M.Sc. – I.T.

Semester I

Data Mining

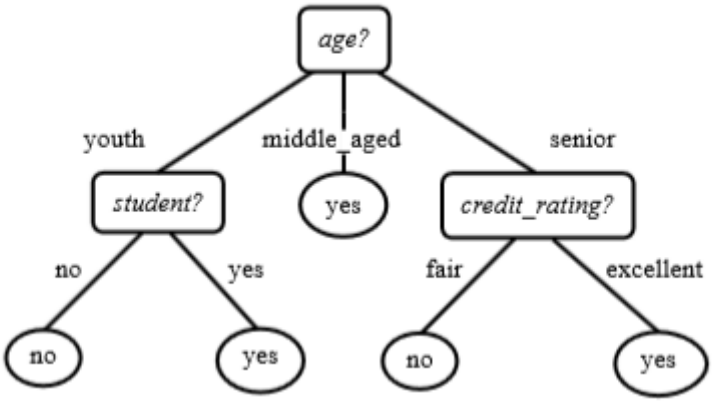
Sr. No.	Title	Sign
1	Prepare the Analysis services for Adventure Works Cycles or (any other database) Build the data mining model structure and built the decision tree with proper decision nodes. And infer atleast five different types of reports	
2	Prepare the Analysis services for Adventure Works Cycles or (any other database) .Build the data mining model structure. Implement the clustering Algorithm.	
3	Prepare the Analysis services for Adventure Works Cycles or (any other database) .Build the data mining model structure and Implement Naïve Bayes Algorithm.	
4	Prepare the Analysis services for Adventure Works Cycles or (any other database) .Build the basic Time series model structure and create the predictions	
5	Prepare the Analysis services for Adventure Works Cycles or (any other database) .Build the basic data mining model and show the implementation of Association algorithm	
6	Using R-Tool, show the analysis for social networking sites.	
7	Consider the suitable data for text mining and Implement the Text Mining technique using R-Tool	
8	Consider the suitable data for Apriori Algorithm and Implement the Apriori Algorithm using R-Tool.	

Practical 1:

Aim: Prepare the Analysis services for Adventure Works Cycles or (any other database) Build the data mining model structure and built the decision tree with proper decision nodes. And infer atleast five different types of reports

Theory:

Decision tree induction is the learning of decision trees from class- labelled training tuples. A decision tree is a flowchart- like tree structure, where each internal node (non leaf node) denotes a test on an attribute ,each branch represents an outcome of the test ,and each leaf node (or terminal node) holds a class label. The topmost node in a tree is the root node. Internal nodes are denoted by rectangles, and leaf nodes are denoted by ovals.



A typical decision tree is shown in Figure. It represents the concept buys computer, that is, it predicts whether a customer at AllElectronics is likely to purchase a computer.

The benefits of having a decision tree are as follows −

- It does not require any domain knowledge.
- It is easy to comprehend.
- The learning and classification steps of a decision tree are simple and fast.

Tree Pruning

Tree pruning is performed in order to remove anomalies in the training data due to noise or outliers. The pruned trees are smaller and less complex.

Tree Pruning Approaches

There are two approaches to prune a tree −

- **Pre-pruning** − The tree is pruned by halting its construction early.
- **Post-pruning** - This approach removes a sub-tree from a fully grown tree.

Cost Complexity

The cost complexity is measured by the following two parameters −

- Number of leaves in the tree, and
- Error rate of the tree.

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition *D*.

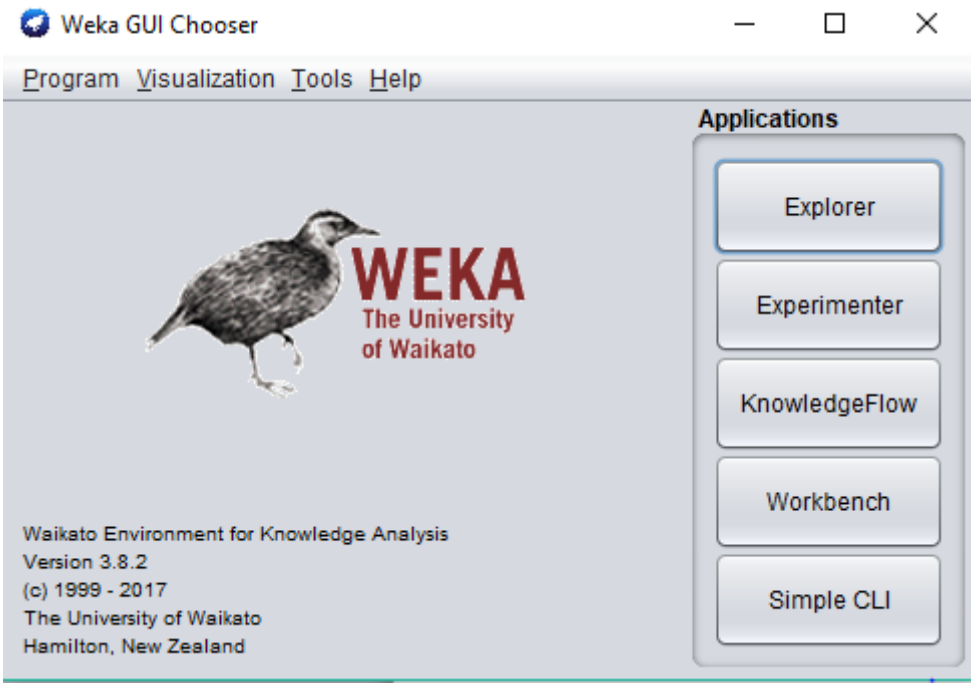
Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split point* or *splitting subset*.

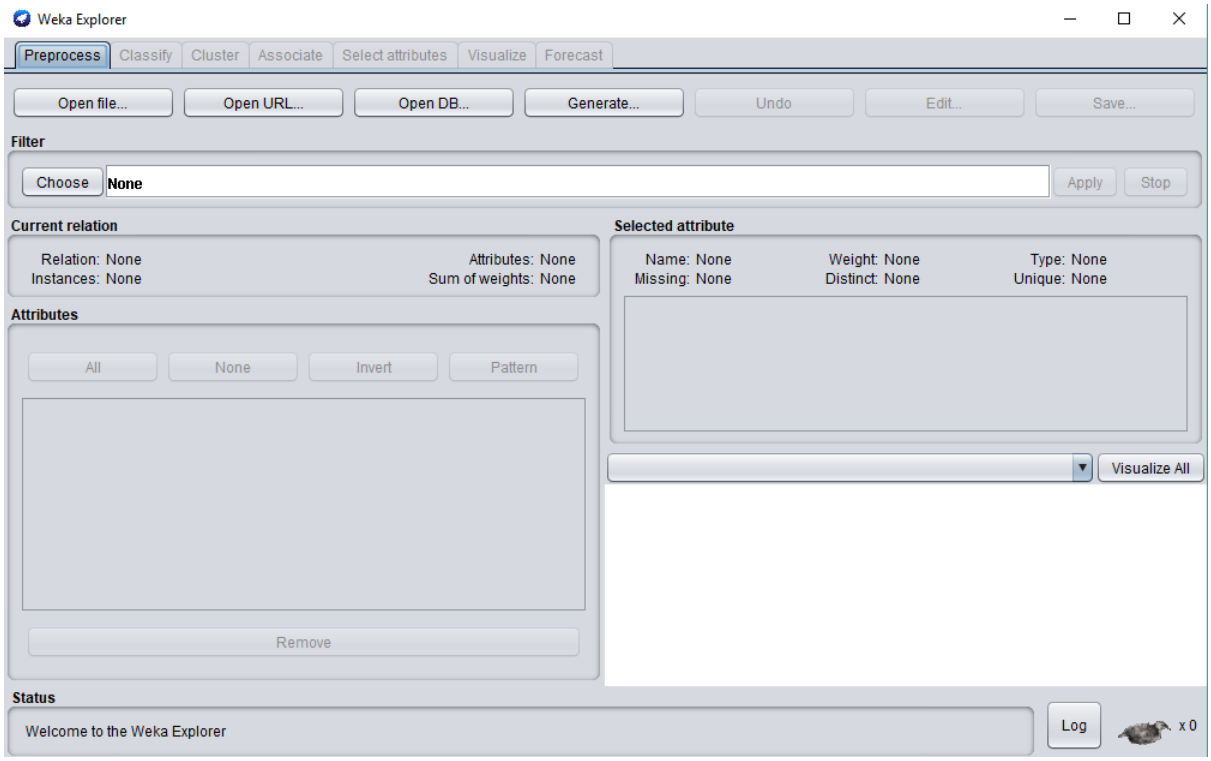
Output: A decision tree.

Practical

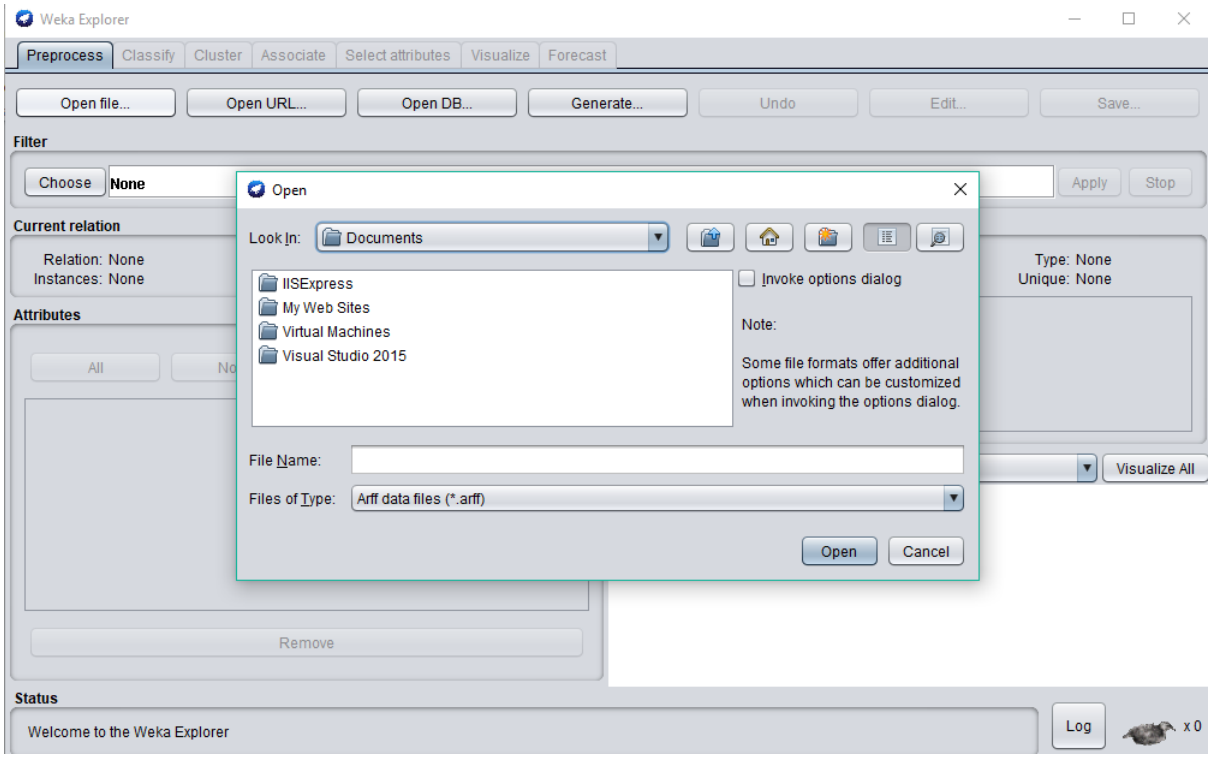
Step 1: Open Weka



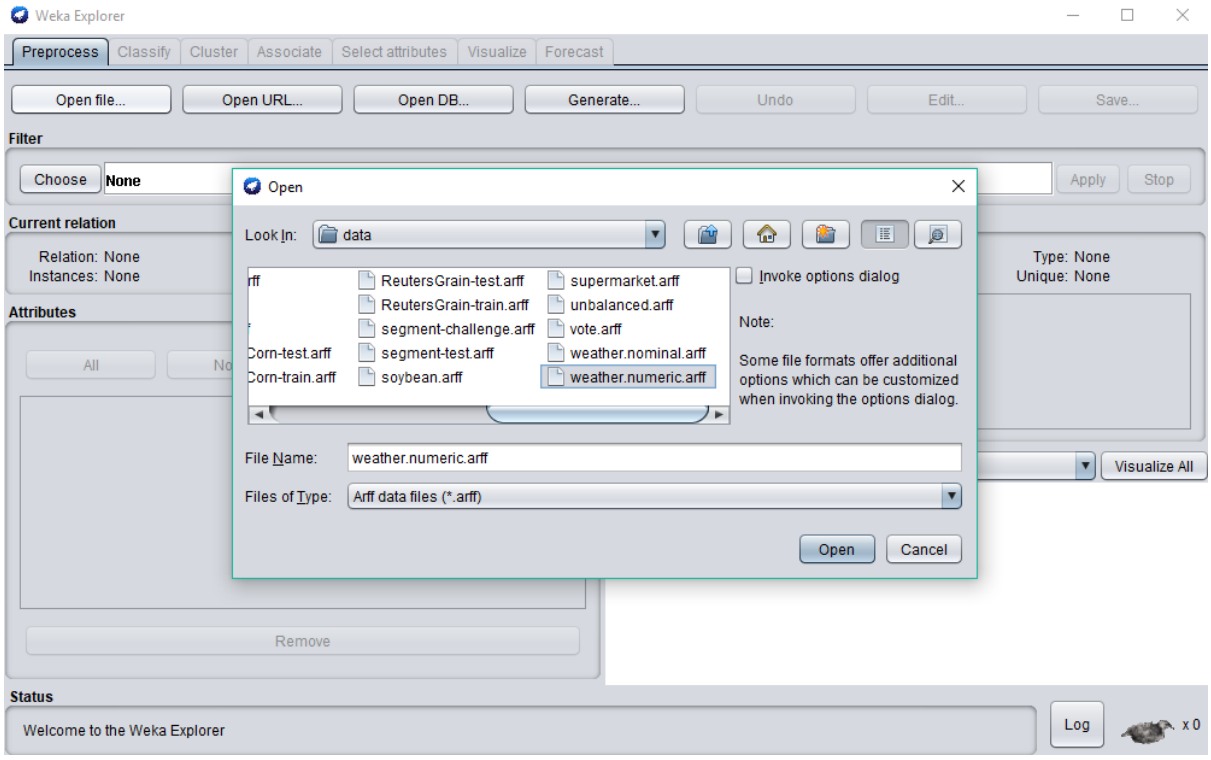
Step 2: Click on Explorer



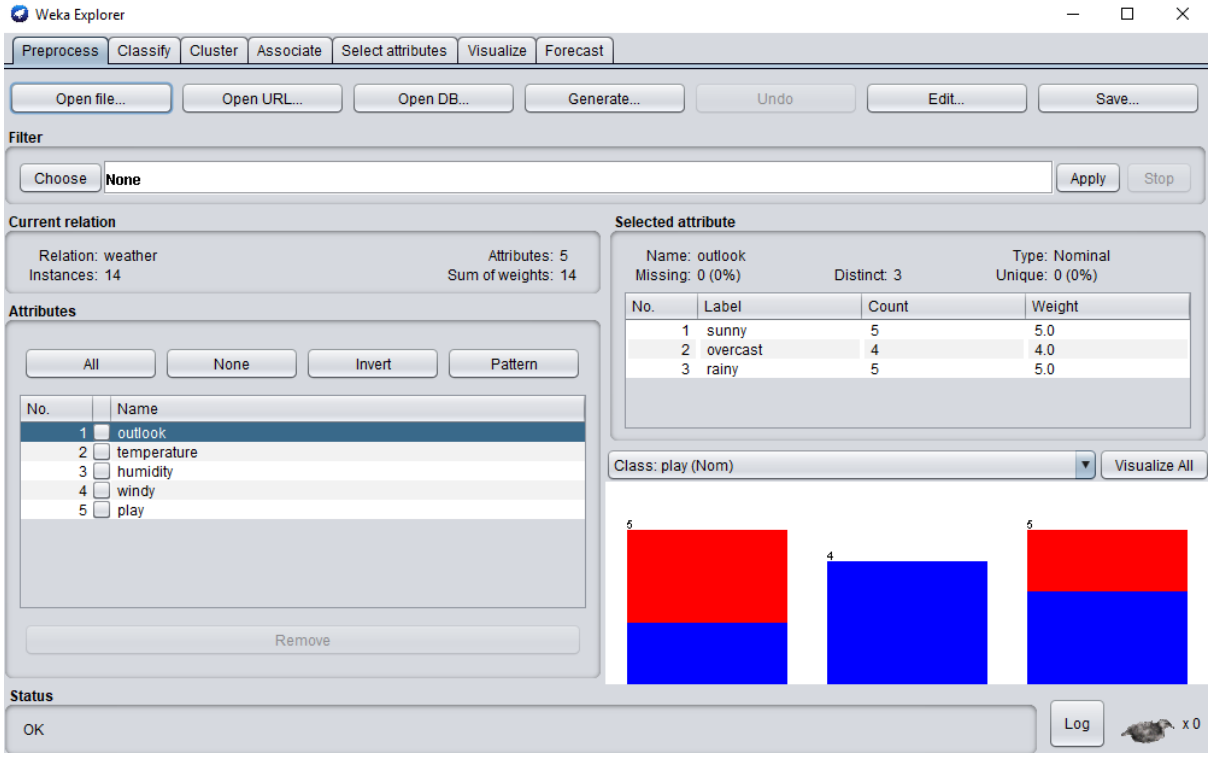
Step 3: Click on Open File Option



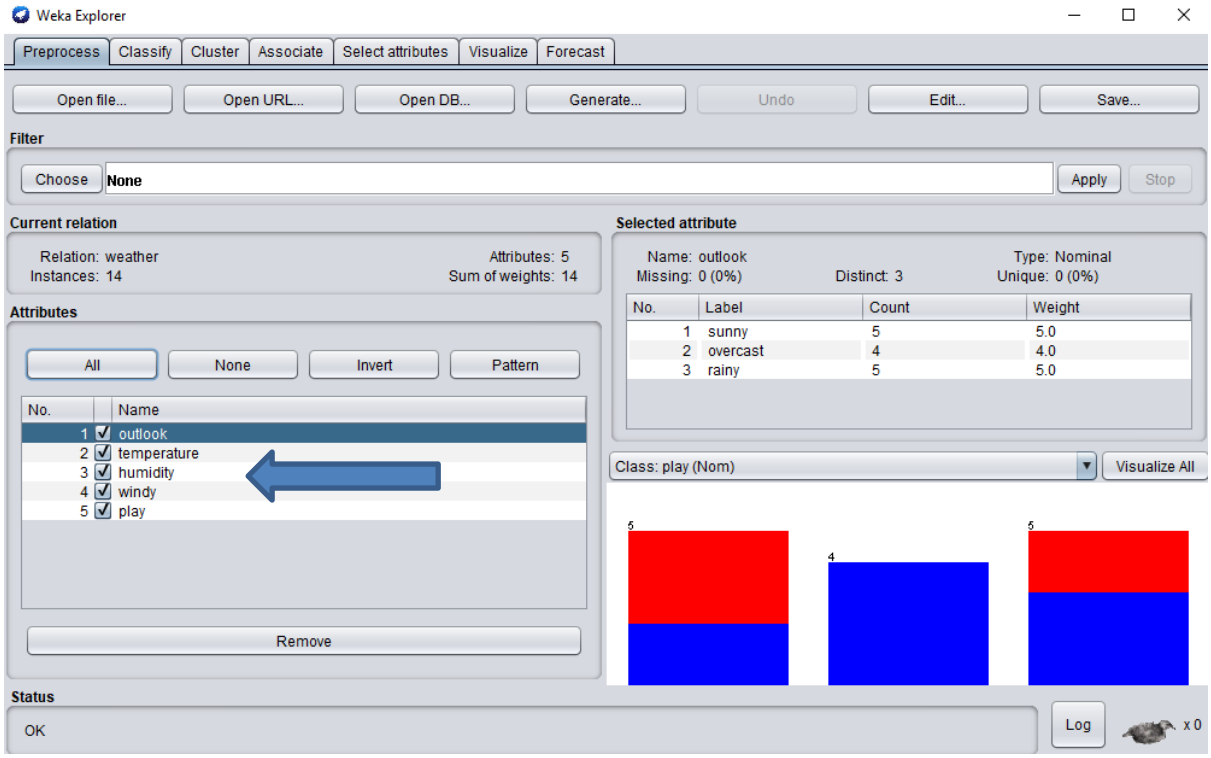
Step 4: Go to C://Program Files/Weka-3-8/data. Select weather.nominal.arff database



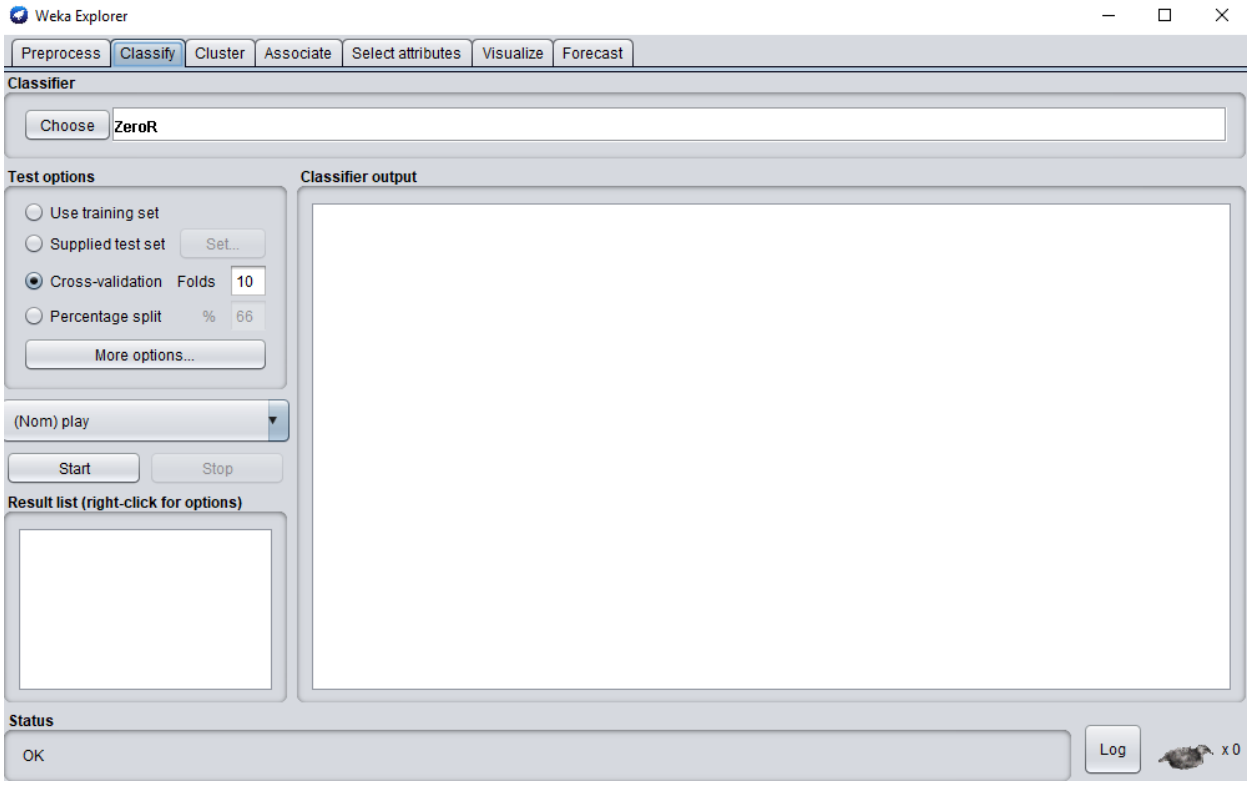
Step 5: Open the database



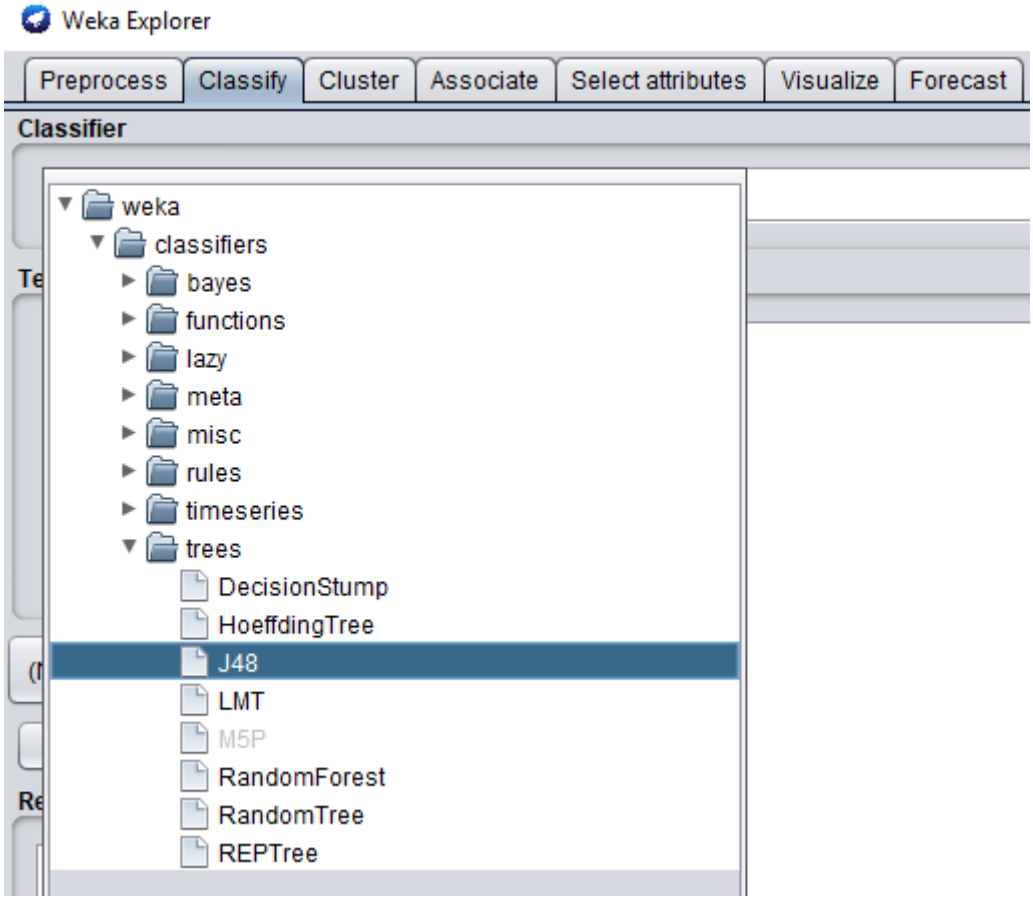
Step 6: Select all the attributes



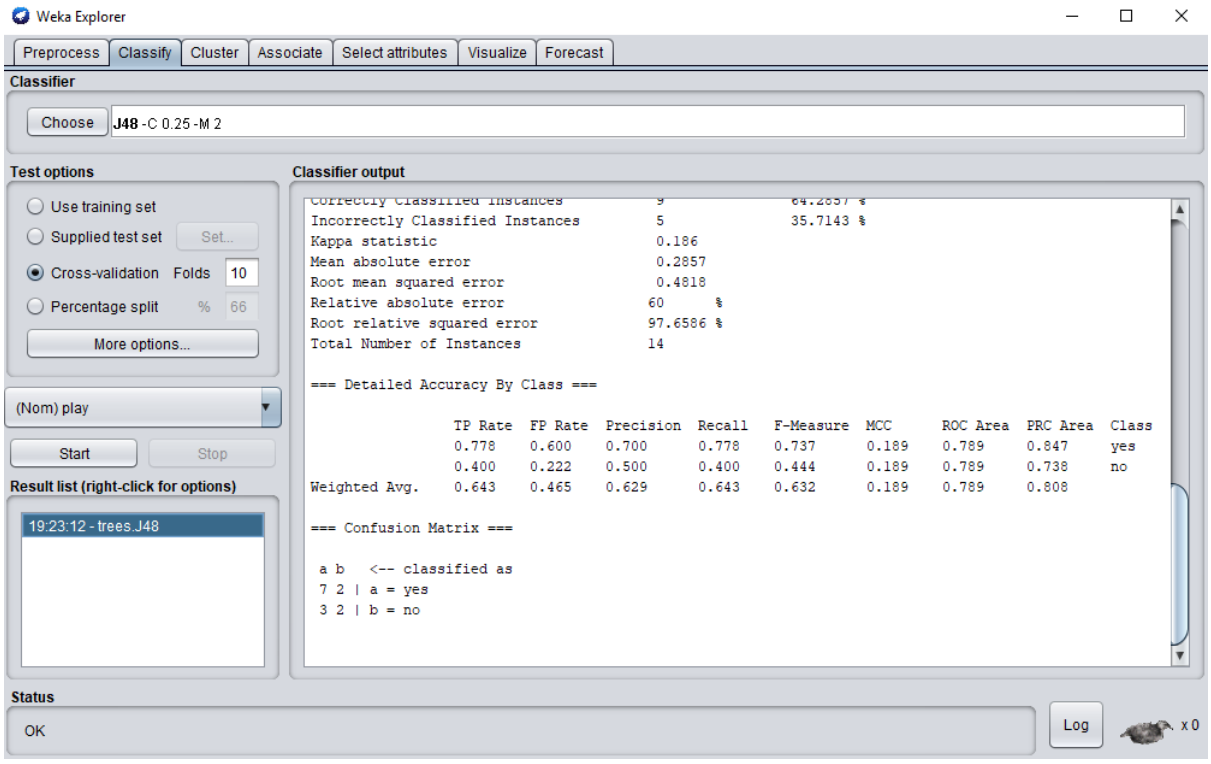
Step 7: Go to Classify tab



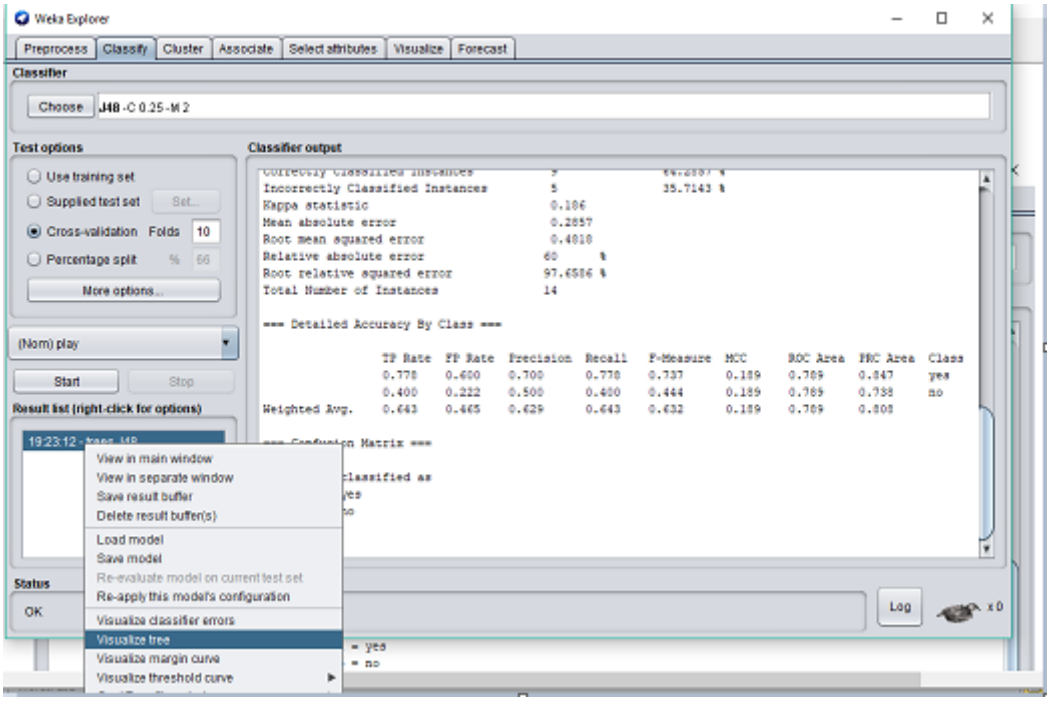
Step 8: Choose algorithm to apply. For that Go to->Choose tab->Trees-> J48



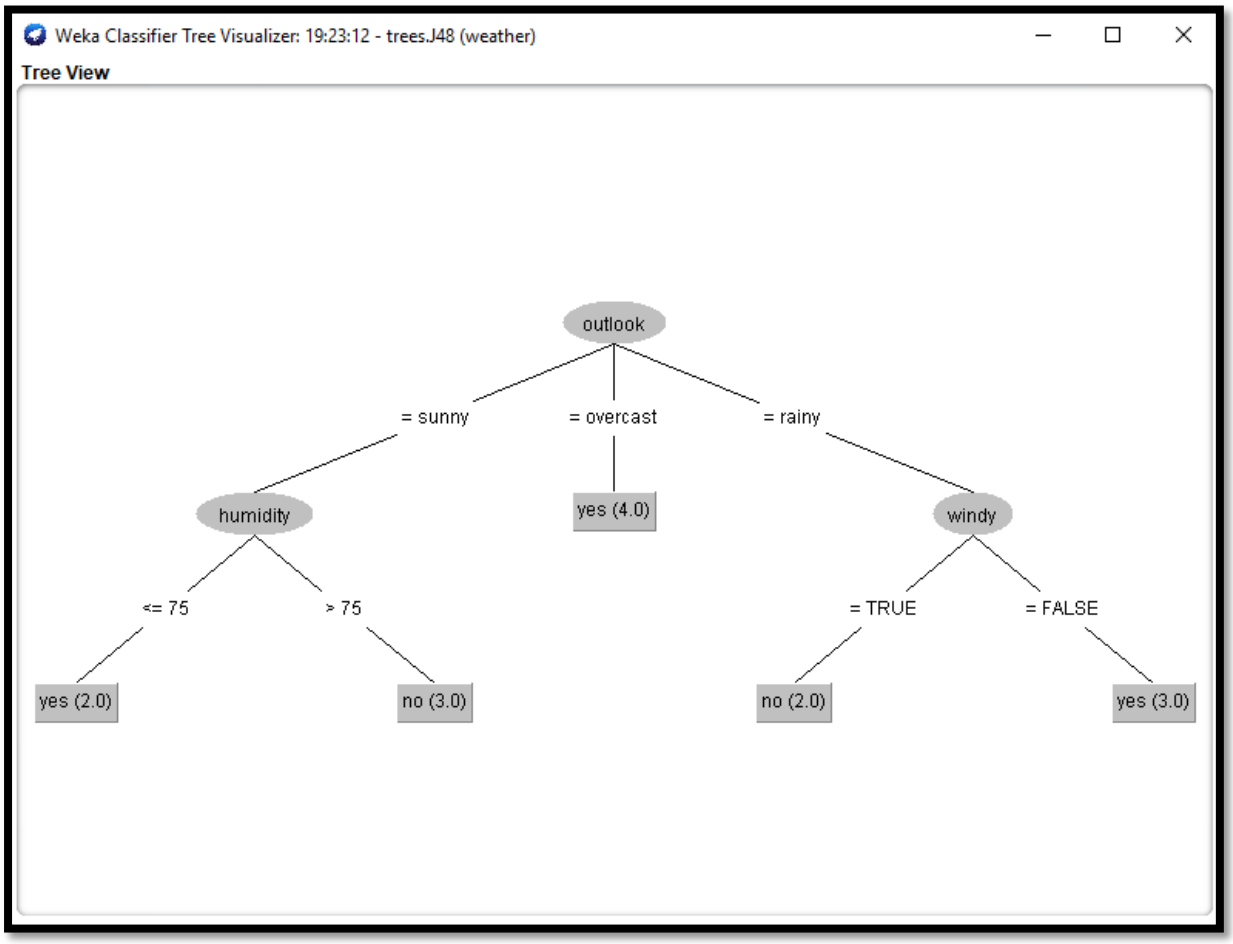
Step 9: Click on Start Button



Step 10: Right click on the created structure and select Visualize Tree option



Step 11: View the Decision tree

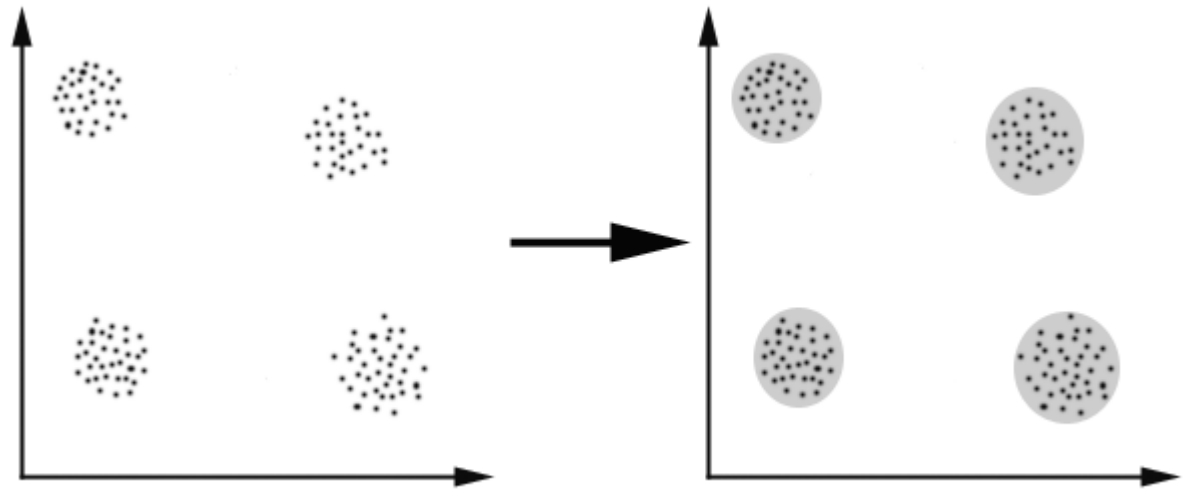


Practical 2:

Aim: Prepare the Analysis services for Adventure Works Cycles or (any other database) .Build the data mining model structure and Implement the clustering Algorithm.

Theory: Clustering can be considered the most important unsupervised learning problem. So, as every other problem of this kind, it deals with finding a structure in a collection of unlabelled data.
A loose definition of clustering could be "the process of organizing objects into groups whose members are similar in some way".
A cluster is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters.

We can show this with a simple graphical example:



In this case we easily identify the 4 clusters into which the data can be divided; the similarity criterion is distance: two or more objects belong to the same cluster if they are "close" according to a given distance (in this case geometrical distance). This is called distance-based *clustering*.

Another kind of clustering is conceptual clustering: two or more objects belong to the same cluster if this one defines a concept *common* to all that objects. In other words, objects are grouped according to their fit to descriptive concepts, not according to simple similarity measures.

The goal of clustering is to determine the intrinsic grouping in a set of unlabelled data. But how to decide what constitutes a good clustering? It can be shown that there is no absolute "best" criterion which would be independent of the final aim of the

clustering. Consequently, it is the user which must supply this criterion, in such a way that the result of the clustering will suit their needs.

Clustering algorithms may be classified as listed below:

- Exclusive Clustering
- Overlapping Clustering
- Hierarchical Clustering
- Probabilistic Clustering

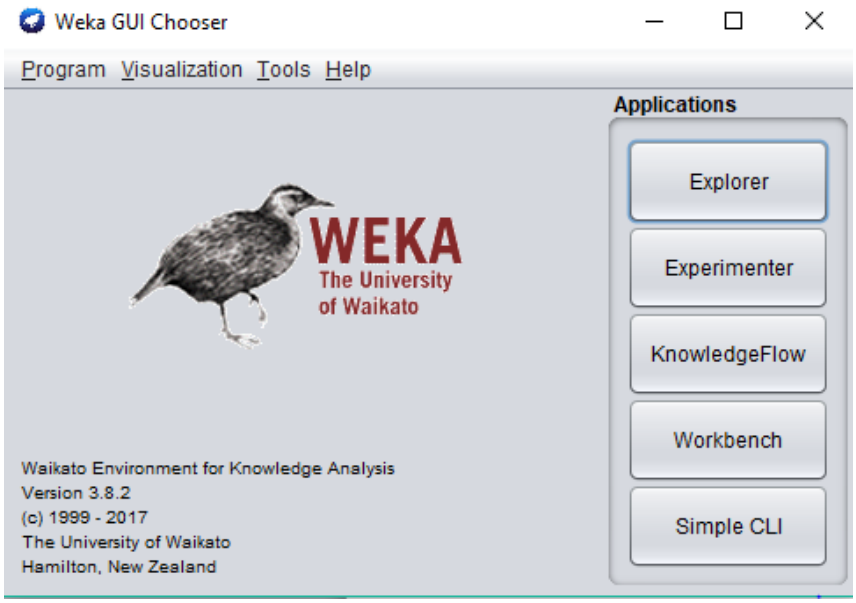
K-means is an *exclusive clustering* algorithm. K-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem.

The algorithm is composed of the following steps:

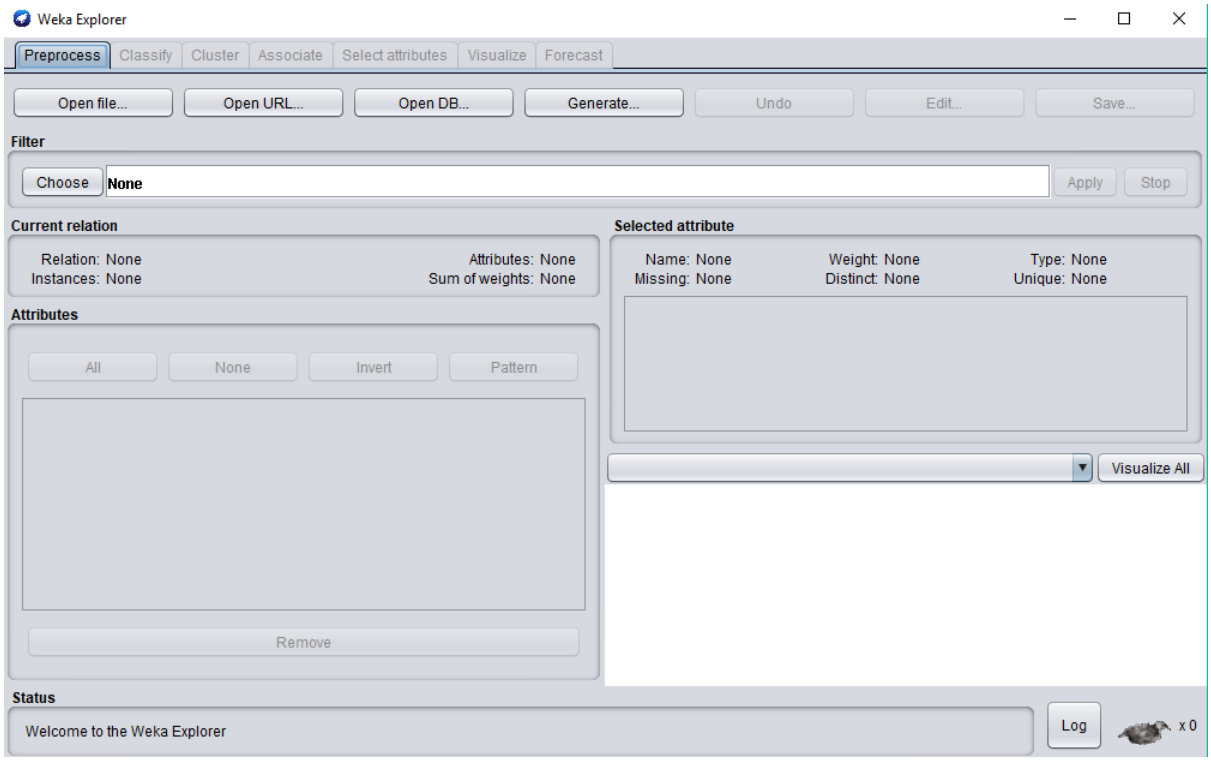
1. *Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.*
2. *Assign each object to the group that has the closest centroid.*
3. *When all objects have been assigned, recalculate the positions of the K centroids.*
4. *Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.*

Practical

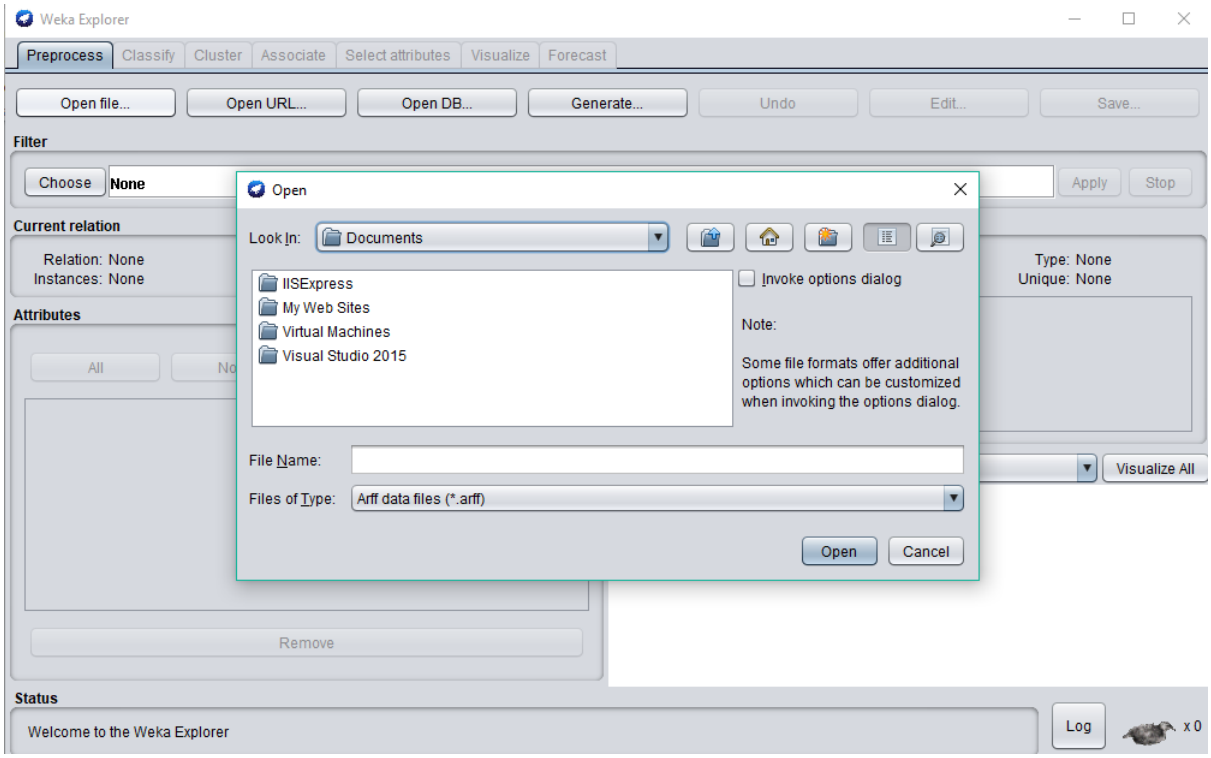
Step 1: Open Weka



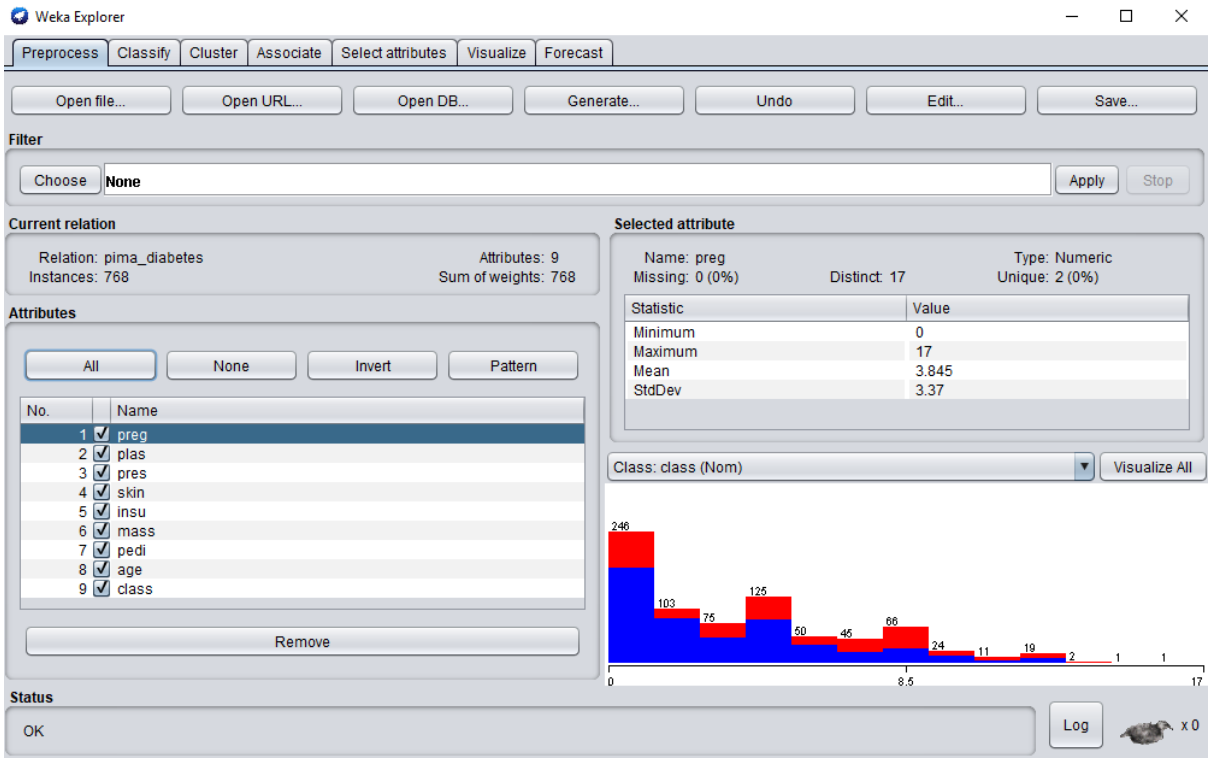
Step 2: Click on Explorer



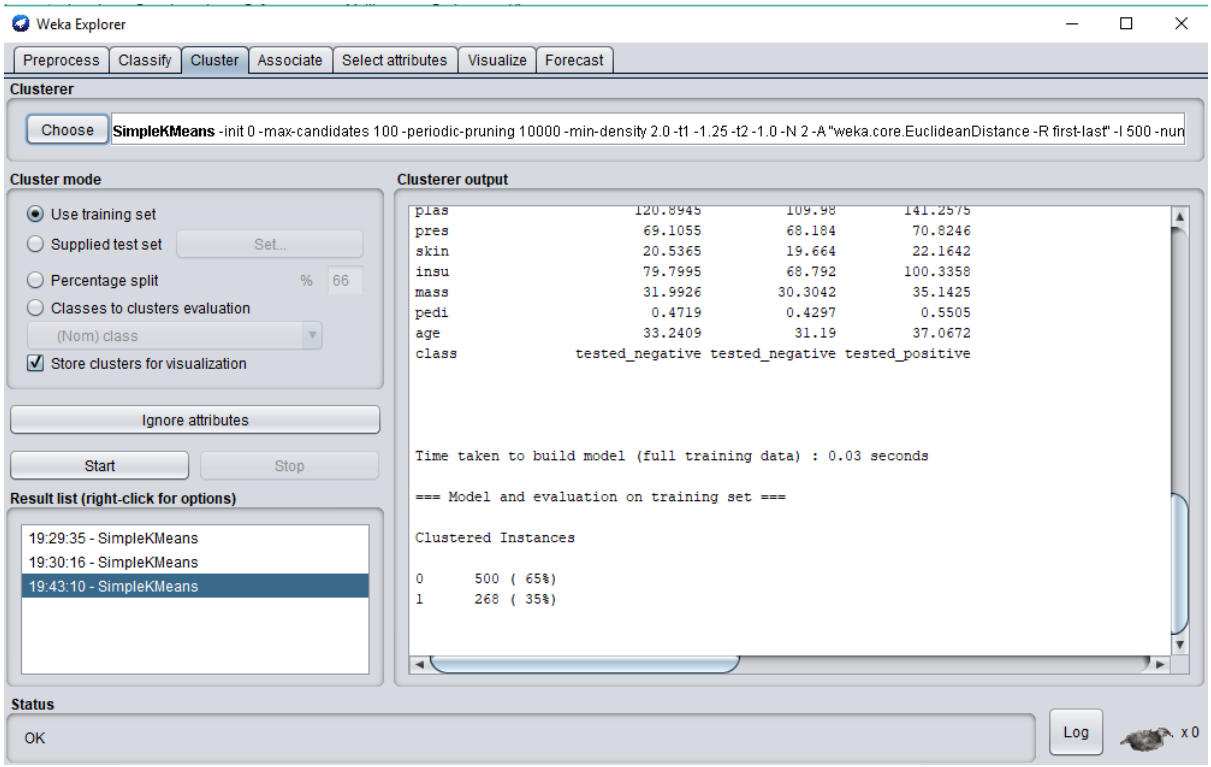
Step 3: Click on Open File Option



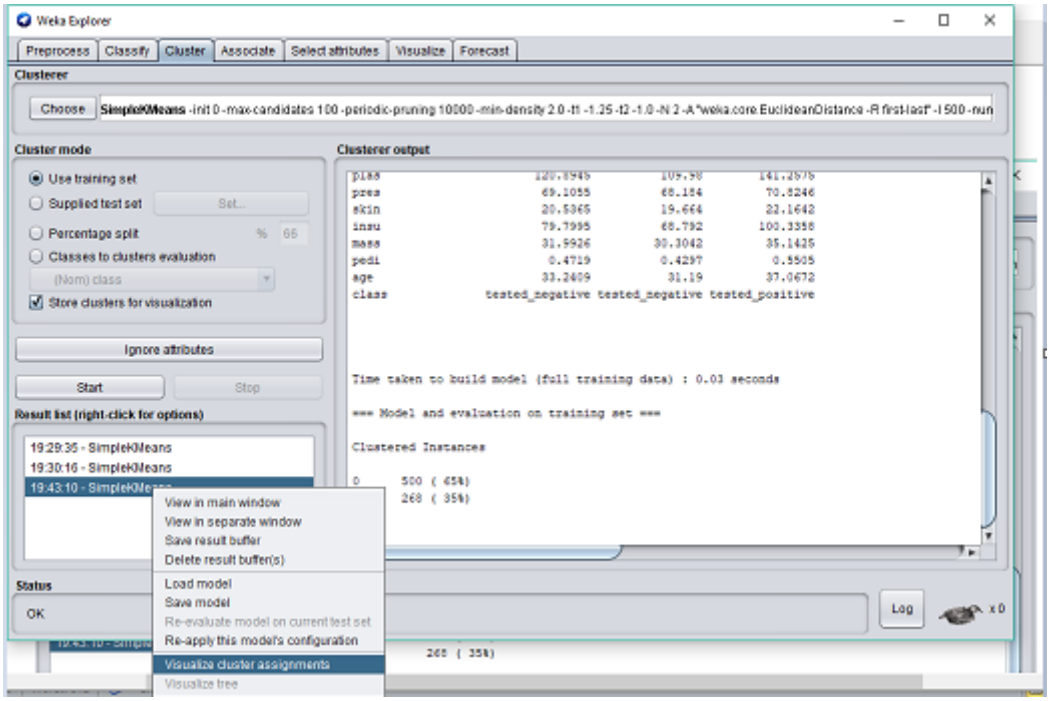
Step 4: Go to C://Program Files/Weka-3-8/data. Select diabetes.arff database



Step 5: Go to Cluster tab and Choose SimpleKMeans algorithm



Step 6: Right Click on created project and select Visualize cluster assignments


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Forecast
Clusterer
Choose
SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
19:29:35 - SimpleKMeans
19:30:16 - SimpleKMeans
19:43:10 - SimpleKMeans
Clusterer output
plas 120.8945 109.98 141.2675
pres 69.1055 68.184 70.8246
skin 20.5365 19.664 22.1642
insu 79.7995 68.792 100.3358
mass 31.9926 30.3042 35.1425
pedi 0.4719 0.4297 0.5505
age 33.2409 31.19 37.0672
class tested_negative tested_negative tested_positive
Time taken to build model (full training data) : 0.03 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 500 (65%)
1 268 (35%)
View in main window
View in separate window
Save result buffer
Delete result buffer(s)
Load model
Save model
Re-evaluate model on current test set
Re-apply this model's configuration
Visualize cluster assignments
Visualize tree
Status
OK
Log
x 0

Practical 3:

Aim: Prepare the Analysis services for Adventure Works Cycles or (any other database) . Build the data mining model structure and Implement Naïve Bayes Algorithm.

Theory:

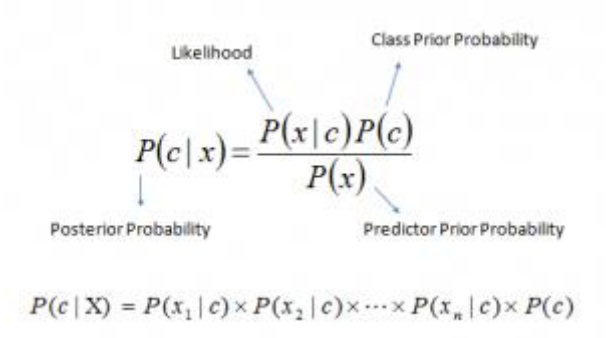
Bayesian classification is based on Bayes' theorem. Simple Bayesian classifier known as the naive Bayesian classifier. Naïve Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence. It is made to simplify the computations involved and, in this sense, is considered "naïve."

What is Naive Bayes algorithm?

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Above,

- $P(c|x)$ is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).
- $P(c)$ is the prior probability of *class*.

- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

How Naive Bayes algorithm works?

Let's understand it using an example. Below I have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let's follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Problem: Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

What are the Pros and Cons of Naive Bayes?

Pros:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

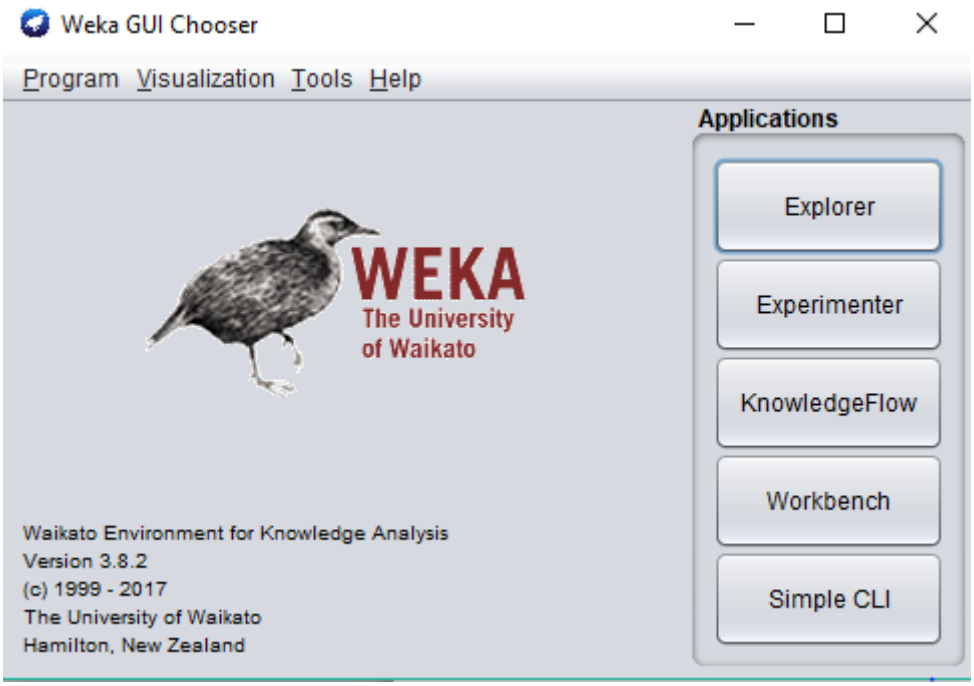
Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency". To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

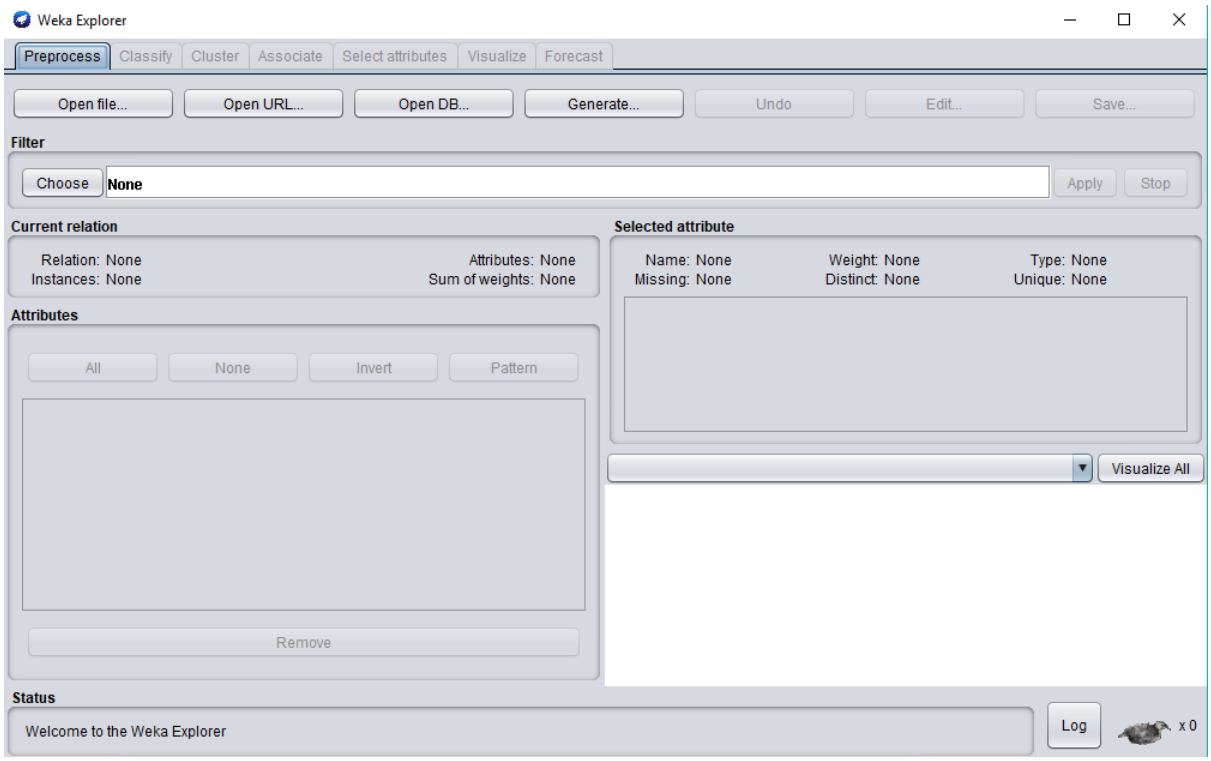
4 Applications of Naive Bayes Algorithms

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

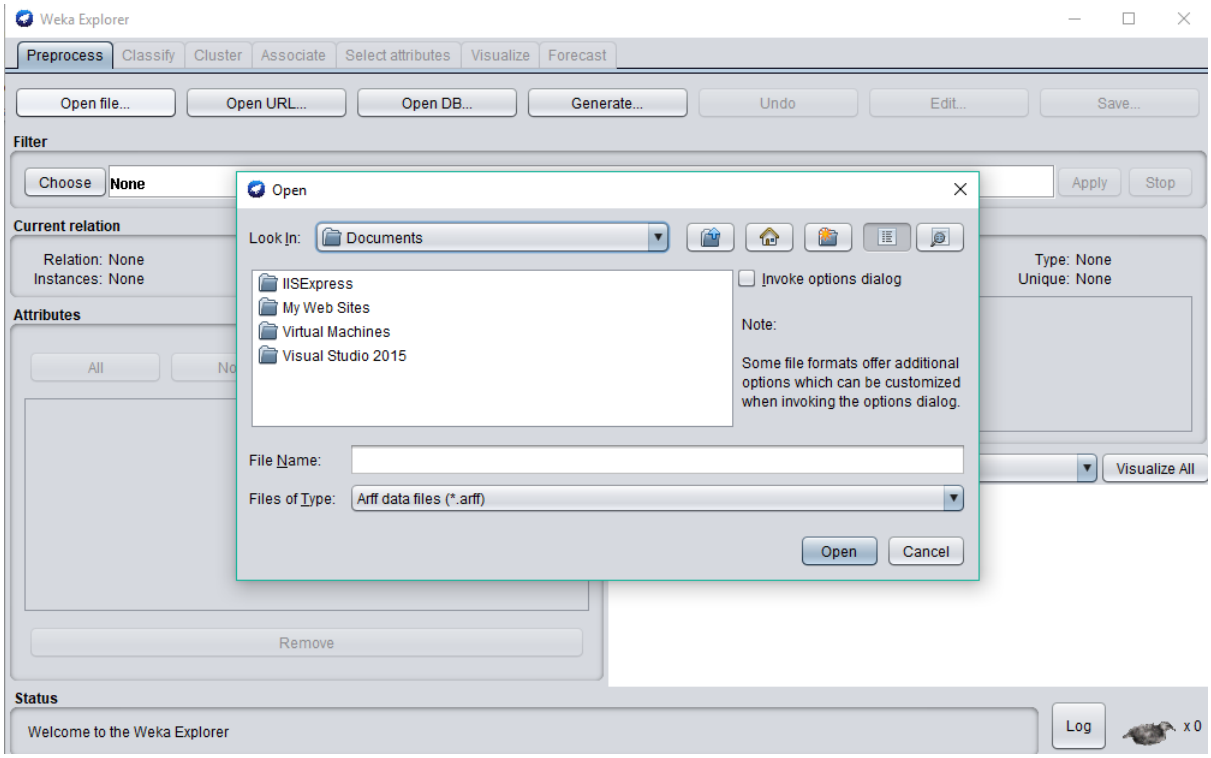
Step 1: Open Weka



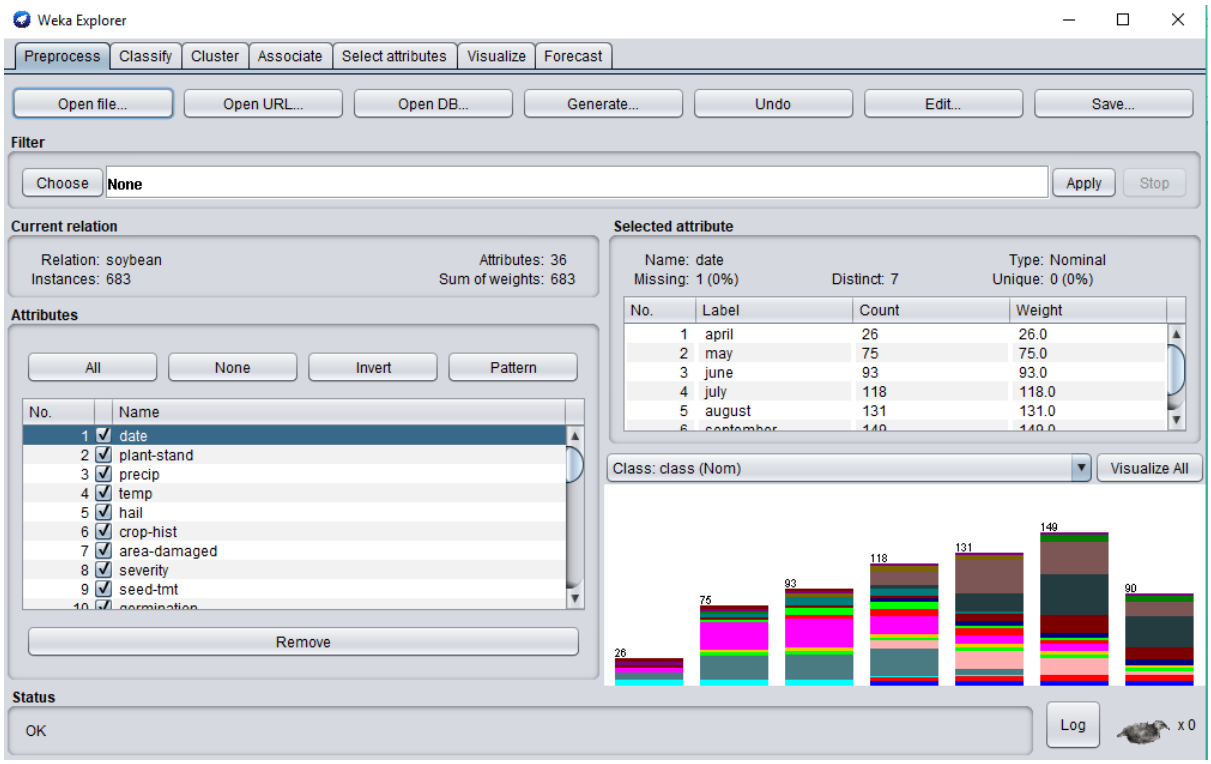
Step 2: Click on Explorer



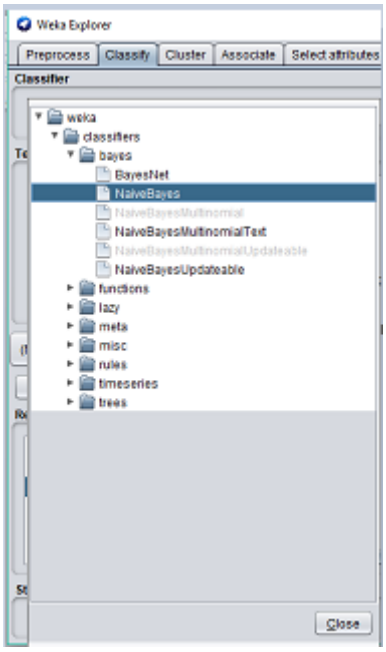
Step 3: Click on Open File Option



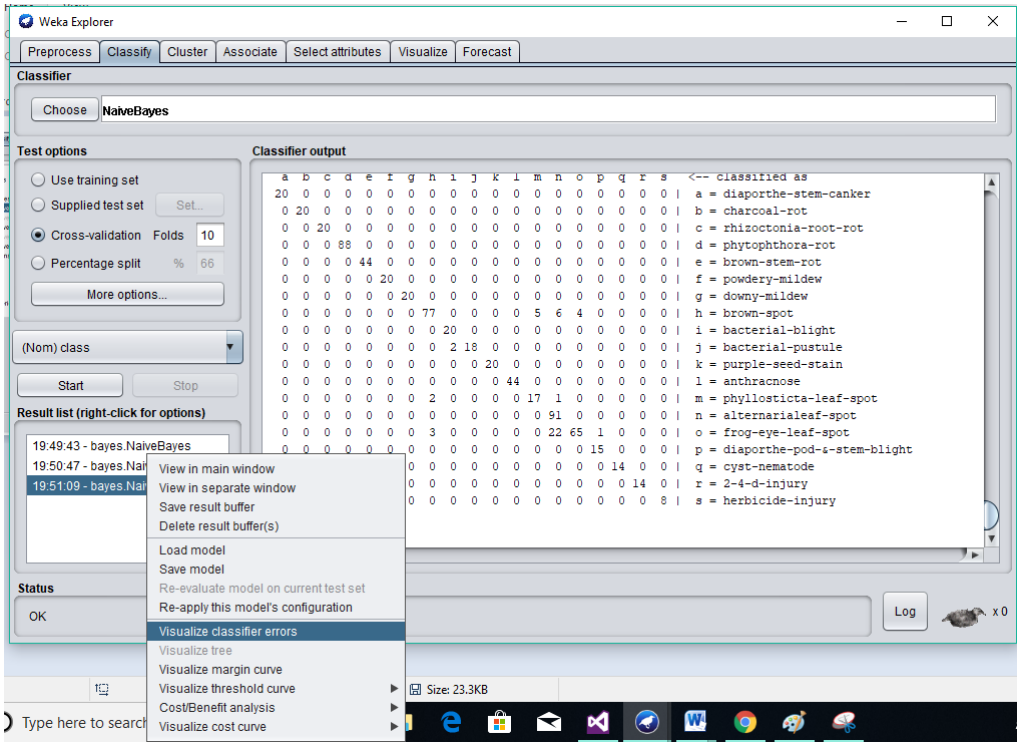
Step 4: Go to C://Program Files/Weka-3-8/data. Select soyabean.arff database



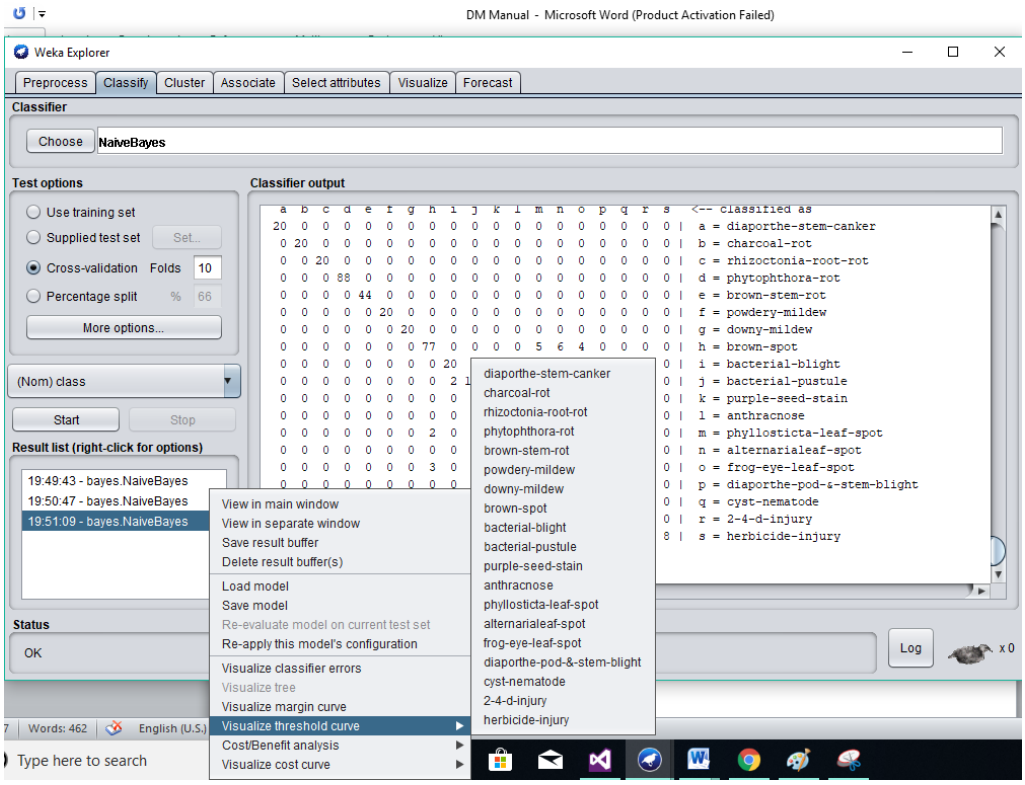
Step 5: Go to Classify tab and Select Naïve Bayes algorithm under Bayes tab



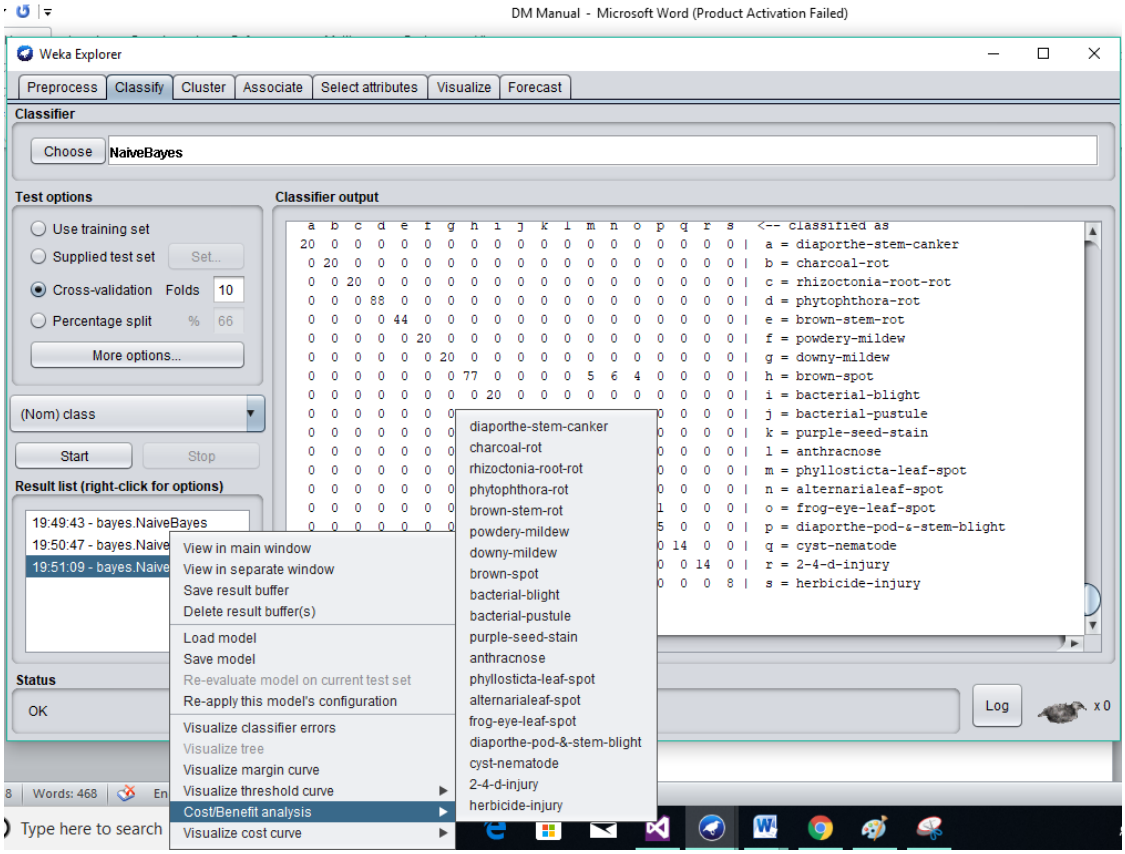
Step 6: Right Click on created structure and select Visualize Classifier Errors



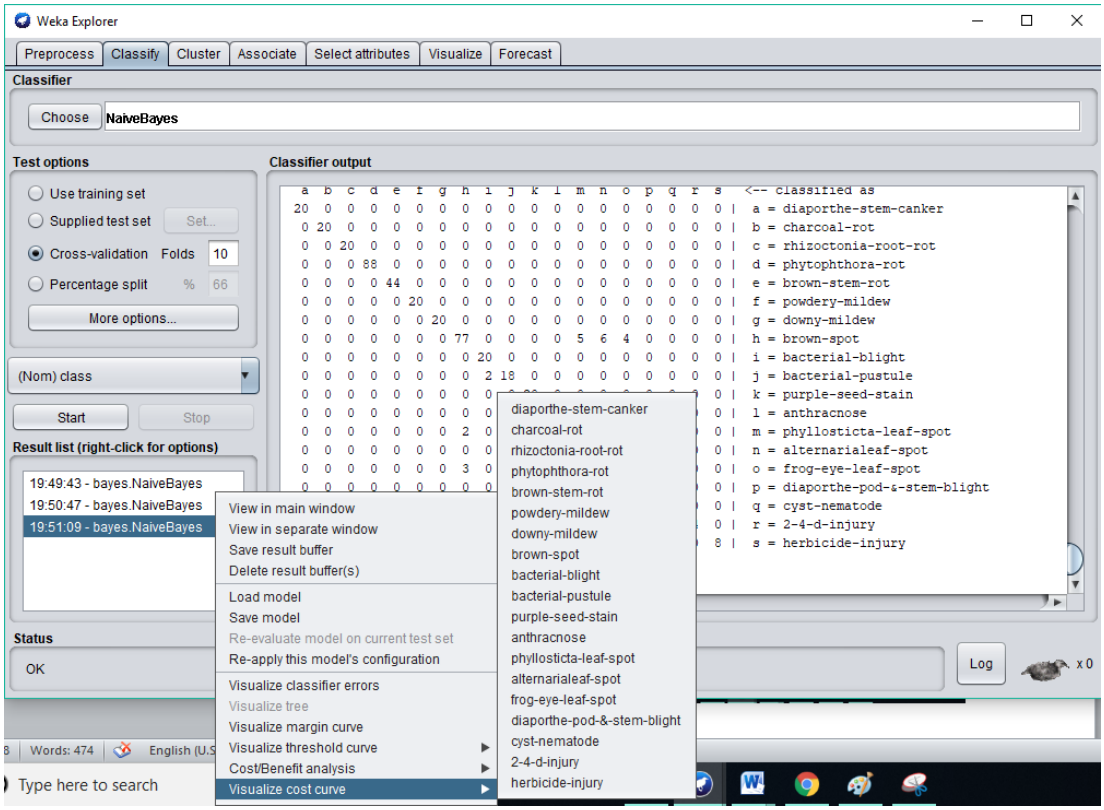
Step 7: Explore Visualize Threshold Curve



Step 8: Explore Cost Benefit Analysis



Step 9: Explore Cost Curve


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Forecast
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
19:49:43 - bayes.NaiveBayes
19:50:47 - bayes.NaiveBayes
19:51:09 - bayes.NaiveBayes
Classifier output
a b c d e f g h i j k l m n o p q r s <-- classified as
a = diaporthe-stem-canker
b = charcoal-rot
c = rhizoctonia-root-rot
d = phytophthora-rot
e = brown-stem-rot
f = powdery-mildew
g = downy-mildew
h = brown-spot
i = bacterial-blight
j = bacterial-pustule
k = purple-seed-stain
l = anthracnose
m = phyllosticta-leaf-spot
n = alternarialeaf-spot
o = frog-eye-leaf-spot
p = diaporthe-pod-&-stem-blight
q = cyst-nematode
r = 2-4-d-injury
s = herbicide-injury
View in main window
View in separate window
Save result buffer
Delete result buffer(s)
Load model
Save model
Re-evaluate model on current test set
Re-apply this model's configuration
Visualize classifier errors
Visualize tree
Visualize margin curve
Visualize threshold curve
Cost/Benefit analysis
Visualize cost curve
diaporthe-stem-canker
charcoal-rot
rhizoctonia-root-rot
phytophthora-rot
brown-stem-rot
powdery-mildew
downy-mildew
brown-spot
bacterial-blight
bacterial-pustule
purple-seed-stain
anthracnose
phyllosticta-leaf-spot
alternarialeaf-spot
frog-eye-leaf-spot
diaporthe-pod-&-stem-blight
cyst-nematode
2-4-d-injury
herbicide-injury
Status
OK
Log
x 0
Words: 474
English (U.S
Type here to search

Practical 4:

Aim: Prepare the Analysis services for Adventure Works Cycles or (any other database).Build the basic Time series model structure and create the predictions

Theory: The Time Series mining function enables forecasting of time series values. Similar to common regression methods, time series algorithms predict a numerical value. In contrast to common regression methods, time series predictions are focused on future values of an ordered series. These predictions are commonly called forecasts.

The time series algorithms are univariate algorithms. This means that the independent variable is a time column or an order column. The forecasts are based on past values. They are not based on other independent columns.

Time series algorithms are different from common regression algorithms because they do not only predict future values but also incorporate seasonal cycles into the forecast.

Time series algorithms

The Time Series mining function provides algorithms that are based on different underlying model assumptions with several parameters. The learning algorithms try to find the best model and the best parameter values for the given data.

If you do not specify a seasonal cycle, it is automatically determined. Also, missing values and non-equidistant time series are automatically interpolated.

The Time Series mining function provides the following algorithms to predict future trends:

- Autoregressive Integrated Moving Average (ARIMA)
- Exponential Smoothing
- Seasonal Trend Decomposition

Which of the algorithms creates the best forecast of your data depends on different model assumptions. You can calculate all forecasts at the same time. The algorithms calculate a detailed forecast including seasonal behaviour of the original time series. With the Time Series Visualizer, you can evaluate and compare the resulting curves.

Autoregressive Integrated Moving Average (ARIMA)

The ARIMA algorithm also incorporates seasonal components. Therefore this algorithm is also referred to as Seasonal ARIMA (SARIMA).

The autoregressive part of the algorithm uses weighted previous values while the moving average part weighs the previously assumed errors of the time series.

The ARIMA algorithm assumes the error to be independent and identically distributed from a normal distribution with zero mean. The basic ARMA model works for stationary time series only. Stationary time series contain equal mean and equal variance for the whole time series. Therefore the integrated part creates stationary series by differentiation.

Exponential Smoothing

Exponential Smoothing can consist of the following components:

- Basic level at a certain point in time.
- Trend.

  The trend can have additive or multiplicative characteristics. Also, it can be damped or non-damped.

- The seasonal component.

Dependant on the data, trend and the seasonal component are optional. There are ARIMA models that correspond to Exponential Smoothing models and vice versa.

Seasonal Trend Decomposition

Seasonal Trend Decomposition fits different seasonal trend functions to the given data and selects the best seasonal trend function according to an error measure. The following trends are used during the training run:

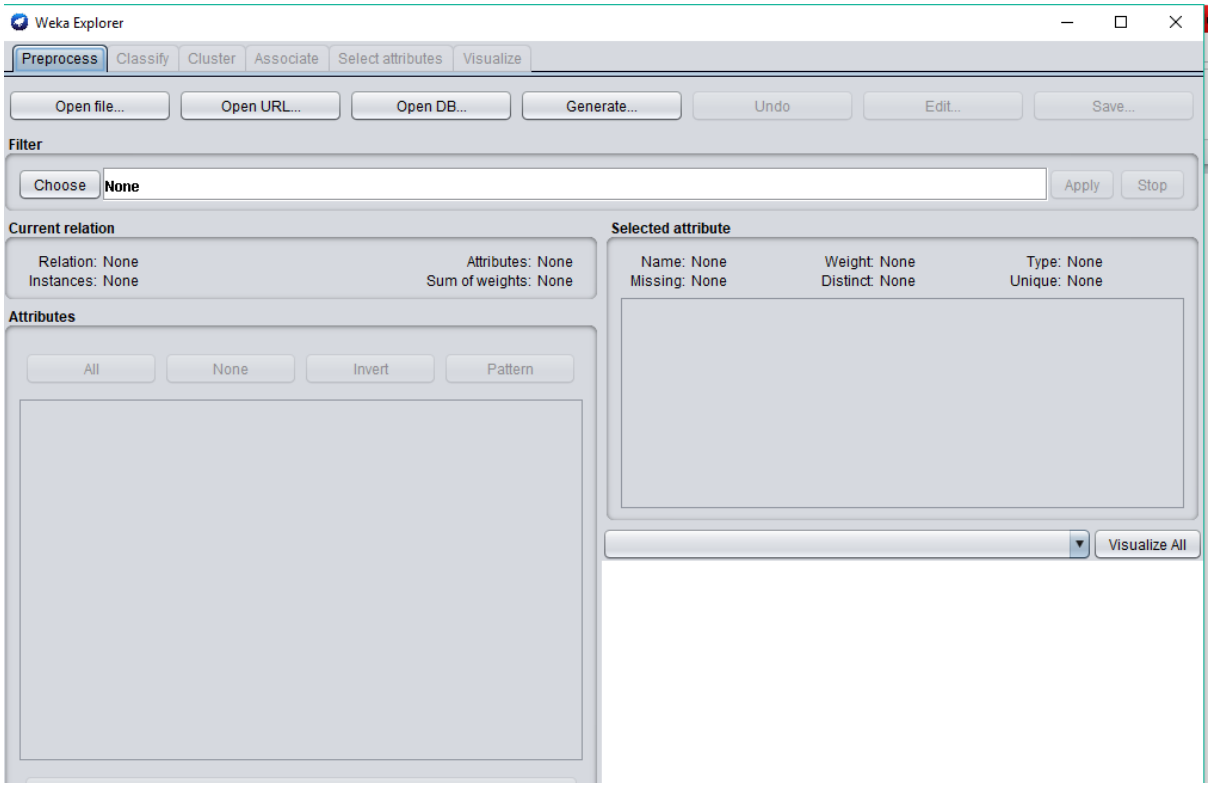
- Linear trend
- Quadratic trend
- Cubic trend
- Logarithmic trend
- Exponential trend
- Hyperbolic trend

The seasonality is incorporated in an additive or multiplicative way.

Practical

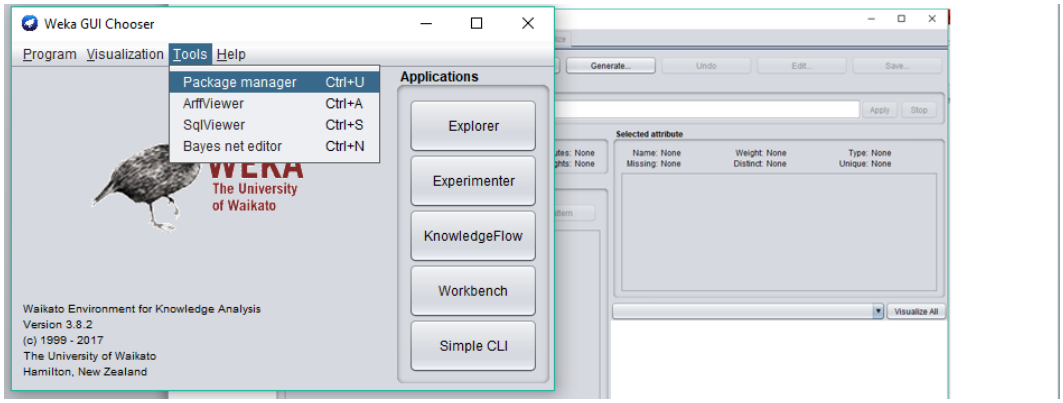
Step 1: Open Weka

Step 2: Click on Explorer

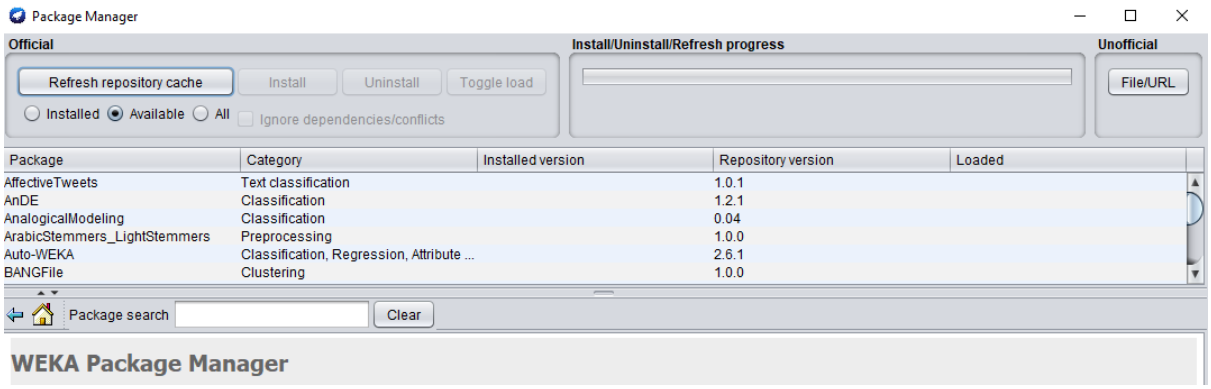


Step 3: For this practical we need to install Forecast, as you can see last tab in Visualize tab and we need to install one more tab Forecast.

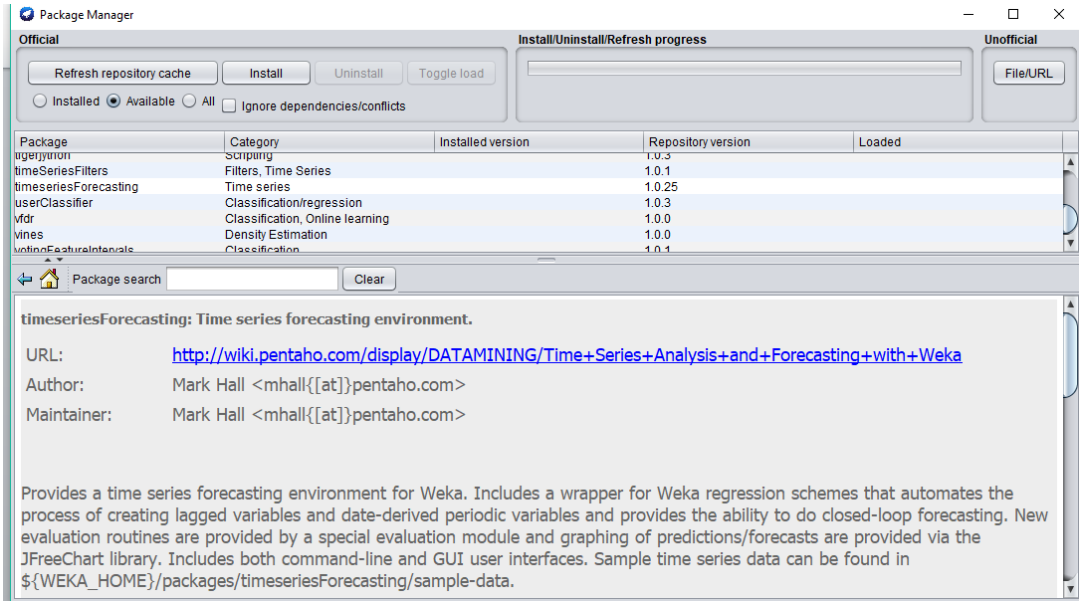
Step 4: Open Weka-> Click on Tools-> Select Package Manager



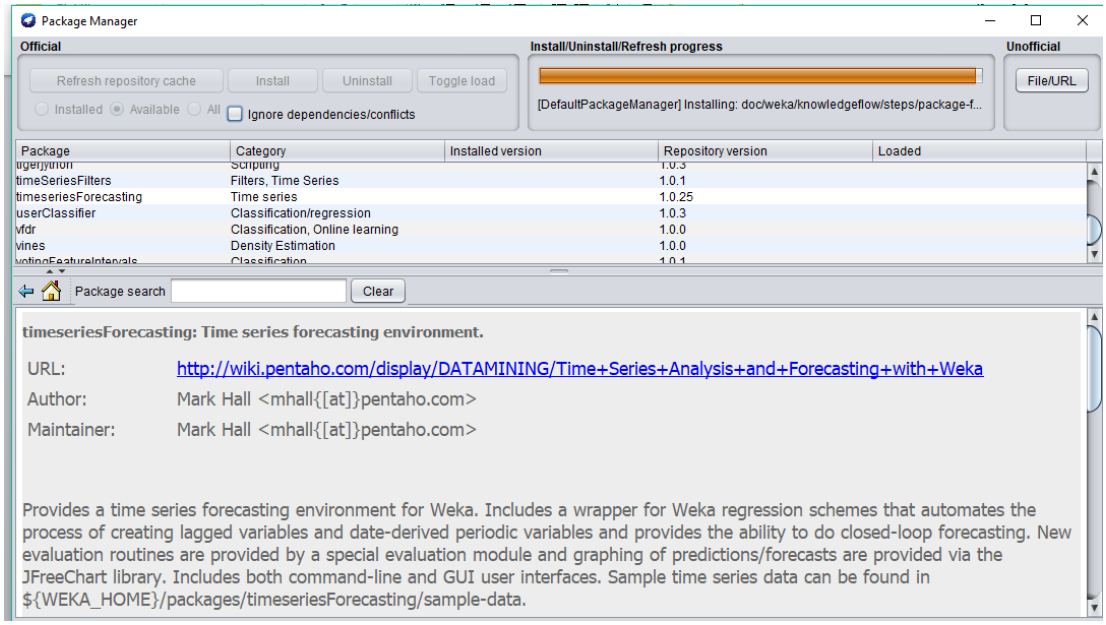
Step 5: Following is the Package Manager through which we need to install



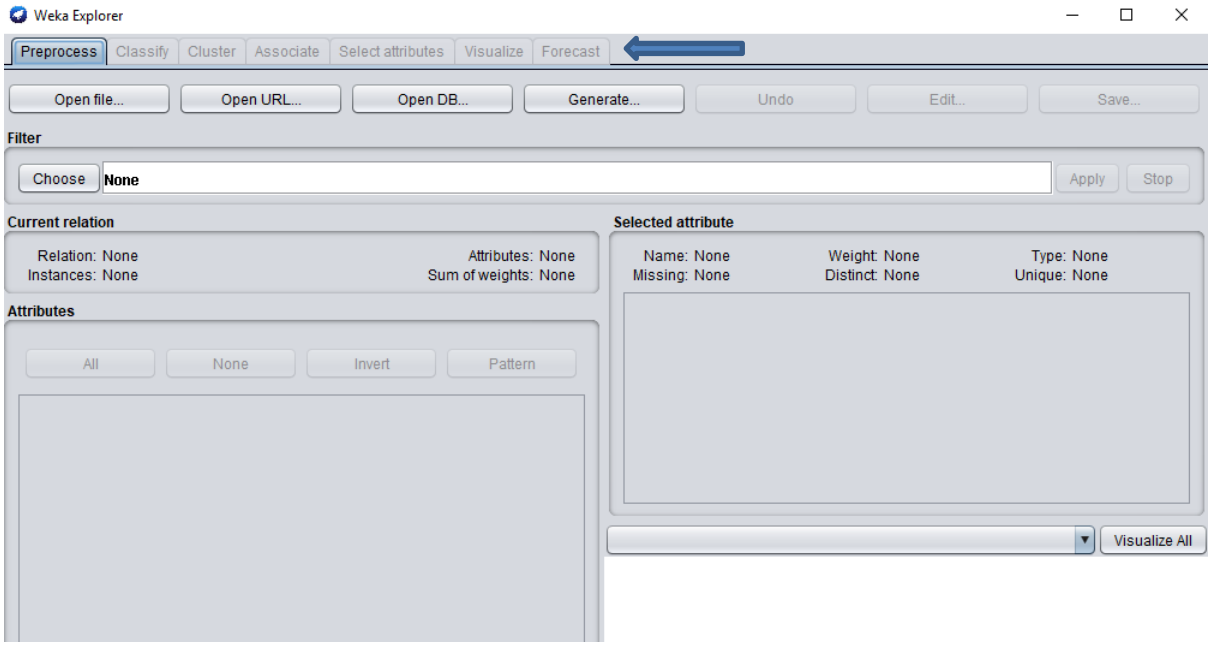
Step 6: Scroll Down and Select timeSeriesForecasting



Step 7: Click on Install



Step 8: Now again go to Explorer and you can find new tab is been added



Step 9: Create the following database. Open Notepad and type the following

@relation AnkaraPopulation

@attribute year numeric

@attribute population numeric

@data

2007, 70586256

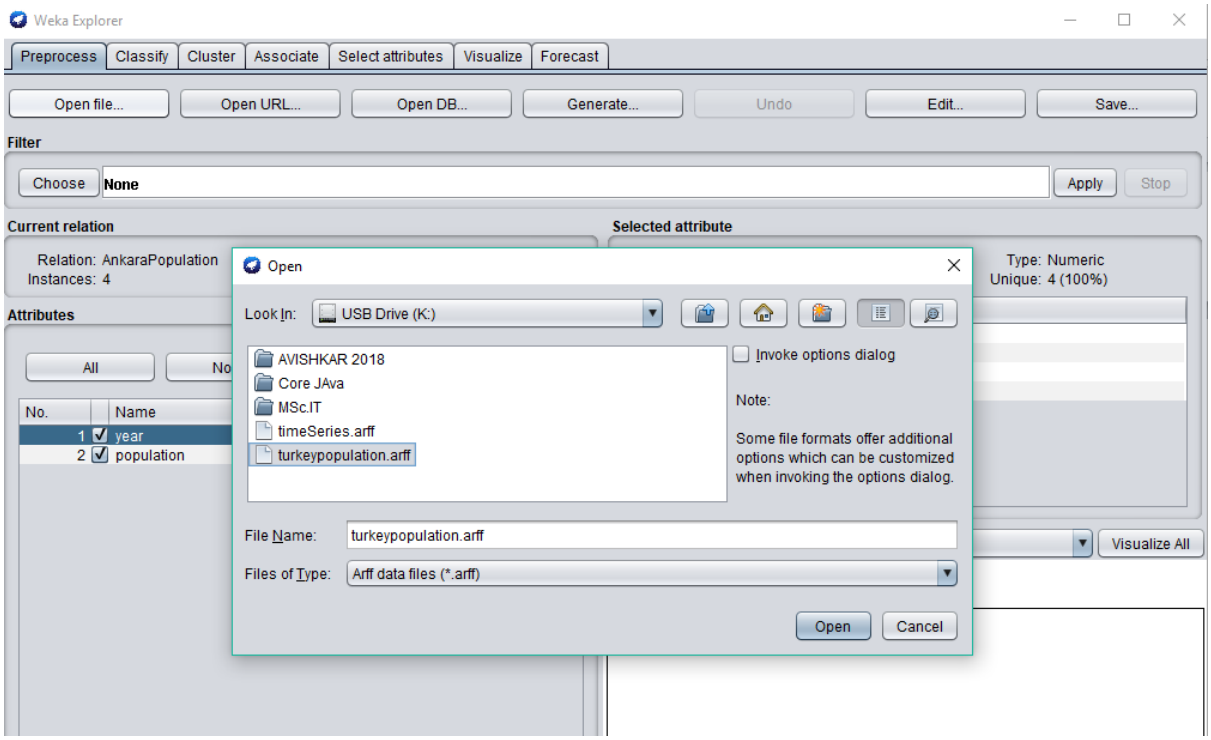
2008,71517100

2009,72561312

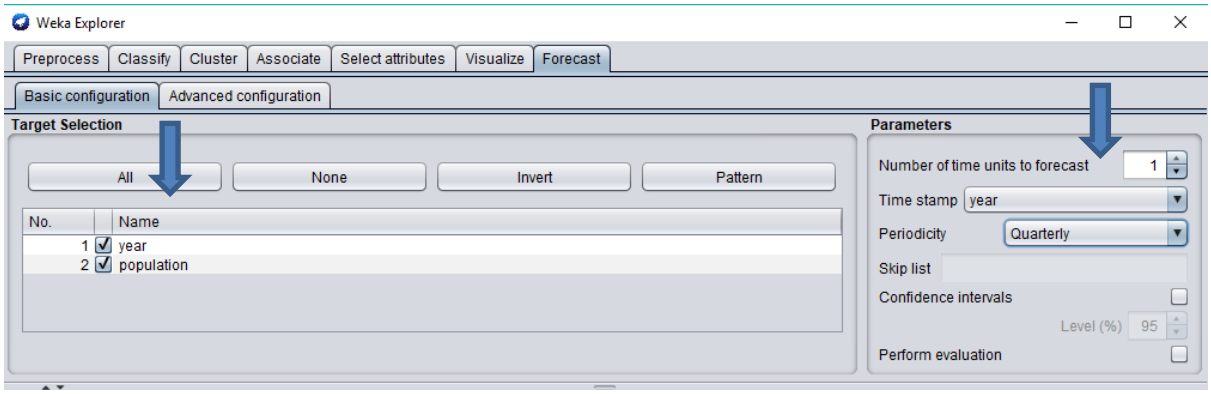
2010,73722988

Save the file with .arff extension

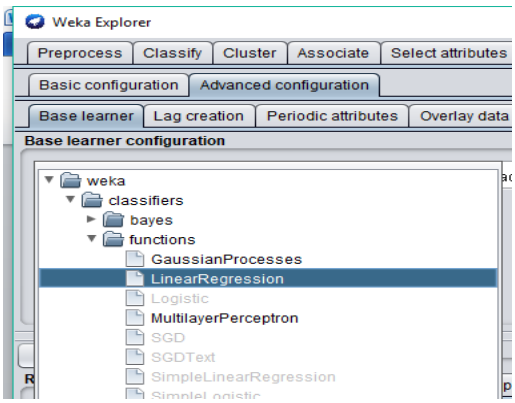
Step 10: Open the created database



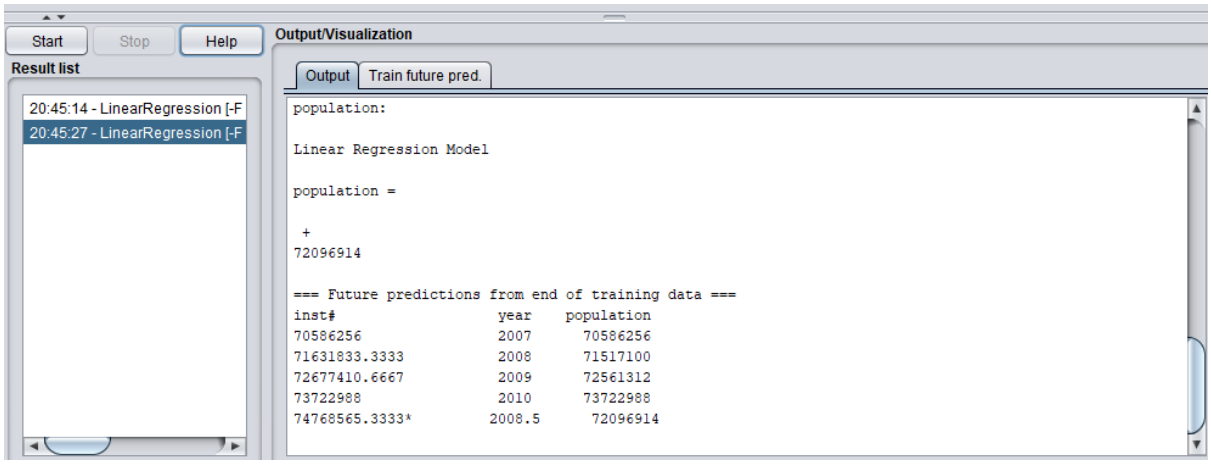
Step 11: Go to Forecast tab and under basic configuration do the following changes:



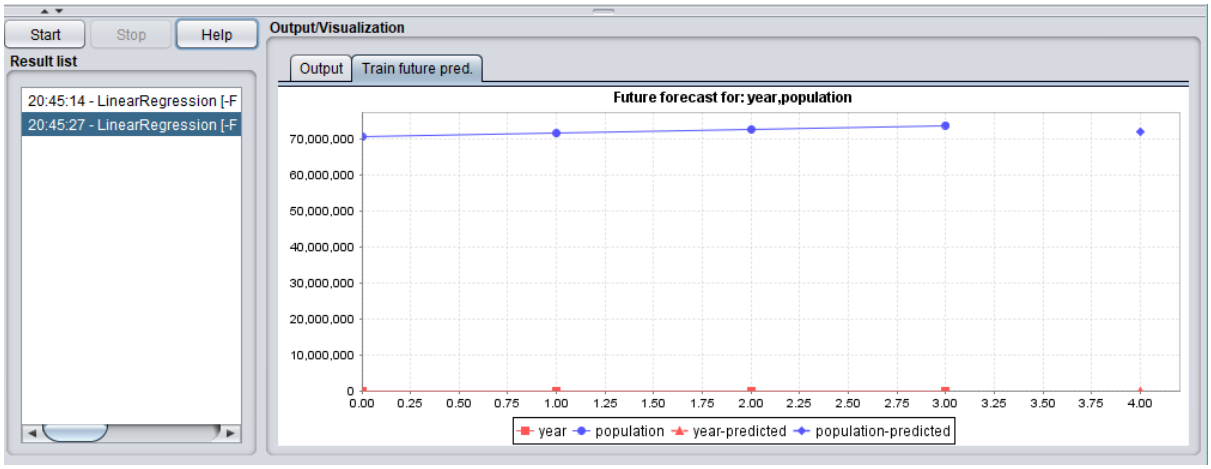
Step 12: Go to Advanced Configuration tab and select LinearRegression Algorithm under functions option.



Step 13: Click on start button



Step 14: Go to Train future pred option.



Practical 5:

Aim: Prepare the Analysis services for Adventure Works Cycles or (any other database) .Build the basic data mining model and show the implementation of Association algorithm. And also apply the DMX queries.

Theory:

Rule support and confidence are two measures of rule interestingness. They respectively reflect the usefulness and certainty of discovered rules. Asupport of 2% for Association Rule means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together. A confidence of 60% means that 60% of the customers who purchased a computer also bought the software. Typically, association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold.

A transaction T is said to contain A if and only if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I$,$B \subset I$,and $A \cap B = \varphi$. The rule $A \Rightarrow B$ holds in the transaction set D with support s, where s is the percentage of transactions in D that contain $A \cup B$ (i.e., the union of sets A and B, or say, both A and B).

This is taken to be the probability, $P(A \cup B)$.1 The rule $A \Rightarrow B$ has confidence c in the transaction set D, where c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, P(B|A). That is,

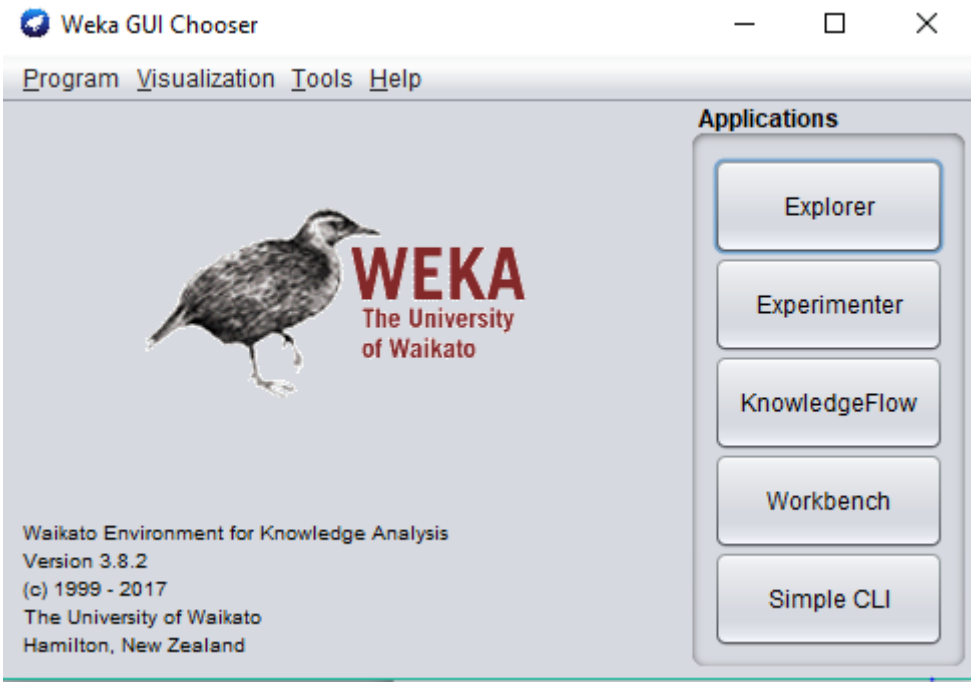
$$\begin{aligned} support(A \Rightarrow B) &= P(A \cup B) \\ confidence(A \Rightarrow B) &= P(B|A). \end{aligned}$$

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

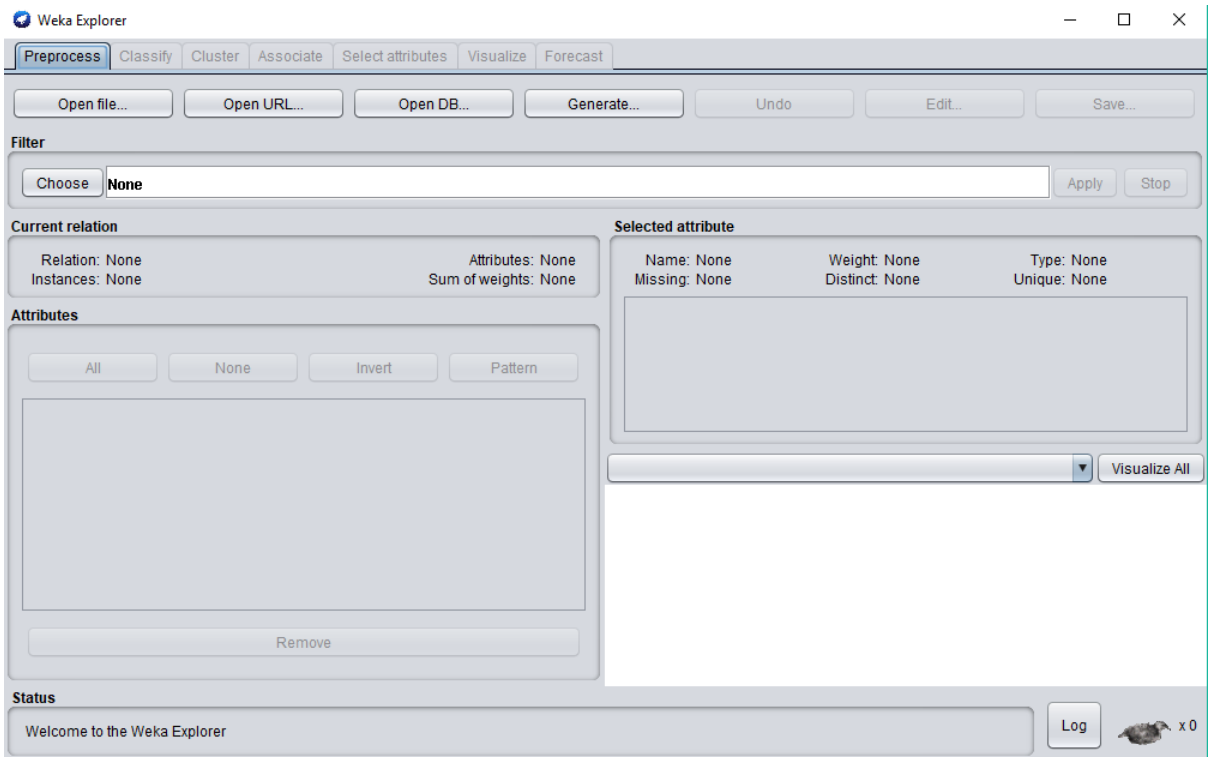
In general, **association rule mining** can be viewed as a two-step process:

1. **Find all frequent itemsets:**
   By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, min sup.
2. **Generate strong association rules from the frequent itemsets:**
   By definition, these rules must satisfy minimum support and minimum confidence.

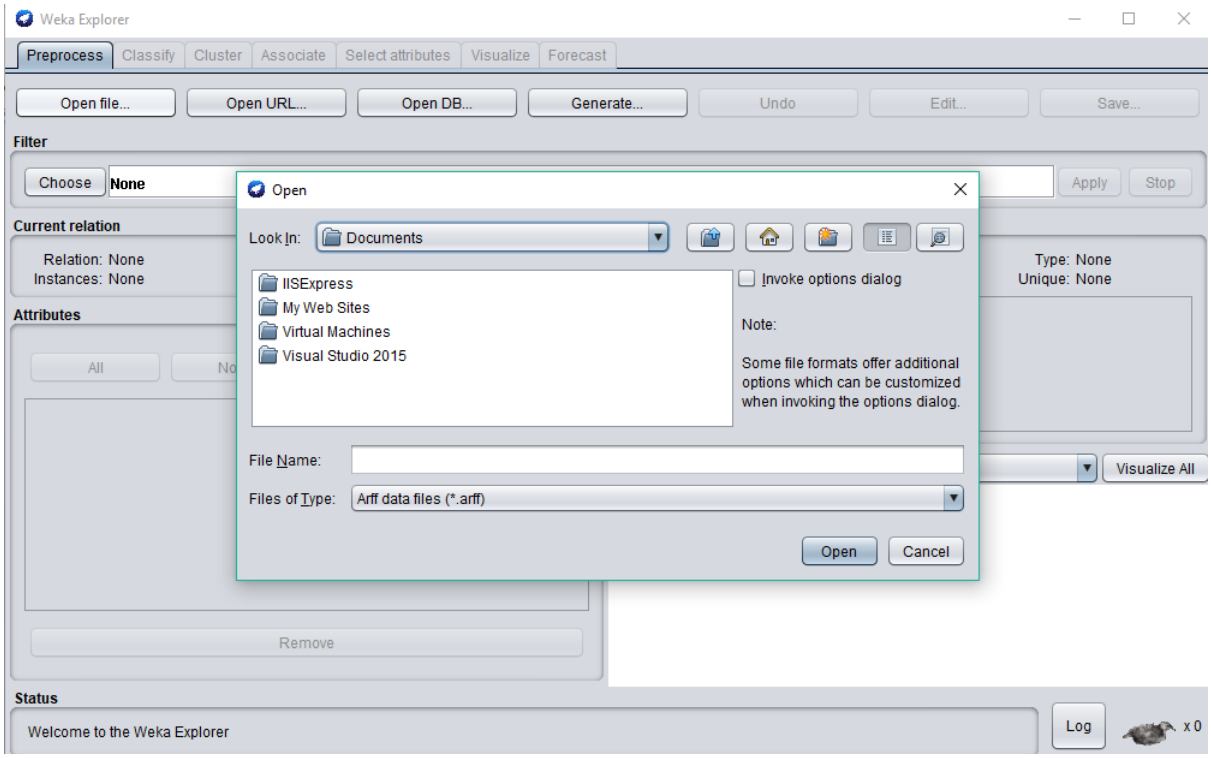
Step 1: Open Weka



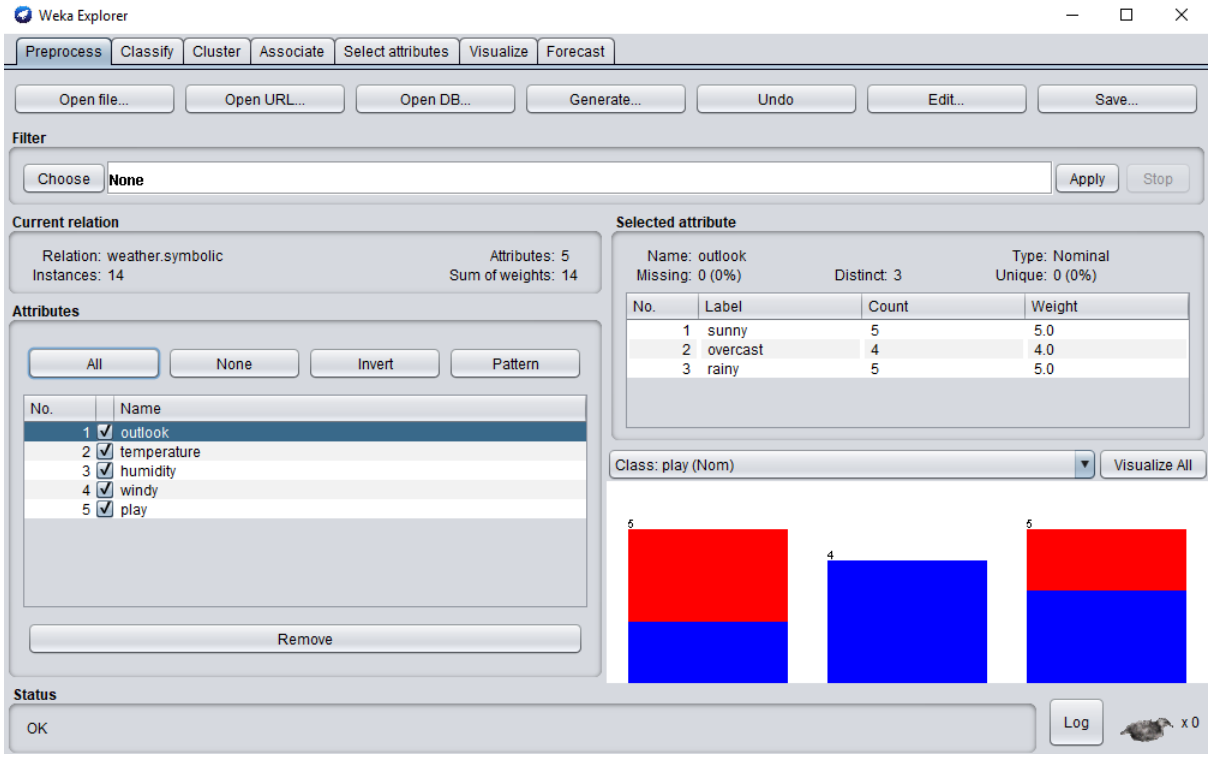
Step 2: Click on Explorer



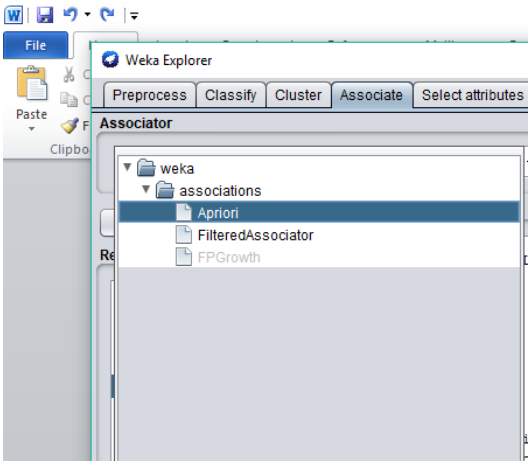
Step 3: Click on Open File Option



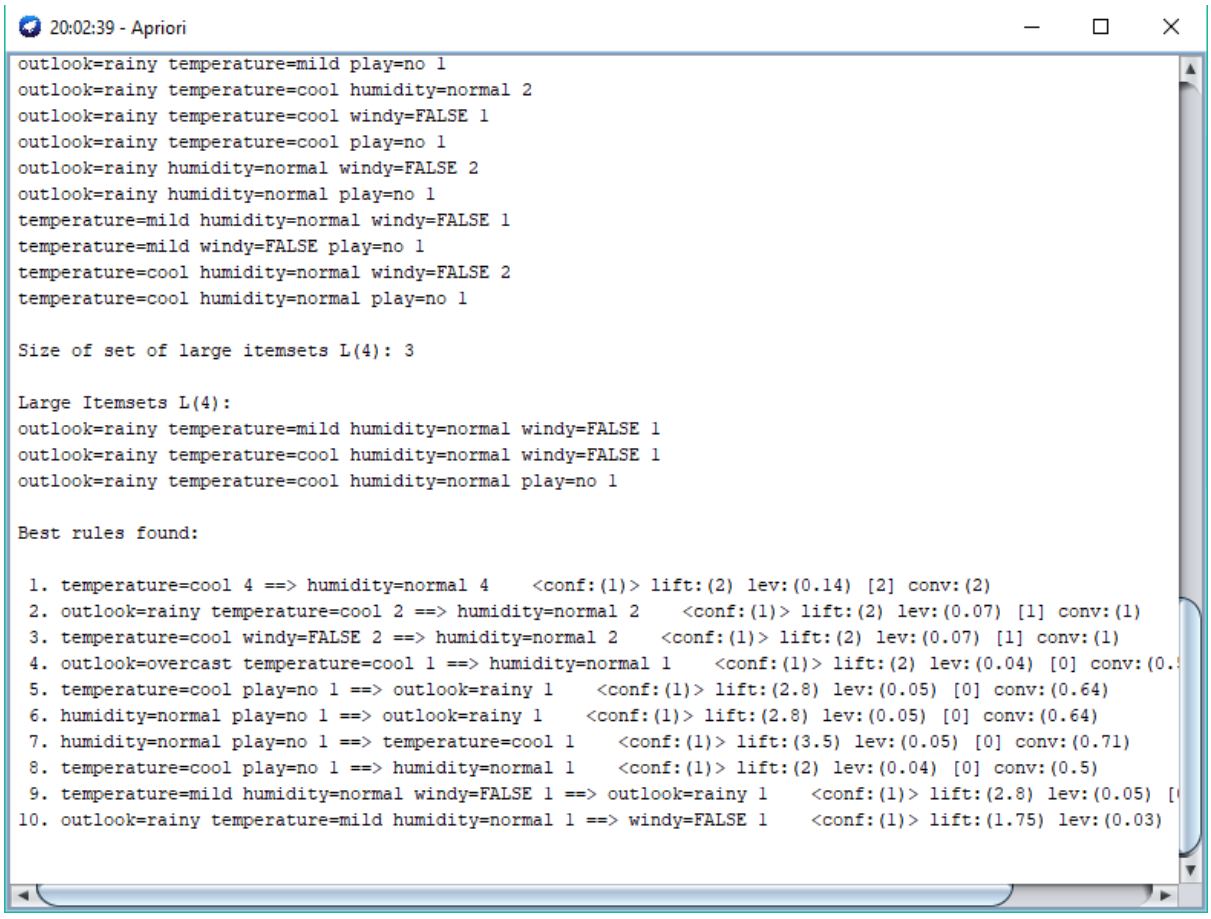
Step 4: Go to C://Program Files/Weka-3-8/data. Select weather.nomial.arff database. Select all the attributes.



Step 5: Go to Associate tab and Choose Apriori Algorithm



Step 6: Right Click on created structure and select View in separate window



Practical 6:

Aim: Using R-Tool , show the analysis for social networking sites.

Theory:

Social network analysis is based on an assumption of the importance of relationships among interacting units.

The social network perspective encompasses theories, models, and applications that are expressed in terms of relational concepts or processes.

Along with growing interest and increased use of network analysis has come a consensus about the central principles underlying the network perspective. In addition to the use of relational concepts, we note the following as being important:

- Actors and their actions are viewed as interdependent rather than independent, autonomous units
- Relational ties (linkages) between actors are channels for transfer or "flow" of resources (either material or nonmaterial)
- Network models focusing on individuals view the network structural environment as providing opportunities for or constraints on individual action
- Network models conceptualize structure (social, economic, political, and so forth) as lasting patterns of relations among actors

The unit of analysis in network analysis is not the individual, but an entity consisting of a collection of individuals and the linkages among them.

Network methods focus on dyads (two actors and their ties), triads (three actors and their ties), or larger systems (subgroups of individuals, or entire networks.

Social network analysis [SNA] is the mapping and measuring of relationships and flows between people, groups, organizations, computers or other information/knowledge processing entities.

The nodes in the network are the people and groups while the links show relationships or flows between the nodes. SNA provides both a visual and a mathematical analysis of complex human systems.

Procedure

The data to analyze is Twitter text dataand it can be downloaded as file "termDocMatrix.rdata" athttp://www.rdatamining.com/data .

1) Load Data

```
> # load termDocMatrix
> load("data/termDocMatrix.rdata")
> # inspect part of the matrix
> termDocMatrix[5:10,1:20]
```

```
           Docs

Terms       1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

data        1 1 0 0 2 0 0 0 0  0  1  2  1  1  1  0  1  0  0  0

examples    0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0

introduction 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  1

mining      0 0 0 0 0 0 0 0 0  0  0  1  1  0  1  0  0  0  0  0

network     0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  1  0  1  1  1

package     0 0 0 1 1 0 0 0 0  0  0  1  0  0  0  0  0  0  0  0
```

2) Transform Data into an Adjacency Matrix

```
> # change it to a Boolean matrix
> termDocMatrix[termDocMatrix>=1] <- 1
> # transform into a term-term adjacency matrix
> termMatrix <- termDocMatrix %*% t(termDocMatrix)
> # inspect terms numbered 5 to 10
> termMatrix[5:10,5:10]
```

```
           Terms

Terms       data examples introduction mining network package

data        53         5           2    34       0       7

examples    5         17           2     5       2       2

introduction 2         2          10     2       2       0
```

mining	34	5	2	47	1	5
network	0	2	2	1	17	1
package	7	2	0	5	1	21

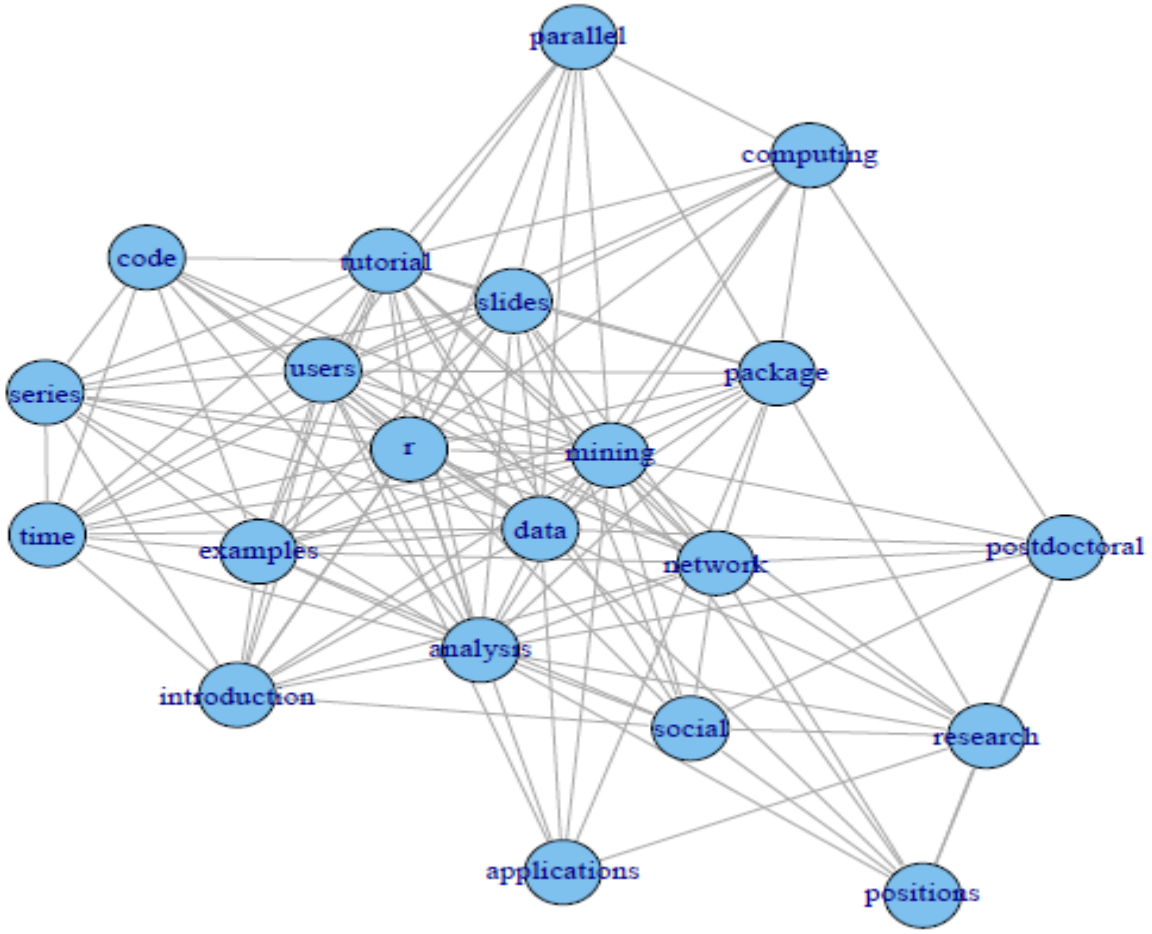
3)Build a Graph

Now we have built a term-term adjacency matrix, where the rows and columns represents terms, and every entry is the number of co-occurrences of two terms. Next we can build a graph with graph.adjacency() from package igraph.

```
> library(igraph)
> # build a graph from the above matrix
> g <- graph.adjacency(termMatrix, weighted=T, mode = “undirected”)
> # remove loops
> g <- simplify(g)
> # set labels and degrees of vertices
> V(g)$label <- V(g)$name
> V(g)$degree <- degree(g)
```

4) Plot a Graph

```
> # set seed to make the layout reproducible
> set.seed(3952)
> layout1 <- layout.fruchterman.reingold(g)
> plot(g, layout=layout1)
```



A different layout can be generated with the first line of code below. The second line produces an interactive plot, which allows us to manually rearrange the layout. Details about other layout options can be obtained by running ?igraph::layout in R.

```
> plot(g, layout=layout.kamada.kawai)
> tkplot(g, layout=layout.kamada.kawai)
```

Practical 7:

Aim: Consider the suitable data for text mining and Implement the Text Mining technique using R-Tool

Theory:

Text mining, which is sometimes referred to "text analytics" is one way to make qualitative or "unstructured" data usable by a computer.

Qualitative data is descriptive data that cannot be measured in numbers and often includes qualities of appearance like color, texture, and textual description. Quantitative data is numerical, structured data that can be measured. However, there is often slippage between qualitative and quantitative categories. For example, a photograph might traditionally be considered "qualitative data" but when you break it down to the level of pixels, which can be measured.

Text mining is defined as the process or practice of examining large collections of written resources in order to generate new information, typically using specialized computer software. It is a subset of the larger field of data mining.

Some applications of text-mining include:

- Enterprise Business Intelligence/Data Mining, Competitive Intelligence
- E-Discovery, Records Management
- National Security/Intelligence
- Scientific discovery, especially Life Sciences
- Search/Information Access
- Social media monitoring

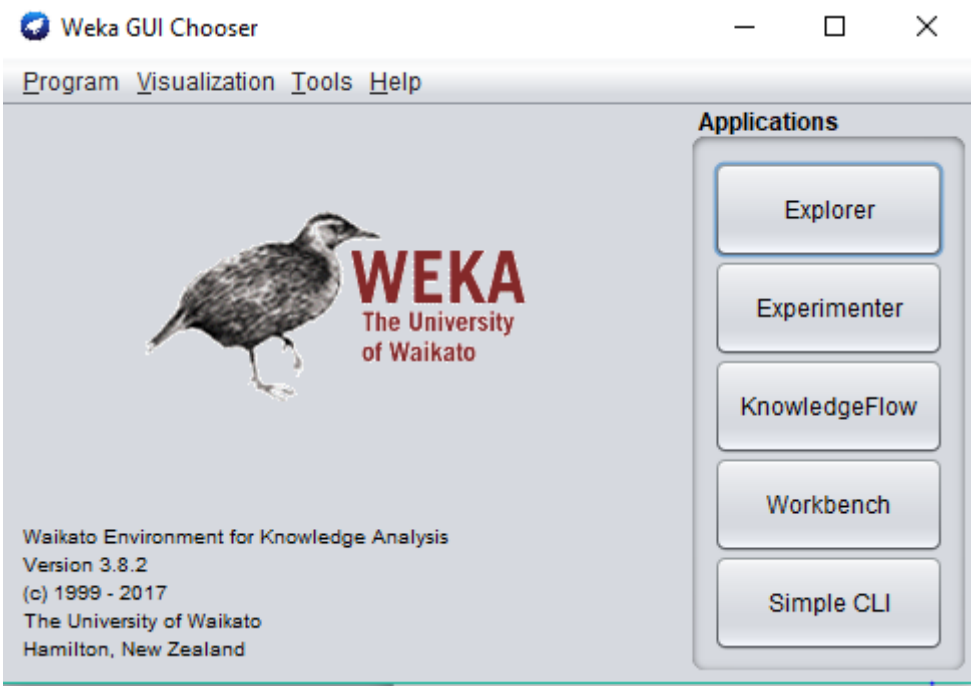
"Text mining involves the application of techniques from areas such as information retrieval, natural language processing, information extraction and data mining. These various stages of a text-mining process can be combined into a single workflow".

- **Information retrieval (IR) systems** match a user's query to documents in a collection or database. The first step in the text mining process is to find the body of documents that are relevant to the research question(s).

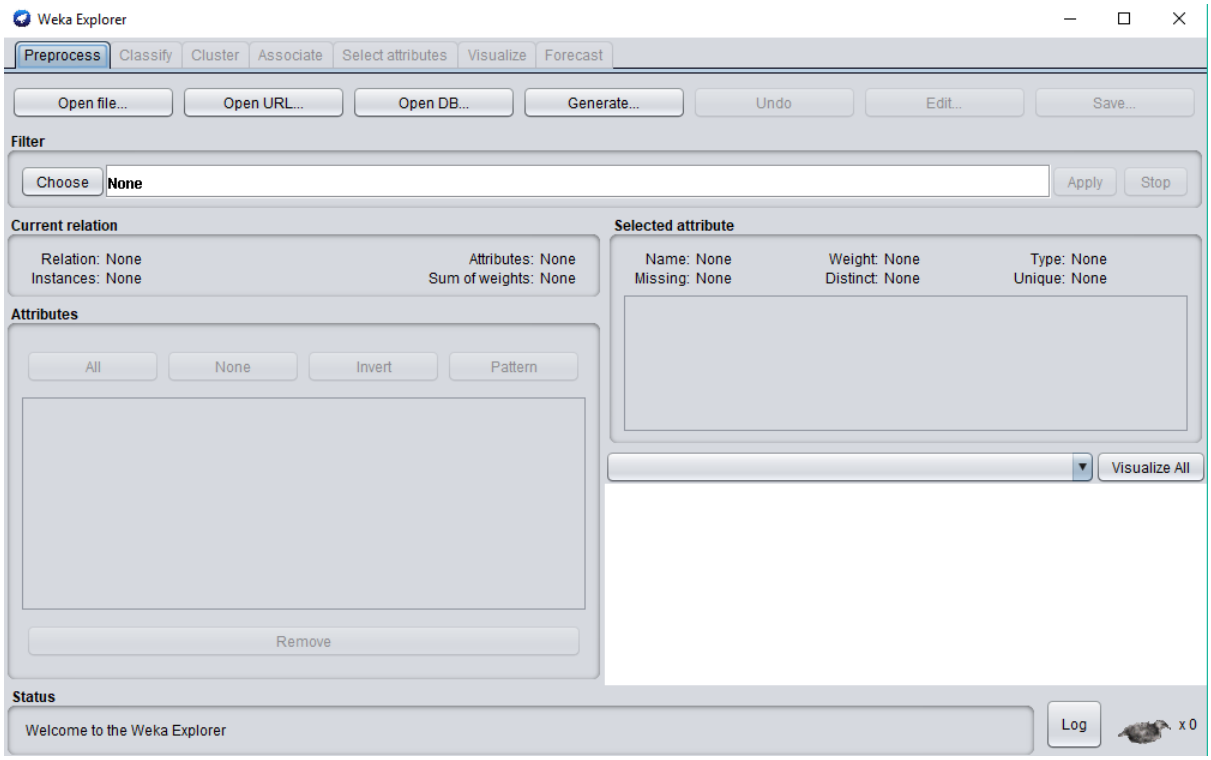
- **Natural language processing (NLP)** analyzes the text in structures based on human speech. It allows the computer to perform a grammatical analysis of a sentence to "read" the text.
- **Information extraction (IE)** involves structuring the data that the NLP system generates.
- **Data mining (DM) i**s the process of identifying patterns in large sets of data, to find that new knowledge.

Practical

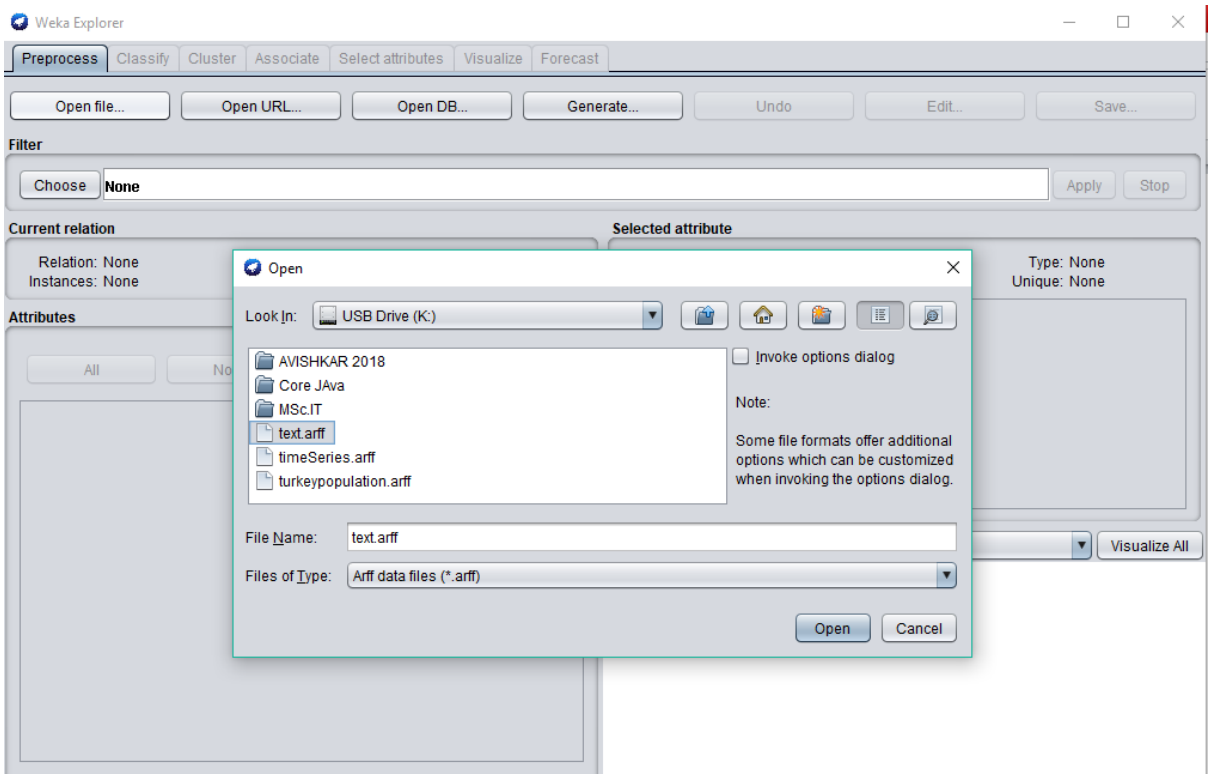
Step 1: Open Weka



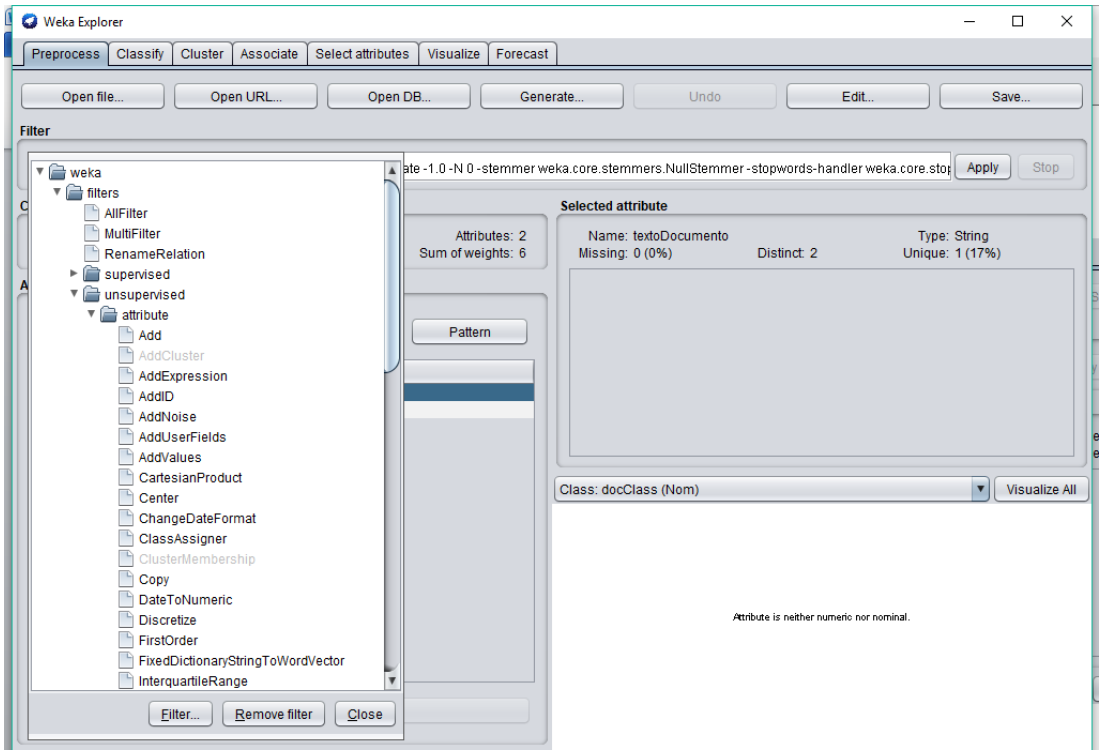
Step 2: Click on Explorer



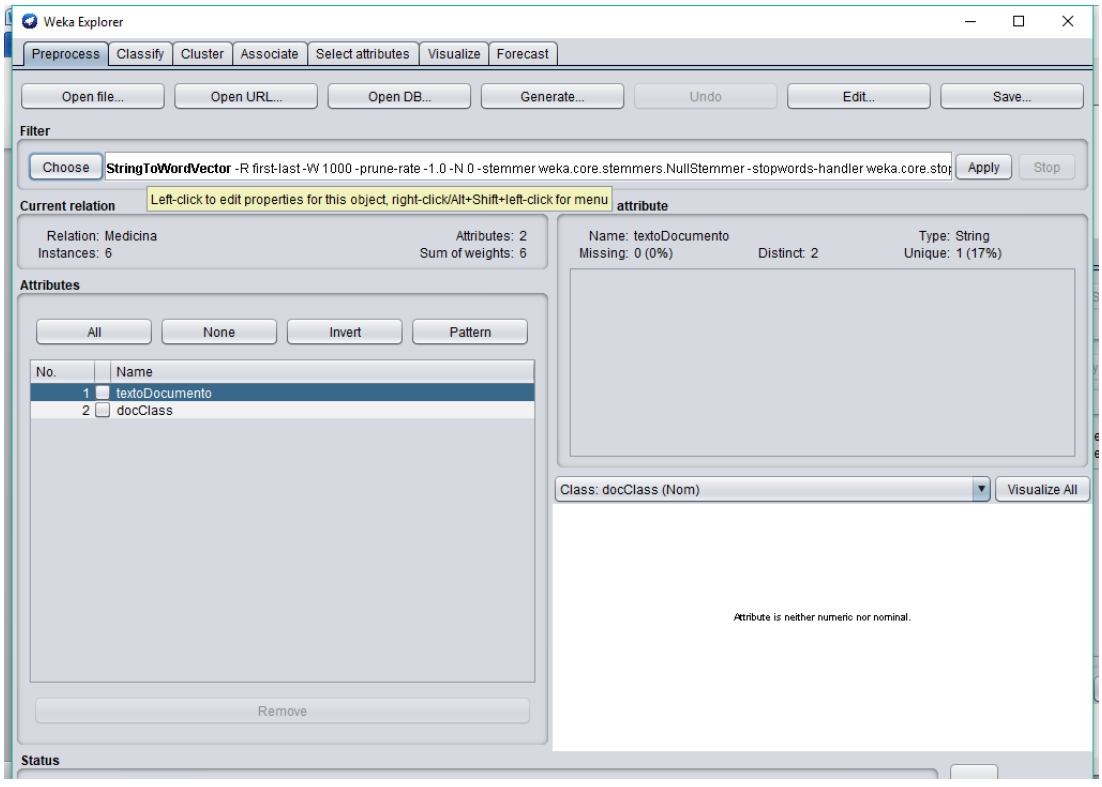
Step 3: Click on Open File Option and browse for created file



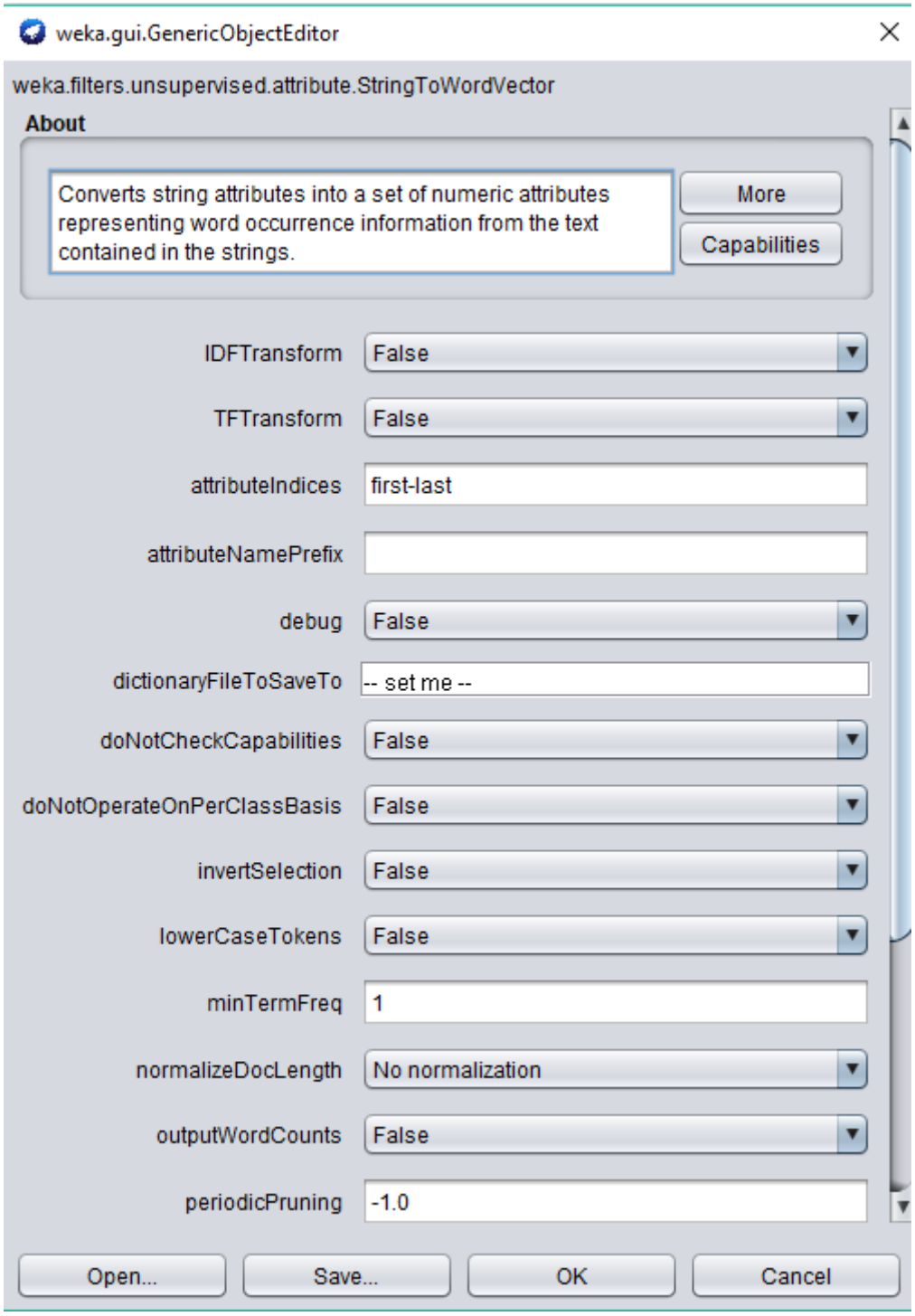
Step 4: Click on Choose button and Select Filter->Unsupervised Learning-> Attributes->StringtoVector option



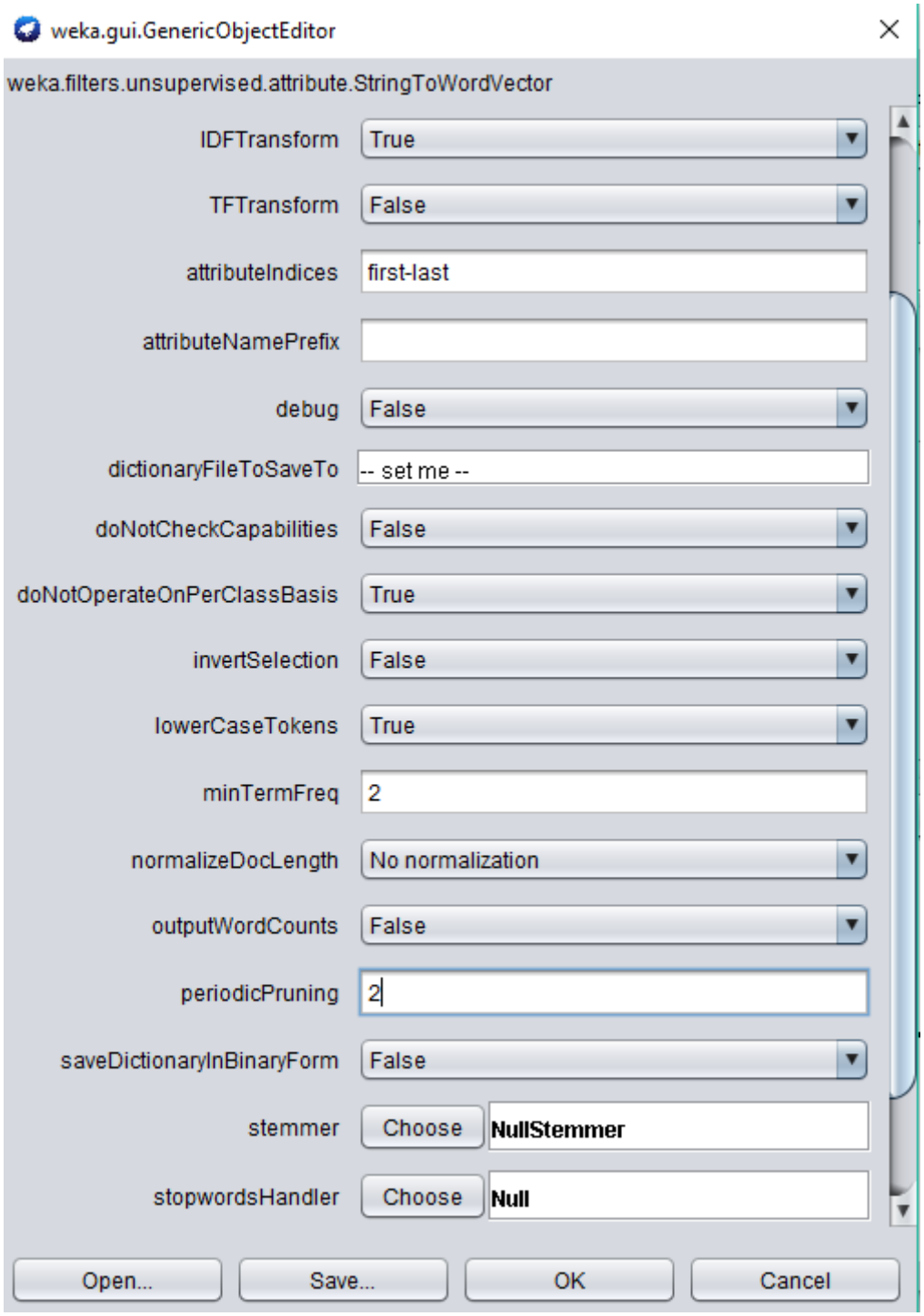
Step 5: Click on the StringtoVector



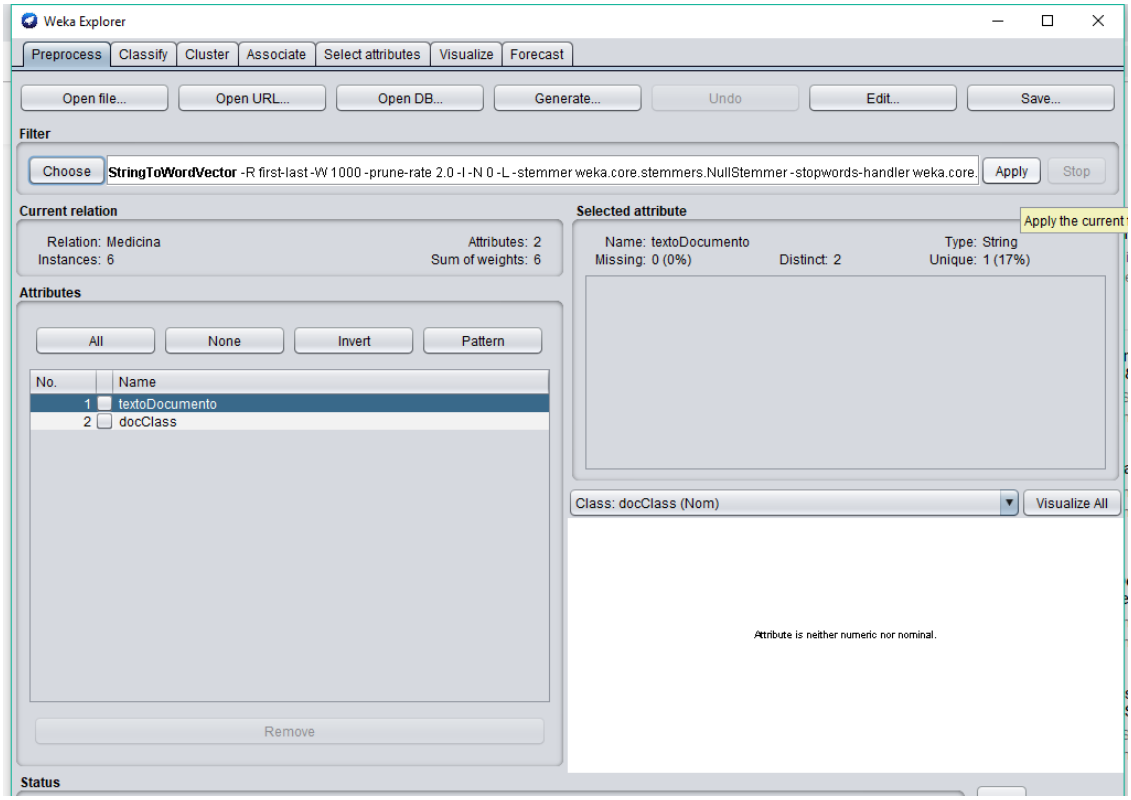
Step 6: ObjectEditor appears



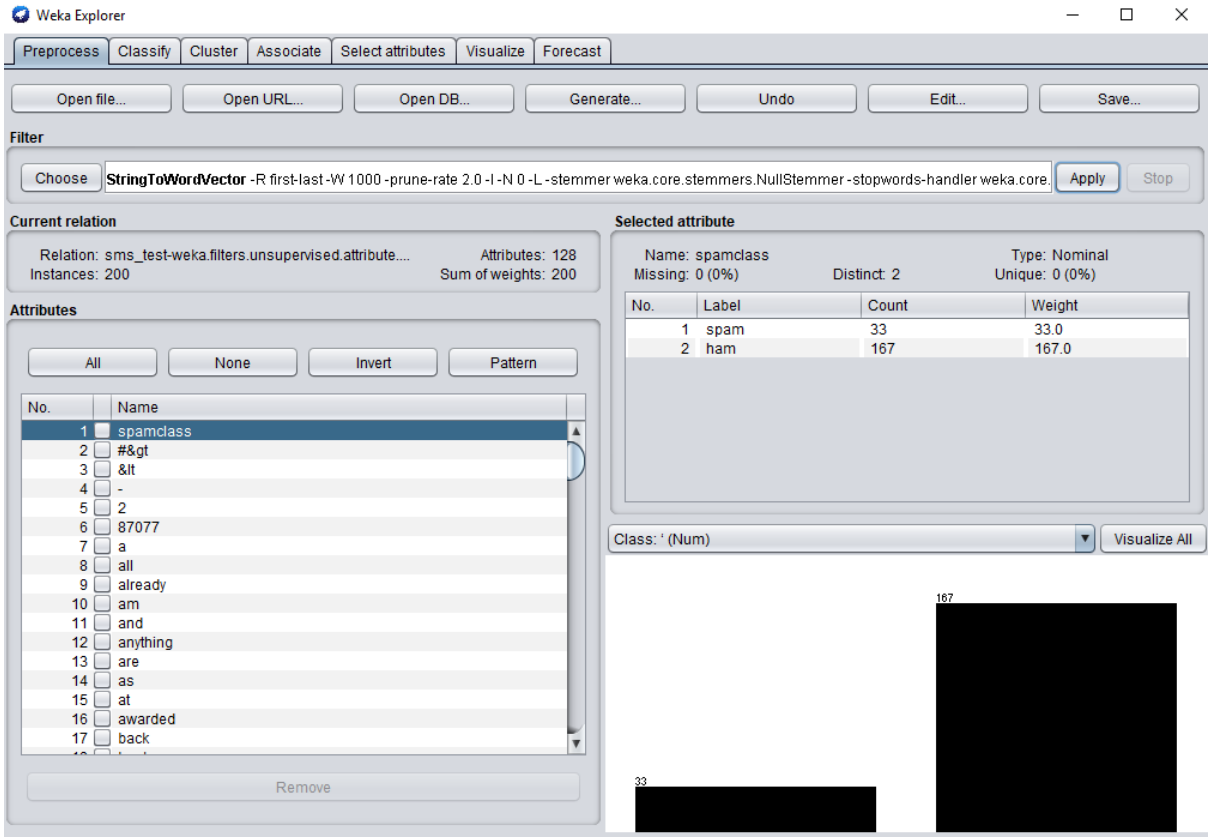
Step 7: Make the following changes



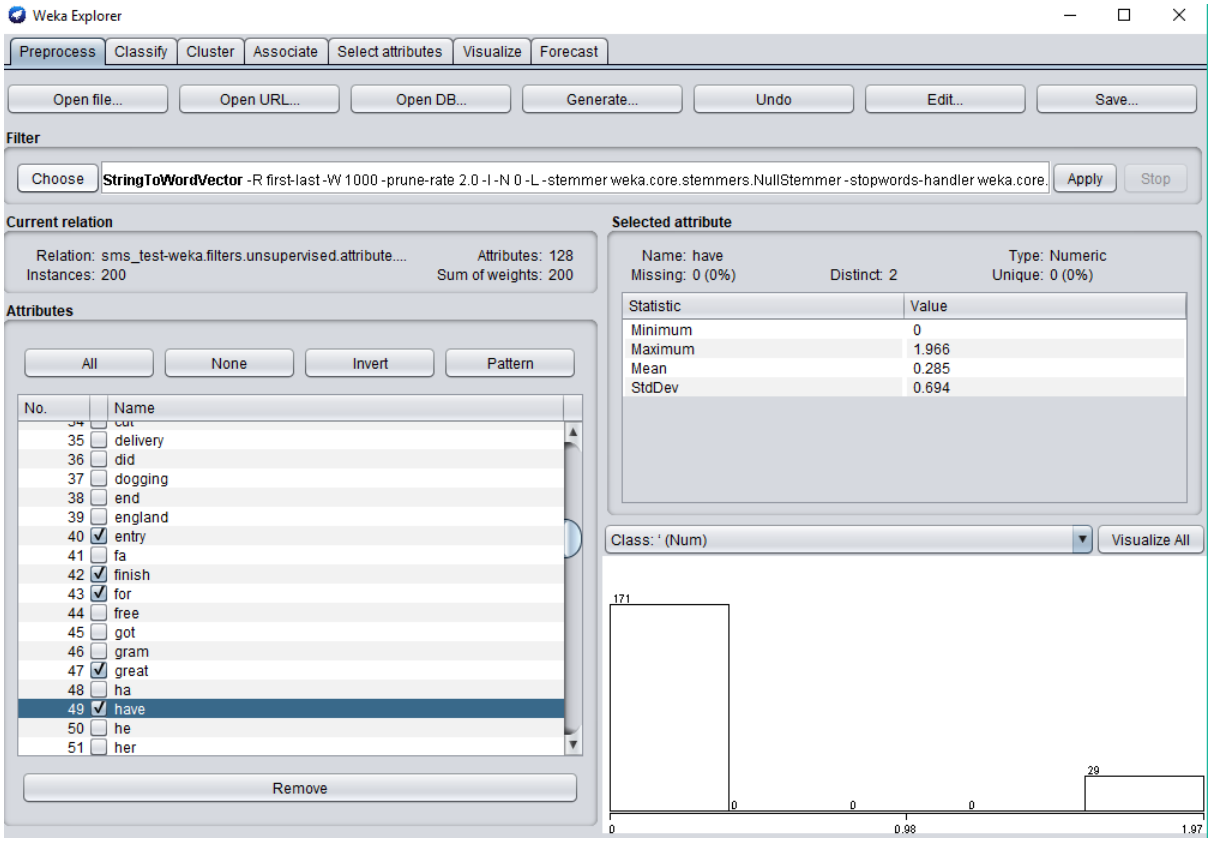
Step 8: Click on Apply



Step 9: Following will appear



Step 10: Select one or two attributes



Practical 8:

Aim: Prepare the Analysis services for Adventure Works Cycles or (any other database) . Build the data mining model and implement Apriori algorithm.

Theory:

Apriori is a seminal algorithm proposed by R.Agrawal and R.Srikant in 1994 formining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses prior knowledge of frequent itemset properties, as we shall see following. Apriori employs an iterative approach known as a level-wise search, where k-itemsets are used to explore(k+1)-itemsets.

To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property, presented below, is used to reduce the search space. We will first describe this property, and then show an example illustrating its use.

Apriori property: All nonempty subsets of a frequent itemset must also be frequent.

A two-step process is followed, consisting of join and prune actions.

The join step: To find Lk, a set of candidate k-itemsets is generated by joining Lk−1 with itself. This set of candidates is denoted Ck.

The prune step : Ck is a super set of Lk , that is , its members may or may not be frequent, but all of the frequentk-itemsets are included in Ck. A scan of the database to determine the count of each candidate in Ck would result in the determination of Lk

Algorithm: Apriori Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- min_sup, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
(1)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ ϕ; k++) {
(3)      Ck = apriori_gen(Lk−1);
(4)      for each transaction t ∈ D { // scan D for counts
(5)         Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)         for each candidate c ∈ Ct
(7)            c.count++;
(8)      }
(9)      Lk = {c ∈ Ck | c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;
```

```
procedure apriori_gen(L_{k-1}:frequent (k-1)-itemsets)
(1)     for each itemset l_1 ∈ L_{k-1}
(2)         for each itemset l_2 ∈ L_{k-1}
(3)             if (l_1[1] = l_2[1]) ∧ (l_1[2] = l_2[2]) ∧ ... ∧ (l_1[k-2] = l_2[k-2]) ∧ (l_1[k-1] < l_2[k-1]) then {
(4)                 c = l_1 ⋈ l_2; // join step: generate candidates
(5)                 if has_infrequent_subset(c, L_{k-1}) then
(6)                     delete c; // prune step: remove unfruitful candidate
(7)                 else add c to C_k;
(8)             }
(9)     return C_k;

procedure has_infrequent_subset(c: candidate k-itemset;
            L_{k-1}: frequent (k-1)-itemsets); // use prior knowledge
(1)     for each (k-1)-subset s of c
(2)         if s ∉ L_{k-1} then
(3)             return TRUE;
(4)     return FALSE;
```

Procedure:

The R package “arules” is used for for implementing Apriori Algorithm.After the user installed the necessaries packages, he must load them. This can be done using the function "library(*package name*)".

Reading the Data:

Transactions can be read from files in the basket format, with the command read.transactions.

```
>tr<-read.transactions("c:/rpracs/testfile.txt",format="basket",sep=",")
```

The object ”tr” is used to store the transactions read from the file named ”testfile”, where each item is separated by a ”,”. ”testfile” could be, for example:

```
A,B,C
B,C
A,B,D
A,B,C,D
A
B
```

One way to visualize the data is inspect(object). For example:

```
>inspect(tr)
```

```
items
```

```
1 {A,
   B,
   C}
2 {B,
   C}
3 {A,
   B,
   D}
4 {A,
   B,
   C,
   D}
5 {A}
6 {B}
```

Additionally, you can visually inspect binary incidence matrices, or plot the frequency of items sets:

>image**(tr)**
>itemFrequencyPlot**(tr**, support = 0.1**)**

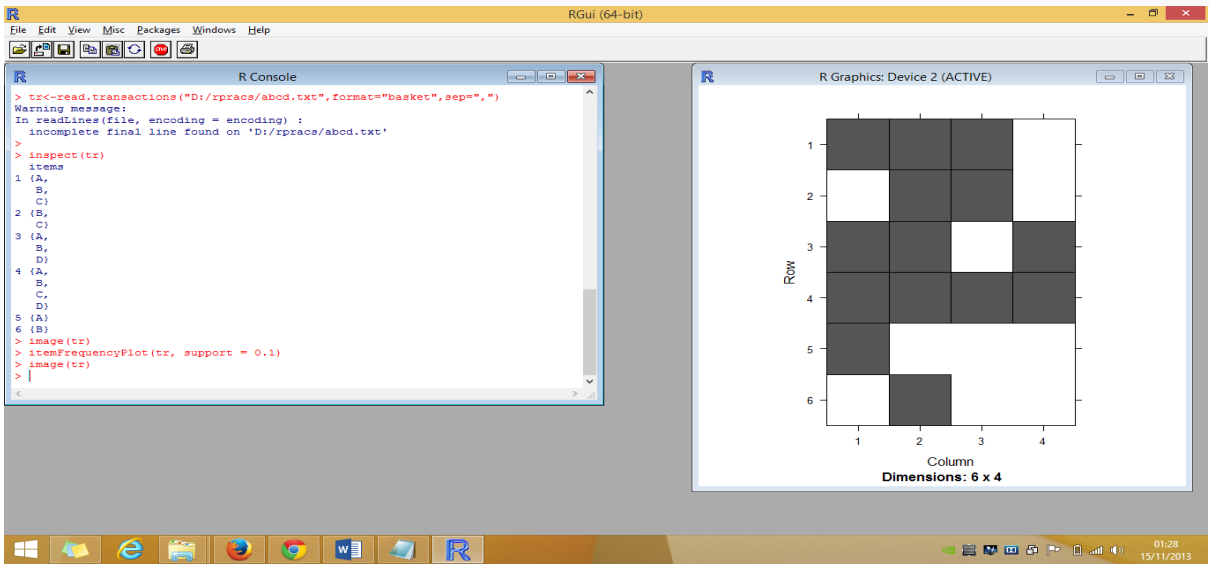


Figure:Output of >image**(tr)**

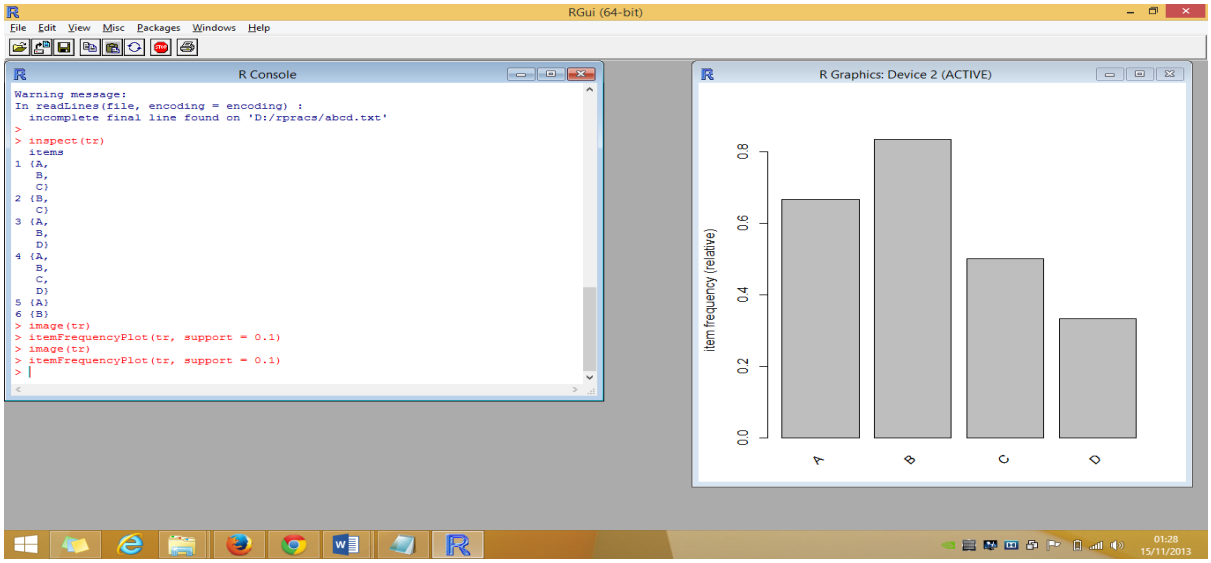


Figure:Output of>itemFrequencyPlot**(tr**, support = 0.1**)**

To show the number of items in transactions read from the file named ”teste” do:

>length**(tr)**
[1] 6

Rules:

The function to mine frequent itemsets, association rules or association hyperedges, using the Apriori algorithm, takes 2 parameters:

Data: the object that contains the data

parameter: a multi-dimensional parameter to set up support and confidence

For example, using the dataset gathered in the previous section:

>rules <- apriori**(tr**, parameter= list**(**supp=0.5, conf=0.5**))**

The rules can be visualized with the command inspect:

>inspect**(**rules**)**

```
    rhs    support confidence lift
1 {} => {C} 0.5000000 0.5000000 1.0
2 {} => {A} 0.6666667 0.6666667 1.0
3 {} => {B} 0.8333333 0.8333333 1.0
4 {C} => {B} 0.5000000 1.0000000 1.2
5 {B} => {C} 0.5000000 0.6000000 1.2
6 {A} => {B} 0.5000000 0.7500000 0.9
7 {B} => {A} 0.5000000 0.6000000 0.9
```

To get a summary of the rules' characteristics, the function "summary" can be used:

```
>summary(rules)
```

```
set of 7 rules

rule length distribution (lhs + rhs):sizes
1 2
3 4

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   2.000   1.571   2.000   2.000

summary of quality measures:
    support         confidence          lift
 Min.   :0.5000   Min.   :0.5000   Min.   :0.900
 1st Qu.:0.5000   1st Qu.:0.6000   1st Qu.:0.950
 Median :0.5000   Median :0.6667   Median :1.000
 Mean   :0.5714   Mean   :0.7071   Mean   :1.029
 3rd Qu.:0.5833   3rd Qu.:0.7917   3rd Qu.:1.100
 Max.   :0.8333   Max.   :1.0000   Max.   :1.200

mining info:
 data ntransactions support confidence
   tr            6     0.5        0.5
```

Outher quality measures of the rules can be displaed with:

```
>interestMeasure(rules, c("support", "chiSquare", "confidence", "conviction",
"cosine", "coverage", "leverage", "lift", "oddsRatio"), tr)
```

```
    support chiSquare confidence conviction    cosine  coverage    leverage
1 0.5000000       NaN  0.5000000  1.0000000 0.7071068 1.0000000  0.00000000
2 0.6666667       NaN  0.6666667  1.0000000 0.8164966 1.0000000  0.00000000
3 0.8333333       NaN  0.8333333  1.0000000 0.9128709 1.0000000  0.00000000
4 0.5000000       1.2  1.0000000        Inf 0.7745967 0.5000000  0.08333333
5 0.5000000       1.2  0.6000000  1.2500000 0.7745967 0.8333333  0.08333333
6 0.5000000       0.6  0.7500000  0.6666667 0.6708204 0.6666667 -0.05555556
7 0.5000000       0.6  0.6000000  0.8333333 0.6708204 0.8333333 -0.05555556
  lift oddsRatio
1  1.0       NaN
2  1.0       NaN
3  1.0       NaN
4  1.2       Inf
5  1.2       Inf
6  0.9         0
```

7 0.9 0

To calculate a single measure and add it to the quality slot:

```
>quality(rules)<- cbind(quality(rules), hyperConfidence =interestMeasure(rules,
method = "hyperConfidence", a))
>inspect(head(SORT(rules, by = "hyperConfidence")))
```

```
lhs    rhs   support confidence lift hyperConfidence
1 {C} => {B} 0.5000000  1.0000000  1.2       0.9789272
2 {B} => {C} 0.5000000  0.6000000  1.2       0.9789272
3 {}  => {C} 0.5000000  0.5000000  1.0       0.0000000
4 {}  => {A} 0.6666667  0.6666667  1.0       0.0000000
5 {}  => {B} 0.8333333  0.8333333  1.0       0.0000000
6 {A} => {B} 0.5000000  0.7500000  0.9       0.0000000
```

Finally, to send the output to a file use:

```
>sink("sink-examp.txt")
>inspect(head(SORT(rules, by = "hyperConfidence")))
```