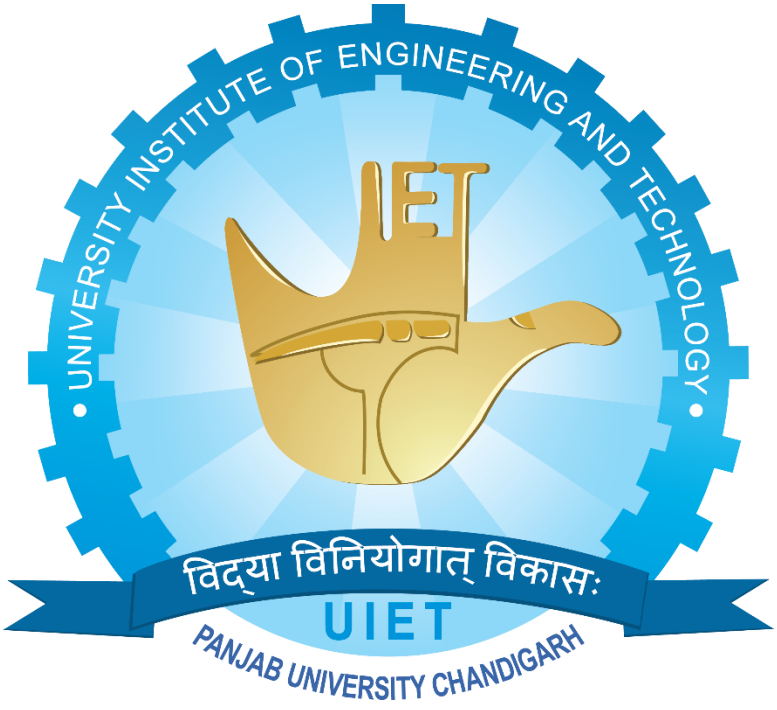
DATA WAREHOUSE AND DATA MINING

PRACTICLE FILE



SUBMITTED BY: Hitesh Chawla, BE IT 6th Sem

ROLL NO: UE198046

SUBMITTED TO: MS VEENU MANGAT (Professor, IT)

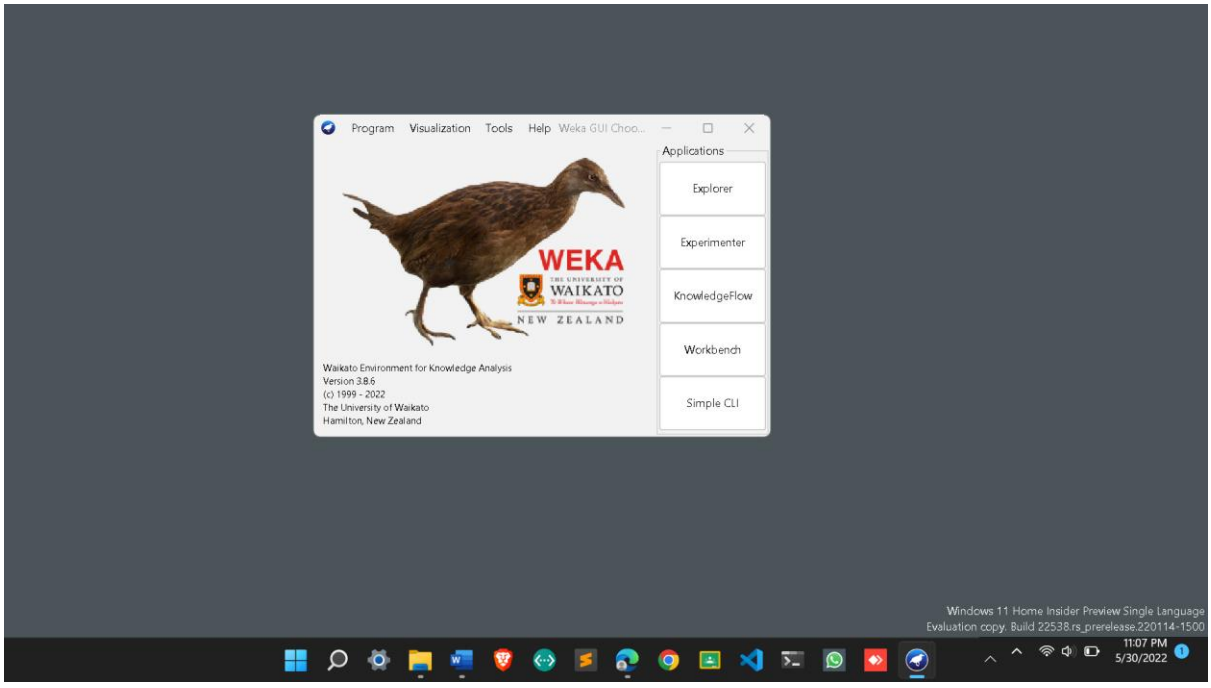
PRACTICLE 1: INTRODUCTION TO WEKA

To install WEKA on your machine, visit WEKA's official website and download the installation file. WEKA supports installation on Windows, Mac OS X and Linux. You just need to follow the instructions on this page to install WEKA for your OS.

The steps for installing as follows –

- Download the installation file.
- Double click on the downloaded **.exe file**.

You will see the following screen on successful installation.



The WEKA GUI Chooser application will start and you would see the following screen –

The GUI Chooser application allows you to run five different types of applications as listed here –

- Explorer
- Experimenter
- KnowledgeFlow
- Workbench
- Simple CLI

On the top, you will see several tabs as listed here −

- Preprocess
- Classify
- Cluster
- Associate
- Select Attributes
- Visualize

Under these tabs, there are several pre-implemented machine learning algorithms. Let us look into each of them in detail now.

Preprocess Tab

Initially as you open the explorer, only the **Preprocess** tab is enabled. The first step in machine learning is to preprocess the data. Thus, in the **Preprocess** option, you will select the data file, process it and make it fit for applying the various machine learning algorithms.

Classify Tab

The **Classify** tab provides you several machine learning algorithms for the classification of your data. To list a few, you may apply algorithms such as Linear Regression, Logistic Regression, Support Vector Machines, Decision Trees, RandomTree, RandomForest, NaiveBayes, and so on. The list is very exhaustive and provides both supervised and unsupervised machine learning algorithms.

Cluster Tab

Under the **Cluster** tab, there are several clustering algorithms provided - such as SimpleKMeans, FilteredClusterer, HierarchicalClusterer, and so on.

Associate Tab

Under the **Associate** tab, you would find Apriori, FilteredAssociator and FPGrowth.

Select Attributes Tab

Select Attributes allows you feature selections based on several algorithms such as ClassifierSubsetEval, PrinicipalComponents, etc.

PRACTICLE 2: ASSOCIATION RULE IN DATA MINING

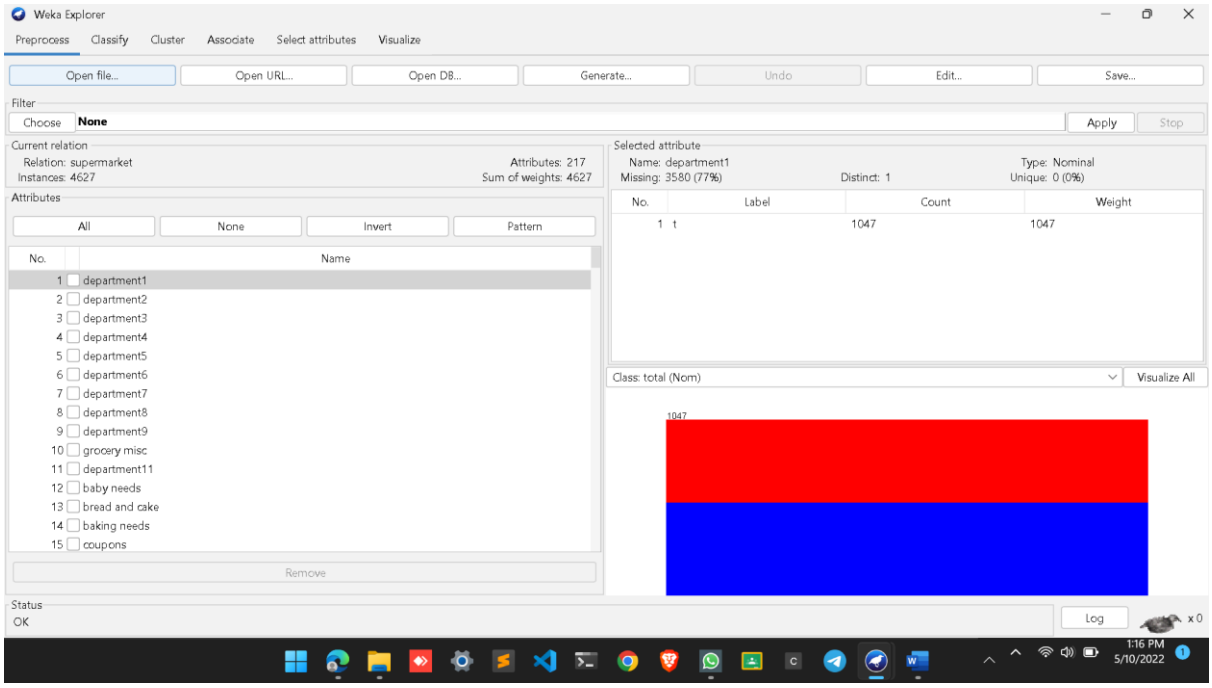
Dataset – Market.arff

Source – https://raw.githubusercontent.com/renatopp/arff-datasets

Reasons for choosing the data set:

- Relatable data set with knowledge of all attributes.
- Complete data with low inconsistency

STEP 1: In the WEKA explorer, open the Preprocess tab, click on the Open file … button and select market.arff database from the installation folder. After the data is loaded you will see the following screen –



I've implemented the two algorithms: Apriori & FP Growth. The database contains 4627 instances and 217 attributes. The data is denormalized. Each attribute is binary and either has a value ("t" for true) or no value ("?" for missing). There is a nominal class attribute called "total" that indicates whether the transaction was less than $100 (low) or greater than $100 (high).

STEP 2: Left click the field of Associator, choose Show Property from the drop down list. The property window of Apriori opens. Weka runs an Apriori-type algorithm to find association rules, but this algorithm is not exact the same one as we discussed in class.

a. The min. support is not fixed. This algorithm starts with min. support as upperBoundMinSupport (default 1.0 = 100%), iteratively decrease it by delta (default 0.05 = 5%). Note that upperBoundMinSupport is decreased by delta before the basic Apriori algorithm is run for the first time.

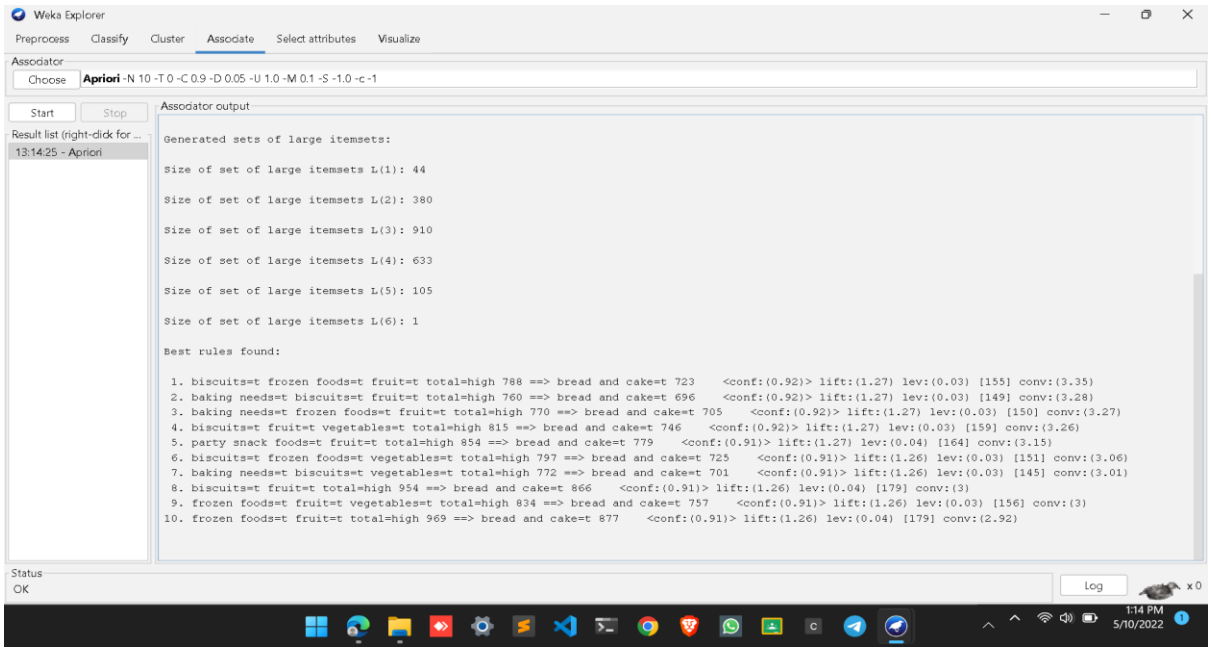
b. The algorithm stops when lowerBoundMinSupport (default 0.1 = 10%) is reached, or required number of rules – numRules (default value 10) have been generated.

c. Rules generated are ranked by metricType (default Confidence). Only rules with score higher than minMetric (default 0.9 for Confidence) are considered and delivered as the output.

d. If you choose to show the all frequent itemsets found, outputItemSets should be set as True.

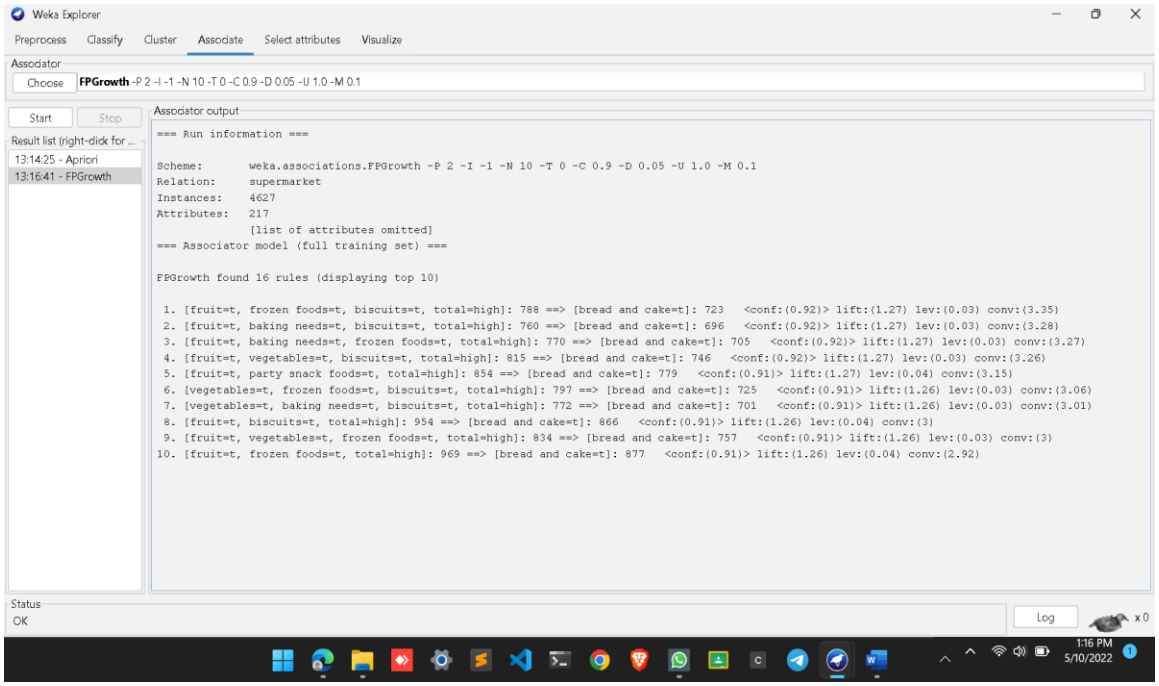
NOTE: The Algorithm starts with min. support as 100% and stops at 15% after running 17 times

STEP 3: Click Start button on the left of the window, the algorithm begins to run. The output is showing in the right window.



Apriori Algorithm Implementation

STEP 4: Left click the field of Associator, choose Show Property from the drop down list. The property window of FP Growth opens. Start it.



FP Growth Implementation

The rules discovered where:

1. biscuits=t frozen foods=t fruit=t total=high 788 ==> bread and cake=t 723 conf:(0.92)
2. baking needs=t biscuits=t fruit=t total=high 760 ==> bread and cake=t 696 conf:(0.92)
3. baking needs=t frozen foods=t fruit=t total=high 770 ==> bread and cake=t 705 conf:(0.92)
4. biscuits=t fruit=t vegetables=t total=high 815 ==> bread and cake=t 746 conf:(0.92)
5. party snack foods=t fruit=t total=high 854 ==> bread and cake=t 779 conf:(0.91)
6. biscuits=t frozen foods=t vegetables=t total=high 797 ==> bread and cake=t 725 conf:(0.91)
7. baking needs=t biscuits=t vegetables=t total=high 772 ==> bread and cake=t 701 conf:(0.91)
8. biscuits=t fruit=t total=high 954 ==> bread and cake=t 866 conf:(0.91)
9. frozen foods=t fruit=t vegetables=t total=high 834 ==> bread and cake=t 757 conf:(0.91)
10. frozen foods=t fruit=t total=high 969 ==> bread and cake=t 877 conf:(0.91)

Observations

1. We can see that all presented rules have a consequent of “bread and cake”.
2. All presented rules indicate a high total transaction amount.
3. “biscuits” an “frozen foods” appear in many of the presented rules.
4. For example, we might want to convince people that buy biscuits, frozen foods and fruit to buy bread and cake so that they result in a high total transaction amount (Rule #1). This may sound plausible, but is flawed reasoning.
5. The product combination does not cause a high total, it is only associated with a high total. Those 723 transactions may have a vast assortment of random items in addition to those in the rule.

PRACTICLE 3: CLASSIFICATION RULE IN DATA MINING

Classification: It is a data analysis task, i.e., the process of finding a model that describes and distinguishes data classes and concepts. Classification is the problem of identifying to which of a set of categories (subpopulations), a new observation belongs to, on the basis of a training set of data containing observations and whose categories membership is known.

Example: Before starting any project, we need to check its feasibility. In this case, a classifier is required to predict class labels such as 'Safe' and 'Risky' for adopting the Project and to further approve it. It is a two-step process such as:

1. **Learning Step (Training Phase)**: Construction of Classification Model
   Different Algorithms are used to build a classifier by making the model learn using the training set available. The model has to be trained for the prediction of accurate results.

2. **Step**: Model used to predict class labels and testing the constructed model on test data and hence estimate the accuracy of the classification rules.

Dataset – Wild_Animals.arff

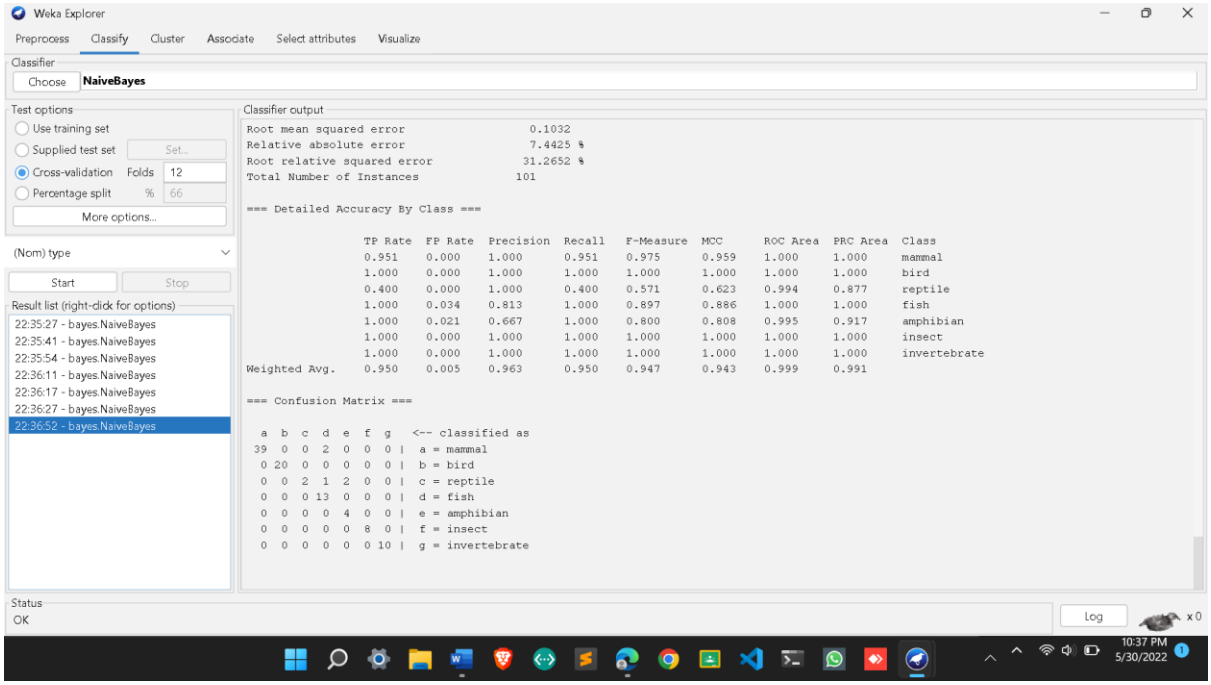
Source – https://raw.githubusercontent.com/renatopp/arff-datasets/master/classification/zoo.arff

STEP 1: Launch WEKA
2. Choose Explorer interface
3. Open file wild_animals.arff

STEP 2: 1. In Classify tab choose weka->classifiers->bayes->naiveBayes
2. The instances to be classified should be supplied in .arff format. Create a new .arff file, similar to wild_animals.arff, with 1 instance (row): hairy,red,large,soft,safe (for example)

3. Use test option: supplied test set. From file menu select your file with an instance to be classified.

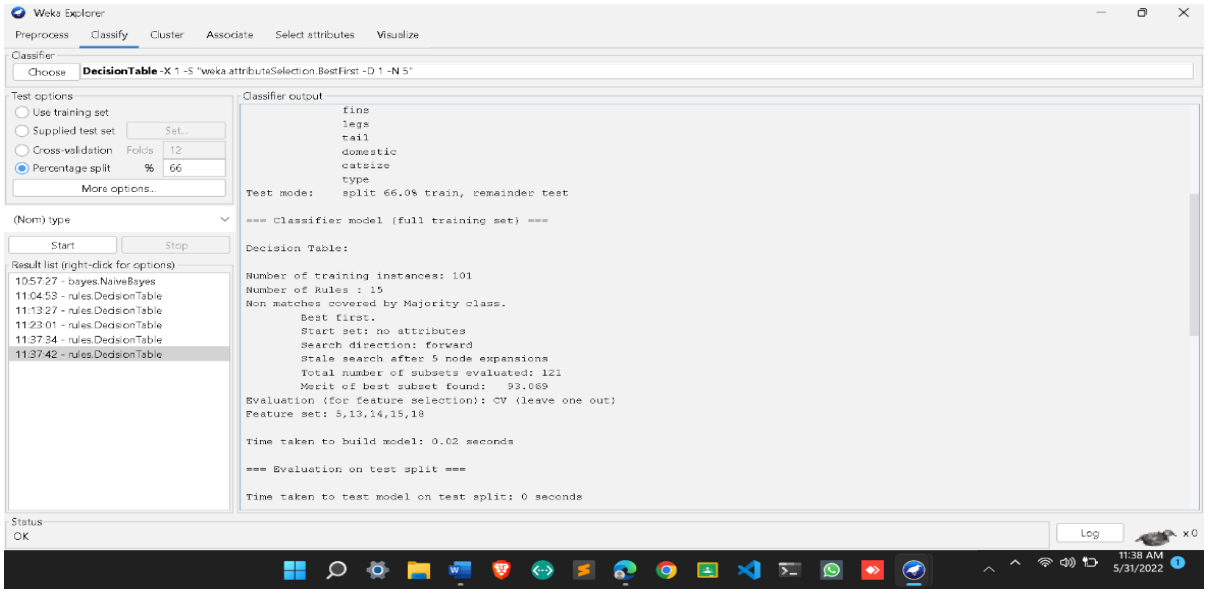
The results appear in the window on the left.



NaiveBayes Algorithm Implementation

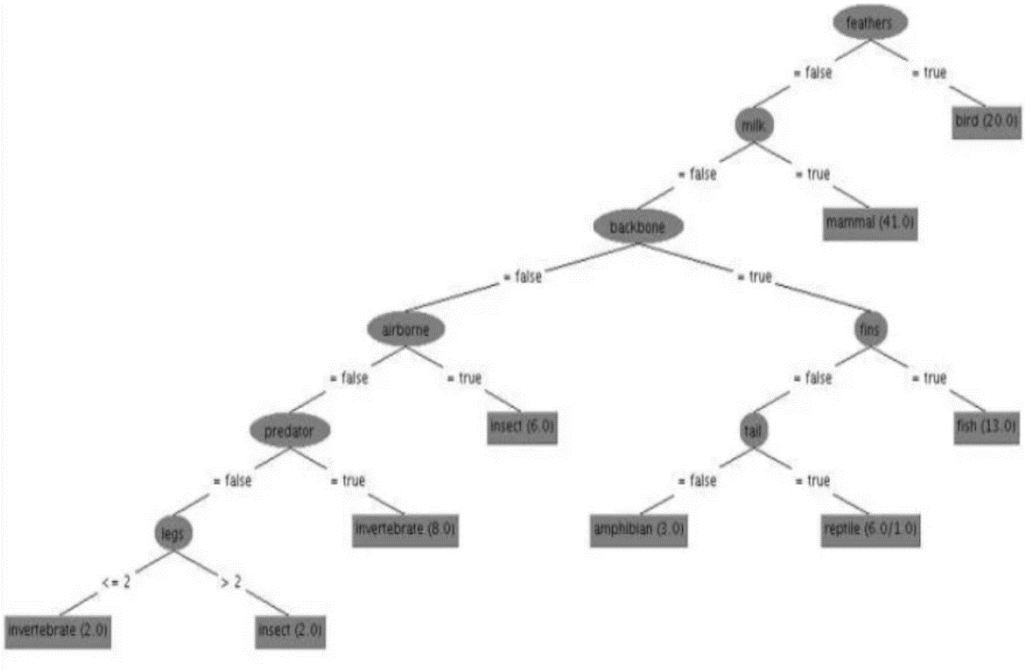
```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.951    0.000    1.000      0.951    0.975      0.959    1.000     1.000     mammal
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     bird
               0.400    0.000    1.000      0.400    0.571      0.623    0.994     0.877     reptile
               1.000    0.034    0.813      1.000    0.897      0.886    1.000     1.000     fish
               1.000    0.021    0.667      1.000    0.800      0.808    0.995     0.917     amphibian
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     insect
               1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     invertebrate
Weighted Avg.  0.950    0.005    0.963      0.950    0.947      0.943    0.999     0.991

=== Confusion Matrix ===

  a  b  c  d  e  f  g   <-- classified as
 39  0  0  2  0  0  0 |  a = mammal
  0 20  0  0  0  0  0 |  b = bird
  0  0  2  1  2  0  0 |  c = reptile
  0  0  0 13  0  0  0 |  d = fish
  0  0  0  0  4  0  0 |  e = amphibian
  0  0  0  0  0  8  0 |  f = insect
  0  0  0  0  0  0 10 |  g = invertebrate
```

STEP 3: Select Classifier tab, choose classifiers->Choose Decision Tables-> (implementation of decision tree algorithm). We will use 66% of animals to train the computer and 34% to evaluate the classifier. For this choose percentage split 66% option.



Decision Tree Algorithm

STEP 4: Build decision tree by clicking on run button. The result appears on the left and as the line in the history list. To see the tree, right-click on the line in the history list and choose visualize tree. In the tree window right-click and adjust the size of the tree using menu options.

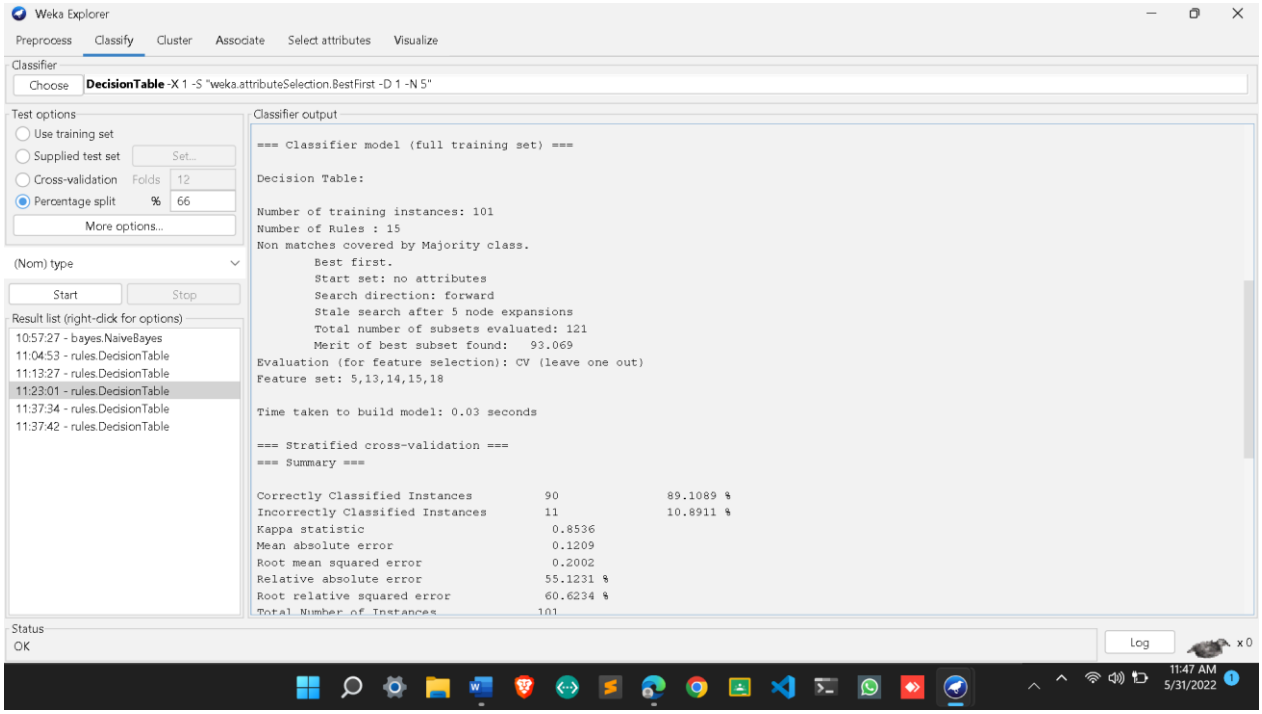


Decision Tree

Observations:

Correctly Classified Instances	96	95.0495 %
Incorrectly Classified Instances	5	4.9505 %
Mean absolute error	0.0163	
Root mean squared error	0.1032	
Relative absolute error	7.4425 %	
Root relative squared error	31.2652 %	
Total Number of Instances	101	

In the results panel below the tree itself you see the estimation of the tree predictive performance.



PRACTICLE 4: CLUSTERING IN DATA MINING

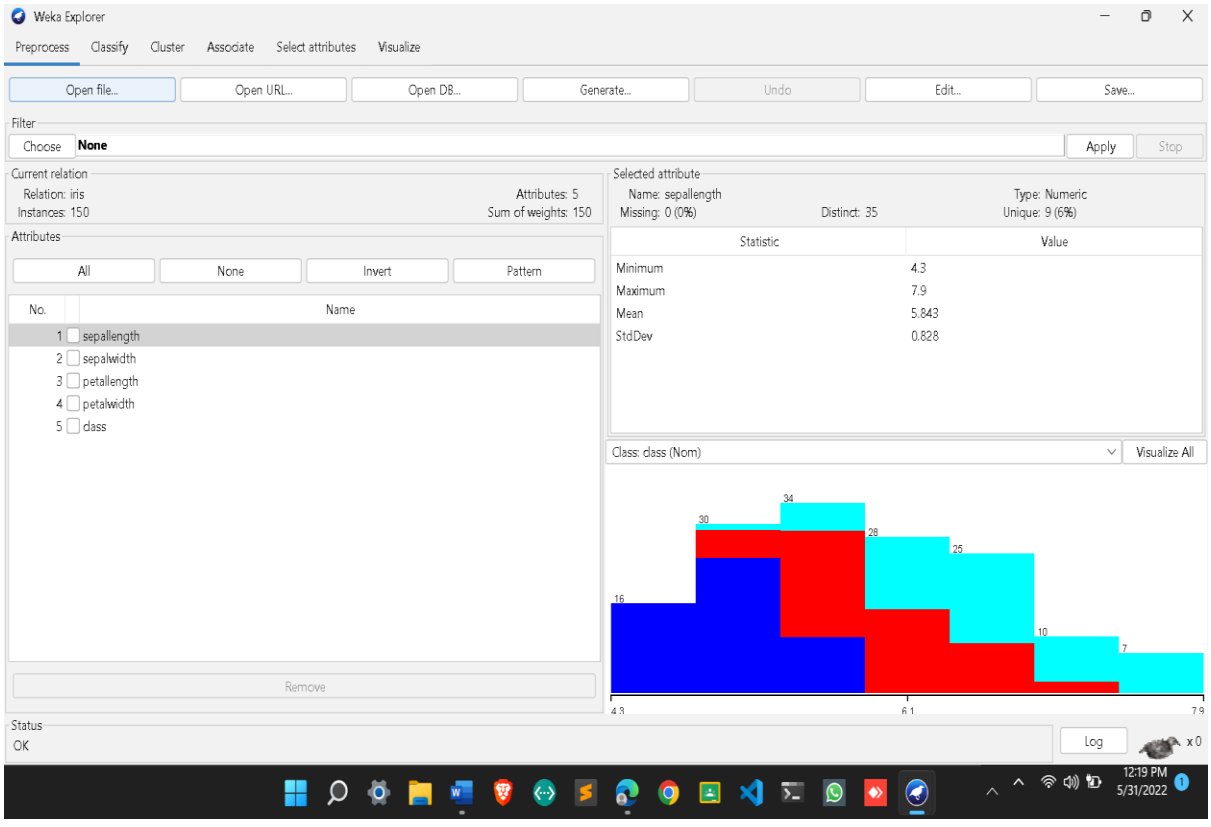
A clustering algorithm finds groups of similar instances in the entire dataset. WEKA supports several clustering algorithms such as EM, FilteredClusterer, HierarchicalClusterer, SimpleKMeans and so on. You should understand these algorithms completely to fully exploit the WEKA capabilities.

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

Dataset – iris.arff

Source – https://storm.cis.fordham.edu/~gweiss/data-mining/weka-data/iris.arff

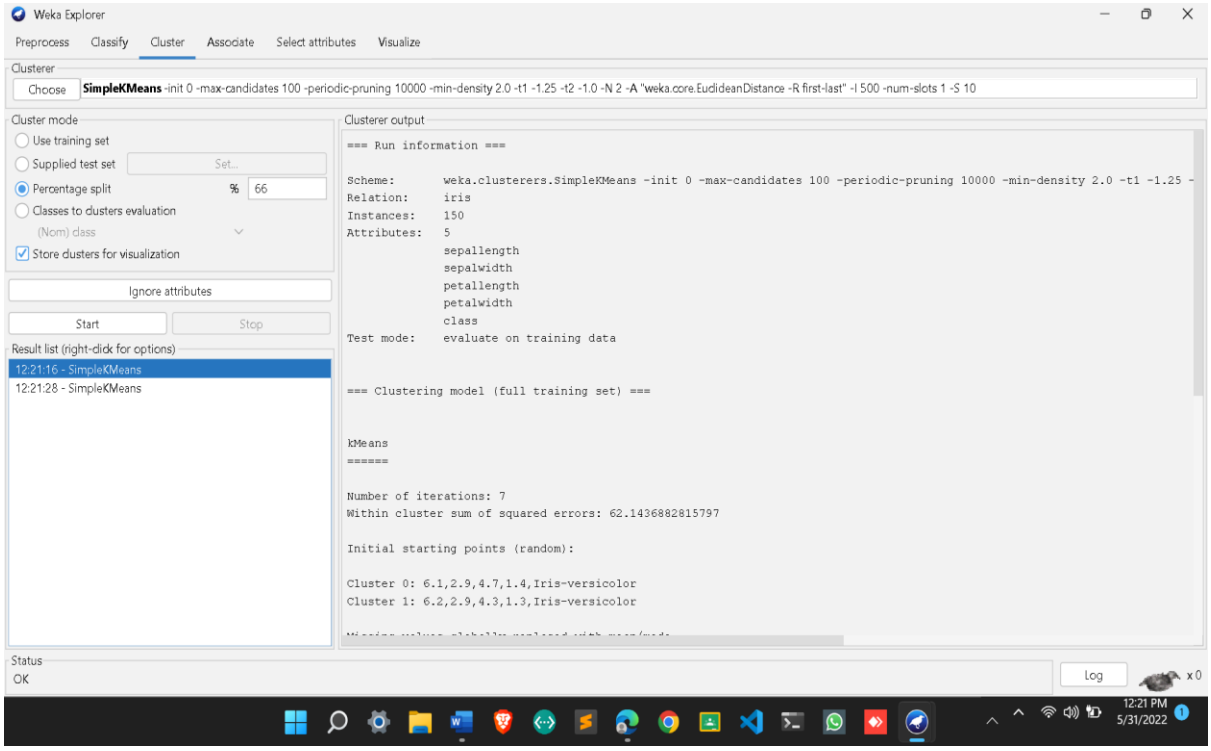
STEP 1 : In the WEKA explorer select the **Preprocess** tab. Click on the **Open file** ... option and select the **iris.arff** file in the file selection dialog. When you load the data, the screen looks like as shown below –



There are 150 instances and 5 attributes. The names of attributes are listed as **sepallength**, **sepalwidth**, **petallength**, **petalwidth** and **class**. The first four attributes are of numeric type while the class is a nominal type with 3 distinct values. Examine each attribute to understand the features of the database

STEP 2: Click on the **Cluster** TAB to apply the clustering algorithms to our loaded data. Select **SimpleKmeans** as the clustering algorithm.

STEP 3: Click on the **Start** button to process the data. After a while, the results will be presented on the screen.



Observations

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

=== Run information ===

Scheme: weka.clusterers.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-slots 1 -S 10
Relation: iris
Instances: 150

```
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    evaluate on training data

Number of iterations: 7

=== Clustering model (full training set) ===
Within cluster sum of squared errors: 62.1436882815797

Initial starting points (random):

Cluster 0: 6.1,2.9,4.7,1.4,Iris-versicolor
Cluster 1: 6.2,2.9,4.3,1.3,Iris-versicolor



Missing values globally replaced with mean/mode
Final cluster centroids:
                                Cluster#
Attribute            Full Data          0          1
                      (150.0)      (100.0)     (50.0)
=================================================================
sepallength            5.8433       6.262      5.006
sepalwidth              3.054       2.872      3.418
petallength            3.7587       4.906      1.464
petalwidth             1.1987       1.676      0.244
class              Iris-setosa Iris-versicolor  Iris-setosa
```

From the output screen, we can observe that −

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