minimum thresholds on support and confidence. The support  $\text{supp}(X)$  of an itemset  $X$  is defined as the proportion of transactions in the data set which contain the itemset. In the example database, the itemset  $\{\text{milk, bread}\}$  has a support of  $2 / 5 = 0.4$  since it occurs in 40% of all transactions (2 out of 5 transactions).

The confidence of a rule is defined. For example, the rule has a confidence of  $0.2 / 0.4 = 0.5$  in the database, which means that for 50% of the transactions containing milk and bread the rule is correct. Confidence can be interpreted as an estimate of the probability  $P(Y | X)$ , the probability of finding the RHS of the rule in transactions under the condition that these transactions also contain the LHS.

### ALGORITHM:

Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database. The problem is usually decomposed into two sub problems. One is to find those itemsets whose occurrences exceed a predefined threshold in the database; those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with the constraints of minimal confidence.

Suppose one of the large itemsets is  $L_k$ ,  $L_k = \{I_1, I_2, \dots, I_k\}$ , association rules with this itemsets are generated in the following way: the first rule is  $\{I_1, I_2, \dots, I_{k-1}\}$  and  $\{I_k\}$ , by checking the confidence this rule can be determined as interesting or not. Then other rule are generated by deleting the last items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine the interestingness of them. Those processes iterated until the antecedent becomes empty.

Since the second subproblem is quite straight forward, most of the researches focus on the first subproblem. The Apriori algorithm finds the frequent sets  $L$  In Database  $D$ .

- Find frequent set  $L_{k-1}$ .
- Join Step.
  - o  $C_k$  is generated by joining  $L_{k-1}$  with itself
- Prune Step.
  - o Any  $(k-1)$  itemset that is not frequent cannot be a subset of a frequent  $k$  itemset, hence should be removed.

Where · ( $C_k$ : Candidate itemset of size  $k$ )

· ( $L_k$ : frequent itemset of size  $k$ )

Apriori Pseudocode

*Apriori* ( $T, \epsilon$ )

$L \leftarrow \{ \text{Large Itemsets that appear in more than } \epsilon \text{ transactions} \}$

$K \leftarrow 2$

while  $L(K) \neq \Phi$

$C(K) \leftarrow \text{Generate}(L_{K-1})$

for transactions  $t \in T$

$C(t) \leftarrow \text{Subset}(C(K), t)$

for candidates  $c \in C(t)$

$\text{count}[c] \leftarrow \text{count}[c] + 1$

$L(K) \leftarrow \{ c \in C(K) \mid \text{count}[c] \geq \epsilon \}$

$K \leftarrow K + 1$

return  $\bigcup_k L(k)$

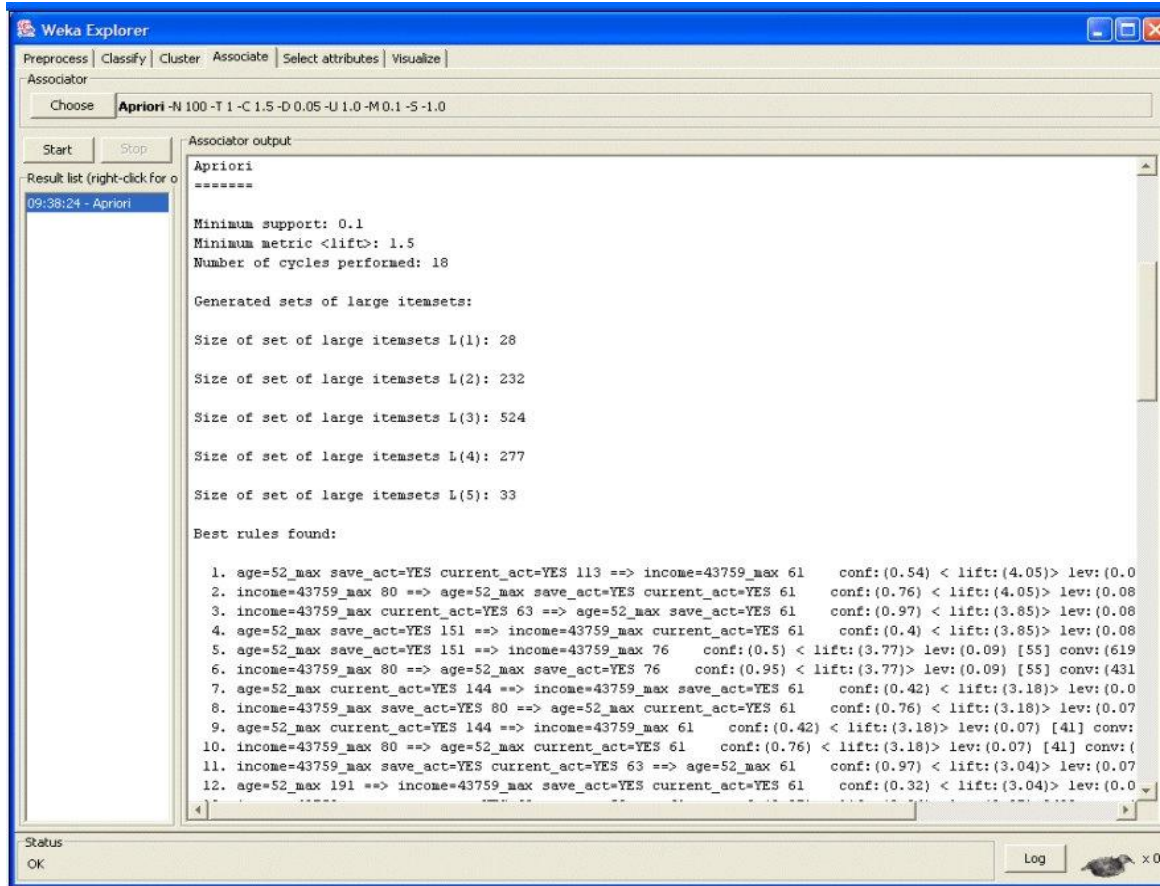
## 2.4 PROCEDURE:

- 1) Given the Bank database for mining.
- 2) Select EXPLORER in WEKA GUI Chooser.
- 3) Load "Bank.csv" in Weka by Open file in Preprocess tab.
- 4) Select only Nominal values.
- 5) Go to Associate Tab.
- 6) Select Apriori algorithm from "Choose" button present in Associator  
weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

7) Select Start button

8) Now we can see the sample rules.

## 2.5 OUTPUT:



The screenshot shows the Weka Explorer interface with the Apriori algorithm running. The 'Associator output' pane displays the following text:

```
Apriori
=====
Minimum support: 0.1
Minimum metric <lift>: 1.5
Number of cycles performed: 18

Generated sets of large itemsets:

Size of set of large itemsets L(1): 28
Size of set of large itemsets L(2): 232
Size of set of large itemsets L(3): 524
Size of set of large itemsets L(4): 277
Size of set of large itemsets L(5): 33

Best rules found:

1. age=52_max save_act=YES current_act=YES 113 ==> income=43759_max 61   conf:(0.54) < lift:(4.05)> lev:(0.0
2. income=43759_max 80 ==> age=52_max save_act=YES current_act=YES 61   conf:(0.76) < lift:(4.05)> lev:(0.08
3. income=43759_max current_act=YES 63 ==> age=52_max save_act=YES 61   conf:(0.97) < lift:(3.85)> lev:(0.08
4. age=52_max save_act=YES 151 ==> income=43759_max current_act=YES 61   conf:(0.4) < lift:(3.85)> lev:(0.08
5. age=52_max save_act=YES 151 ==> income=43759_max 76   conf:(0.5) < lift:(3.77)> lev:(0.09) [55] conv:(619
6. income=43759_max 80 ==> age=52_max save_act=YES 76   conf:(0.95) < lift:(3.77)> lev:(0.09) [55] conv:(431
7. age=52_max current_act=YES 144 ==> income=43759_max save_act=YES 61   conf:(0.42) < lift:(3.18)> lev:(0.0
8. income=43759_max save_act=YES 80 ==> age=52_max current_act=YES 61   conf:(0.76) < lift:(3.18)> lev:(0.07
9. age=52_max current_act=YES 144 ==> income=43759_max 61   conf:(0.42) < lift:(3.18)> lev:(0.07) [41] conv:(
10. income=43759_max 80 ==> age=52_max current_act=YES 61   conf:(0.76) < lift:(3.18)> lev:(0.07) [41] conv:(
11. income=43759_max save_act=YES current_act=YES 63 ==> age=52_max 61   conf:(0.97) < lift:(3.04)> lev:(0.07
12. age=52_max 191 ==> income=43759_max save_act=YES current_act=YES 61   conf:(0.32) < lift:(3.04)> lev:(0.0
```

## EXPERIMENT-3

### 3.1 OBJECTIVE:

\*What attributes do you think might be crucial in making the bank assessment?

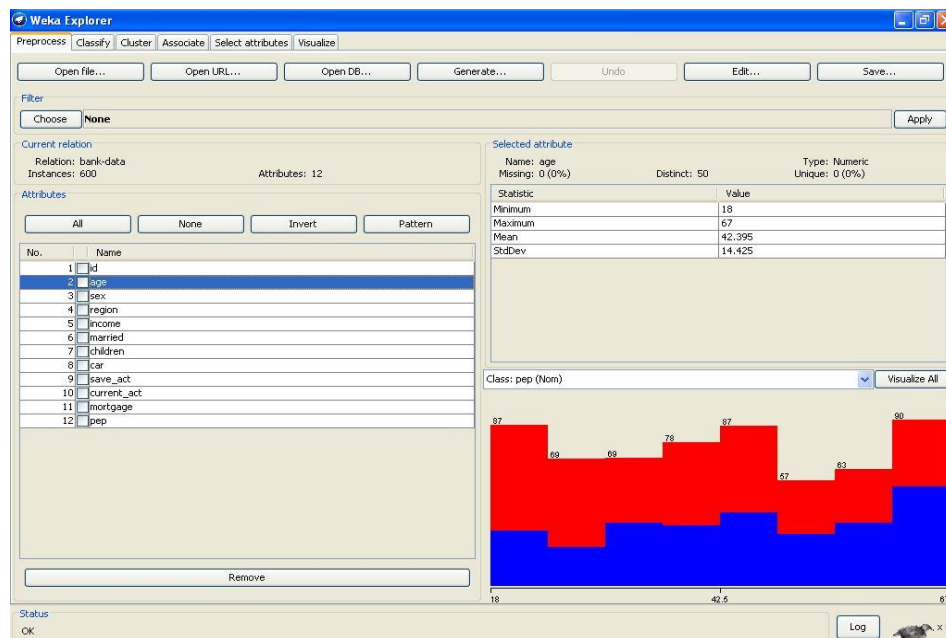
### 3.2 RESOURCES:

Weka mining tool

### 3.3 PROCEDURE:

- 1) Open the Weka GUI Chooser.
- 2) Select EXPLORER present in Applications.
- 3) Select Preprocess Tab.
- 4) Go to OPEN file and browse the file that is already stored in the system “bank.csv”.
- 5) Clicking on any attribute in the left panel will show the basic statistics on that selected attribute.

### 3.4 OUTPUT:



## EXPERIMENT-4

### 4.1 OBJECTIVE:

One type of model that you can create is a decision tree. Train a decision tree using the complete dataset as the training data. Report the model obtained after training.

### 4.2 RESOURCES:

Weka mining tool

### 4.3 THEORY:

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks. A classification task begins with a data set in which the class assignments are known. For example, a classification model that predicts credit risk could be developed based on observed data for many loan applicants over a period of time.

In addition to the historical credit rating, the data might track employment history, home ownership or rental, years of residence, number and type of investments, and so on. Credit rating would be the target, the other attributes would be the predictors, and the data for each customer would constitute a case.

Classifications are discrete and do not imply order. Continuous, floating point values would indicate a numerical, rather than a categorical, target. A predictive model with a numerical target uses a regression algorithm, not a classification algorithm. The simplest type of classification problem is binary classification. In binary classification, the target attribute has only two possible values: for example, high credit rating or low credit rating. Multiclass targets have more than two values: for example, low, medium, high, or unknown credit rating. In the model build (training) process, a classification algorithm finds relationships between the values of the predictors and the values of the target. Different classification algorithms use different techniques for finding relationships. These relationships are summarized in a model, which can then be applied to a different data set in which the class assignments are unknown.

Classification models are tested by comparing the predicted values to known target values in a set of test data. The historical data for a classification project is typically divided into two data sets: one for building the model; the other for testing the model. Scoring a classification model results in class assignments and probabilities for each case. For example, a model that classifies customers as low, medium, or high value would also predict the probability of each classification for each customer. Classification has many applications in customer segmentation, business modeling, marketing, credit analysis, and biomedical and drug response modeling.

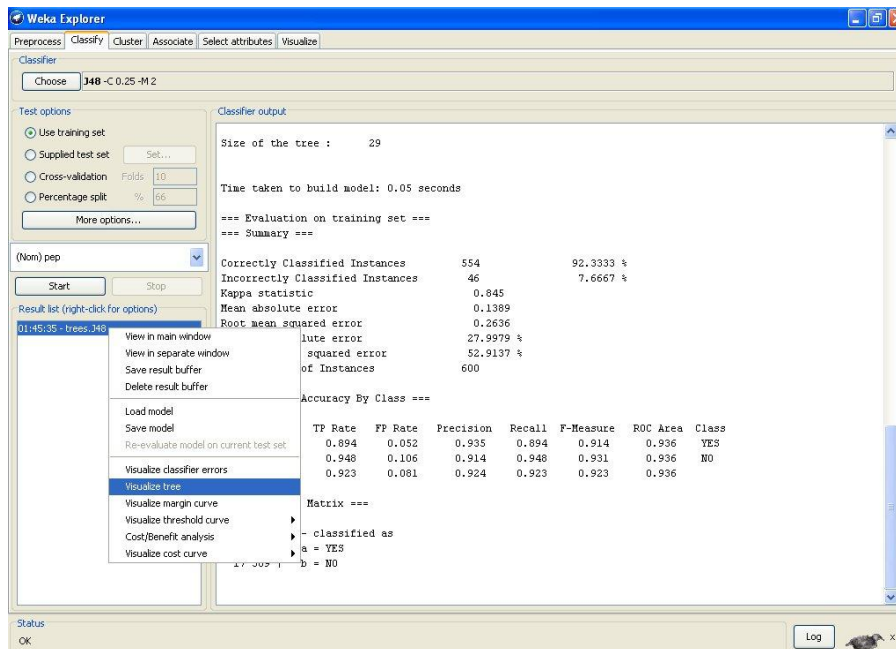
**Different Classification Algorithms:** Oracle Data Mining provides the following algorithms for classification:

- Decision Tree - Decision trees automatically generate rules, which are conditional statements that reveal the logic used to build the tree.
- Naive Bayes - Naive Bayes uses Bayes' Theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in the historical data.

#### 4.4 PROCEDURE:

- 1) Open Weka GUI Chooser.
- 2) Select EXPLORER present in Applications.
- 3) Select Preprocess Tab.
- 4) Go to OPEN file and browse the file that is already stored in the system “bank.csv”.
- 5) Go to Classify tab.
- 6) Here the c4.5 algorithm has been chosen which is entitled as j48 in Java and can be selected by clicking the button choose and select tree j48
- 7) Select Test options “Use training set”
- 8) if need select attribute.
- 9) Click Start.
- 10) Now we can see the output details in the Classifier output.
- 11) Right click on the result list and select “visualize tree”option .

#### 4.5 OUTPUT:



The screenshot displays the Weka Explorer interface. The 'Classifier' tab is active, showing the 'j48 -C 0.25 -M 2' algorithm. The 'Test options' section is set to 'Use training set'. The 'Classifier output' pane shows the following details:

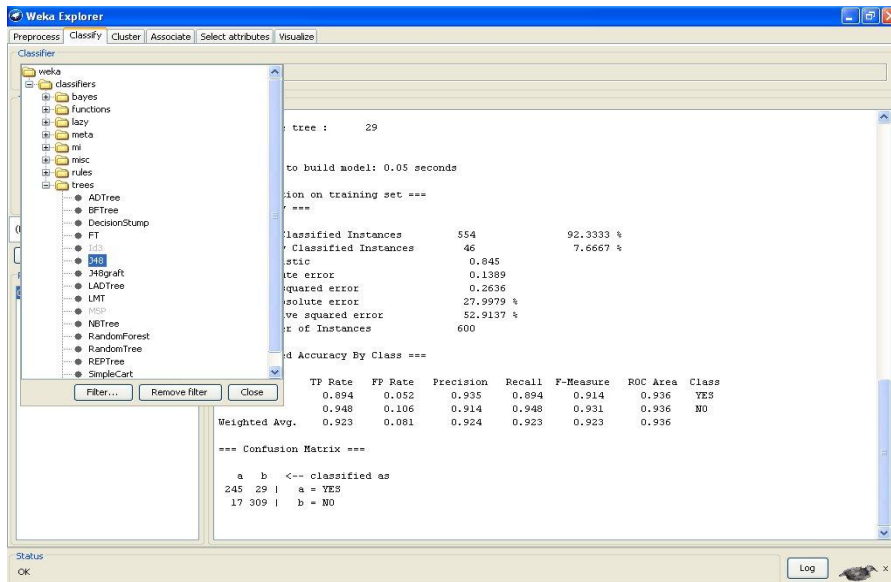
- Size of the tree : 29
- Time taken to build model: 0.05 seconds
- === Evaluation on training set ===
- === Summary ===
- Correctly Classified Instances: 554 (92.3333 %)
- Incorrectly Classified Instances: 46 (7.6667 %)
- Rappa statistic: 0.845
- Mean absolute error: 0.1389
- Root mean squared error: 0.2636
- lute error: 27.9979 %
- squared error: 52.9137 %
- of Instances: 600

The 'Accuracy By Class' table is as follows:

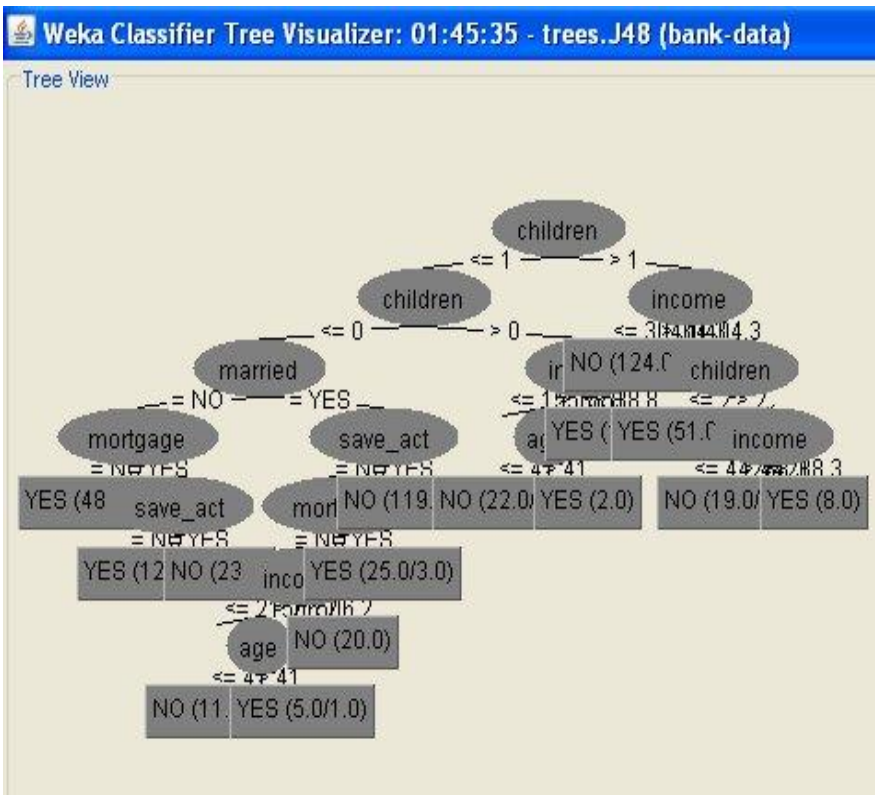
TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.894	0.052	0.935	0.894	0.914	0.936	YES
0.948	0.106	0.914	0.948	0.931	0.936	NO
0.923	0.081	0.924	0.923	0.923	0.936	

The 'Result list' shows a single entry: '01:45:35 - trees.j48'. A right-click context menu is open over this entry, with 'Visualize tree' selected. The 'Matrix' section shows the confusion matrix:

```
Matrix ===
- classified as
a = YES
b = NO
```



The decision tree constructed by using the implemented C4.5 algorithm



## EXPERIMENT-5

### 5.1 OBJECTIVE:

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 you cannot get 100 % training accuracy?

### 5.2 RESOURCES:

Weka mining tool

### 5.3 THEORY:

Naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even though these features depend on the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.

An advantage of the naive Bayes classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix. The naive Bayes probabilistic model :

The probability model for a classifier is a conditional model

$P(C|F_1, \dots, F_n)$  over a dependent class variable  $C$  with a small number of outcomes or *classes*, conditional on several feature variables  $F_1$  through  $F_n$ . The problem is that if the number of features  $n$  is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.

Using Bayes' theorem, we write

$$P(C|F_1, \dots, F_n) = \frac{p(C)p(F_1, \dots, F_n|C)}{p(F_1, \dots, F_n)}$$

In plain English the above equation can be written as

$$\text{Posterior} = \frac{\text{prior} * \text{likelihood}}{\text{evidence}}$$

In practice we are only interested in the numerator of that fraction, since the denominator does not depend on  $C$  and the values of the features  $F_i$  are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model  $p(C, F_1, \dots, F_n)$  which can be rewritten as follows, using repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C, F_1, \dots, F_n) &= p(C) p(F_1, \dots, F_n|C) = p(C) p(F_1|C) p(F_2, \dots, F_n|C, F_1, F_2) \\ &= p(C) p(F_1|C) p(F_2|C, F_1) p(F_3, \dots, F_n|C, F_1, F_2) \\ &= p(C) p(F_1|C) p(F_2|C, F_1) p(F_3, \dots, F_n|C, F_1, F_2, F_3, \dots, F_{n-1}) \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature  $F_i$  is conditionally independent of every other feature  $F_j$  for  $j \neq i$ .



This means that  $p(F_i|C, F_j) = p(F_i|C)$  and so the joint model can be expressed as  $p(C, F_1, \dots, F_n) = p(C) p(F_1|C) p(F_2|C) \dots = p(C) \pi \prod p(F_i|C)$

This means that under the above independence assumptions, the conditional distribution over the class variable  $C$  can be expressed like this:

$$p(C|F_1, \dots, F_n) = p(C) \pi \prod p(F_i|C) Z$$

where  $Z$  is a scaling factor dependent only on  $F_1, \dots, F_n$ , i.e., a constant if the values of the feature variables are known.

Models of this form are much more manageable, since they factor into a so called *class prior*  $p(C)$  and independent probability distributions  $p(F_i|C)$ . If there are  $k$  classes and if a model for each  $p(F_i|C=c)$  can be expressed in terms of  $r$  parameters, then the corresponding naive Bayes model has  $(k - 1) + n r k$  parameters. In practice, often  $k = 2$  (binary classification) and  $r = 1$  (Bernoulli variables as features) are common, and so the total number of parameters of the naive Bayes model is  $2n + 1$ , where  $n$  is the number of binary features used for prediction

$$P(h/D) = P(D/h) P(h) P(D)$$

- $P(h)$  : Prior probability of hypothesis  $h$
- $P(D)$  : Prior probability of training data  $D$
- $P(h/D)$  : Probability of  $h$  given  $D$
- $P(D/h)$  : Probability of  $D$  given  $h$

Naïve Bayes Classifier : Derivation

- $D$  : Set of tuples
- Each Tuple is an „ $n$ “ dimensional attribute vector
- $X : (x_1, x_2, x_3, \dots, x_n)$
- Let there be „ $m$ “ Classes :  $C_1, C_2, C_3 \dots C_m$
- NB classifier predicts  $X$  belongs to Class  $C_i$  iff
- $P(C_i/X) > P(C_j/X)$  for  $1 \leq j \leq m, j \neq i$
- Maximum Posteriori Hypothesis
- $P(C_i/X) = P(X/C_i) P(C_i) / P(X)$
- Maximize  $P(X/C_i) P(C_i)$  as  $P(X)$  is constant

Naïve Bayes Classifier : Derivation

- With many attributes, it is computationally expensive to evaluate  $P(X/C_i)$
- Naïve Assumption of “class conditional independence”
- $P(X/C_i) = \prod P(x_k / C_i)$

k = 1

$$P(X/C_i) = P(x_1/C_i) * P(x_2/C_i) * \dots * P(x_n/C_i)$$

#### 5.4 PROCEDURE:

- 1) Given the Bank database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system "bank.csv".
- 6) Go to Classify tab.
- 7) Choose Classifier "Tree"
- 8) Select "NBTree" i.e., Navie Baysian tree.
- 9) Select Test options "Use training set"
- 10) if need select attribute.
- 11) Now start weka.
- 12) Now we can see the output details in the Classifier output.

#### 5.5 OUTPUT:

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances      554            92.3333 %

Incorrectly Classified Instances      46            7.6667 %

Kappa statistic                            0.845

Mean absolute error                      0.1389

Root mean squared error                0.2636

Relative absolute error                27.9979 %

Root relative squared error            52.9137 %

Total Number of Instances            600

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.894	0.052	0.935	0.894	0.914	0.936	YES
0.948	0.106	0.914	0.948	0.931	0.936	NO
Weighted Avg.						
0.923	0.081	0.924	0.923	0.923	0.936	

=== Confusion Matrix ===

A	b
245	29
17	309

<-- classified as  
a = YES , b = NO

## EXPERIMENT-6

### 6.1 OBJECTIVE:

\*Find out the correctly classified instances, root mean squared error, kappa statistics, and mean absolute error for weather data set?

### 6.2 RESOURCES:

Weka mining tool

### 6.3 THEORY:

Naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even though these features depend on the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.

An advantage of the naive Bayes classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix. The naive Bayes probabilistic model :

The probability model for a classifier is a conditional model

$P(C|F_1, \dots, F_n)$  over a dependent class variable  $C$  with a small number of outcomes or *classes*, conditional on several feature variables  $F_1$  through  $F_n$ . The problem is that if the number of features  $n$  is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.

Using Bayes' theorem, we write

$$P(C|F_1, \dots, F_n) = \frac{P(C)P(F_1, \dots, F_n|C)}{P(F_1, \dots, F_n)}$$

In plain English the above equation can be written as

$$\text{Posterior} = \frac{(\text{prior} * \text{likelihood})}{\text{evidence}}$$

In practice we are only interested in the numerator of that fraction, since the denominator does not depend on  $C$  and the values of the features  $F_i$  are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model  $p(C, F_1, \dots, F_n)$  which can be rewritten as follows, using repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C, F_1, \dots, F_n) &= p(C) p(F_1, \dots, F_n|C) = p(C) p(F_1|C) p(F_2, \dots, F_n|C, F_1, F_2) \\ &= p(C) p(F_1|C) p(F_2|C, F_1) p(F_3, \dots, F_n|C, F_1, F_2) \\ &= p(C) p(F_1|C) p(F_2|C, F_1) p(F_3, \dots, F_n|C, F_1, F_2) \dots p(F_n|C, F_1, F_2, F_3, \dots, F_{n-1}) \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature  $F_i$  is conditionally independent of every other feature  $F_j$  for  $j \neq i$ .

This means that  $p(F_i|C, F_j) = p(F_i|C)$  and so the joint model can be expressed as  $p(C, F_1, \dots, F_n) = p(C) p(F_1|C) p(F_2|C) \dots = p(C) \pi \prod p(F_i|C)$

This means that under the above independence assumptions, the conditional distribution over the class variable  $C$  can be expressed like this:

$$p(C|F_1, \dots, F_n) = p(C) \pi \prod p(F_i|C) Z$$

where  $Z$  is a scaling factor dependent only on  $F_1, \dots, F_n$ , i.e., a constant if the values of the feature variables are known.

Models of this form are much more manageable, since they factor into a so called *class prior*  $p(C)$  and independent probability distributions  $p(F_i|C)$ . If there are  $k$  classes and if a model for each  $p(F_i|C=c)$  can be expressed in terms of  $r$  parameters, then the corresponding naive Bayes model has  $(k - 1) + n r k$  parameters. In practice, often  $k = 2$  (binary classification) and  $r = 1$  (Bernoulli variables as features) are common, and so the total number of parameters of the naive Bayes model is  $2n + 1$ , where  $n$  is the number of binary features used for prediction

$$P(h/D) = P(D/h) P(h) P(D)$$

- $P(h)$  : Prior probability of hypothesis  $h$
- $P(D)$  : Prior probability of training data  $D$
- $P(h/D)$  : Probability of  $h$  given  $D$
- $P(D/h)$  : Probability of  $D$  given  $h$

Naïve Bayes Classifier : Derivation

- $D$  : Set of tuples
- Each Tuple is an „ $n$ “ dimensional attribute vector
- $X : (x_1, x_2, x_3, \dots, x_n)$
- Let there be „ $m$ “ Classes :  $C_1, C_2, C_3 \dots C_m$
- NB classifier predicts  $X$  belongs to Class  $C_i$  iff
- $P(C_i/X) > P(C_j/X)$  for  $1 \leq j \leq m, j \neq i$
- Maximum Posteriori Hypothesis
- $P(C_i/X) = P(X/C_i) P(C_i) / P(X)$
- Maximize  $P(X/C_i) P(C_i)$  as  $P(X)$  is constant

Naïve Bayes Classifier : Derivation

- With many attributes, it is computationally expensive to evaluate  $P(X/C_i)$
- Naïve Assumption of “class conditional independence”
- $P(X/C_i) = \prod P(x_k / C_i)$

k = 1

$$P(X/C_i) = P(x_1/C_i) * P(x_2/C_i) * \dots * P(x_n/C_i)$$

#### 6.4 PROCEDURE:

- 1) Given the Bank database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system "bank.csv".
- 6) Go to Classify tab.
- 7) Choose Classifier "Tree"
- 8) Select "NBTree" i.e., Navie Baysiean tree.
- 9) Select Test options "Use training set"
- 10) if need select attribute.
- 11) Now start weka.
- 12) Now we can see the output details in the Classifier output.

#### 6.5 OUTPUT:

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	554	92.3333 %
Incorrectly Classified Instances	46	7.6667 %
Kappa statistic	0.845	
Mean absolute error	0.1389	
Root mean squared error	0.2636	
Relative absolute error	27.9979 %	
Root relative squared error	52.9137 %	
Total Number of Instances	600	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.894	0.052	0.935	0.894	0.914	0.936	YES
0.948	0.106	0.914	0.948	0.931	0.936	NO
Weighted Avg.						
0.923	0.081	0.924	0.923	0.923	0.936	

=== Confusion Matrix ===

A	b
245	29
17	309

<-- classified as  
a = YES , b = NO

## **EXPERIMENT-7**

### **7.1 OBJECTIVE:**

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

### **7.2 RESOURCES:**

Weka Mining tool

### **7.3 PROCEDURE:**

- 1) In Test options, select the Supplied test set radio button
- 2) Click Set
- 3) Choose the file which contains records that were not in the training set we used to create the model.
- 4) Click Start(WEKA will run this test data set through the model we already created. )
- 5) Compare the output results with that of the 4th experiment

### **7.4 OUTPUT:**

This can be experienced by the different problem solutions while doing practice.

The important numbers to focus on here are the numbers next to the "Correctly Classified Instances" (92.3 percent) and the "Incorrectly Classified Instances" (7.6 percent). Other important numbers are in the "ROC Area" column, in the first row (the 0.936); Finally, in the "Confusion Matrix," it shows the number of false positives and false negatives. The false positives are 29, and the false negatives are 17 in this matrix.

Based on our accuracy rate of 92.3 percent, we say that upon initial analysis, this is a good model.

One final step to validating our classification tree, which is to run our test set through the model and ensure that accuracy of the model

Comparing the "Correctly Classified Instances" from this test set with the "Correctly Classified Instances" from the training set, we see the accuracy of the model, which indicates that the model will not break down with unknown data, or when future data is applied to it.



## EXPERIMENT-8

### 8.1 OBJECTIVE:

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 your accuracy increase/decrease? Why?

### 8.2 RESOURCES:

Weka mining tool

### 8.3 THEORY:

Decision tree learning, used in data mining and machine learning, uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. In these tree structures, leaves represent classifications and branches represent conjunctions of features that lead to those classifications. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data but not decisions; rather the resulting classification tree can be an input for decision making. This page deals with decision trees in data mining.

Decision tree learning is a common method used in data mining. The goal is to create a model that predicts the value of a target variable based on several input variables. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. In data mining, trees can be described also as the combination of mathematical and computational techniques to aid the description, categorization and generalization of a given set of data.

Data comes in records of the form:

$$(x, y) = (x_1, x_2, x_3, \dots, x_k, y)$$

The dependent variable, Y, is the target variable that we are trying to understand, classify or generalize. The vector x is comprised of the input variables, x<sub>1</sub>, x<sub>2</sub>, x<sub>3</sub> etc., that are used for that task.

### 8.4 PROCEDURE:

- 1) Given the Bank database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system "bank.csv".

- 6) Go to Classify tab.
- 7) Choose Classifier “Tree”
- 8) Select J48
- 9) Select Test options “Cross-validation”.
- 10) Set “Folds” Ex:10
- 11) if need select attribute.
- 12) now Start weka.
- 13)now we can see the output details in the Classifier output.
- 14)Compare the output results with that of the 4<sup>th</sup> experiment
- 15) check whether the accuracy increased or decreased?

## 8.5 OUTPUT:

The screenshot shows the Weka Explorer interface. The Classifier tab is active, and the classifier is set to J48 -C 0.25 -M 2. The Test options are set to Cross-validation with 10 folds. The Classifier output window displays the following results:

```

Size of the tree :      29

Time taken to build model: 0.02 seconds

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

Correctly Classified Instances   539      89.8333 %
Incorrectly Classified Instances  61      10.1667 %
Kappa statistic                  0.7942
Mean absolute error              0.167
Root mean squared error          0.305
Relative absolute error          33.6511 %
Root relative squared error      61.2344 %
Total Number of Instances       600

=== Detailed Accuracy By Class ===
                TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
                -----  -----  -
                0.861    0.071    0.911     0.861    0.886     0.883    YES
                0.929    0.139    0.889     0.929    0.909     0.883    NO
Weighted Avg.   0.898    0.108    0.899     0.898    0.898     0.883

=== Confusion Matrix ===
  a  b  <-- classified as
236 38 | a = YES
 23 303 | b = NO
  
```

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	539	89.8333 %
Incorrectly Classified Instances	61	10.1667 %
Kappa statistic	0.7942	
Mean absolute error	0.167	
Root mean squared error	0.305	

Relative absolute error      33.6511 %  
Root relative squared error      61.2344 %  
Total Number of Instances      600

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.861	0.071	0.911	0.861	0.886	0.883	YES
0.929	0.139	0.889	0.929	0.909	0.883	NO
Weighted Avg.						
0.898	0.108	0.899	0.898	0.898	0.883	

=== Confusion Matrix ===

a b <-- classified as

236 38 | a = YES

23 303 | b = NO

## EXPERIMENT-9

### 9.1 OBJECTIVE

Check to see if the data shows a bias against "foreign workers" (attribute 20), or "personal -status" (attribute 9). One way to do this (perhaps rather simple minded) is to remove these attributes from the dataset and see if the decision tree created in those cases is significantly different from the full dataset case which you have already done. To remove an attribute you can use the preprocess tab in Weka's GUI Explorer. Did removing these attributes have any significant effect? Discuss.

### 9.2 RESOURCES:

Weka mining tool

### 9.3 PROCEDURE:

- 1) Given the Bank database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system "bank.csv".
- 6) In the "Filter" panel, click on the "Choose" button. This will show a popup window with list available filters.
- 7) Select "weka.filters.unsupervised.attribute.Remove"
- 8) Next, click on text box immediately to the right of the "Choose" button
- 9) In the resulting dialog box enter the index of the attribute to be filtered out (Make sure that the "invert Selection" option is set to false )
- 10) Then click "OK" . Now, in the filter box you will see "Remove -R 1"
- 11) Click the "Apply" button to apply this filter to the data. This will remove the "id" attribute and create a new working relation
- 12) To save the new working relation as an ARFF file, click on save button in the top panel.
- 13) Go to OPEN file and browse the file that is newly saved (attribute deleted file)
- 14) Go to Classify tab.
- 15) Choose Classifier "Tree"
- 16) Select j48 tree
- 17) Select Test options "Use training set"
- 18) If need select attribute.
- 19) Now start weka.
- 20) Now we can see the output details in the Classifier output.
- 21) Right click on the result list and select "visualize tree" option .
- 22) Compare the output results with that of the 4<sup>th</sup> experiment
- 23) Check whether the accuracy increased or decreased.
- 24) Check whether removing these attributes have any significant effect.

## 9.4 OUTPUT:

The screenshot shows the Weka Explorer interface with the 'Remove' dialog open for the attribute 'age'. The 'Current relation' is 'bank-data-weka.filters.unsupervised.attribute.Remove-R1' with 600 instances and 11 attributes. The 'Selected attribute' is 'age', which is numeric with 50 distinct values and 0 missing values. A summary table for the attribute is shown:

Statistic	Value
Minimum	18
Maximum	67
Mean	42.395
StdDev	14.425

Below the table is a histogram for the 'age' attribute, showing a distribution of values from 18 to 67. The histogram bars are colored in red and blue, with values labeled above them: 87, 69, 69, 78, 87, 67, 83, 90.

The screenshot shows the Weka Explorer interface with the 'Classifier' output window open. The classifier used is 'J48 -C 0.25 -M 2'. The output shows the size of the tree is 29 and the time taken to build the model is 0 seconds. The evaluation on the training set is summarized as follows:

```

=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances      554      92.3333 %
Incorrectly Classified Instances    46       7.6667 %
Kappa statistic                    0.845
Mean absolute error                 0.1389
Root mean squared error             0.2636
Relative absolute error             27.9979 %
Root relative squared error         52.9137 %
Total Number of Instances          600
  
```

Detailed Accuracy By Class:

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.894	0.052	0.935	0.894	0.914	0.936	YES
	0.940	0.106	0.914	0.940	0.931	0.936	NO
Weighted Avg.	0.923	0.081	0.924	0.923	0.923	0.936	

Confusion Matrix:

```

=== Confusion Matrix ===
 a b <-- classified as
245 29 | a = YES
17 309 | b = NO
  
```

The 'Result list' on the left shows the execution times for the classifier on the training set: 01:45:35 - trees.J48, 02:07:49 - trees.J48, 02:50:48 - trees.J48, and 02:51:44 - trees.J48.

## EXPERIMENT-10

### 10.1 OBJECTIVE:

\*Load the „weather.arff“ dataset in Weka and run the ID3 classification algorithm. What problem do you have and what is the solution?

### 10.2 RESOURCES:

Weka Mining tool

### 10.3 PROCEDURE:

- 1) In Test options, select the Supplied test set radio button
- 2) click Set
- 3) Choose the file which contains records that were not in the training set we used to create the model.
- 4) Click Start (WEKA will run this test data set through the model we already created. )
- 5) Compare the output results with that of the 4th experiment

### 10.4 OUTPUT:

This can be experienced by the different problem solutions while doing practice.

The important numbers to focus on here are the numbers next to the "Correctly Classified Instances" (92.3 percent) and the "Incorrectly Classified Instances" (7.6 percent). Other important numbers are in the "ROC Area" column, in the first row (the 0.936); finally, in the "Confusion Matrix," it shows the number of false positives and false negatives. The false positives are 29, and the false negatives are 17 in this matrix.

Based on our accuracy rate of 92.3 percent, we say that upon initial analysis, this is a good model.

One final step to validating our classification tree, which is to run our test set through the model and ensure that accuracy of the model

Comparing the "Correctly Classified Instances" from this test set with the "Correctly Classified Instances" from the training set, we see the accuracy of the model, which indicates that the model will not break down with unknown data, or when future data is applied to it.

## EXPERIMENT-11

### 11.1 OBJECTIVE:

Another question might be, do you really need to input so many attributes to get good results? Maybe only a few would do. For example, you could try just having attributes 2, 3, 5, 7, 10, 17 (and 21, the class attribute (naturally)). 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).

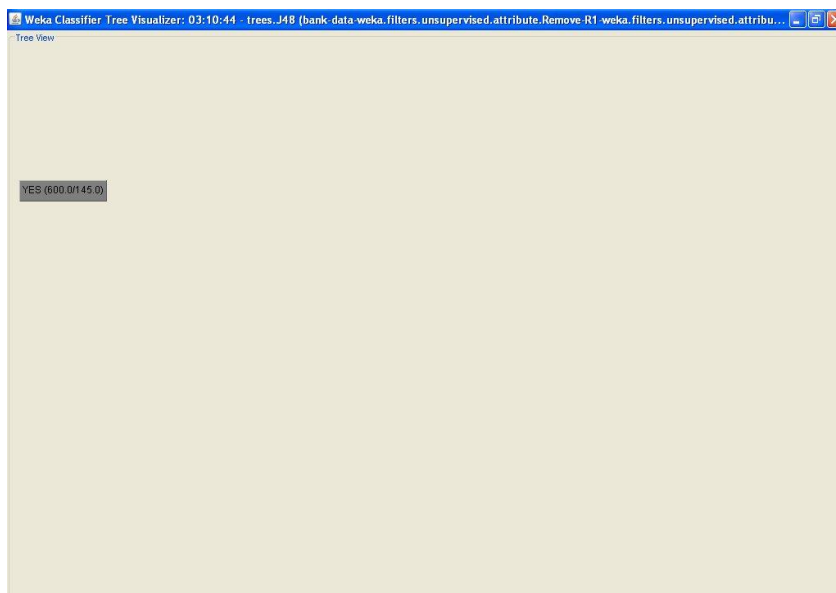
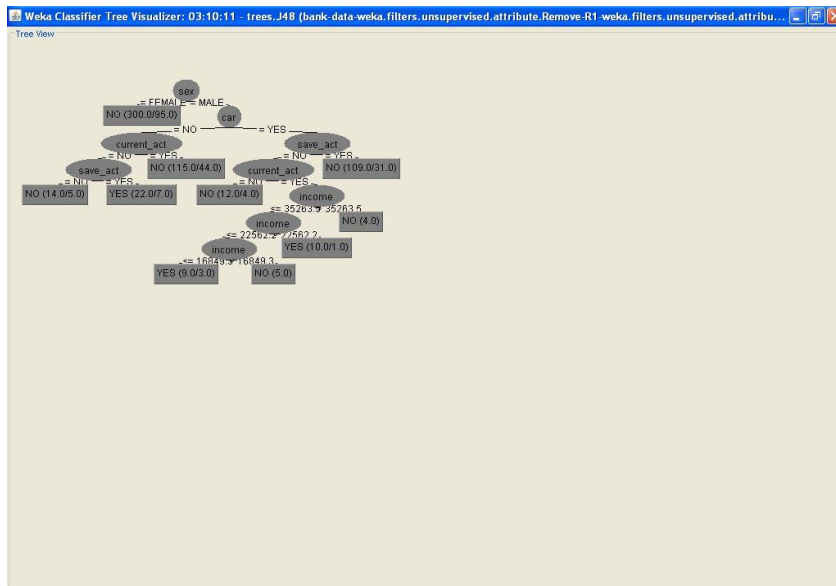
### 11.2 RESOURCES:

Weka mining tool.

### 11.3 PROCEDURE:

- 1) Given the Bank database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system “bank.csv”.
- 6) Select some of the attributes from attributes list which are to be removed. With this step only the attributes necessary for classification are left in the attributes panel.
- 7) The go to Classify tab.
- 8) Choose Classifier “Tree”
- 9) Select j48
- 10) Select Test options “Use training set”
- 11) If need select attribute.
- 12) Now start Weka.
- 13) Now we can see the output details in the Classifier output.
- 14) Right click on the result list and select “visualize tree” option.
- 15) Compare the output results with that of the 4<sup>th</sup> experiment.
- 16) Check whether the accuracy increased or decreased?
- 17) Check whether removing these attributes have any significant effect.

## 11.4 OUTPUT:





## EXPERIMENT-12

### 12.1 OBJECTIVE:

Sometimes, the cost of rejecting an applicant who actually has a good credit (case 1) might be higher than accepting an applicant who has bad credit (case 2). Instead of counting the misclassifications equally in both cases, give a higher cost to the first case (say cost 5) and lower cost to the second case. You can do this by using a cost matrix in Weka. Train your Decision Tree again and report the Decision Tree and cross-validation results. Are they significantly different from results obtained in problem 6 (using equal cost)?

### 12.2 RESOURCES:

Weka mining tool

### 12.3 PROCEDURE:

- 1) Given the Bank database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system "bank.csv".
- 6) Go to Classify tab.
- 7) Choose Classifier "Tree"
- 8) Select j48
- 9) Select Test options "Training set".
- 10) Click on "more options".
- 11) Select cost sensitive evaluation and click on set button
- 12) Set the matrix values and click on resize. Then close the window.
- 13) Click Ok
- 14) Click start.
- 15) We can see the output details in the Classifier output
- 16) Select Test options "Cross-validation".
- 17) Set "Folds" Ex: 10
- 18) if need select attribute.
- 19) Now start weka.
- 20) Now we can see the output details in the Classifier output.
- 21) Compare results of 15<sup>th</sup> and 20<sup>th</sup> steps.
- 22) Compare the results with that of experiment 6.

## 12.4 OUTPUT:

**Classifier evaluation options**

- Output model
- Output per-class stats
- Output entropy evaluation measures
- Output confusion matrix
- Store predictions for visualization
- Output predictions
- Output additional attributes:
- Cost-sensitive evaluation
- Random seed for XVal / % Split:
- Preserve order for % Split
- Output source code

OK

**Classifier output**

Time taken to build model: 0 seconds

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

Correctly Classified Instances	539	89.8333 %
Incorrectly Classified Instances	61	10.1667 %
Kappa statistic	0.7942	
Total Cost	61	
Average Cost	0.1017	
Mean absolute error	0.167	
Root mean squared error	0.305	
Relative absolute error	33.6511 %	
Root relative squared error	61.2344 %	
Total Number of Instances	600	

Detailed Accuracy By Class ==

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
Weighted Avg.	0.861	0.071	0.911	0.861	0.886	0.883	YES
	0.929	0.139	0.889	0.929	0.909	0.883	NO
Weighted Avg.	0.898	0.108	0.899	0.898	0.898	0.883	

Confusion Matrix ==

```

a b <-- classified as
38 | a = YES
303 | b = NO
  
```

**Classifier output**

Time taken to build model: 0 seconds

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

Correctly Classified Instances	539	89.8333 %
Incorrectly Classified Instances	61	10.1667 %
Kappa statistic	0.7942	
Total Cost	61	
Average Cost	0.1017	
Mean absolute error	0.167	
Root mean squared error	0.305	
Relative absolute error	33.6511 %	
Root relative squared error	61.2344 %	
Total Number of Instances	600	

Detailed Accuracy By Class ==

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
Weighted Avg.	0.861	0.071	0.911	0.861	0.886	0.883	YES
	0.929	0.139	0.889	0.929	0.909	0.883	NO
Weighted Avg.	0.898	0.108	0.899	0.898	0.898	0.883	

Confusion Matrix ==

```

a b <-- classified as
236 38 | a = YES
23 303 | b = NO
  
```

**Result list (right-click for options)**

- 03:23:24 - trees.J48
- 03:28:09 - trees.J48
- 03:29:49 - trees.J48
- 03:31:12 - trees.J48**

## **EXPERIMENT-13**

### **13.1 OBJECTIVE:**

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

### **13.2 RESOURCES:**

Weka mining tool

### **13.3 PROCEDURE:**

This will be based on the attribute set, and the requirement of relationship among attribute we want to study. This can be viewed based on the database and user requirement.

## EXPERIMENT-14

### 14.1 OBJECTIVE:

\*Run the J48 and 1Bk classifiers using-the cross-validation strategy with various fold levels. Compare the accuracy results. Hold out strategy with three percentage levels. Compare the accuracy results.

### 14.2 RESOURCES:

Weka mining tool

### 14.3 THEORY:

Reduced-error pruning

- Each node of the (over-fit) tree is examined for pruning
  - A node is pruned (removed) only if the resulting pruned tree performs no worse than the original over the validation set
  - Pruning a node consists of
    - Removing the sub-tree rooted at the pruned node
    - Making the pruned node a leaf node
    - Assigning the pruned node the most common classification of the training instances attached to that node
  - Pruning nodes iteratively
    - Always select a node whose removal most increases the DT accuracy over the validation set
    - Stop when further pruning decreases the DT accuracy over the validation set
- IF (Children=yes)  $\wedge$  (income=>30000)  
THEN (car=Yes)

### 14.4 PROCEDURE:

- 1) Given the Bank database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system “bank.csv”.
- 6) Select some of the attributes from attributes list
- 7) Go to Classify tab.
- 8) Choose Classifier “Tree”
- 9) Select “NBTree” i.e., Navie Bayesian tree.
- 10) Select Test options “Use training set”
- 11) Right click on the text box besides choose button, select show properties
- 12) Now change unprune mode “false” to “true”.
- 13) Change the reduced error pruning % as needed.
- 14) If need select attribute.
- 15) Now start Weka.
- 16) Now we can see the output details in the Classifier output.

17) Right click on the result list and select “visualize tree” option.

## 14.5 OUTPUT:

```
11:01:10 - trees.J48
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
-----
children = YES
| income <= 30099.3
| | car = YES: NO (50.0/15.0)
| | car = NO
| | | married = YES
| | | | income <= 13106.6: NO (9.0/2.0)
| | | | income > 13106.6
| | | | | mortgage = YES: YES (12.0/3.0)
| | | | | mortgage = NO
| | | | | | income <= 18923: YES (9.0/3.0)
| | | | | | income > 18923: NO (10.0/3.0)
| | | | | married = NO: NO (22.0/6.0)
| | | income > 30099.3: YES (59.0/7.0)
children = NO
| married = YES
| | mortgage = YES
| | | region = INNER_CITY
| | | | income <= 39547.8: YES (12.0/3.0)
| | | | income > 39547.8: NO (4.0)
| | | region = RURAL: NO (3.0/1.0)
| | | region = TOWN: NO (9.0/2.0)
| | | region = SUBURBAN: NO (4.0/1.0)
| | mortgage = NO: NO (57.0/9.0)
| married = NO
| | mortgage = YES
| | | age <= 39
| | | | age <= 28: NO (4.0)
| | | | age > 28: YES (5.0/1.0)
| | | age > 39: NO (11.0)
| | mortgage = NO: YES (20.0/1.0)

Number of Leaves : 17
Size of the tree : 31
```

## EXPERIMENT-15

### 15.1 OBJECTIVE:

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 (you can do this in Weka) and report the Decision Tree you obtain? Also, report your accuracy using the pruned model. Does your accuracy increase?

### 15.2 RESOURCES:

Weka mining tool

### 15.3 THEORY:

Reduced-error pruning

- ❖ Each node of the (over-fit) tree is examined for pruning
  - ❖ A node is pruned (removed) only if the resulting pruned tree performs no worse than the original over the validation set
    - Pruning a node consists of
      - Removing the sub-tree rooted at the pruned node
      - Making the pruned node a leaf node
      - Assigning the pruned node the most common classification of the training instances attached to that node
    - Pruning nodes iteratively
      - Always select a node whose removal most increases the DT accuracy over the validation set
      - Stop when further pruning decreases the DT accuracy over the validation set
- IF (Children=yes)  $\wedge$  (income=>30000)  
THEN (car=Yes)

### 15.4 PROCEDURE:

- 1) Given the Bank database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system "bank.csv".
- 6) Select some of the attributes from attributes list
- 7) Go to Classify tab.
- 8) Choose Classifier "Tree"
- 9) Select "NBTree" i.e., Navie Bayesian tree.
- 10) Select Test options "Use training set"
- 11) Right click on the text box besides choose button, select show properties
- 12) Now change unprune mode "false" to "true".
- 13) Change the reduced error pruning % as needed.

- 14) If need select attribute.
- 15) Now start weka.
- 16) Now we can see the output details in the Classifier output.
- 17) Right click on the result list and select "visualize tree" option.

### 15.5 OUTPUT:

```
11:01:10 - trees_J48
Test mode: 10-fold cross-validation

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

J48 pruned tree
-----

children = YES
| income <= 30099.3
| | car = YES: NO (50.0/15.0)
| | car = NO
| | | married = YES
| | | | income <= 13106.6: NO (9.0/2.0)
| | | | income > 13106.6
| | | | | mortgage = YES: YES (12.0/3.0)
| | | | | mortgage = NO
| | | | | | income <= 18923: YES (9.0/3.0)
| | | | | | income > 18923: NO (10.0/3.0)
| | | | | married = NO: NO (22.0/5.0)
| | | | | income > 30099.3: YES (59.0/7.0)
| | | | | married = YES
| | | | | | mortgage = YES
| | | | | | | region = INNER_CITY
| | | | | | | | income <= 39547.8: YES (12.0/3.0)
| | | | | | | | income > 39547.8: NO (4.0)
| | | | | | | region = RURAL: NO (3.0/1.0)
| | | | | | | region = TOWN: NO (9.0/2.0)
| | | | | | | region = SUBURBAN: NO (4.0/1.0)
| | | | | | mortgage = NO: NO (57.0/9.0)
| | | | | married = NO
| | | | | | mortgage = YES
| | | | | | | age <= 39
| | | | | | | | age <= 28: NO (4.0)
| | | | | | | | age > 28: YES (5.0/1.0)
| | | | | | | age > 39: NO (11.0)
| | | | | | mortgage = NO: YES (20.0/1.0)

Number of Leaves : 17
Size of the tree : 31
```

## EXPERIMENT-16

### 16.1 OBJECTIVE:

(Extra Credit): How can you convert a Decision Trees into "if –then -else rules". Make up your own small Decision Tree consisting of 2 - 3 levels and convert it 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 dataset? 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 and OneR.

### 16.2 RESOURCES:

Weka mining tool.

### 16.3 PROCEDURE:

- 1) Given the Bank database for mining.
  - 2) Use the Weka GUI Chooser.
  - 3) Select EXPLORER present in Applications.
  - 4) Select Preprocess Tab.
  - 5) Go to OPEN file and browse the file that is already stored in the system "bank.csv".
  - 6) Select some of the attributes from attributes list
  - 7) Go to Classify tab.
  - 8) Choose Classifier "Trees Rules"
  - 9) Select "J48".
  - 10) Select Test options "Use training set"
  - 11) If need select attribute.
  - 12) Now start weka.
  - 13) Now we can see the output details in the Classifier output.
  - 14) Right click on the result list and select "visualize tree" option .
- (or)

➤ `java weka.classifiers.trees.J48 -t c:\temp\bank.arff`

### Procedure for "OneR":

- 1) Given the Bank database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system "bank.csv".
- 6) Select some of the attributes from attributes list
- 7) Go to Classify tab.
- 8) Choose Classifier "Rules"
- 9) Select "OneR".
- 10) Select Test options "Use training set"



- 11) if need select attribute.
- 12) Now start weka.
- 13) Now we can see the output details in the Classifier output.

**Procedure for “PART”:**

- 1) Given the Bank database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system “bank.csv”.
- 6) Select some of the attributes from attributes list
- 7) Go to Classify tab.
- 8) Choose Classifier “Rules”.
- 9) Select “PART”.
- 10) Select Test options “Use training set”
- 11) if need select attribute.
- 12) Now start weka.
- 13) Now we can see the output details in the Classifier output.

Attribute relevance with respect to the class – relevant attribute (*science*)

IF accounting=1 THEN class=A (Error=0, Coverage = 7 instance)

IF accounting=0 THEN class=B (Error=4/13, Coverage = 13 instances)

**12.4 OUTPUT:**

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Classifier: Choose **J48 -C 0.25-M 2**

Test options

- Use training set
- Supplied test set
- Cross-validation Folds **4**
- Percentage split % **25**

(Nom) remark

Result list (right-click for options)

- 08:50:19 - trees.J48

Classifier output

```

Number of Leaves : 1
Size of the tree : 1

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 391 65.1667 %
Incorrectly Classified Instances 209 34.8333 %
Kappa statistic 0
Mean absolute error 0.454
Root mean squared error 0.4764
Relative absolute error 99.9552 %
Root relative squared error 99.9999 %
Total Number of Instances 600

=== Detailed Accuracy By Class ===
          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
          0 0 0 0 0 0.497 good
          1 1 0.652 1 0.789 0.497 bad
Weighted Avg. 0.652 0.652 0.425 0.652 0.514 0.497

=== Confusion Matrix ===
  a  b  <-- classified as
0 209 | a = good
0 391 | b = bad

```

Status: OK  x 0

J48

One R

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Classifier: Choose **OneR -B 6**

Test options

- Use training set
- Supplied test set
- Cross-validation Folds **4**
- Percentage split % **25**

(Nom) remark

Result list (right-click for options)

- 08:50:19 - trees.J48
- 08:52:54 - rules.OneR

Classifier output

```

ID12681 -> good
ID12685 -> good
ID12694 -> bad
(600/600 instances correct)

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 209 34.8333 %
Incorrectly Classified Instances 391 65.1667 %
Kappa statistic 0
Mean absolute error 0.6517
Root mean squared error 0.8073
Relative absolute error 143.4746 %
Root relative squared error 169.4322 %
Total Number of Instances 600

=== Detailed Accuracy By Class ===
          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
          1 1 0.348 1 0.517 0.5 good
          0 0 0 0 0 0.5 bad
Weighted Avg. 0.348 0.348 0.121 0.348 0.18 0.5

=== Confusion Matrix ===
  a  b  <-- classified as
209 0 | a = good
391 0 | b = bad

```

Status: OK  x 0

# PART

The screenshot shows the Weka Explorer interface. The 'Classifier' dropdown is set to 'PART -M 2 -C 0.25 -Q 1'. The 'Test options' section has 'Cross-validation' selected with 'Folds' set to 4. The 'Classifier output' pane displays the following results:

```
: bad (600.0/209.0)
Number of Rules : 1
Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 391 65.1667 %
Incorrectly Classified Instances 209 34.8333 %
Kappa statistic 0
Mean absolute error 0.454
Root mean squared error 0.4764
Relative absolute error 99.9552 %
Root relative squared error 99.9999 %
Total Number of Instances 600

=== Detailed Accuracy By Class ===
      TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
      0 0 0 0 0 0.497 good
      1 1 0.652 1 0.789 0.497 bad
Weighted Avg. 0.652 0.652 0.425 0.652 0.514 0.497

=== Confusion Matrix ===
 a b <-- classified as
 0 209 | a = good
 0 391 | b = bad
```

The 'Result list' on the left shows three entries: '08:50:19 - trees.J48', '08:52:54 - rules.OneR', and '08:53:37 - rules.PART', with the last one selected.

## EXPERIMENT-17

### 17.1 OBJECTIVE:

\*Run J48 and Naïve Bayes classifiers on the following datasets and determine the accuracy:

- 1.vehicle.arff
- 2.kr-vs-kp.arff
- 3.glass.arff
- 4.wave-form-5000.arff

On which datasets does the Naïve Bayes perform better?

### 17.2 RESOURCES:

Weka mining tool.

### 17.3 PROCEDURE:

- 1) Given the Bank database for mining.
  - 2) Use the Weka GUI Chooser.
  - 3) Select EXPLORER present in Applications.
  - 4) Select Preprocess Tab.
  - 5) Go to OPEN file and browse the file that is already stored in the system “bank.csv”.
  - 6) Select some of the attributes from attributes list
  - 7) Go to Classify tab.
  - 8) Choose Classifier “Trees Rules”
  - 9) Select “J48”.
  - 10) Select Test options “Use training set”
  - 11) If need select attribute.
  - 12) Now start weka.
  - 13) Now we can see the output details in the Classifier output.
  - 14) Right click on the result list and select ” visualize tree “option .
- (or)

➤ `java weka.classifiers.trees.J48 -t c:\temp\bank.arff`

### Procedure for “OneR”:

- 1) Given the Bank database for mining.
- 2) Use the Weka GUI Chooser.
- 3) Select EXPLORER present in Applications.
- 4) Select Preprocess Tab.
- 5) Go to OPEN file and browse the file that is already stored in the system “bank.csv”.
- 6) Select some of the attributes from attributes list
- 7) Go to Classify tab.
- 8) Choose Classifier “Rules”
- 9) Select “OneR”.
- 10) Select Test options “Use training set”
- 11) if need select attribute.

12) Now start weka.

13) Now we can see the output details in the Classifier output.

### Procedure for “PART”:

1) Given the Bank database for mining.

2) Use the Weka GUI Chooser.

3) Select EXPLORER present in Applications.

4) Select Preprocess Tab.

5) Go to OPEN file and browse the file that is already stored in the system “bank.csv”.

6) Select some of the attributes from attributes list

7) Go to Classify tab.

8) Choose Classifier “Rules”.

9) Select “PART”.

10) Select Test options “Use training set”

11) if need select attribute.

12) Now start weka.

13) Now we can see the output details in the Classifier output.

Attribute relevance with respect to the class – relevant attribute (*science*)

IF accounting=1 THEN class=A (Error=0, Coverage = 7 instance)

IF accounting=0 THEN class=B (Error=4/13, Coverage = 13 instances)

### 17.4 OUTPUT:

J48

The screenshot shows the Weka Explorer interface with the Classifier tab selected. The classifier chosen is J48 - C 0.25 - M 2. The test options are set to 'Use training set' with 4 folds. The classifier output is displayed in the main window, showing the following details:

```
Number of Leaves : 1
Size of the tree : 1
Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 391      65.1667 %
Incorrectly Classified Instances 209      34.8333 %
Kappa statistic 0
Mean absolute error 0.454
Root mean squared error 0.4764
Relative absolute error 99.9552 %
Root relative squared error 99.9999 %
Total Number of Instances 600

=== Detailed Accuracy By Class ===
          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
          0         0         0           0         0         0.497    good
          1         1         0.652       1         0.789     0.497    bad
Weighted Avg.  0.652  0.652  0.425  0.652  0.514  0.497

=== Confusion Matrix ===
 a  b  <-- classified as
0 209 | a = good
0 391 | b = bad
```

# One R

**Weka Explorer**  
 Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Classifier: Choose **OneR -B 6**

Test options:  
 Use training set  
 Supplied test set (Set...)  
 Cross-validation Folds **4**  
 Percentage split % **25**  
 More options...

(Nom) remark: **Start** Stop

Result list (right-click for options):  
 08:50:19 - trees.J48  
 08:52:54 - rules.OneR

**Classifier output:**

```

ID12681 -> good
ID12685 -> good
ID12694 -> bad
(600/600 instances correct)

Time taken to build model: 0.02 seconds

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

Correctly Classified Instances      209      34.8333 %
Incorrectly Classified Instances    391      65.1667 %
Kappa statistic                    0
Mean absolute error                0.6517
Root mean squared error            0.8073
Relative absolute error            143.4746 %
Root relative squared error        169.4322 %
Total Number of Instances          600

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
          1      1      0.348      1      0.517      0.5      good
          0      0      0          0      0          0.5      bad
Weighted Avg.  0.348  0.348  0.121  0.348  0.18      0.5

=== Confusion Matrix ===

  a  b  <-- classified as
209  0  |  a = good
 391  0  |  b = bad
  
```

Status: OK

# PART

**Weka Explorer**  
 Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Classifier: Choose **PART -M 2 -C 0.25 -Q 1**

Test options:  
 Use training set  
 Supplied test set (Set...)  
 Cross-validation Folds **4**  
 Percentage split % **25**  
 More options...

(Nom) remark: **Start** Stop

Result list (right-click for options):  
 08:50:19 - trees.J48  
 08:52:54 - rules.OneR  
 08:53:37 - rules.PART

**Classifier output:**

```

: bad (600.0/209.0)

Number of Rules : 1

Time taken to build model: 0.02 seconds

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

Correctly Classified Instances      391      65.1667 %
Incorrectly Classified Instances    209      34.8333 %
Kappa statistic                    0
Mean absolute error                0.454
Root mean squared error            0.4764
Relative absolute error            99.9552 %
Root relative squared error        99.9999 %
Total Number of Instances          600

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
          0      0      0          0      0          0.497      good
          1      1      0.652      1      0.789      0.497      bad
Weighted Avg.  0.652  0.652  0.425  0.652  0.514  0.497

=== Confusion Matrix ===

  a  b  <-- classified as
 209  1  |  a = good
  391  1  |  b = bad
  
```

Status: OK