

Plot RoC Curves

Compare classification results of ID3, J48, Naïve-Bayes and k-NN classifiers for each dataset, and deduce which classifier is performing best and poor for each dataset and justify.

While working on a classification model, we feel a need of a metric which can show us how our model is performing. A metric which can also give a graphical representation of the performance will be very helpful.

ROC curve can efficiently give us the score that how our model is performing in classifying the labels. We can also plot graph between False Positive Rate and True Positive Rate with this ROC(Receiving Operating Characteristic) curve. The area under the ROC curve give is also a metric.

The true-positive rate is also known as sensitivity, recall or *probability of detection*. The false-positive rate is also known as *probability of false alarm*.

```
[1] import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import matplotlib.pyplot as plt
```

```
[2] url = "https://raw.githubusercontent.com/Statology/Python-Guides/main/default.csv"
data = pd.read_csv(url)
```

```
[3] X = data[['student', 'balance', 'income']]
Y = data['default']
```

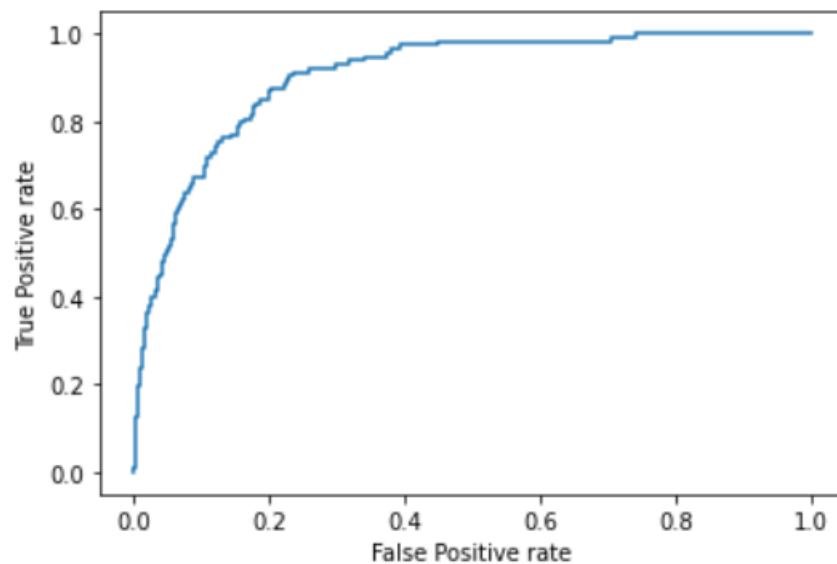
```
[4] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state = 0)
```

```
▶ log_regression = LogisticRegression()
log_regression.fit(X_train, Y_train)
```

```
📄 LogisticRegression()
```

```
[6] Y_pred_proba = log_regression.predict_proba(X_test)[::,1]  
fpr, tpr, _ = metrics.roc_curve(Y_test, Y_pred_proba)
```

```
[7] plt.plot(fpr,tpr)  
plt.ylabel('True Positive rate')  
plt.xlabel('False Positive rate')  
plt.show()
```



ENLIGHTENS THE NESCIENCE

Week – 9 Write a program of cluster analysis using simple k-means algorithm Python programming language

Aim:To Write a program of cluster analysis using simple k-means algorithm Python programming language.

Description:

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of kmeans clustering.

What is K-Means Algorithm?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabelled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if

K=2, there will be two clusters, and
for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabelled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabelled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

Program:

```
from copy import deepcopy
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
```

```
[4] df = pd.read_csv('/content/IRIS.csv')
```

```
print(df)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
..
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

[150 rows x 5 columns]

```
[6] df["species"] = pd.Categorical(df["species"])
df["species"] = df["species"].cat.codes
```

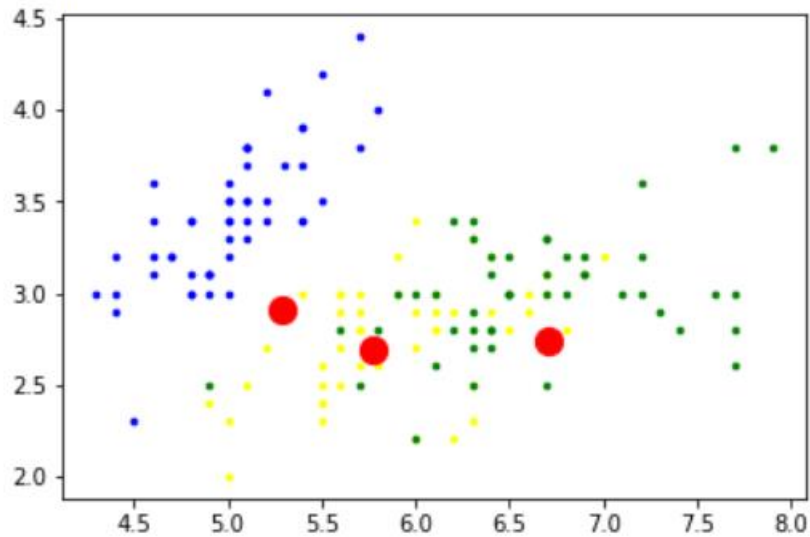
```
[7] data = df.values[:, 0:4]
category = df.values[:, 4]
```

```
k = 3
n = data.shape[0] #Training data
c = data.shape[1] #Number of features in data
```

```
[10] mean = np.mean(data, axis = 0)
std = np.std(data, axis = 0)
centers = np.random.randn(k,c) * std + mean
```

```
▶ colors = ['blue', 'yellow', 'green']  
for i in range(n):  
    plt.scatter(data[i,0], data[i,1], s = 7, color= colors[int(category[i])])  
plt.scatter(centers[:,0], centers[:,1], c = 'r' , s=150)
```

↳ <matplotlib.collections.PathCollection at 0x7f76ed7cbee0>



Pie chart:

A **pie chart** is a graphical representation technique that displays data in a circular-shaped graph. It is a composite static chart that works best with few variables.

Pie charts are often used to represent sample data—with data points belonging to a combination of different categories. Each of these categories is represented as a “slice of the pie.”

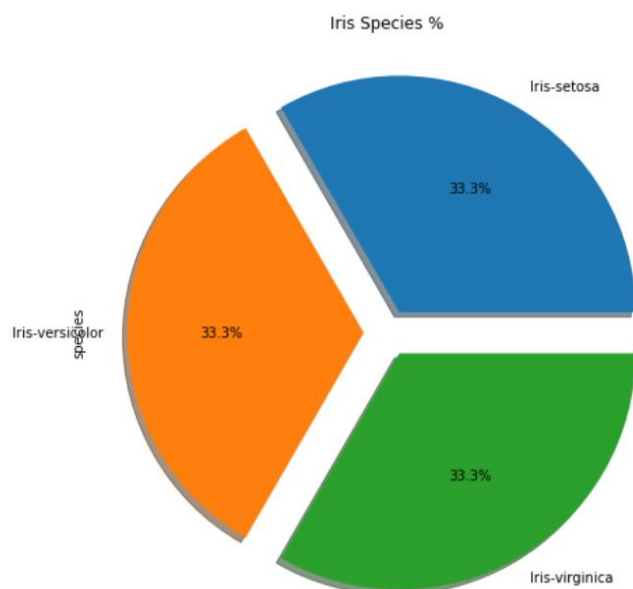
The size of each slice is directly proportional to the number of data points that belong to a particular category.

When the data comprises distinctive parts, a pie chart is best suited to represent it. The aim of using a pie chart is to compare the contribution of each part to the whole data.

Program:

```
[8] import pandas as pd
import matplotlib.pyplot as plt
iris = pd.read_csv('/content/IRIS.csv')
```

```
[13] ax = plt.subplots(1,1,figsize=(10,8))
iris['species'].value_counts().plot.pie(explode=[0.1,0.1,0.1], autopct='%1.1f%%', shadow=True,figsize=(10,8))
plt.title("Iris Species %")
plt.show()
```





```
print(iris)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
..
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

[150 rows x 5 columns]



Week – 8

Demonstrate performing clustering of data sets

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

Explore other clustering techniques available in Weka.

Explore visualization features of Weka to visualize the clusters.

Derive interesting insights and explain.

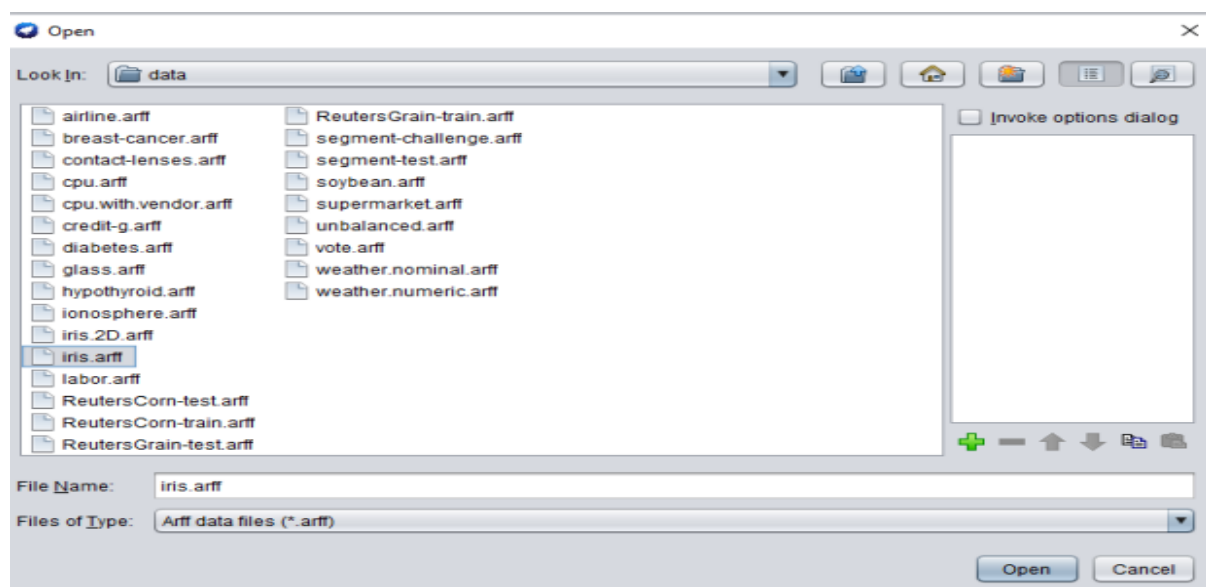
K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of kmeans clustering.

Here K defines the number of pre-defined clusters that need to be created in the process, as if

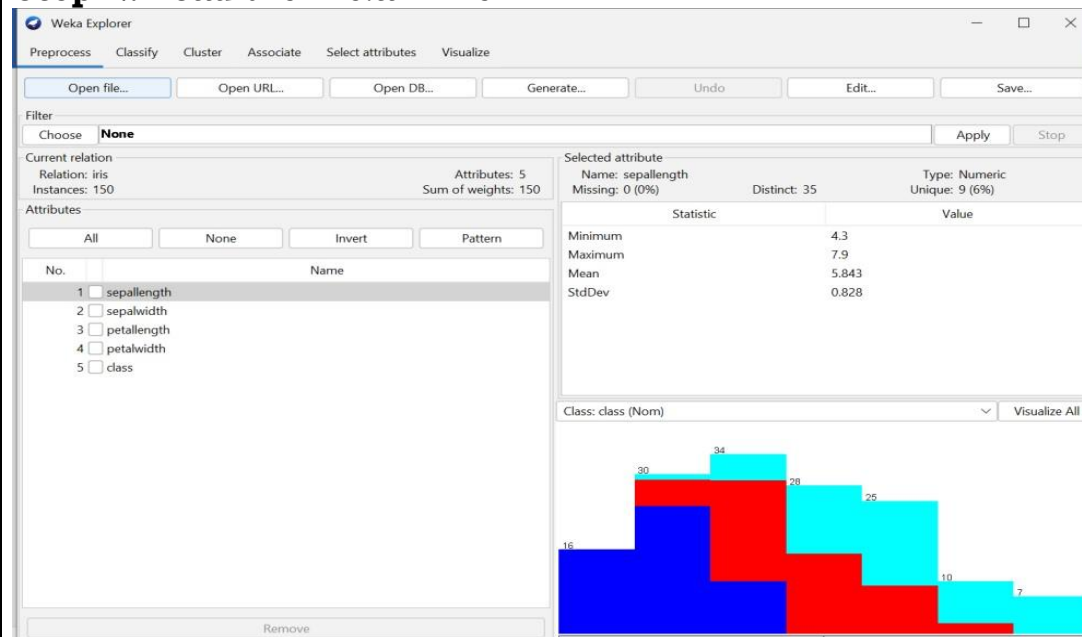
K=2, there will be two clusters, and
for K=3, there will be three clusters, and so on.

Steps to be followed:

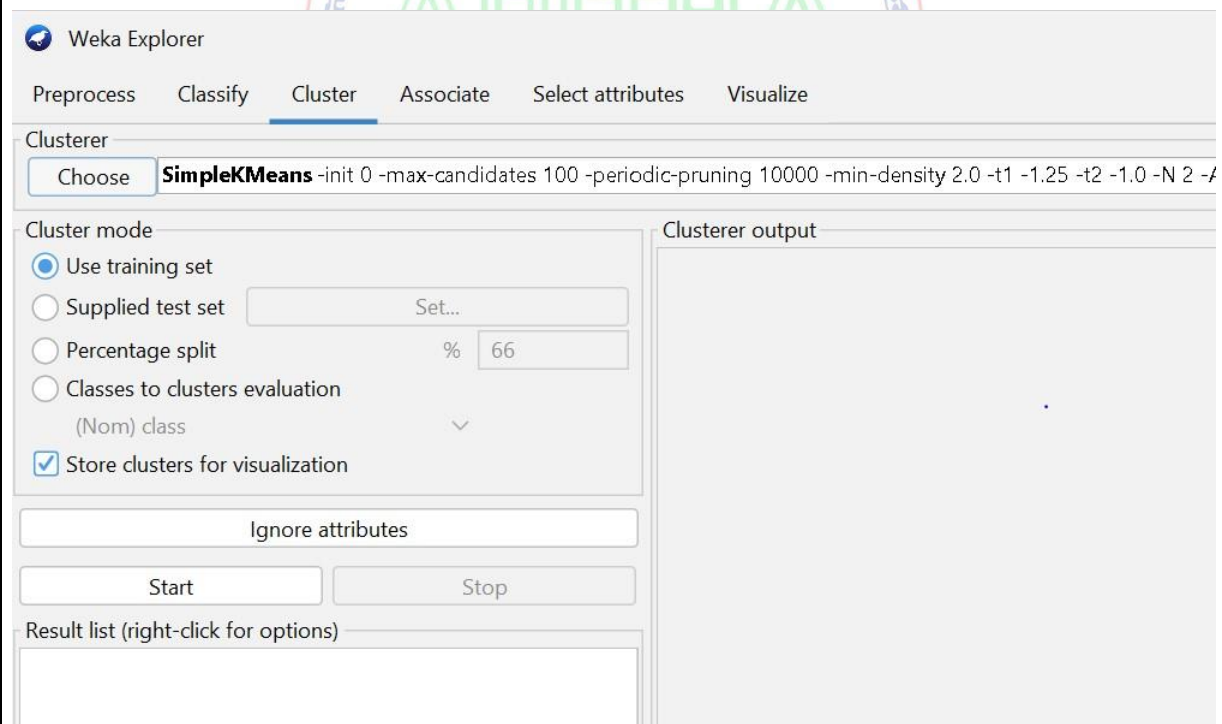
Step 1: In the preprocessing interface, open the Weka Explorer and load the required dataset, and we are taking the iris.arff dataset.



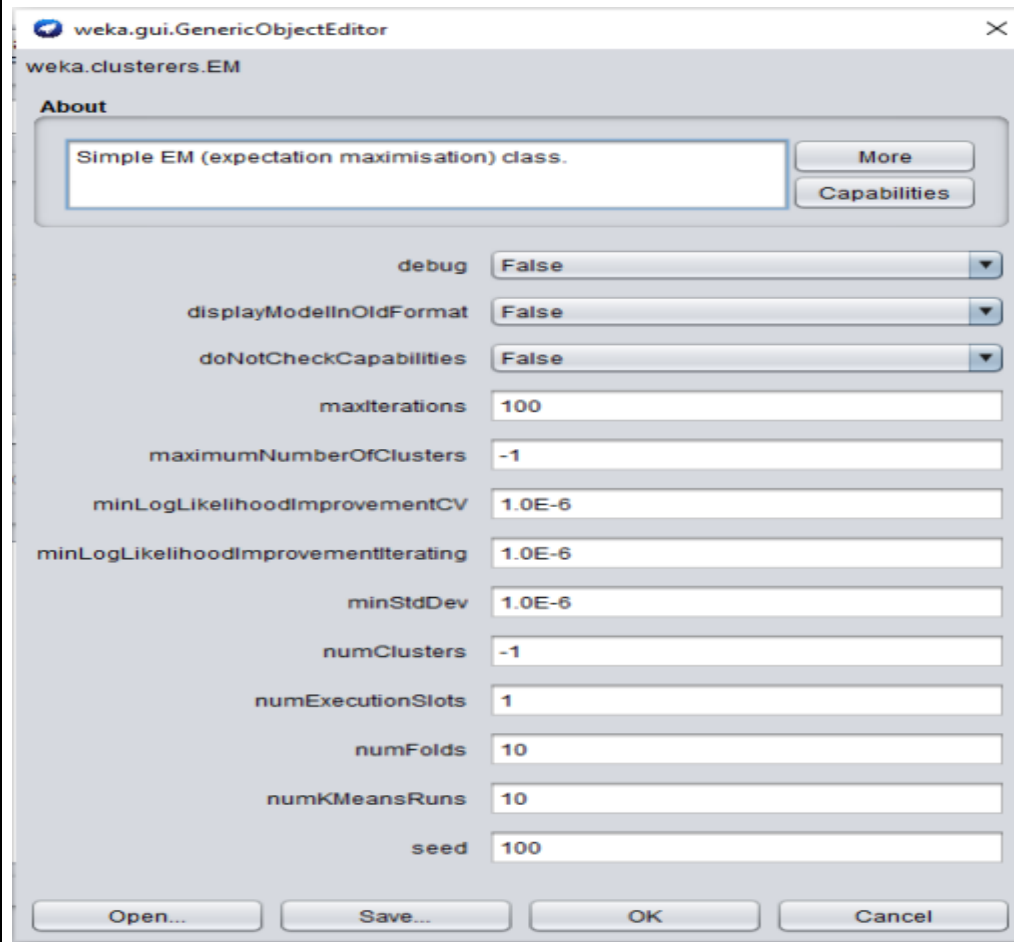
Step 2: Load the iris.arff file



Step 3: Find the 'cluster' tab in the explorer and press the choose button to execute clustering. A dropdown list of available clustering algorithms appears as a result of this step and selects the **simple-k means algorithm**.



Step 4: Then, to the right of the choose icon, press the text button to bring up the popup window shown in the screenshots. We enter three for the number of clusters in this window and leave the seed value alone. The seed value is used to generate a random number that is used to make internal assignments of instances of cluster. Here make numClusters as '3' and seed as '100'



Step 5: One of the choices has been chosen. We must ensure that they are in the '**cluster mode**' panel before running the clustering algorithm.

The choice to use a training set is selected, and then the 'start' button is pressed. The screenshots below display the process and the resulting window.

Cluster mode

☒ Use training set

☐ Supplied test set

☐ Percentage split %

☐ Classes to clusters evaluation

☒ Store clusters for visualization

Step 6: The centroid of each cluster is shown in the result window, along with statistics on the number and percent of instances allocated to each cluster. Each cluster centroid is represented by a mean vector. This cluster be used to describe a cluster.

Number of clusters selected by cross validation: 4
Number of iterations performed: 16

Attribute	Cluster			
	0 (0.32)	1 (0.33)	2 (0.2)	3 (0.14)
=====				
sepal length				
mean	5.897	5.006	6.9426	6.1304
std. dev.	0.5279	0.3489	0.498	0.2943
sepal width				
mean	2.7519	3.418	3.1103	2.8088
std. dev.	0.3103	0.3772	0.2952	0.2361
petal length				
mean	4.2267	1.464	5.8559	5.0993
std. dev.	0.445	0.1718	0.4626	0.2462
petal width				
mean	1.3134	0.244	2.1495	1.8254
std. dev.	0.1864	0.1061	0.232	0.2152
class				
Iris-setosa	1	51	1	1
Iris-versicolor	48.1125	1	1.0182	3.8693
Iris-virginica	2.0983	1	31.0375	19.8641
[total]	51.2108	53	33.0557	24.7335

Step 7: Another way to grasp the characteristics of each cluster is to visualize them. To do so, right-click the result set on the result. Selecting to visualize cluster assignments from the list Column



Week – 6

Demonstrate ZeroR technique on Iris dataset (by using necessary preprocessing technique(s)) and share your observations (using WEKA)

Aim: To Week – 6 Demonstrate ZeroR technique on Iris dataset (by using necessary preprocessing technique(s)) and share your observations (using WEKA)

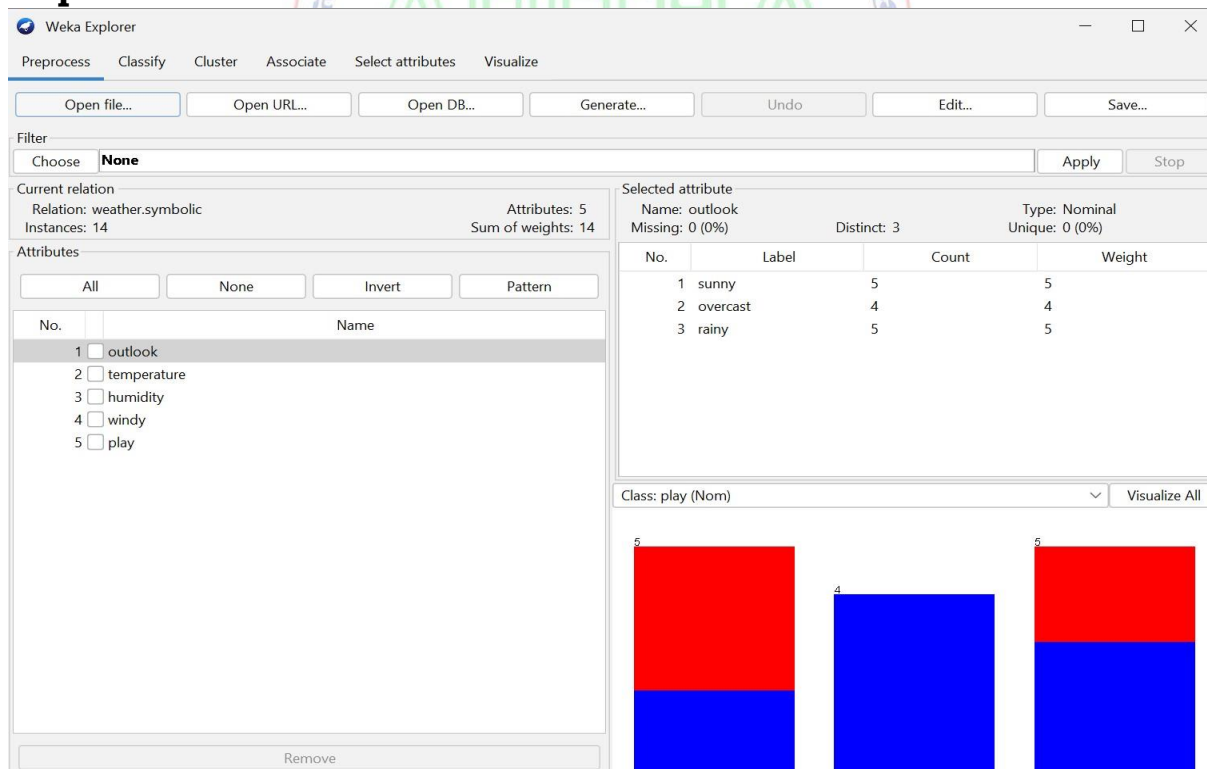
Description:

ZeroR is the simplest classification method which relies on the target and ignores all predictors. ZeroR classifier simply predicts the majority category (class). Although there is no predictability power in ZeroR, it is useful for determining a baseline performance as a benchmark for other classification methods.

Setting Test Data

We will use the preprocessed weather data file from the previous lesson. Open the saved file by using the Open file ... option under the Preprocess tab, click on the Classify tab, and you would see the following screen –

Step 1: Load weather data set



Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Open file... Open URL... Open DB... Generate... Undo Edit... Save...

Filter: Choose **None** Apply Stop

Current relation: Relation: weather.symbolic Instances: 14 Attributes: 5 Sum of weights: 14

Attributes: All None Invert Pattern

No.	Name
1	<input type="checkbox"/> outlook
2	<input type="checkbox"/> temperature
3	<input type="checkbox"/> humidity
4	<input type="checkbox"/> windy
5	<input type="checkbox"/> play

Remove

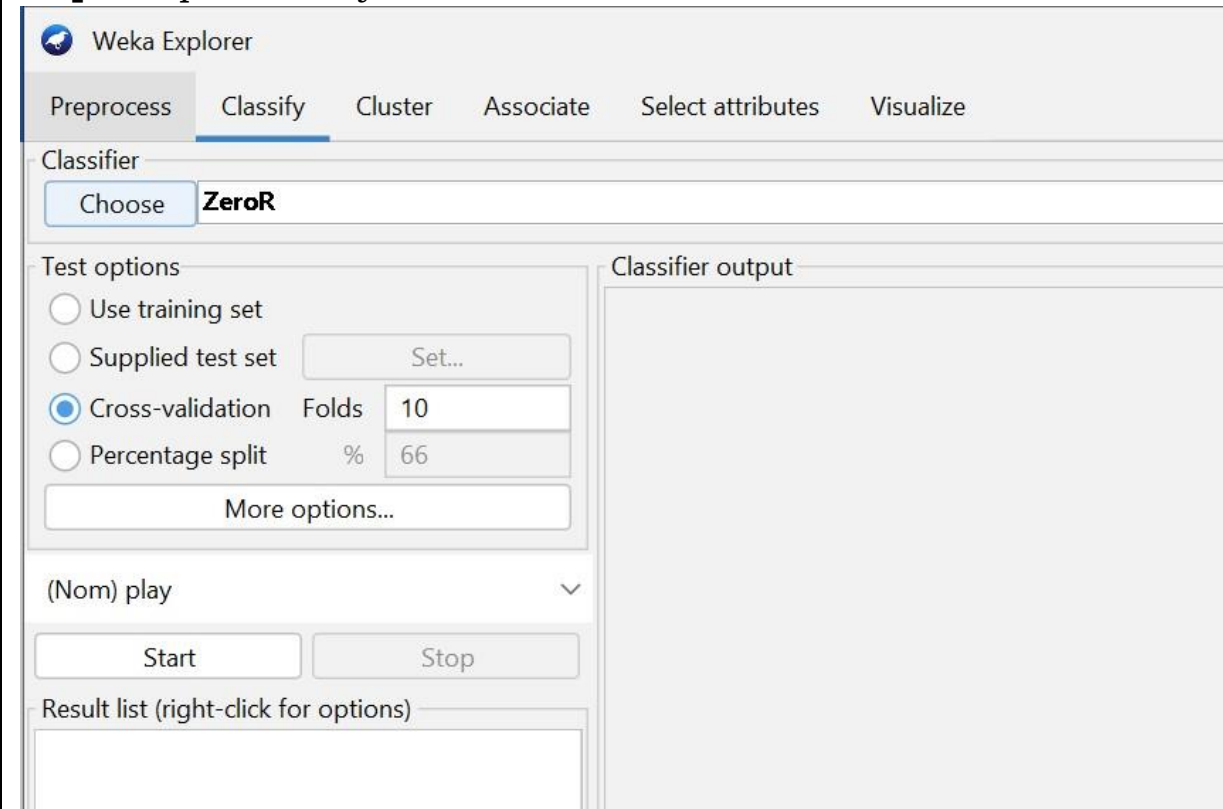
Selected attribute: Name: outlook Missing: 0 (0%) Distinct: 3 Type: Nominal Unique: 0 (0%)

No.	Label	Count	Weight
1	sunny	5	5
2	overcast	4	4
3	rainy	5	5

Class: play (Nom) Visualize All

Visualization: Bar chart showing the distribution of 'play' values for each 'outlook' category. The y-axis represents the count (0 to 5). The x-axis represents the 'outlook' categories: sunny, overcast, and rainy. The bars are colored red for 'play = yes' and blue for 'play = no'.

Step 2: Open classify



The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The 'Classifier' section has 'ZeroR' chosen. The 'Test options' section shows 'Cross-validation' selected with 'Folds' set to 10. The 'Classifier output' section is empty. The 'Result list' section is also empty.

Test options available in classify

Training set

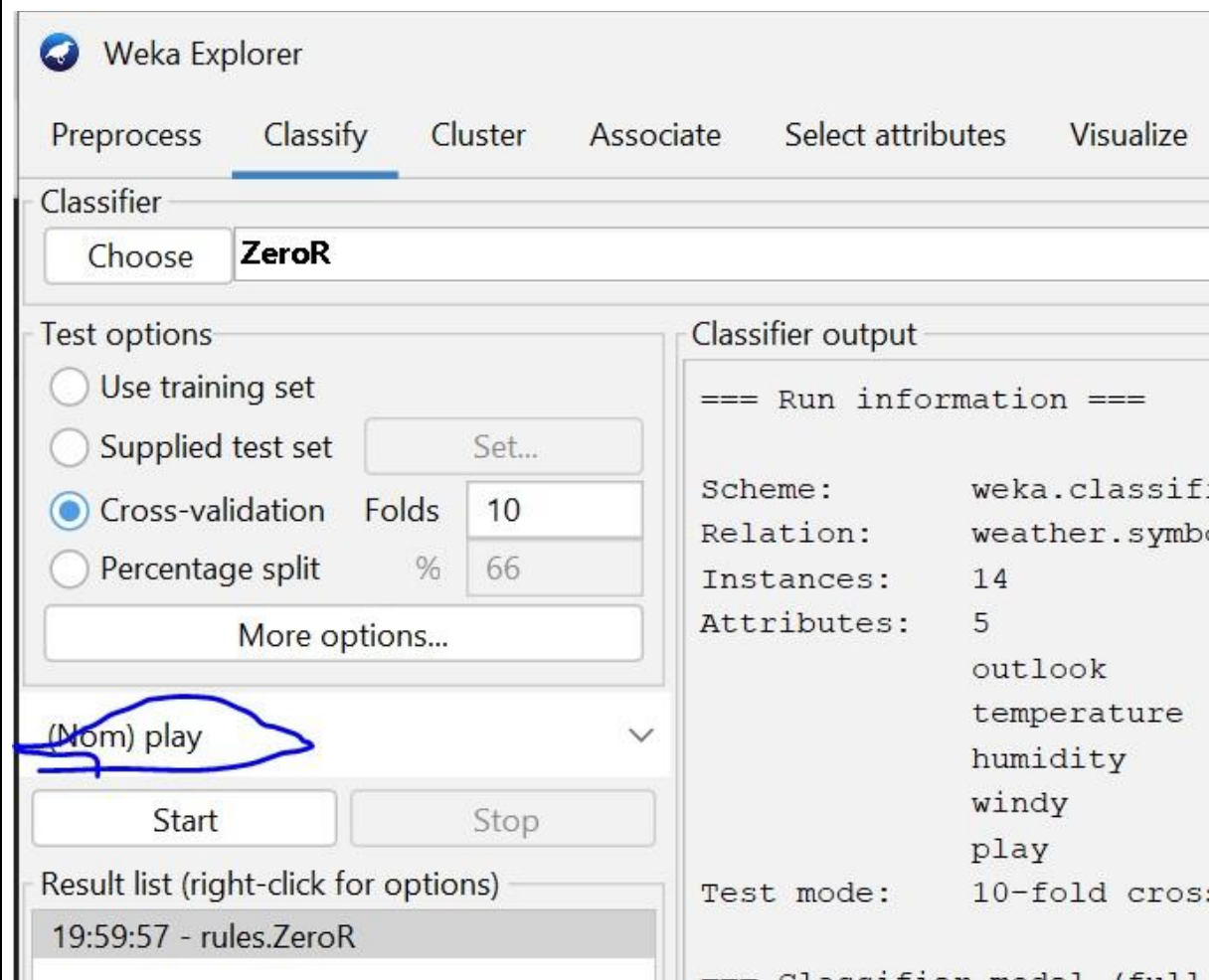
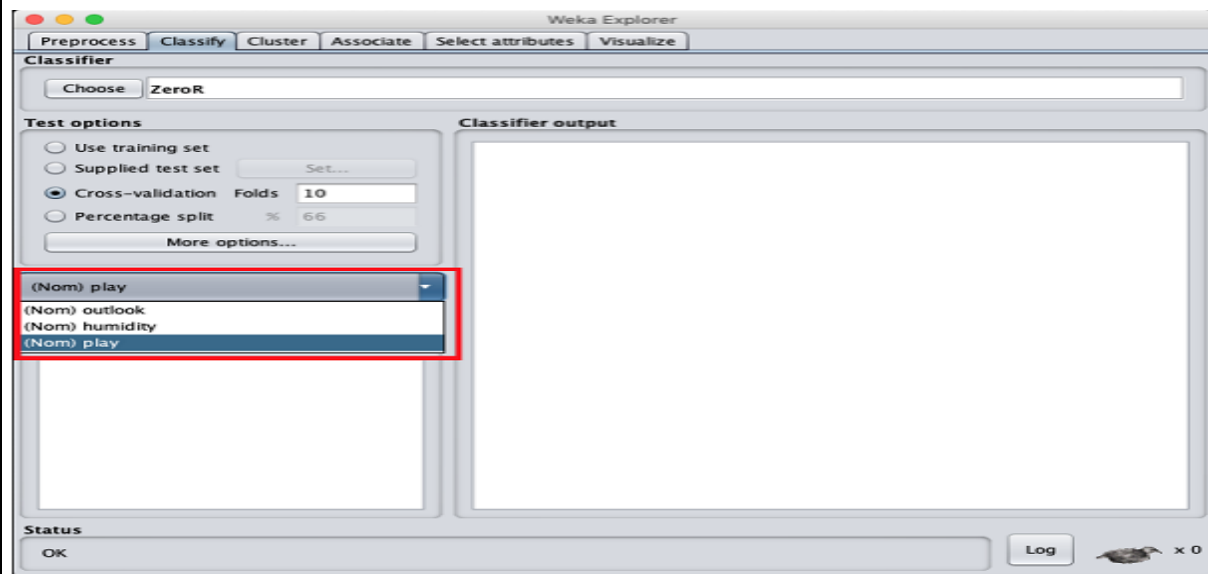
Supplied test set

Cross-validation

Percentage split

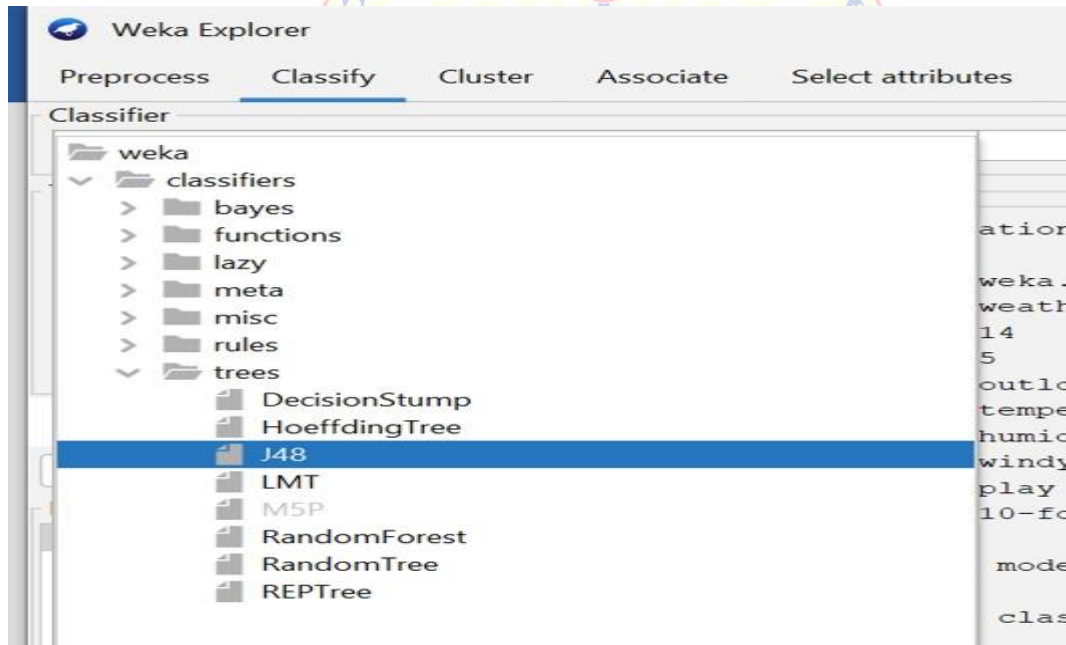
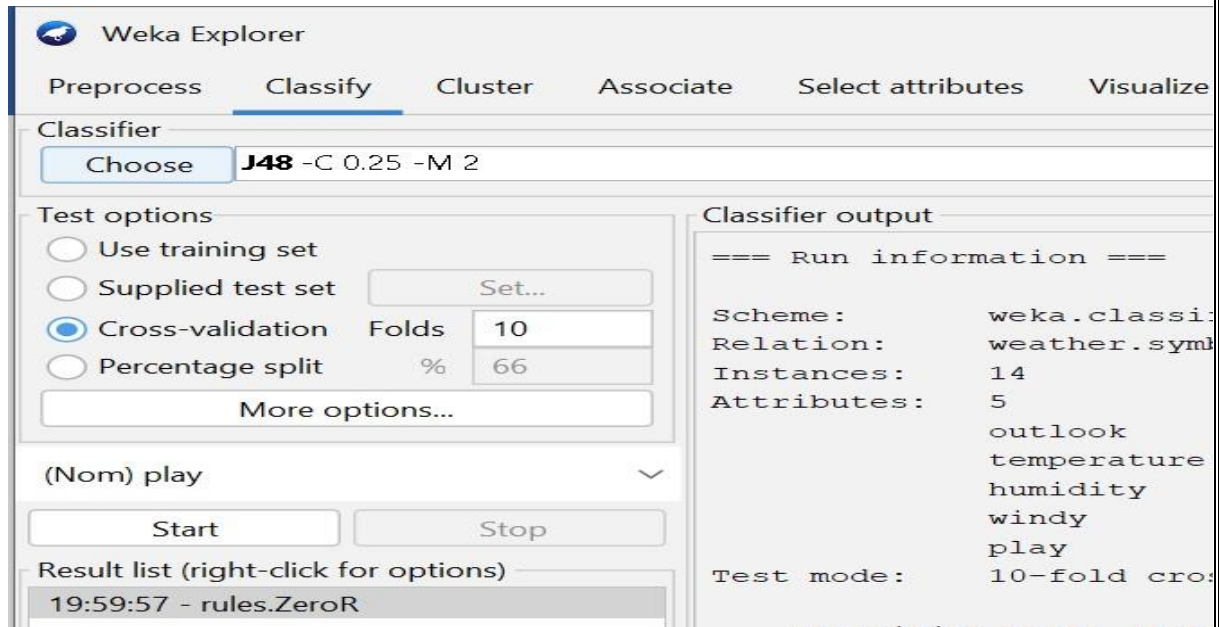
Under **cross-validation**, you can set the number of folds in which entire data would be split and used during each iteration of training. In the **percentage split**, you will split the data between training and testing using the set split percentage.

Step 3: Now, keep the default “play” option for the output class –

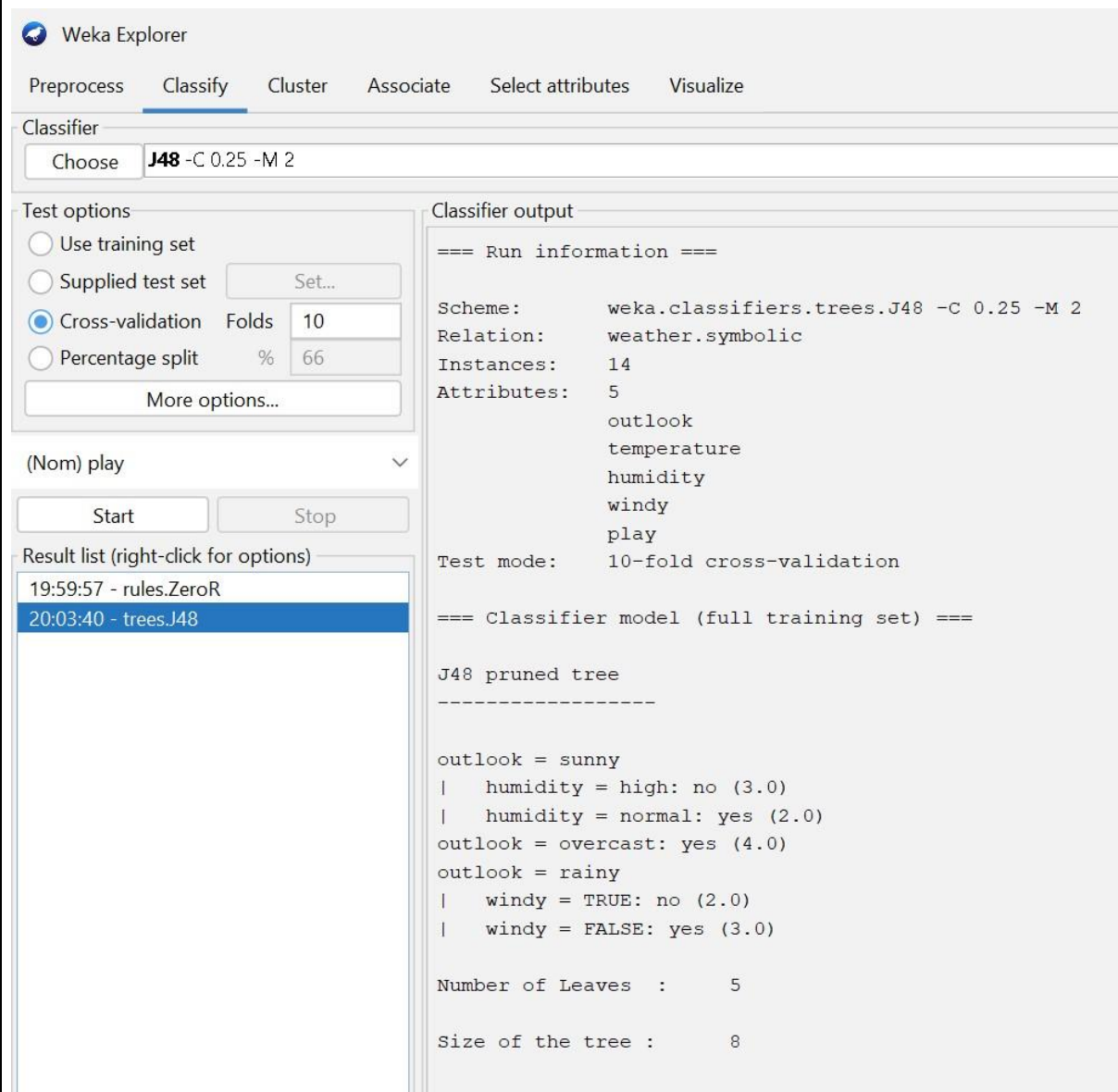


Step 4: Next, you will select the classifier. Click on “**choose**” button and Select Classifier as “J48”

Weka → classifier → trees → J48



Step 5: Click on the “**Start**” button to start the classification process.



Weka Explorer

Preprocess **Classify** Cluster Associate Select attributes Visualize

Classifier

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

Test options

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds **10**

☐ Percentage split % **66**

More options...

(Nom) play v

Start Stop

Result list (right-click for options)

- 19:59:57 - rules.ZeroR
- 20:03:40 - trees.J48**

Classifier output

```

=== Run information ===

Scheme:      weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:    weather.symbolic
Instances:   14
Attributes:  5
              outlook
              temperature
              humidity
              windy
              play

Test mode:   10-fold cross-validation

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

J48 pruned tree
-----

outlook = sunny
|  humidity = high: no (3.0)
|  humidity = normal: yes (2.0)
outlook = overcast: yes (4.0)
outlook = rainy
|  windy = TRUE: no (2.0)
|  windy = FALSE: yes (3.0)

Number of Leaves   :      5

Size of the tree   :      8
  
```

```

Size of the tree :      8

Time taken to build model: 0.17 seconds

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

Correctly Classified Instances      7           50   %
Incorrectly Classified Instances    7           50   %
Kappa statistic                     -0.0426
Mean absolute error                  0.4167
Root mean squared error              0.5984
Relative absolute error              87.5   %
Root relative squared error         121.2987 %
Total Number of Instances          14

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.556    0.600    0.625     0.556    0.588      -0.043   0.633    0.758     yes
                0.400    0.444    0.333     0.400    0.364      -0.043   0.633    0.457     no
Weighted Avg.    0.500    0.544    0.521     0.500    0.508      -0.043   0.633    0.650

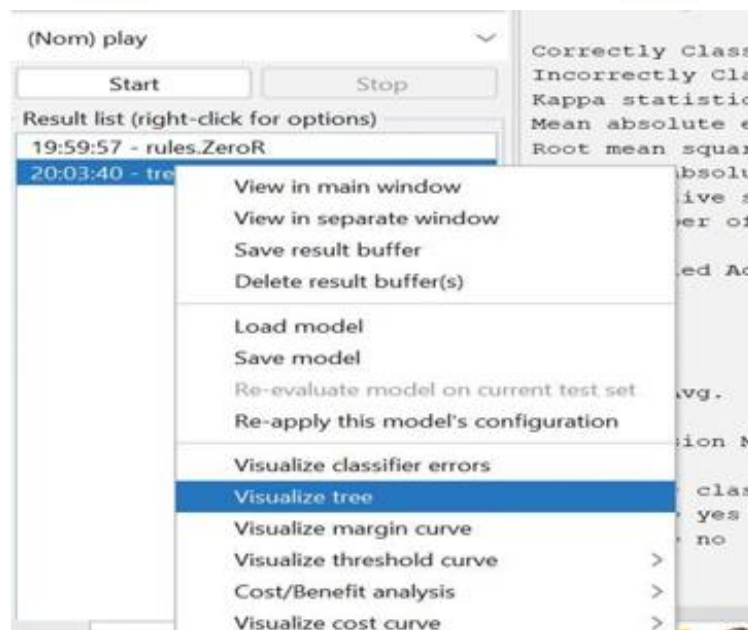
=== Confusion Matrix ===

 a b   <-- classified as
 5 4 | a = yes
 3 2 | b = no

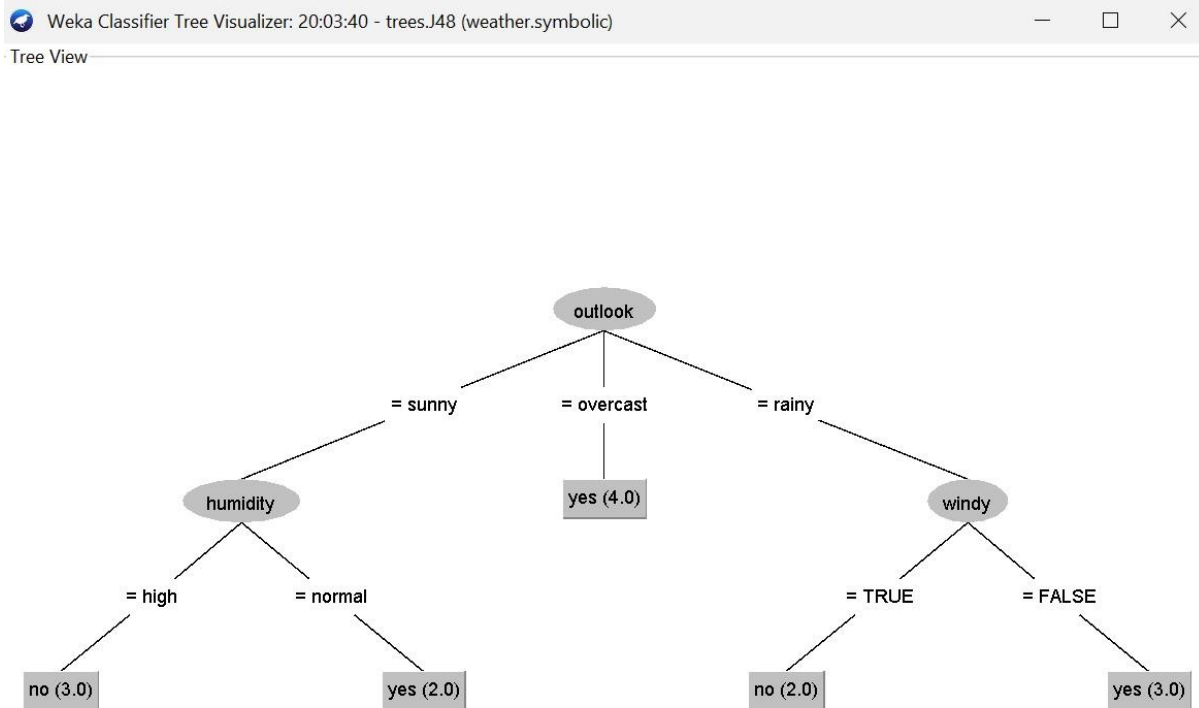
```

Step 6: To see the visual representation of the results, right click on the result in the Result list box.

Now select visualise tree option



Step 7: Obtained tree structure for weather dataset



ENLIGHTENS THE NESCIENCE

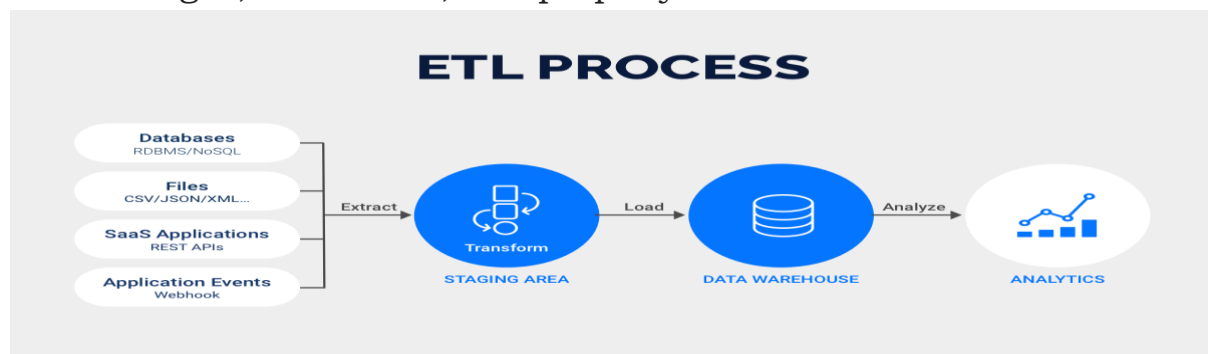
Week – 1

Creation of a Data Warehouse. Build Data Warehouse/Data Mart (using open source tools like Pentaho Data Integration Tool, Pentaho Business Analytics; or other data warehouse tools like Microsoft-SSIS, Informatica, Business Objects, etc.,) Design multi-dimensional data models namely Star, Snowflake and Fact Constellation schemas for any one enterprise (ex. Banking, Insurance, Finance, Healthcare, manufacturing, Automobiles, sales etc). Write ETL scripts and implement using data warehouse tools. Perform Various OLAP operations such as slice, dice, roll up, drill up and pivot.

ETL(Extract, Transform, and Load) is a process that extracts data from multiple source systems, changes it and then puts it into the Data Warehouse system. It's easy to building a Data warehouse as simple as pulling data from numerous sources and feeding it into a Data warehouse database.

The ETL process, which is technically complex, involves active participation from a variety of stakeholders, including developers, analysts, testers, and senior executives.

To preserve its value as a decision-making tool, the data warehouse system must develop in sync with business developments. ETL is a regular (daily, weekly, monthly) process of a data warehouse system that must be agile, automated, and properly documented.



Steps for ETL process :**Step 1: Extraction**

Data is extracted from the source system and placed in the staging area during extraction. If any transformations are required, they are performed in the staging area so that the performance of the source system is not harmed.

Rollback will be difficult if damaged data is transferred directly from the source into the Data warehouse database.

Data warehouses can combine systems with different hardware, database management systems, operating systems, and communication protocols.

Data warehouses must combine systems with disparate DBMS, hardware, operating systems, and communication protocols. Sources may include legacy programs such as mainframes, customized applications, point-of-contact devices such as ATMs and call switches, text files, spreadsheets, ERP, data from vendors and partners, and so on. Before extracting data and loading it physically, a logical data map is required.

Step 2 : Transformation

The data retrieved from the source server is raw and unusable in its original state, it must be cleaned, mapped, and transformed. It is a key ETL concept in which you apply a collection of functions to extracted data. Direct move or pass through **data** is the type of data that does not require any transformation.

Step 3 : Loading

The final stage in the ETL process is to load data into the target data warehouse database.

A large volume of data is loaded in a relatively short period of time in a typical data warehouse. As a result, the load process should be optimized for performance.

In the occurrence of a load failure, recovery procedures should be put in place so that operations can restart from the point of failure without compromising data integrity.

Data Warehouse administrators must monitor, continue, and stop loads based on server performance.

Types of Loading:

Initial Load — filling all of the Data Warehouse tables

Incremental Load — implementing ongoing modifications as needed on a regular basis

Full Refresh — clearing the contents of one or more tables and reloading them with fresh data

Load verification

- Check that the key field data is not missing or null.
- Modelling views based on target tables should be tested.
- Examine the combined values and computed measures.
- Data checks in the dimension and history tables.
- Examine the BI reports on the loaded fact and dimension table.

Step 1 Create database Data Warehouse

Step 2 Create Customer dimension table in Data Warehouse which will hold customer personal details. Fill the Customer dimension with sample Values

Step 3 Create basic level of Product Dimension table without considering any Category or Subcategory Fill the Product dimension with sample Values

Step 4 Create Store Dimension table which will hold details related stores available across various places. Fill the Store Dimension with sample Values

Step 5 Create Dimension Sales Person table which will hold details related stores available across various places. Fill the Dimension Sales Person with sample values:

Step 6 Create Date Dimension table which will create and populate date data divided on various levels.

Step 7 Create Time Dimension table which will create and populate Time data for the entire day with various time buckets.

Step 8 Create Fact table to hold all your transactional entries of previous day sales with appropriate foreign key columns which refer to primary key column of your dimensions;



```
CreatedatabaseSales_DW
Create table DimCustomer (
CustomerID int primary key identity,
CustomerAltIDvarchar(10) not null,
CustomerNamevarchar(50),
Gender varchar(20)
)
Insert into DimCustomer(CustomerAltID, CustomerName, Gender)
values
('IMI-001', 'Henry Ford', 'M'),
('IMI-002', 'Bill Gates', 'M'),
('IMI-003', 'Muskan Shaikh', 'F'),
('IMI-004', 'Richard Thrubin', 'M'),
('IMI-005', 'Emma Wattson', 'F');

Create table DimProduct
(
ProductKey int primary key identity,
ProductAltKeyvarchar(10)not null,
ProductName varchar(100),
ProductActualCost money,
ProductSalesCost money
)
Insert into DimProduct (ProductAltKey, ProductName,
ProductActualCost, ProductSalesCost)values
('ITM-001', 'Wheat Floor 1kg', 5.50, 6.50),
('ITM-002', 'Rice Grains 1kg', 22.50, 24),
('ITM-003', 'SunFlower Oil 1 ltr', 42, 43.5),
('ITM-004', 'Nirma Soap', 18, 20),
('ITM-005', 'Arial Washing Powder 1kg', 135, 139);
)

Create table DimStores
(
StoreID int primary key identity,
StoreAltIDvarchar(10)not null,
StoreNamevarchar(100),
StoreLocationvarchar(100),
City varchar(100),
State varchar(100),
Country varchar(100) )
```



Insert into

DimStores(StoreAltID,StoreName,StoreLocation,City,State,Country
)values

('LOC-A1','X-Mart','S.P. RingRoad','Ahmedabad','Guj','India'),
('LOC-A2','X-Mart','Maninagar','Ahmedabad','Guj','India'),
('LOC-A3','X-Mart','Sivranjani','Ahmedabad','Guj','India');

Create table DimSalesPerson

(
SalesPersonID int primary key identity,
SalesPersonAltIDvarchar(10)not null,
SalesPersonNamevarchar(100),
StoreID int,
City varchar(100),
State varchar(100),
Country varchar(100)
)

Insert into

DimSalesPerson(SalesPersonAltID,SalesPersonName,StoreID,City,State
,Country)values

('SP-DMSPR1','Ashish',1,'Ahmedabad','Guj','India'),
('SP-DMSPR2','Ketan',1,'Ahmedabad','Guj','India'),
('SP-DMNGR1','Srinivas',2,'Ahmedabad','Guj','India'),
('SP-DMNGR2','Saad',2,'Ahmedabad','Guj','India'),
('SP-DMSVR1','Jasmin',3,'Ahmedabad','Guj','India'),
('SP-DMSVR2','Jacob',3,'Ahmedabad','Guj','India');
)