

Date:

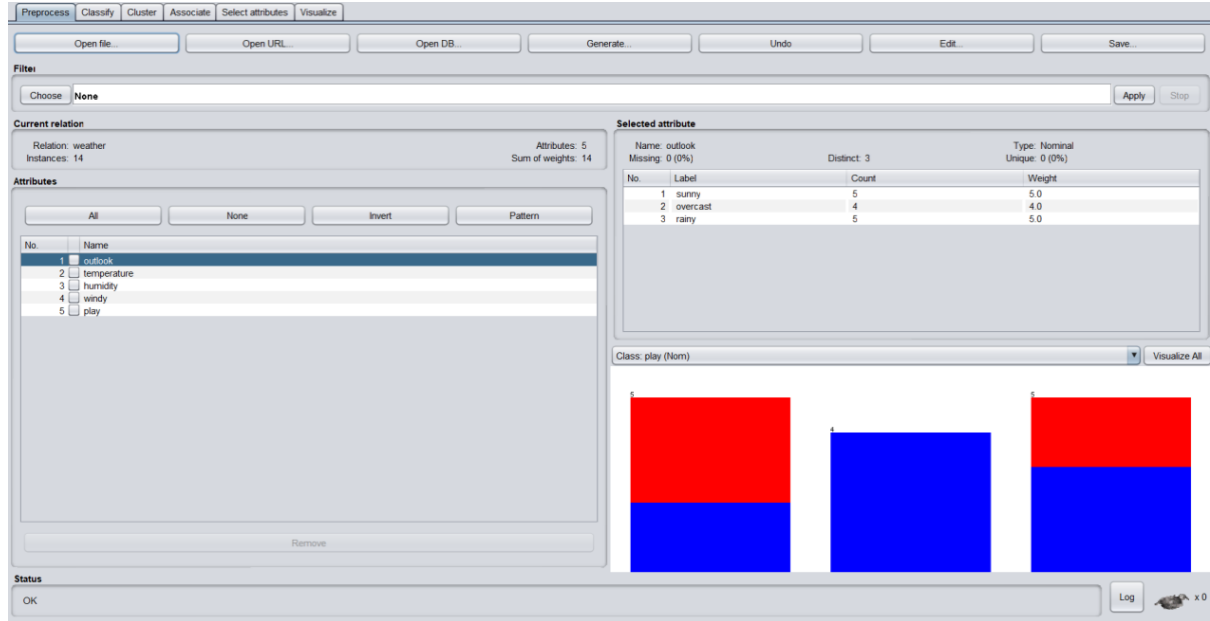
**Experiment-6: Demonstrate ZeroR technique on Iris dataset (by using necessary preprocessing technique(s)) and share your observations (using WEKA)**

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

**Description:**

**Setting Test Data**

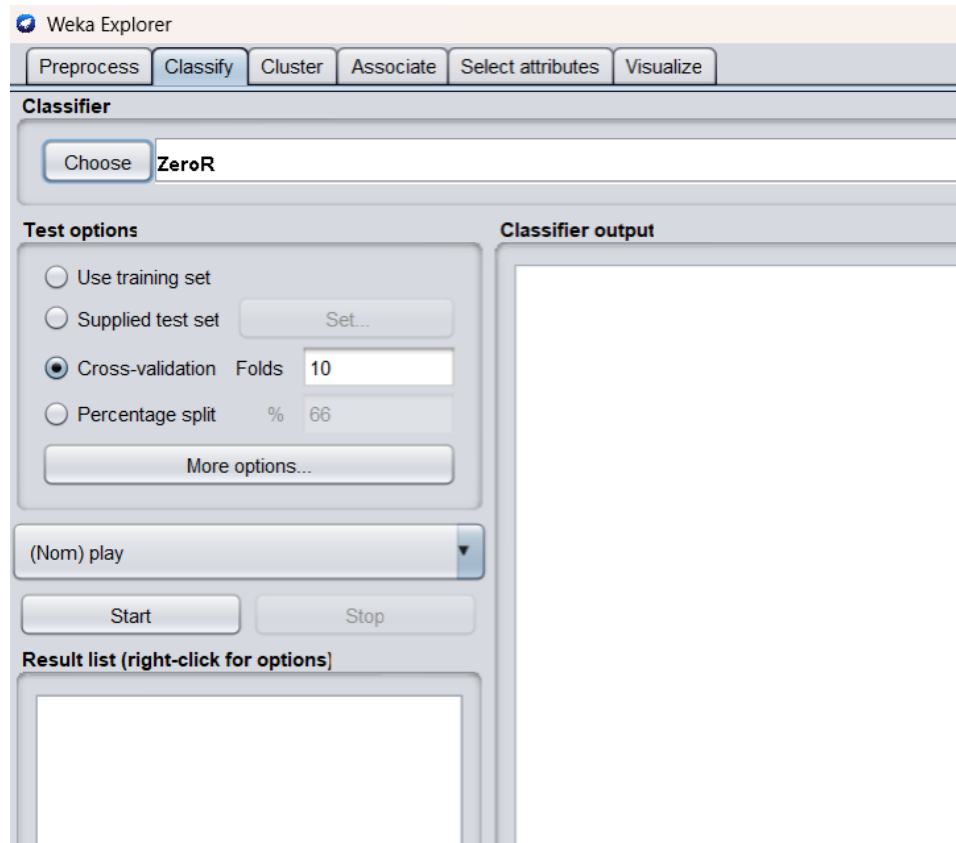
Load the weather dataset into the weka.



The screenshot shows the WEKA GUI with the 'Preprocess' tab selected. The 'Current relation' is 'weather' with 14 instances. The 'Attributes' list on the left includes 'outlook', 'temperature', 'humidity', 'windy', and 'play'. The 'Selected attribute' panel on the right shows 'outlook' with 3 distinct values: 'sunny' (count 5, weight 5.0), 'overcast' (count 4, weight 4.0), and 'rainy' (count 5, weight 5.0). The 'Class: play (Nom)' is selected, and a bar chart visualization is displayed at the bottom right, showing the distribution of 'play' for each 'outlook' category.

No.	Label	Count	Weight
1	sunny	5	5.0
2	overcast	4	4.0
3	rainy	5	5.0

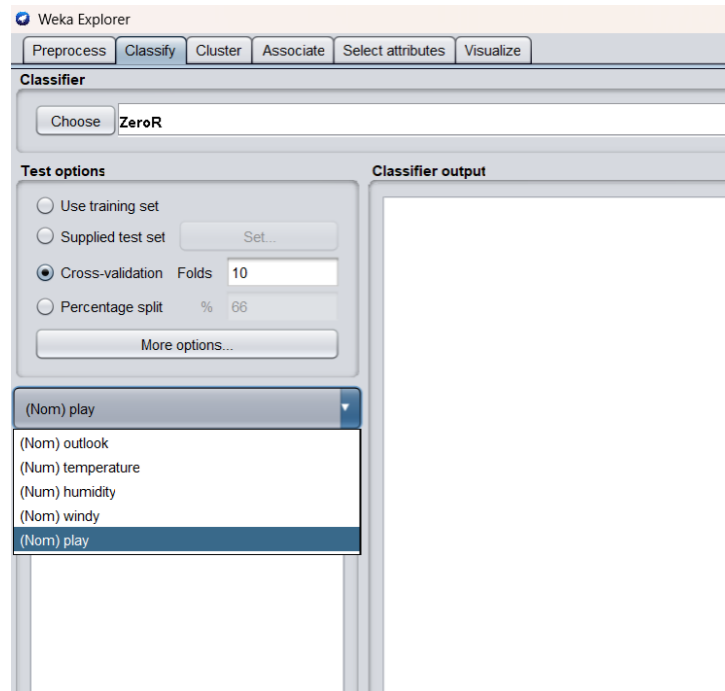
We will use the preprocessed weather data file from the previous lesson. Open the saved file by using the Open file ... option under the Preprocess tab, click on the Classify tab, and you would see the following screen –



Before you learn about the available classifiers, let us examine the Test options. You will notice four testing options as listed below –

- Training set
- Supplied test set
- Cross-validation
- Percentage split

Unless you have your own training set or a client supplied test set, you would use cross-validation or percentage split options. 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. Now, keep the default play option for the output class –



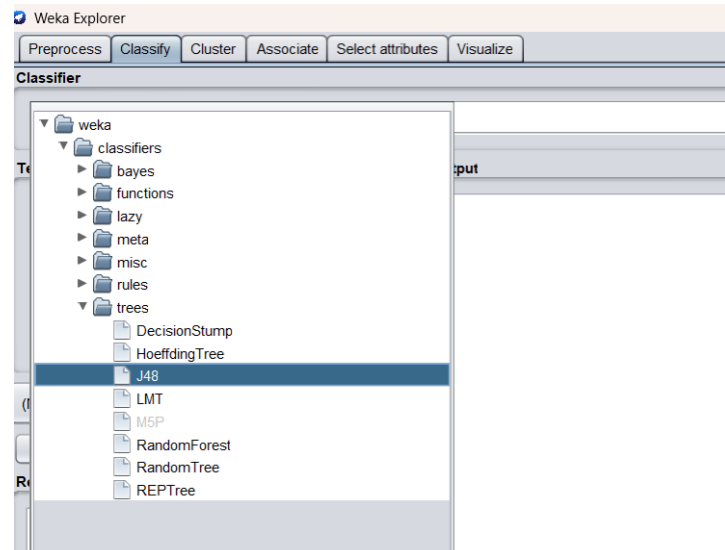
Next, you will select the classifier.

Selecting Classifier

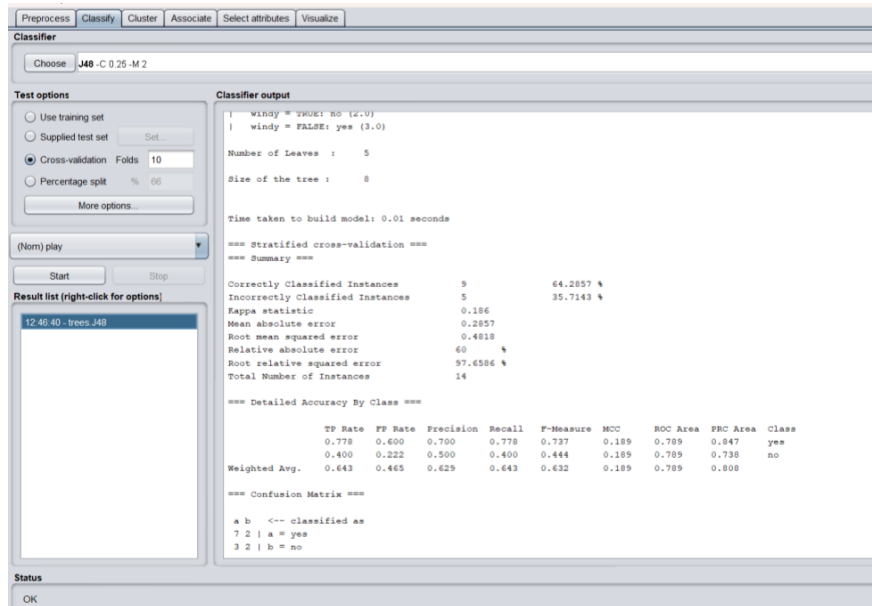
Click on the Choose button and select the following classifier –

weka→classifiers>trees>J48

This is shown in the screenshot below-



Click on the Start button to start the classification process. After a while, the classification results would be presented on your screen as shown here –

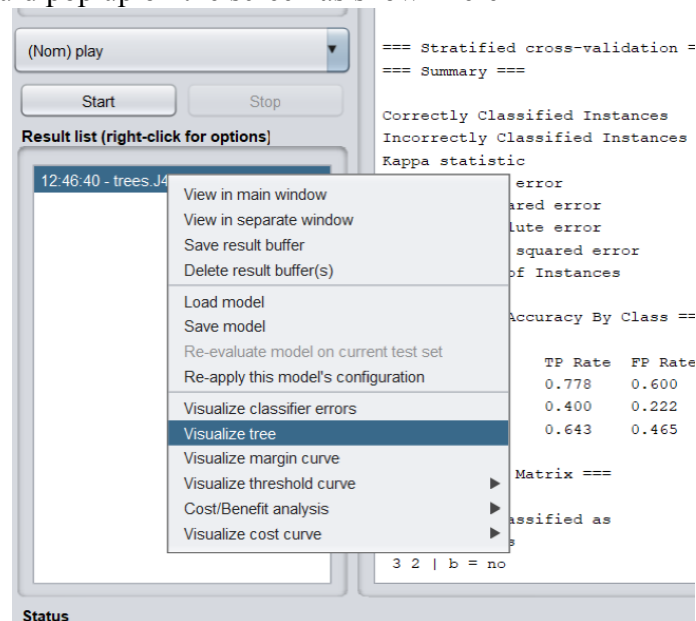


Let us examine the output shown on the right hand side of the screen.

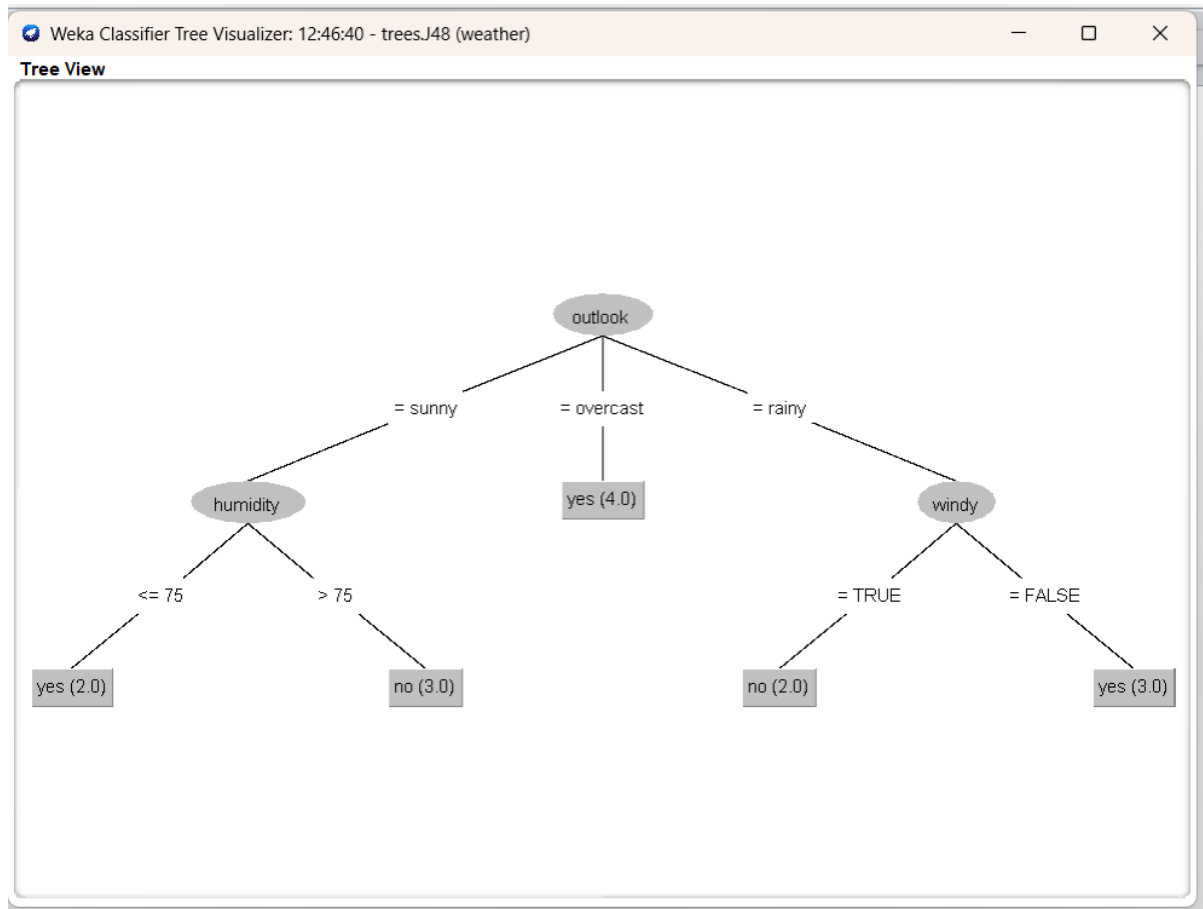
It says the size of the tree is 6. You will very shortly see the visual representation of the tree. In the Summary, it says that the correctly classified instances as 2 and the incorrectly classified instances as 3, It also says that the Relative absolute error is 110%. It also shows the Confusion Matrix. Going into the analysis of these results is beyond the scope of this tutorial. However, you can easily make out from these results that the classification is not acceptable and you will need more data for analysis, to refine your features selection, rebuild the model and so on until you are satisfied with the model's accuracy. Anyway, that's what WEKA is all about. It allows you to test your ideas quickly.

### Visualize Results

To see the visual representation of the results, right click on the result in the Result list box. Several options would pop up on the screen as shown here –



Select Visualize tree to get a visual representation of the traversal tree as seen in the screenshot below –



Date:

**Experiment-7: Write a program of Naive Bayesian classification using Python programming language****AIM:** Write a program of Naive Bayesian classification using Python programming language.**Description:**

Naive Bayes is among one of the most simple and powerful algorithms for classification based on Bayes' Theorem with an assumption of independence among predictors. Naive Bayes model is easy to build and particularly useful for very large data sets. There are two parts to this algorithm:

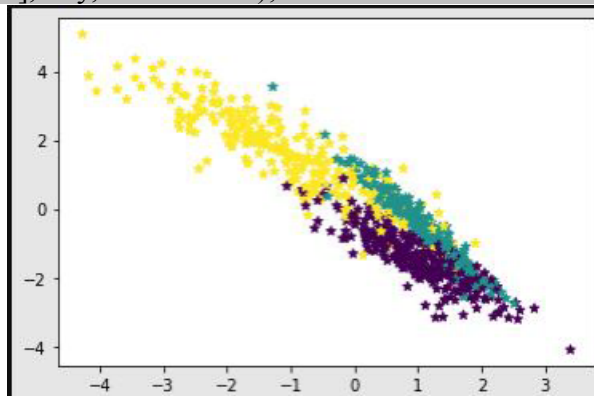
- Naive

- Bayes

The Naive Bayes classifier assumes that the presence of a feature in a class is unrelated to any other feature. 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 a particular fruit is an apple or an orange or a banana and that is why it is known as —Naive.

**Program:**

```
from sklearn.datasets import make_classification
X, y = make_classification(
    n_features=6,
    n_classes=3,
    n_samples=800,
    n_informative=2,
    random_state=1,
    n_clusters_per_class=1,
)
import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=y, marker="*");
```



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.33, random_state=125
)
from sklearn.naive_bayes import GaussianNB
# Build a Gaussian Classifier
model = GaussianNB()
# Model training
```

**Date:**

```
model.fit(X_train, y_train)
# Predict Output
predicted = model.predict([X_test[6]])
print("Actual Value:", y_test[6])
print("Predicted Value:", predicted[0])
```

Actual Value: 0

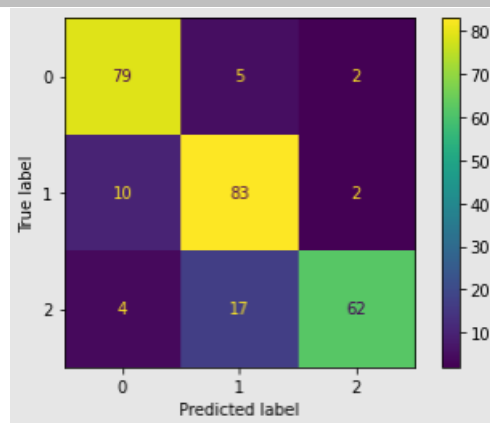
Predicted Value: 0

```
from sklearn.metrics import (
accuracy_score,
confusion_matrix,
ConfusionMatrixDisplay,
f1_score,
)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted")
print("Accuracy:", accuracy)
print("F1 Score:", f1)
```

Accuracy: 0.8484848484848485

F1 Score: 0.8491119695890328

```
labels = [0,1,2]
cm = confusion_matrix(y_test, y_pred, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot();
```





**Expt. No.:**

**Page No.: 46**

**Date:**

**Experiment-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.**

**AIM:** 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.

**Description:**

**Selecting a Clusterer**

By now you will be familiar with the process of selecting and configuring objects. Clicking on the clustering scheme listed in the Clusterer box at the top of the window brings up a GenericObjectEditor dialog with which to choose a new clustering scheme.

**Cluster Modes**

The Cluster mode box is used to choose what to cluster and how to evaluate the results. The first three options are the same as for classification: Use training set, Supplied test set and Percentage split (Section 5.3.1)—except that now the data is assigned to clusters instead of trying to predict a specific class. The fourth mode, Classes to clusters evaluation, compares how well the chosen clusters match up with a pre-assigned class in the data. The drop-down box below this option selects the class, just as in the Classify panel. An additional option in the Cluster mode box, the Store clusters for visualization tick box, determines whether or not it will be possible to visualize the clusters once training is complete. When dealing with datasets that are so large that memory becomes a problem it may be helpful to disable this option.

**Ignoring Attributes**

Often, some attributes in the data should be ignored when clustering. The Ignore attributes button brings up a small window that allows you to select which attributes are ignored. Clicking on an attribute in the window highlights it, holding down the SHIFT key selects a range of consecutive attributes, and holding down CTRL toggles individual attributes on and off. To cancel the selection, back out with the Cancel button. To activate it, click the Select button. The next time clustering is invoked, the selected attributes are ignored.

**Working with Filters**

The Filtered Clusterer meta-clusterer offers the user the possibility to apply filters directly before the clusterer is learned. This approach eliminates the manual application of a filter in the Preprocess panel, since the data gets processed on the fly. Useful if one needs to try out different filter setups.

**Learning Clusters**

The Cluster section, like the Classify section, has Start/Stop buttons, a result text area and a result list. These all behave just like their classification counterparts. Right-clicking an entry





**Expt. No.:**

**Page No.: 47**

**Date:**

in the result list brings up a similar menu, except that it shows only two visualization options: Visualize cluster assignments and Visualize tree. The latter is grayed out when it is not applicable.

### **Steps for run K-mean Clustering algorithms in WEKA**

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Preprocessing tab button.
4. Click on open file button.
5. Choose WEKA folder in C drive.
6. Select and Click on data option button.
7. Choose iris data set and open file.
8. Click on cluster tab and Choose k-mean and select use training set test option.
9. Click on start button.

### **Output:**

==== Run information ====

Scheme:weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last"

-I 500

-S 10

Relation: iris

Instances: 150

Attributes: 5

sepalength

sepalwidth

petallength

petalwidth

class

Test mode:evaluate on training data

==== Model and evaluation on training set ====

kMeans

=====

Number of iterations: 7

Within cluster sum of squared errors: 62.1436882815797

Missing values globally replaced with mean/mode

Cluster centroids:

Cluster#

Attribute Full Data 0 1

(150) (100) (50)

=====

sepalength 5.8433 6.262 5.006

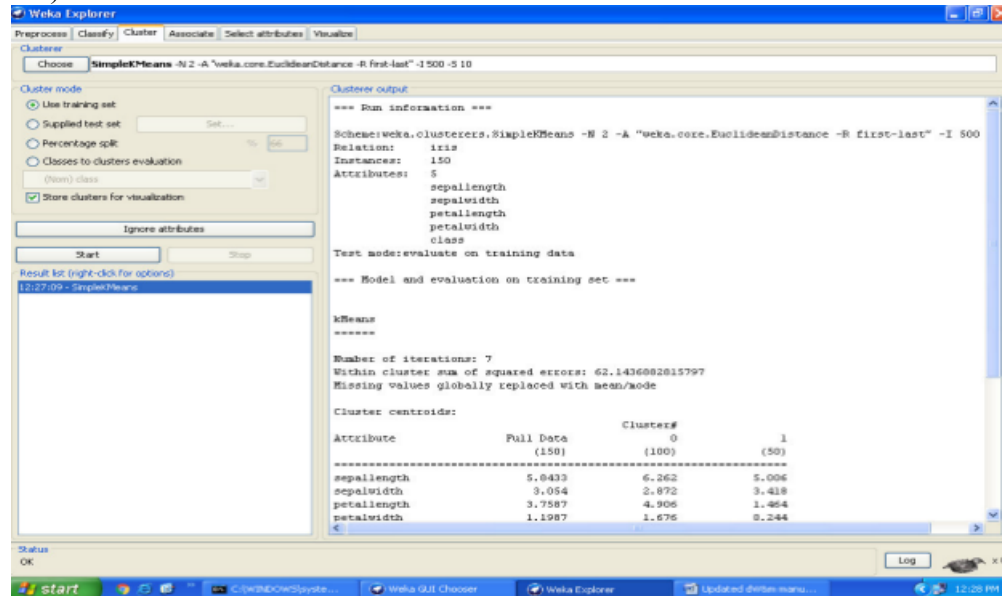
sepalwidth 3.054 2.872 3.418

petallength 3.7587 4.906 1.464

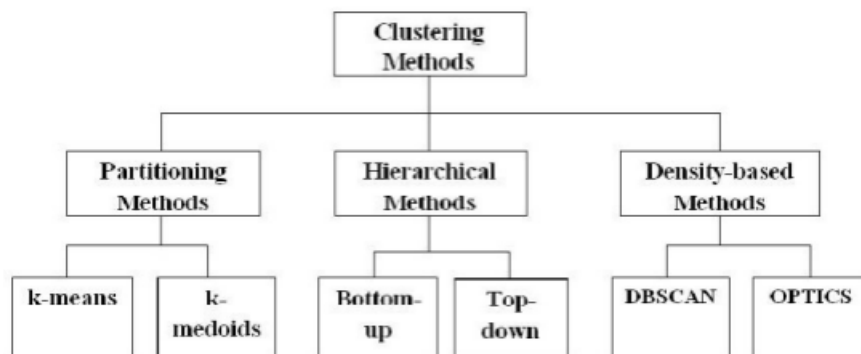
petalwidth 1.1987 1.676 0.244

class Iris-setosa Iris-versicolor Iris-setosa

Time taken to build model (full training data) : 0 seconds



## Clustering Algorithms And Techniques in WEKA, They are



## Visualize Features

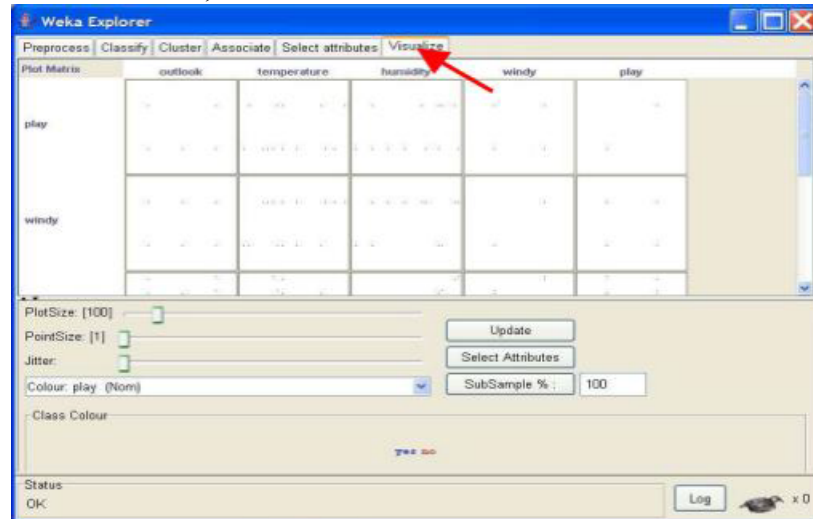
WEKA's visualization allows you to visualize a 2-D plot of the current working relation.

Visualization is very useful in practice, it helps to determine difficulty of the learning problem. WEKA can

visualize single attributes (1-d) and pairs of attributes (2-d), rotate 3-d visualizations (Xgobi-style). WEKA has "Jitter" option to deal with nominal attributes and to detect "hidden" data points.

Access To Visualization From The Classifier, Cluster And Attribute Selection Panel Is Available From A Popup Menu. Click The Right Mouse Button Over An Entry In The Result List To Bring Up The Menu. You Will Be Presented With Options For Viewing Or Saving The Text Output And --- Depending On The Scheme --- Further Options For Visualizing Errors, Clusters, Trees Etc.

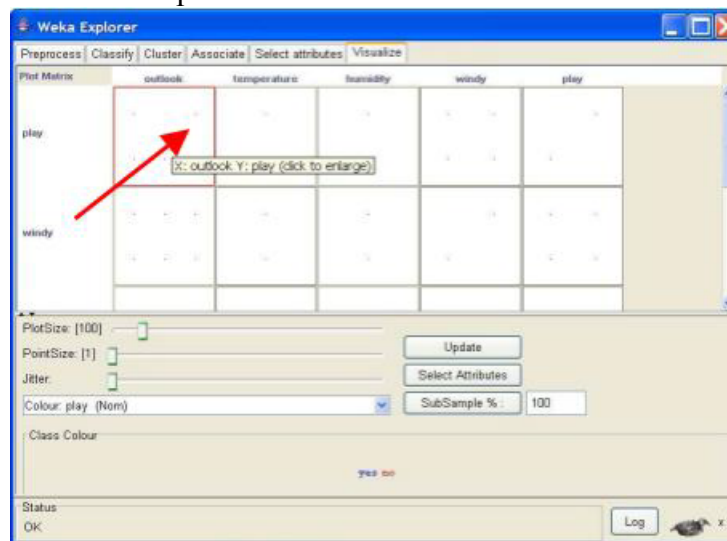
To open Visualization screen, click 'Visualize' tab



Select a square that corresponds to the attributes you would like to visualize. For example, let's choose 'outlook' for X – axis and 'play' for Y – axis. Click anywhere inside the square that corresponds to 'play o

### Changing the View:

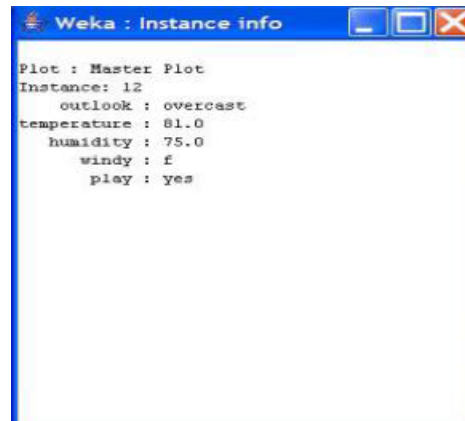
In the visualization window, beneath the X-axis selector there is a drop-down list, 'Colour', for choosing the color scheme. This allows you to choose the color of points based on the attribute selected. Below the plot area, there is a legend that describes what values the colors correspond to. In your example, red represents 'no', while blue represents 'yes'. For better visibility you should change the color of label 'yes'. Left-click on 'yes' in the 'Class colour' box and select lighter color from the color palette. n the left and 'outlook' at the top.



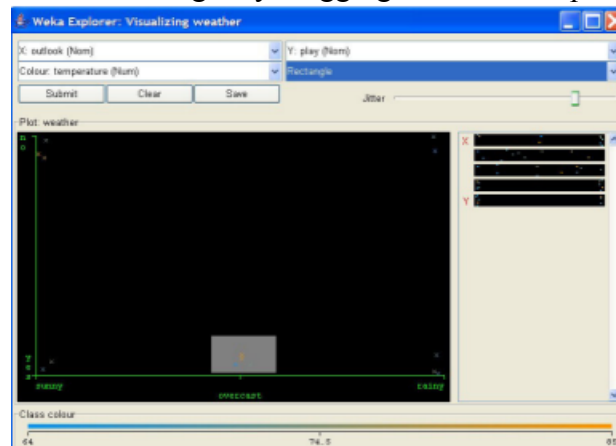
**Date:****Selecting Instances**

Sometimes it is helpful to select a subset of the data using visualization tool. A special case is the 'UserClassifier', which lets you to build your own classifier by interactively selecting instances. Below the Y – axis there is a drop-down list that allows you to choose a selection method. A group of points on the graph can be selected in four ways [2]:

1. Select Instance. Click on an individual data point. It brings up a window listing attributes of the point. If more than one point will appear at the same location, more than one set of attributes will be shown.



2. Rectangle. You can create a rectangle by dragging it around the point.

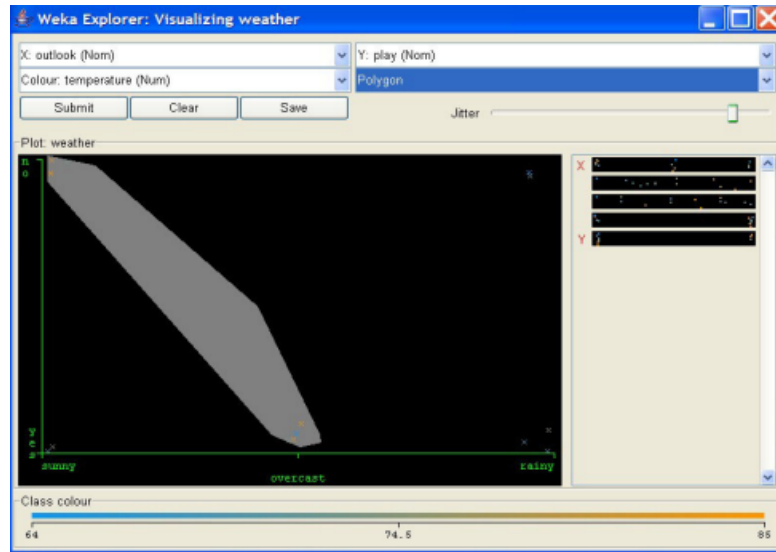


3. Polygon. You can select several points by building a free-form polygon. Left-click on the graph to add vertices to the polygon and right-click to complete it.

Expt. No.:

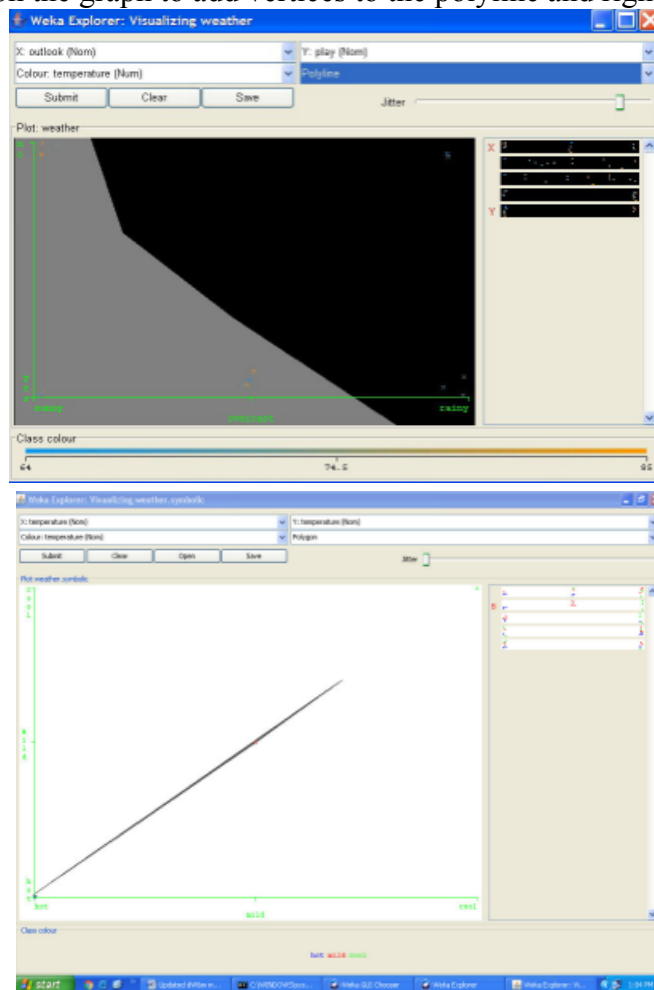
Date:

Page No.: 51



**Date:**

4. Polyline. To distinguish the points on one side from the once on another, you can build a polyline. Left-click on the graph to add vertices to the polyline and right-click to finish.



**Date:**

**Experiment-9: Write a program of cluster analysis using simple k-means algorithm  
Python programming language**

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

**Description:**

K-means is an unsupervised learning method for clustering data points. The algorithm iteratively divides data points into K clusters by minimizing the variance in each cluster. First, each data point is randomly assigned to one of the K clusters. Then, we compute the centroid (functionally the center) of each cluster, and reassign each data point to the cluster with the closest centroid. We repeat this process until the cluster assignments for each data point are no longer changing.


K-means clustering requires us to select K, the number of clusters we want to group the data into. The elbow method lets us graph the inertia (a distance-based metric) and visualize the point at which it starts decreasing linearly. This point is referred to as the "elbow" and is a good estimate for the best value for K based on our data.

**Program:**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
import seaborn as sns
from sklearn.cluster import KMeans
```

```
iris = datasets.load_iris()
Data = pd.DataFrame(iris.data, columns = iris.feature_names)
print(Data)
```

```


  sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0          5.1          3.5          1.4          0.2
1          4.9          3.0          1.4          0.2
2          4.7          3.2          1.3          0.2
3          4.6          3.1          1.5          0.2
4          5.0          3.6          1.4          0.2
..          ...          ...          ...          ...
145         6.7          3.0          5.2          2.3
146         6.3          2.5          5.0          1.9
147         6.5          3.0          5.2          2.0
148         6.2          3.4          5.4          2.3
149         5.9          3.0          5.1          1.8

[150 rows x 4 columns]

```

```
x=Data.iloc[:,0:3].values
#Applying Kmeans classifier
kmeans = KMeans(n_clusters=3,init = 'k-means++', max_iter = 100, n_init = 10,
random_state = 0)
y_kmeans = kmeans.fit_predict(x)
```

**Date:**

# Visualising the clusters - On the first two columns

```
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 7, c = 'yellow', label = 'Iris-setosa')
```

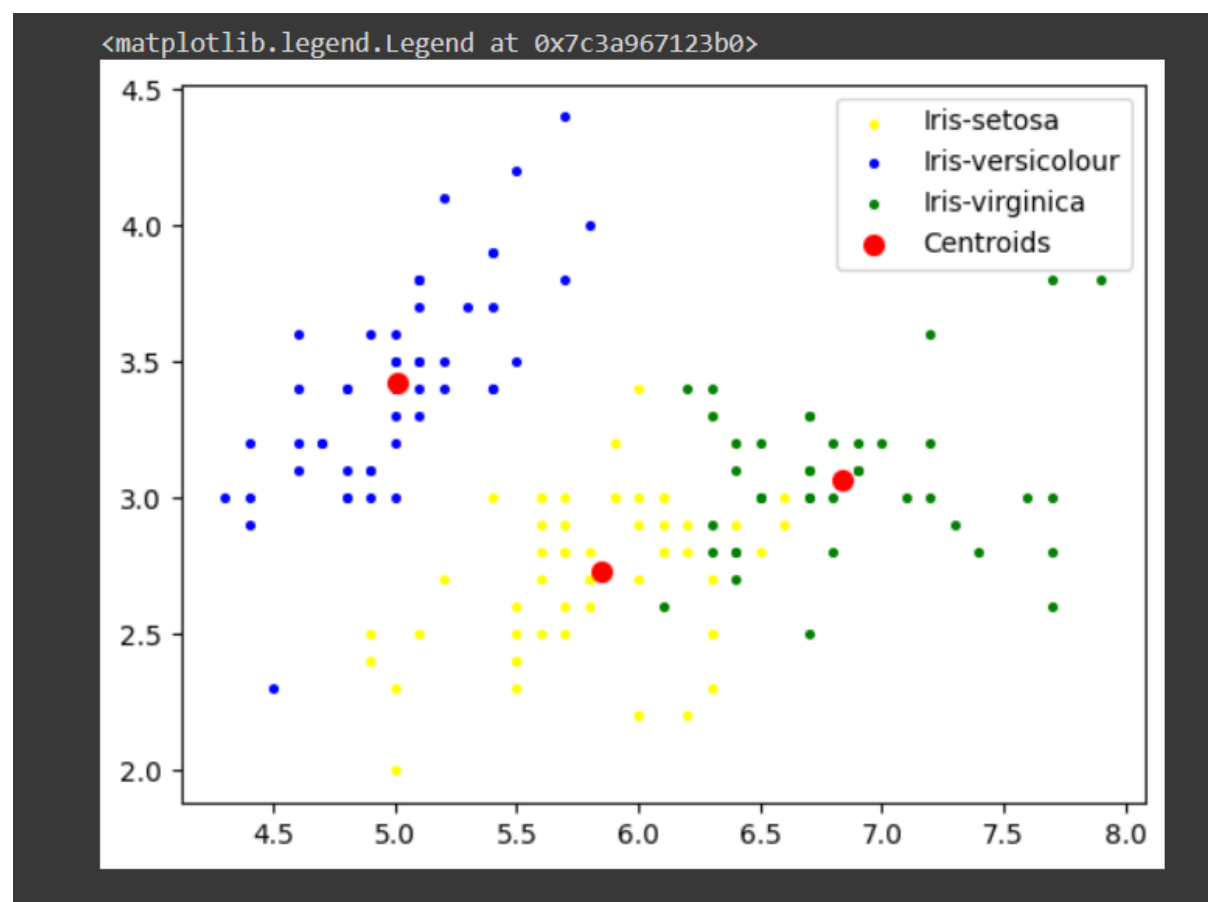
```
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 7, c = 'blue', label = 'Iris-versicolour')
```

```
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 7, c = 'green', label = 'Iris-virginica')
```

# Plotting the centroids of the clusters

```
plt.scatter(kmeans.cluster_centers_[0], kmeans.cluster_centers_[1], s = 50, c = 'red', label = 'Centroids')
```

```
plt.legend()
```







**Expt. No.:**

**Page No.: 55**

**Date:**

**Experiment-10: Write a Python program to generate frequent item sets / association rules using Apriori algorithm**

**AIM:** Write a Python program to generate frequent item sets / association rules using Apriori algorithm.

**Description:**

Apriori Algorithm is a Machine Learning algorithm utilized to understand the patterns of relationships among the various products involved. The most popular use of the algorithm is to suggest products based on the items already in the user's shopping cart. Walmart specifically has utilized the algorithm in recommending items to its users.

**Program:**

```
pip install apyori
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting apyori
  Downloading apyori-1.1.2.tar.gz (8.6 kB)
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... done
  Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5973 sha256=c462245b6df8fdc41b68c07fd37624233d9e0ecc83bc6331dd632a87b18626c0
  Stored in directory: /root/.cache/pip/wheels/cb/f6/e1/57973c631d27efd1a2f375bd6a83b2a616c4021f24aab84080
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

```
import numpy as np
import pandas as pd
from apyori import apriori
data=pd.read_csv('/content/TRANSCATION.csv')
data
```

	WINE	CHIPS	BREAD	BUTTER	MILK	APPLE
0	WINE	NaN	BREAD	BUTTER	MILK	NaN
1	NaN	NaN	BREAD	BUTTER	MILK	NaN
2	NaN	CHIPS	NaN	NaN	NaN	APPLE
3	WINE	CHIPS	BREAD	BUTTER	MILK	APPLE
4	WINE	CHIPS	NaN	NaN	MILK	NaN
5	WINE	CHIPS	BREAD	BUTTER	NaN	APPLE
6	WINE	CHIPS	NaN	NaN	MILK	NaN
7	WINE	NaN	BREAD	NaN	NaN	APPLE
8	WINE	NaN	BREAD	BUTTER	MILK	NaN
9	NaN	CHIPS	BREAD	BUTTER	NaN	APPLE
10	WINE	NaN	NaN	BUTTER	MILK	APPLE
11	WINE	CHIPS	BREAD	BUTTER	MILK	NaN
12	WINE	NaN	BREAD	NaN	MILK	APPLE
13	WINE	NaN	BREAD	BUTTER	MILK	APPLE
14	WINE	CHIPS	BREAD	BUTTER	MILK	APPLE
15	NaN	CHIPS	BREAD	BUTTER	MILK	APPLE
16	NaN	CHIPS	NaN	BUTTER	MILK	APPLE
17	WINE	CHIPS	BREAD	BUTTER	MILK	APPLE

data.shape

```
(21, 6)
```

```
records=[]
for i in range(0,21):
records.append([str(data.values[i,j]) for j in range(0,6)])
ass_rules=apriori(records,min_support=0.5,confidence=0.7)
results=list(ass_rules)
print(len(results))
```

```
18
```