# Exercise 1

#### Problem Statement:

Create a dataset using ARFF and CSV formats and load into the Weka Explorer.

#### Description of the Problem:

We need to create an Employee Table with training data set which includes attributes like name, id, salary, experience, gender, phone number.

#### Implementation on Weka:

**Steps for creating and loading .arff file**

1. Open Start  All Programs  Accessories Notepad
2. Type the following training dataset for Employee Table. @relation employee

@attribute name {x,y,z,a,b} @attribute id numeric

@attribute salary {low,medium,high} @attribute exp numeric

@attribute gender {male,female} @attribute phone numeric

@data x,101,low,2,male,250311 y,102,high,3,female,251665 z,103,medium,1,male,240238 a,104,low,5,female,200200 b,105,high,2,male,240240

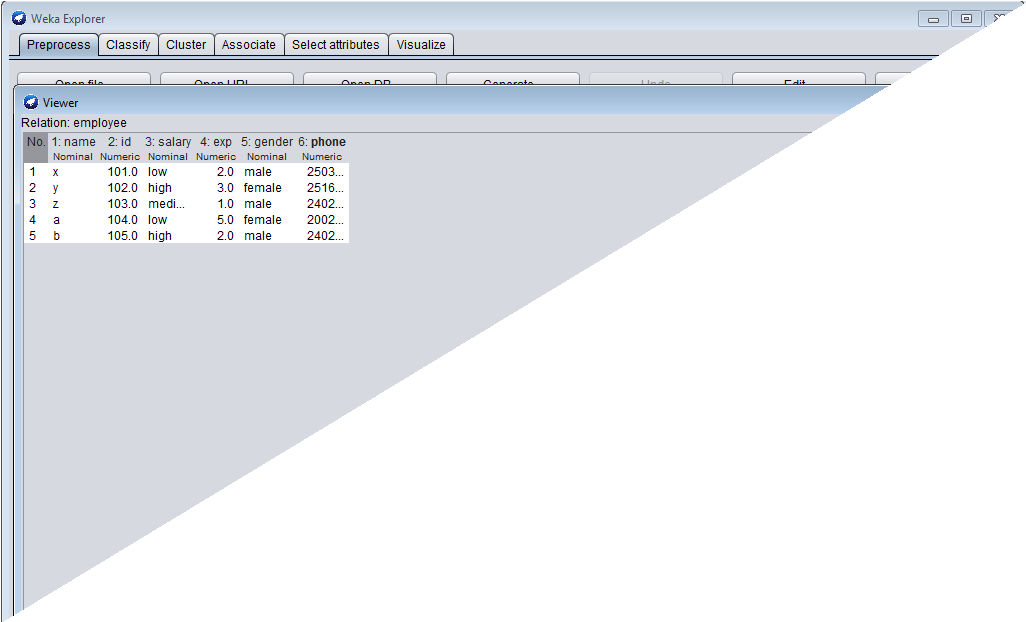
1. Save the file with .arff extension.
2. Open Start  Programs  weka-3.8.6.
3. Click on weka-3.8.6; Weka dialog box is displayed on the screen.
4. In that dialog box there are four tabs; Click on Explorer tab.
5. In the Weka Explorer, click on ‘open file’ and select the arff file.
6. Click on edit button to display employee dataset.

**Steps for creating and loading .csv file**

1. Start Microsoft Excel and add data to a new spreadsheet. For example, type “32,” “19” and “8” in cells “A1,” “A2” and “A3,” respectively.
2. Click the “File” tab on the ribbon and then choose “Save As.” Click the arrow next to “Save as Type” and choose “CSV (Comma Delimited)” from the drop-down list.
3. Change the file name to one you prefer. Select the location to save the file, then click the

“Save” button. Click "OK" to save.

1. Open Start  Programs  Weka.
2. Click on **Weka**, then Weka dialog box is displayed on the screen.
3. In that dialog box there are four modes, click on **Explorer**.
4. Explorer shows many options. In that click on **‘open file’** and select the csv file
5. Click on **edit button** which shows employee table on Weka.



#### Result:

Datasets have been created successfully in arff and csv file format.

# Exercise 2

#### Problem Statement:

Perform the following preprocessing filters on 'Weather' dataset.

(i) Add (ii) Remove (iii) Discretize (iv) Replace Missing values (v) Normalize.

#### Description of the Problem:

Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format. Steps involved are data cleaning, data integration, data transformation, data reduction.

Replacing missing values is one of the steps in data cleaning. Discretization and Normalization are steps in data transformation. Normalization is the process of transforming numerical data onto a new scale. Discretization is the process where the raw values of a numeric attribute (e.g., age) are replaced by interval labels (e.g., 0–10, 11–20, etc.) or conceptual labels (e.g., youth, adult, senior).

#### Implementation on Weka:

##### Creation of Weather Table:

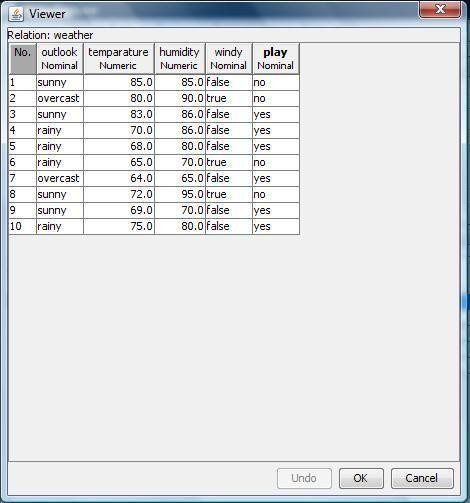
Procedure:

1. Open Start  Programs  Accessories  Notepad
2. Type the following training data set with the help of Notepad for Weather Table. @relation weather

@attribute outlook {sunny, rainy, overcast} @attribute temperature numeric

@attribute humidity numeric @attribute windy {true, false} @attribute play {yes, no} @data sunny,85.0,85.0,false,no overcast,80.0,90.0,true,no sunny,83.0,86.0,false,yes rainy,70.0,86.0,false,yes rainy,68.0,80.0,false,yes rainy,65.0,70.0,true,no overcast,64.0,65.0,false,yes sunny,72.0,95.0,true,no sunny,69.0,70.0,false,yes rainy,75.0,80.0,false,yes

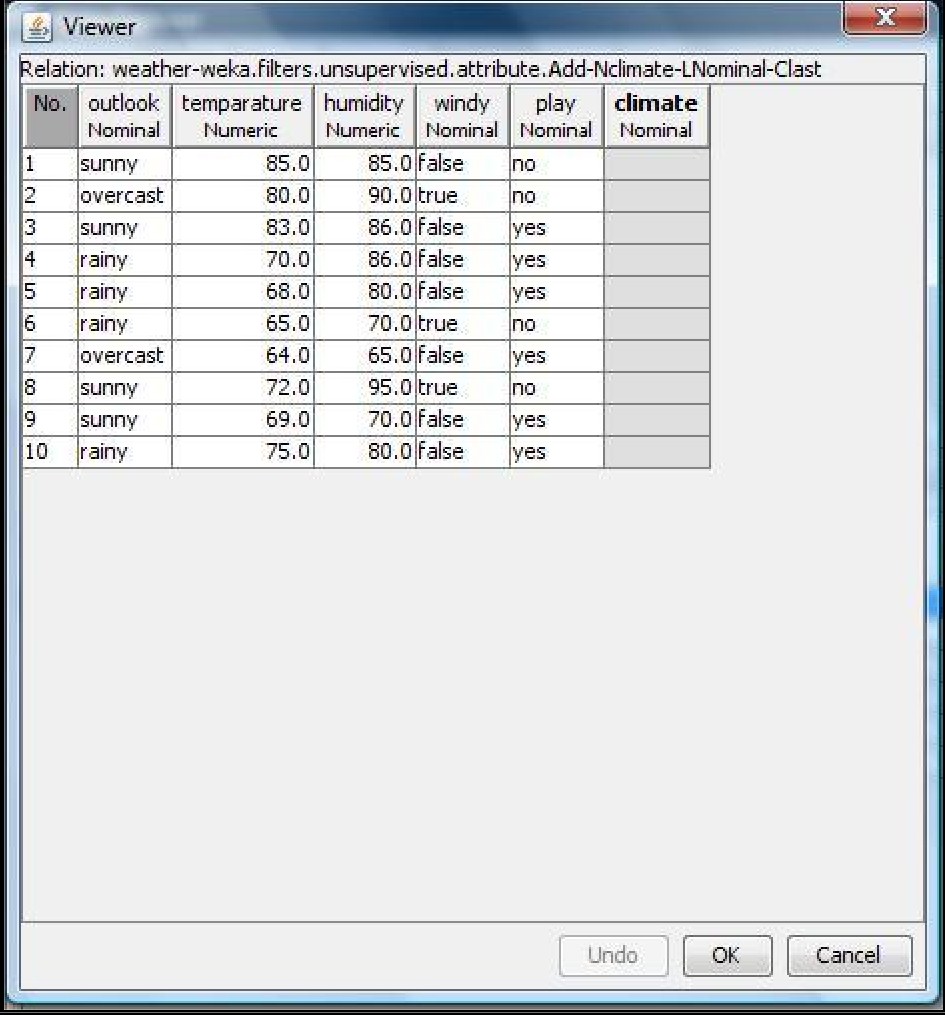
1. After that the file is saved with **.arff** file format.
2. Minimize the arff file and then open Start  Programs  Weka.
3. Click on **Weka**, then Weka dialog box is displayed on the screen.
4. In that dialog box there are four modes, click on **explorer**.
5. Explorer shows many options. In that click on **‘open file’** and select the arff file
6. Click on **edit** shows weather **button** which table on weka.



##### Add Pre-Processing Technique: Procedure:

1. Start  Programs  Weka  Weka
2. Click on **explorer.**
3. Click on **open file.**
4. Select **Weather.arff** file and click on open.
5. Click on **Choose button** and select the **Filters option**.
6. In Filters, we have **Supervised** and **Unsupervised data**.
7. Click on **Unsupervised data**.
8. Select the attribute **Add**.
9. A new window is opened.
10. In that we enter attribute index, type, data format, nominal label values for **Climate**.
11. Click on **OK**.
12. Press the **Apply button**, then a new attribute is added to the Weather Table.
13. **Save** the file.
14. Click on the **Edit button**, it shows a new Weather Table on Weka.

##### Weather Table after adding new attribute CLIMATE:



**Remove Pre-Processing Technique: Procedure:**

1. Start  All Programs  Weka
2. Click on **Explorer.**
3. Click on **open file.**
4. Select **Weather.arff** file and click on open.
5. Click on **Choose button** and select the **Filters option**.
6. In Filters, we have **Supervised** and **Unsupervised data**.
7. Click on **Unsupervised data**.
8. Select the attribute **Remove**.
9. Select the attributes **windy, play** to Remove.
10. Click **Remove button** and then **Save**.
11. Click on the **Edit button**, it shows a new Weather Table on Weka.

##### Weather Table after removing attributes WINDY, PLAY:

**Normalize Pre-Processing Technique: Procedure:**

1. Start  Programs  Weka
2. Click on **explorer.**
3. Click on **open file.**
4. Select **Weather.arff** file and click on open.
5. Click on **Choose button** and select the **Filters option**.
6. In Filters, we have **Supervised** and **Unsupervised data**.
7. Click on **Unsupervised data**.
8. Select the attribute **Normalize**.
9. Select the attributes **temparature, humidity** to Normalize.
10. Click on **Apply button** and then **Save**.
11. Click on the **Edit button**, it shows a new Weather Table with normalized values on Weka.

##### Weather Table after Normalizing TEMPARATURE, HUMIDITY:

**Discretize Pre-Processing Technique: Procedure:**

1. Start  Programs  Weka
2. Click on **explorer.**
3. Click on **open file.**
4. Select **Weather.arff** file and click on open.
5. Click on **Choose button** and select the **Filters option**.
6. In Filters, we have **Supervised** and **Unsupervised data**.
7. Click on **Unsupervised data**.
8. Select the attribute **Discretize**.
9. Select the attributes **temperature, humidity** to Normalize.
10. Click on **Apply button** and then **Save**.
11. Click on the **Edit button**, it shows a new Weather Table with discretized values on Weka.

#### Result:

The weather dataset has been created, edited, missing values have been replaced, normalized and discretized successfully using Weka.

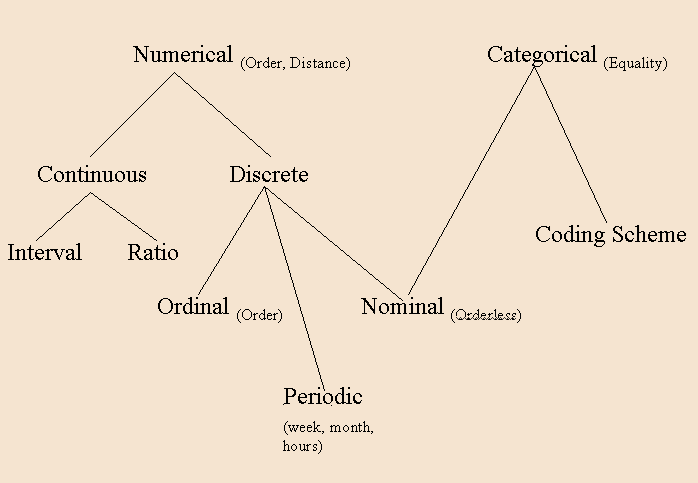
# Exercise 3

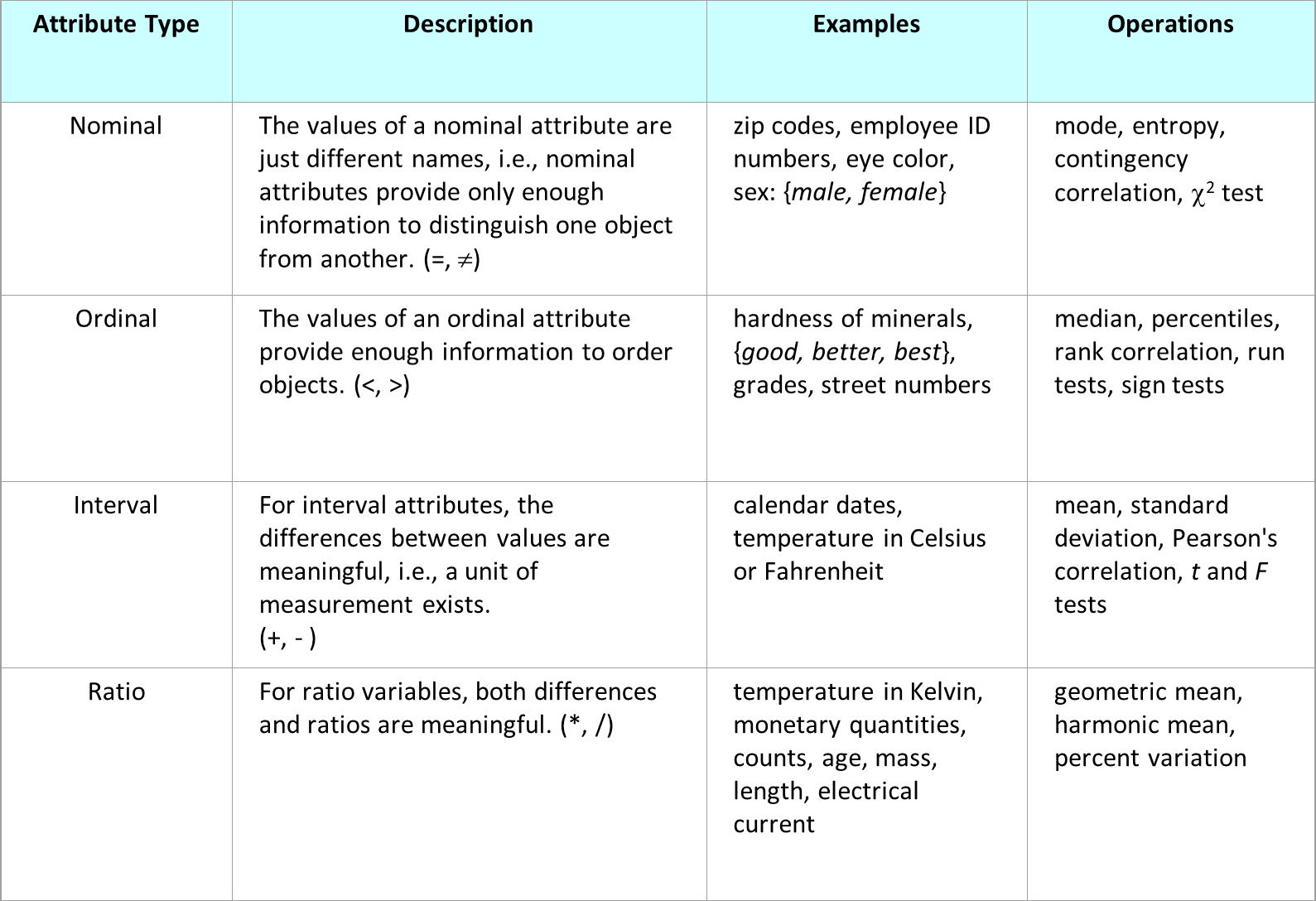
#### Problem Statement:

List all the categorical attributes and the real-valued attributes separately in 'German credit' data set.

#### Description of the Problem:

The types of attributes are as follows:

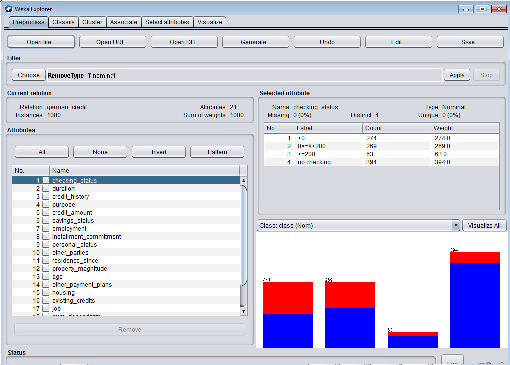


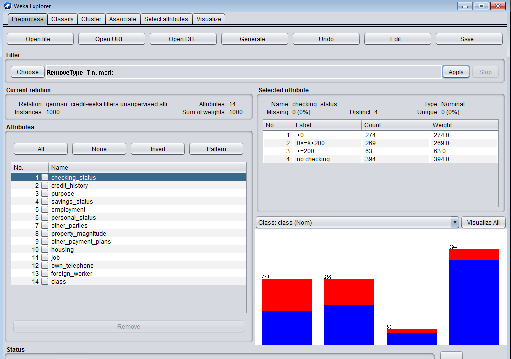


#### Implementation on Weka:

##### Procedure: Steps:

1. Start  Programs  Weka
2. Click on **explorer.**
3. Click on **open file.**
4. Select **credit-g.arff** file and click on open.
5. Click on **Choose button** and select the **Filters option**.
6. In Filters, we have **Supervised** and **Unsupervised data**.
7. Click on **Unsupervised data**.
8. Select the attribute Remove Type.
9. Select attribute Type delete nominal attributes then it gives all numerical attributes.
10. Select attribute Type delete numerical attributes then it gives all nominal attributes.





#### Result:

The categorical attributes and the real-valued attributes in 'German credit' data set are listed separately using Weka.

* 1. **Problem Statement:**

# EXERCISE 4

Generate strong Association rules by using Apriori algorithm on ‘contactlenses.arff' dataset

with Min\_Sup=60% and Min\_Conf=80%.

#### Description of the Problem:

Association Rule is a popular method for discovering interesting relations between variables in large databases. It is used to identify the most frequent item sets or frequent patterns that occur frequently in a database. To perform association rule mining we need two parameters like:

*Support*:- It is the total probability of the item sets that occurred in the transaction. *Confidence*:-it is the ratio of probability of two item sets A, B and the probability of the item set A.

Association rule mining is used in different applications like basket analysis, catalog design,

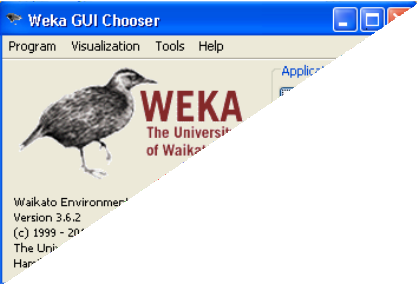
clustering, classification etc…

Apriori Algorithm: - To implement the association rule we use this algorithm. Steps to follow while implementing the association rule are:

* + 1. Find the frequent item sets where the items have the minimum support count.
    2. Use the frequent item sets to generate association rules.
    3. Generate the candidates by performing join operation.
    4. Perform pruning operation i.e. if any one of the candidates that have a subset is not frequent in the set then delete it.
    5. Also calculate support count for each candidate and delete those candidates whose minimum support count is not satisfied.

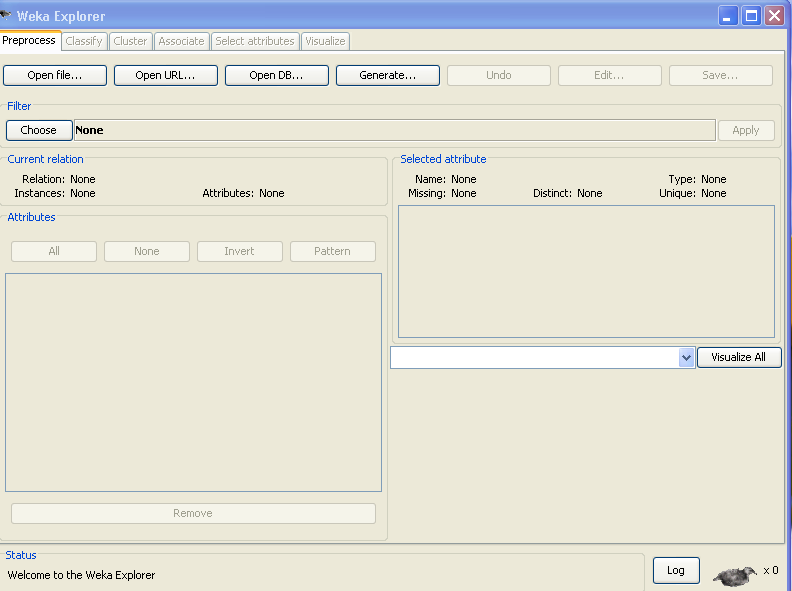
#### Implementation on Weka:

**Step 1:** Open Weka.

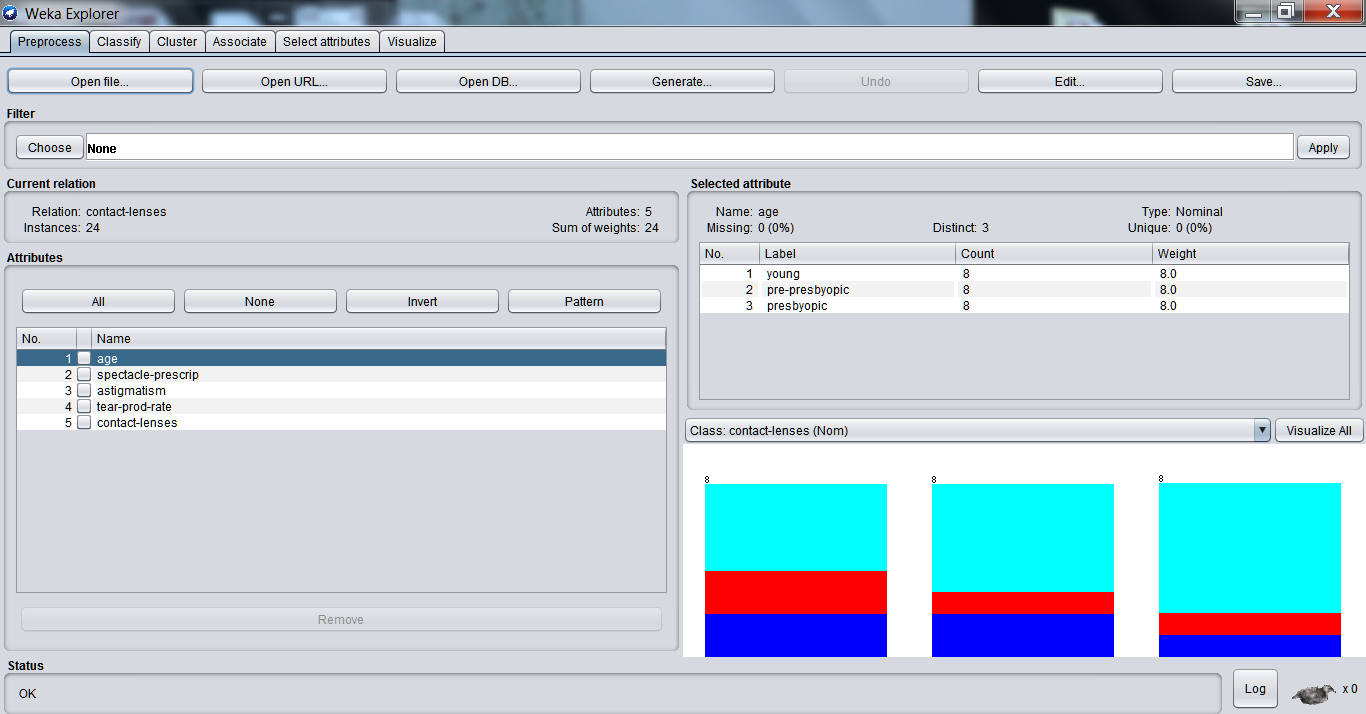


**Step 2:** click on “explorer” option then “weka explorer window” will be displayed as shown

below.

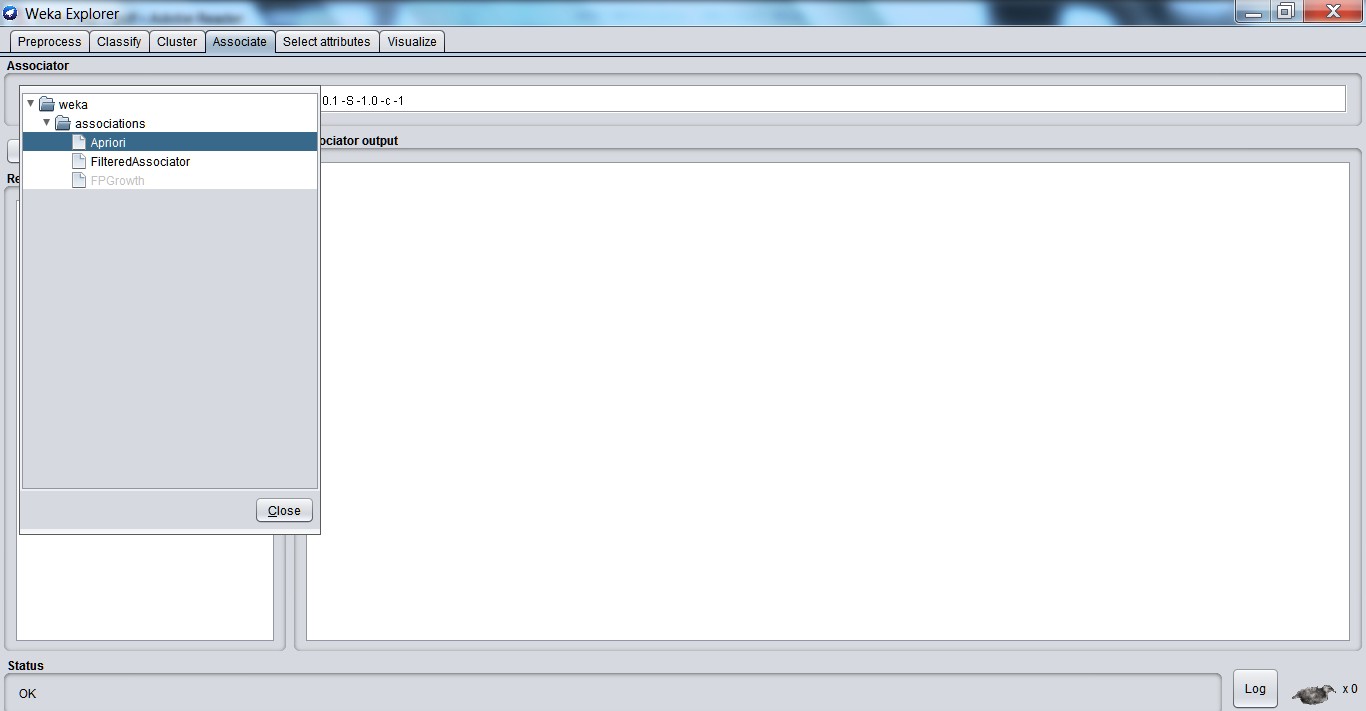


**Step 3:** click on the “open file…” option and select the “contactlenses” dataset.

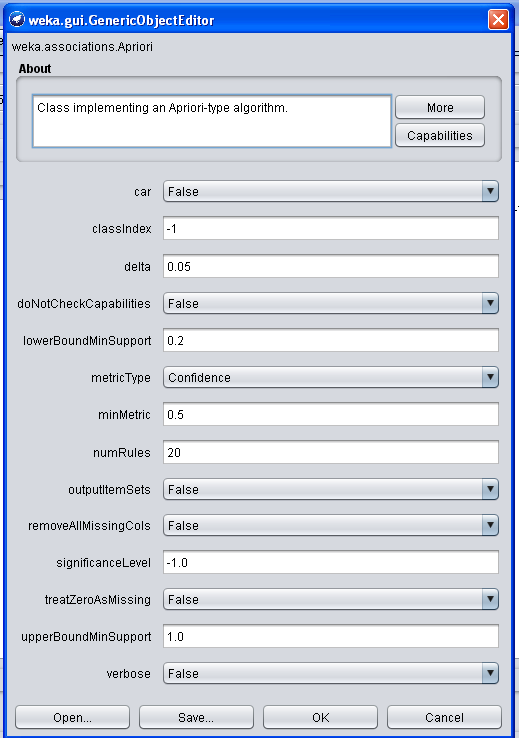


**Step 4:** Click on “Associate” tab. Click on “choose” option and select “apriori”. The window

is display as shown below.



**Step 5:** Click on the text field next to “choose” option. Set the parameters for Apriori and click on “OK” option.



**Step 6:** Click on “Start” option. The output obtained is as follows for the selected dataset “contactlenses” and the parameters as set above.

#### Result:

=== Run information ===

Scheme: weka.associations.Apriori -N 20 -T 0 -C 0.5 -D 0.05 -U 1.0 -M 0.2 -S -1.0 -c -1 Relation: contact-lenses

Instances: 24

Attributes: 5

age

spectacle-prescrip astigmatism

tear-prod-rate contact-lenses

=== Associator model (full training set) === Apriori

=======

Minimum support: 0.25 (6 instances) Minimum metric <confidence>: 0.5 Number of cycles performed: 15

Generated sets of large itemsets:

Size of set of large itemsets L(1): 10 Size of set of large itemsets L(2): 18 Size of set of large itemsets L(3): 4

Best rules found:

1. tear-prod-rate=reduced 12 ==> contact-lenses=none 12 <conf:(1)> lift:(1.6) lev:(0.19) [4] conv:(4.5)
2. spectacle-prescrip=myope tear-prod-rate=reduced 6 ==> contact-lenses=none 6

<conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)

1. spectacle-prescrip=hypermetrope tear-prod-rate=reduced 6 ==> contact-lenses=none 6

<conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)

1. astigmatism=no tear-prod-rate=reduced 6 ==> contact-lenses=none 6 <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
2. astigmatism=yes tear-prod-rate=reduced 6 ==> contact-lenses=none 6 <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
3. spectacle-prescrip=myope contact-lenses=none 7 ==> tear-prod-rate=reduced 6

<conf:(0.86)> lift:(1.71) lev:(0.1) [2] conv:(1.75)

1. astigmatism=no contact-lenses=none 7 ==> tear-prod-rate=reduced 6 <conf:(0.86)> lift:(1.71) lev:(0.1) [2] conv:(1.75)
2. contact-lenses=none 15 ==> tear-prod-rate=reduced 12 <conf:(0.8)> lift:(1.6) lev:(0.19) [4] conv:(1.88)
3. age=presbyopic 8 ==> contact-lenses=none 6 <conf:(0.75)> lift:(1.2) lev:(0.04) [1] conv:(1)
4. spectacle-prescrip=hypermetrope contact-lenses=none 8 ==> tear-prod-rate=reduced 6

<conf:(0.75)> lift:(1.5) lev:(0.08) [2] conv:(1.33)

1. astigmatism=yes contact-lenses=none 8 ==> tear-prod-rate=reduced 6 <conf:(0.75)> lift:(1.5) lev:(0.08) [2] conv:(1.33)
2. spectacle-prescrip=hypermetrope 12 ==> contact-lenses=none 8 <conf:(0.67)> lift:(1.07) lev:(0.02) [0] conv:(0.9)
3. astigmatism=yes 12 ==> contact-lenses=none 8 <conf:(0.67)> lift:(1.07) lev:(0.02) [0] conv:(0.9)
4. spectacle-prescrip=myope 12 ==> contact-lenses=none 7 <conf:(0.58)> lift:(0.93) lev:(-0.02) [0] conv:(0.75)
5. astigmatism=no 12 ==> contact-lenses=none 7 <conf:(0.58)> lift:(0.93) lev:(-0.02) [0] conv:(0.75)
6. contact-lenses=none 15 ==> spectacle-prescrip=hypermetrope 8 <conf:(0.53)> lift:(1.07) lev:(0.02) [0] conv:(0.94)
7. contact-lenses=none 15 ==> astigmatism=yes 8 <conf:(0.53)> lift:(1.07) lev:(0.02) [0] conv:(0.94)
8. astigmatism=no 12 ==> spectacle-prescrip=myope 6 <conf:(0.5)> lift:(1) lev:(0) [0] conv:(0.86)
9. spectacle-prescrip=myope 12 ==> astigmatism=no 6 <conf:(0.5)> lift:(1) lev:(0) [0] conv:(0.86)
10. astigmatism=yes 12 ==> spectacle-prescrip=myope 6 <conf:(0.5)> lift:(1) lev:(0) [0] conv:(0.86)

Association Rules have been successfully generated using Apriori algorithm using Weka.

* 1. **Problem Statement:**

# EXERCISE 5 (i)

1. Implement the Classification using Decision Tree algorithm on 'Weather' dataset. Draw the confusion matrix and report the model with accuracy.

#### Description of the Problem:

Decision Trees are an important type of algorithm for predictive modeling machine learning. Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Advantages of Decision Trees:

* Decision trees are simple to understand, interpret, visualize.
* Decision trees implicitly perform variable screening or feature selection.
* Decision trees require little data preparation.
* Decision trees can handle both numerical and categorical data.
* Decision trees can also handle single-class and multi-class classification problems.
* Nonlinear relationships between parameters do not affect tree performance.
* Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by Boolean logic.
* The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
* Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
* Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

Disadvantages of Decision Trees:

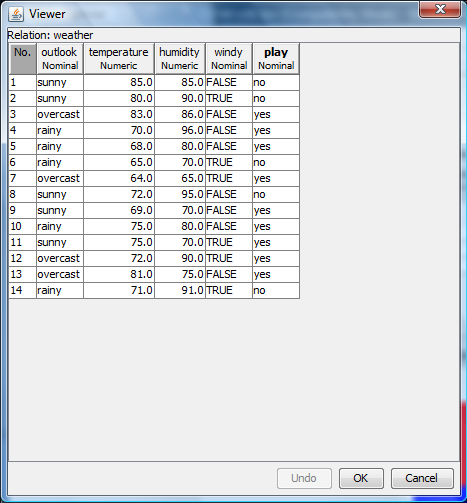
* Decision-tree learners can create over-complex trees that do not generalize the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
* Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This is called variance, which needs to be lowered by methods like bagging, boosting etc.
* Predictions of decision trees are neither smooth nor continuous, but are piecewise constant approximations. Therefore, they are not good at extrapolation.
* The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot

guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.

* There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
* Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the data set prior to fitting with the decision tree.

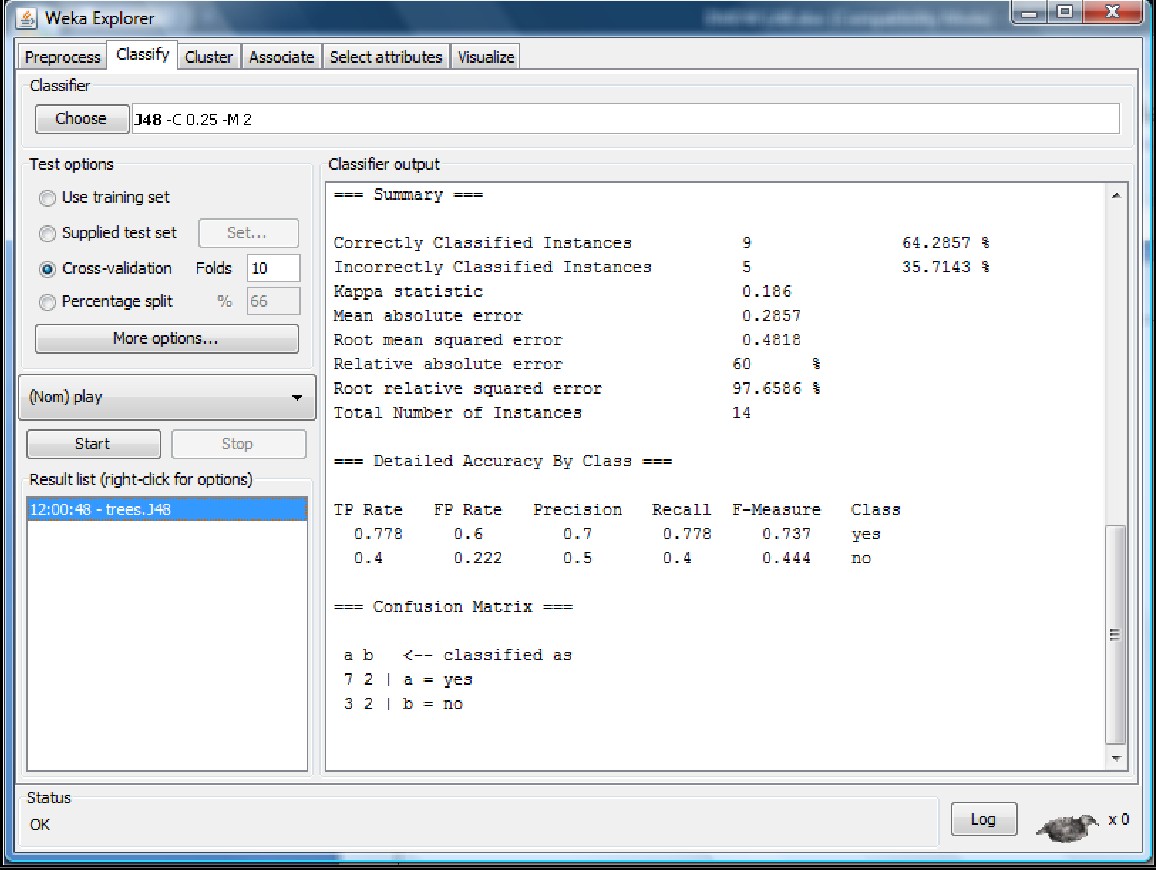
#### Implementation in Weka:

1. Start  Programs  Weka
2. Click on **explorer.**
3. Click on **open file.**
4. Select **Weather.arff** file and click on open.
5. Click on edit button which shows weather table on weka.

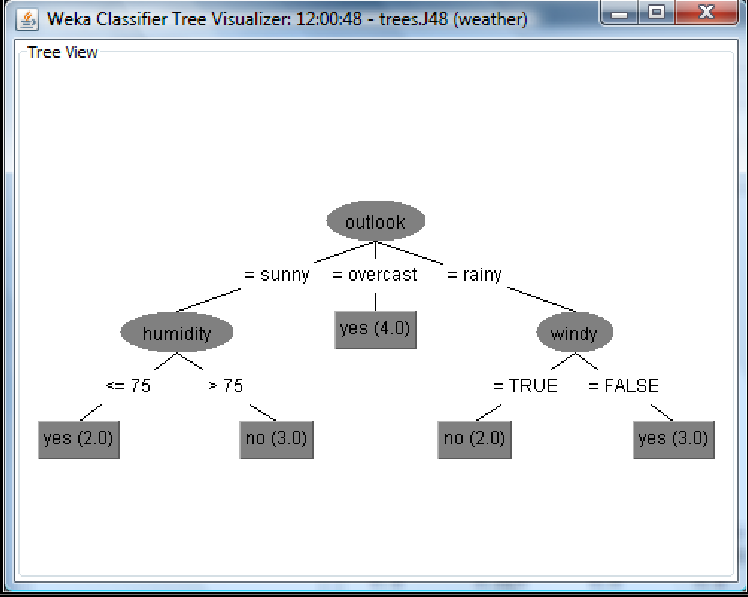
**Training Data Set Weather Table**

##### Procedure for Constructing Decision Tree:

1. Open Start  Programs  Weka
2. Open **explorer**.
3. Click on **open file** and select **weather.arff**
4. Select **Classifier option** on the top of the Menu bar.
5. Select **Choose button** and click on **Tree option**.
6. Click on **J48.**
7. Click on **Start button** and output will be displayed on the **right side** of the window.
8. Select the **result list** and **right click** on result list and select **Visualize Tree option**.
9. Then **Decision Tree** will be displayed on **new window**.



#### Result:



Decision Tree has been successfully constructed for weather dataset using Weka.

**EXERCISE 5 (ii)**

#### Problem Statement:

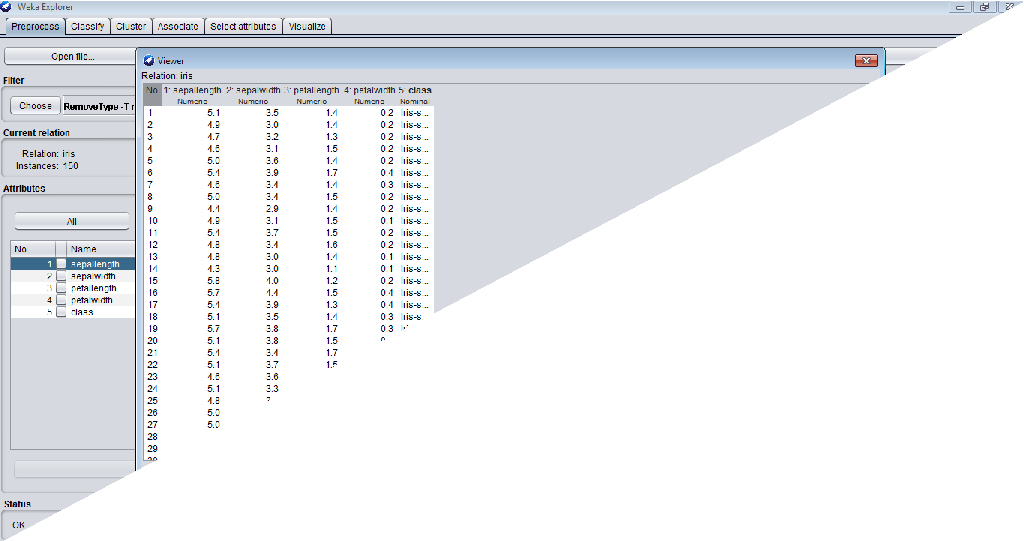
1. Implement Bayesian Classification and analyze the results on 'iris' Dataset.

#### Description of the Problem:

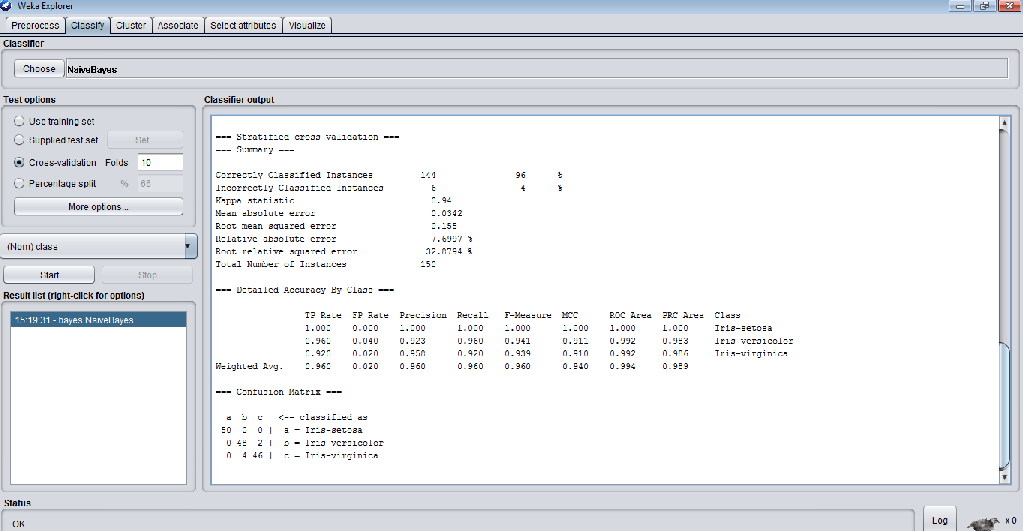
* Bayesian Learning provides a probabilistic approach to inference.
* It is based on the assumption that the quantities of interest are governed by probability distributions and that optimal decisions can be made by reasoning about these probabilities together with observed data.
* Bayesian learning algorithms calculate explicit probabilities for hypotheses.
* Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct. This provides a more flexible approach to learning than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example.
* Prior knowledge can be combined with observed data to determine the final probability of a hypothesis. In Bayesian learning, prior knowledge is provided by asserting
  + 1. a prior probability for each candidate hypothesis, and
    2. a probability distribution over observed data for each possible hypothesis.
* Bayesian methods can accommodate hypotheses that make probabilistic predictions (e.g., hypotheses such as "this pneumonia patient has a 93% chance of complete recovery").
* New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.
* They require initial knowledge of many probabilities. When these probabilities are not known in advance they are often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.
* They require significant computational cost.
* They can provide a standard of optimal decision making against which other practical methods can be measured.

#### Implementation in Weka:

1. Open Start  Programs  Weka
2. Open **explorer**.
3. Click on **open file** and select **iris.arff**
4. Select **Classify option** on the top of the Menu bar.
5. Select **Choose button** and click on **weka****classifier****bayes****NaiveBayes**.
6. Click on **Start button** and output will be displayed on the **right side** of the window.



#### Result:



Bayesian Classifier has been successfully constructed for iris dataset using Weka.

### EXERCISE 6

#### Problem Statement:

1. Implement Simple Linear Regression on an 'Employee' dataset.

#### Description of the Problem:

Numeric prediction is the task of predicting continuous values for given input. For example, it is required to predict the salary of employees with 10 years of experience, or tomorrow’s temperature. The most widely used approach for numeric prediction is regression. Regression analysis can be used to model the relationship between a set of predictor variables and a response variable (which is continuous-valued). The response variable is also referred to as the predicted attribute.

Regression analysis is a good choice when all of the predictor variables are continuous valued as well. Many problems can be solved by linear regression, and even more can be tackled by applying transformations to the variables so that a nonlinear problem can be converted to a linear one. Several software packages exist to solve regression problems. Examples include SAS, SPSS, and S-Plus.

Simple Linear regression analysis involves a response variable, *y*, and a single predictor variable, *x*. It is the simplest form of regression, and models *y* as a linear function of *x*. That is,

*y* = *b* + *wx*

where the variance of *y* is assumed to be constant, and *b* and *w* are regression coefficients specifying the Y-intercept and slope of the line, respectively.

#### Implementation in Weka:

##### Procedure:

1. Open Start  Programs  Weka
2. Open **explorer**.
3. Click on **open file** and select **Employee.arff**
4. Select **Classify option** on the top of the Menu bar.
5. Select **Choose button** and click on wekaclassifiersfunctionsLinear Regression.
6. Click on **Start button** and output will be displayed on the **right side** of the window.

#### Result:

=== Run information ===

Scheme: weka.classifiers.functions.LinearRegression -S 0 -R 1.0E-8 -num-decimal-places 3 Relation: Employee

Instances: 10

Attributes: 2 Experience Salary Test mode: 10-fold cross-validation

=== Classifier model (full training set) === Linear Regression Model

Salary = 3.537 \* Experience + 23.209 Time taken to build model: 0 seconds

=== Cross-validation ===

=== Summary === Correlation coefficient 0.9474 Mean absolute error 6.0423

##### Root mean squared error 7.2002

Relative absolute error 29.3632 % Root relative squared error 29.7254 % Total Number of Instances 10

Linear Regressor has been successfully constructed for employee dataset using Weka.

### EXERCISE 6

#### Problem Statement:

1. Demonstrate the simple k-Means clustering algorithm on 'iris' dataset.

#### Description of the Problem:

**Clustering is** the process of grouping a set of data objects into multiple groups or *clusters* so that objects within a cluster have high similarity, but are very dissimilar to objects in other clusters. Dissimilarities and similarities are assessed based on the attribute values describing the objects and often involve distance measures. Clustering as a data mining tool has its roots in many application areas such as biology, security, business intelligence, and Web search.

**Algorithm: *k***-**means.** The *k*-means algorithm for partitioning, where each cluster‟s center

is represented by the mean value of the objects in the cluster. **Input:** *k*: the number of clusters, *D*: a data set containing *n* objects. **Output:** A set of *k* clusters.

##### Method:

(1) Arbitrarily choose *k* objects from *D* as the initial cluster centers;

2) **Repeat**

1. (Re) assign each object to the cluster to which the object is the most similar,based on the mean value of the objects in the cluster;
2. update the cluster means, that is, calculate the mean value of the objects for each cluster;
3. **until** no change;

#### Implementation in Weka:

##### 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 iris data set and open file.
6. Click on cluster tab and Choose k-mean and select use training set test option.
7. Click on start button.

#### Result:

=== Run information ===

Scheme: weka.clusterers.EM -I 100 -N 2 -X 10 -max -1 -ll-cv 1.0E-6 -ll-iter 1.0E-6 -M 1.0E- 6 -K 10 -num-slots 1 -S 100

Relation: iris Instances: 150

Attributes: 5

sepallength sepalwidth petallength petalwidth

class Test mode: evaluate on training data

=== Clustering model (full training set) === EM== Number of clusters: 2

Number of iterations performed: 2 Cluster

Attribute 0 1

(0.33) (0.67)

====================================

sepallength

mean 5.006 6.262

std. dev. 0.3489 0.6595

sepalwidth

mean 3.418 2.872

std. dev. 0.3772 0.3311

petallength

mean 1.464 4.906

std. dev. 0.1718 0.8214

petalwidth

mean 0.244 1.676

std. dev. 0.1061 0.4226

class

Iris-setosa 51 1

Iris-versicolor 1 51

Iris-virginica 1 51

[total] 53 103

Time taken to build model (full training data) : 0.02 seconds

=== Model and evaluation on training set === Clustered Instances

0 50 ( 33%)

1 100 ( 67%)

Log likelihood: -3.06314

Supervised Learning has been successfully performed using simple k-Means clustering algorithm on 'iris' dataset using Weka.