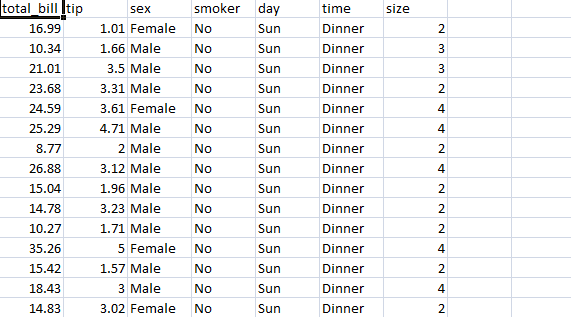
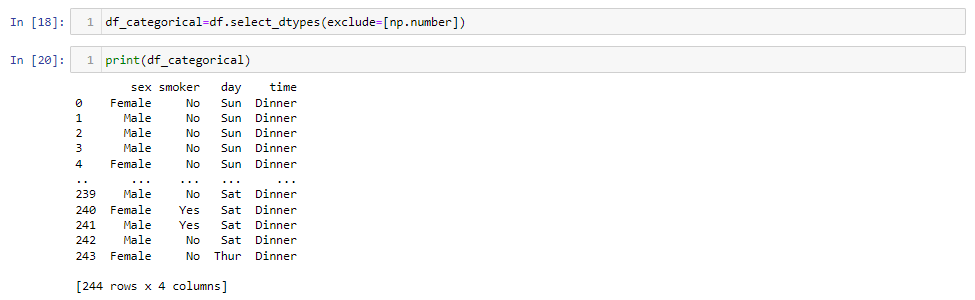
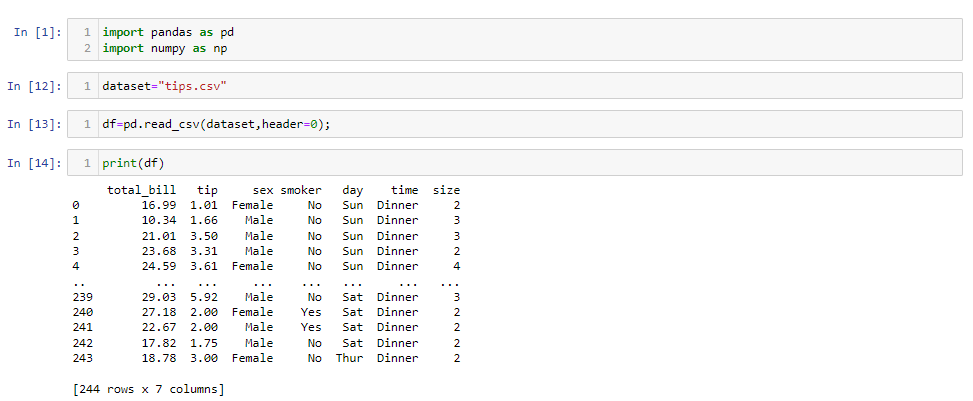
**CATEGORICAL TO NUMERIC DATA CONVERSION**

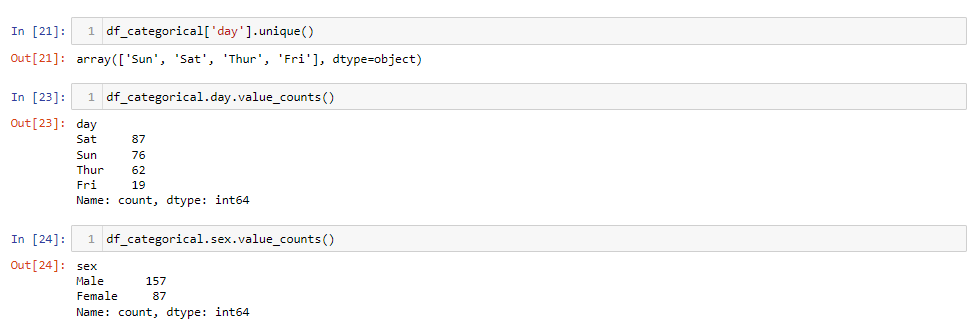
**AIM:** To convert categorical variables in a dataset in a “.csv” file into numerical data using python

**PROCEDURE:**

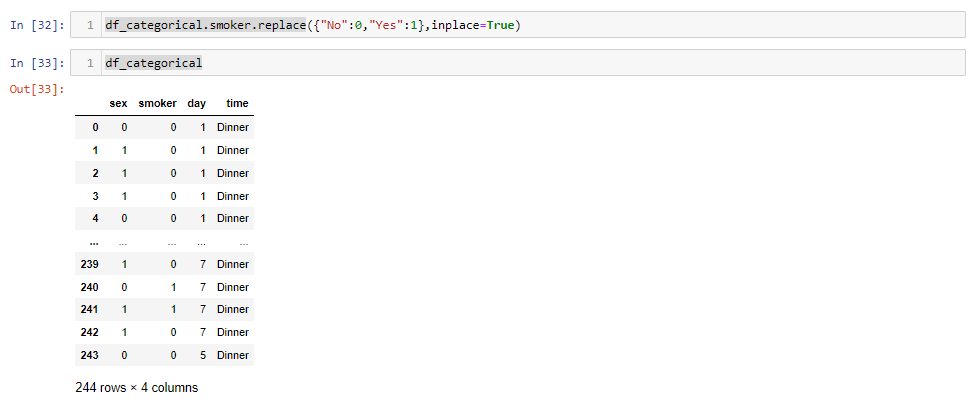
1. Open the jupyter notebook and load file.
2. Check the categorical values in given data set.
3. Find unique values of grade, gender, employed in the given data set.
4. Count each attribute present in Data Set.
5. Replace categorical values to numerical values.
6. Store in another data set.



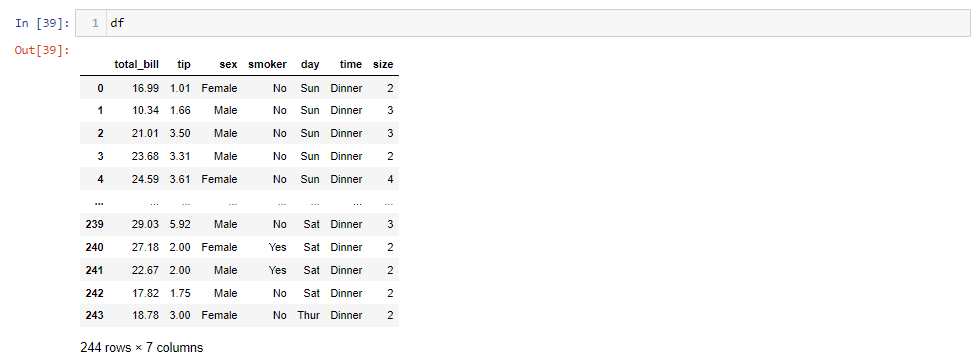












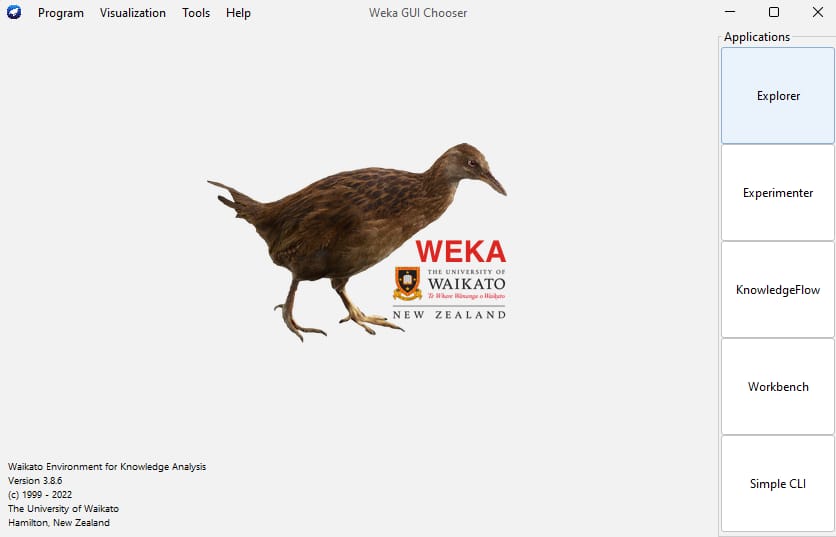


**VISUALIZATION OF DECISION TREE**

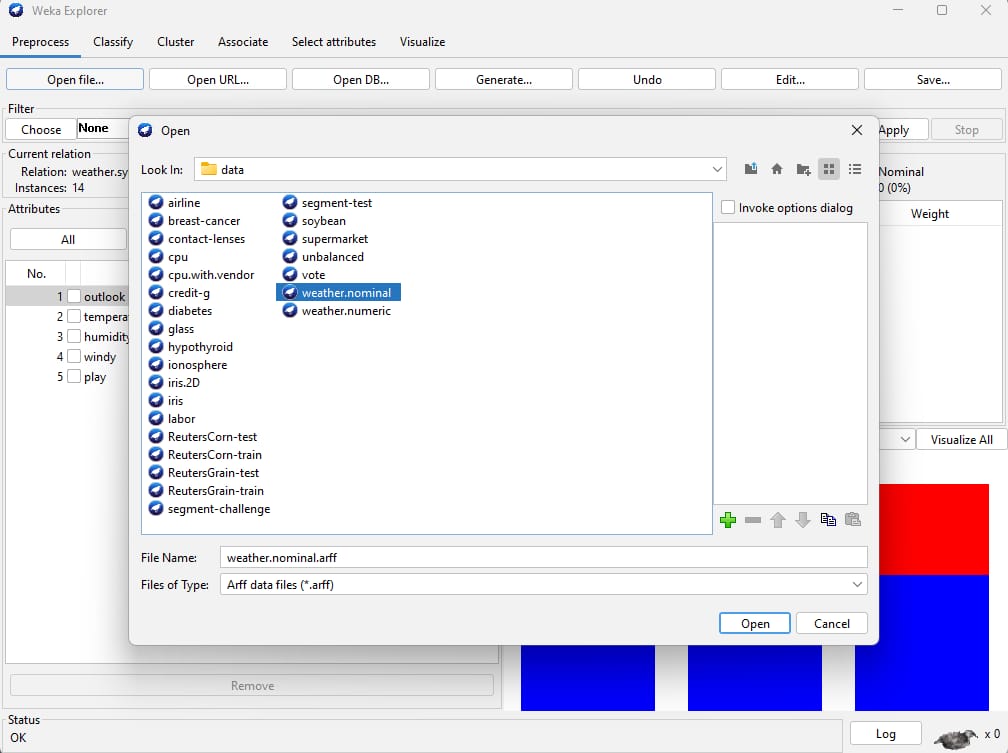
**AIM: To visualize the decision tree using ‘WEKA’ tool**

**PROCEDURE:**

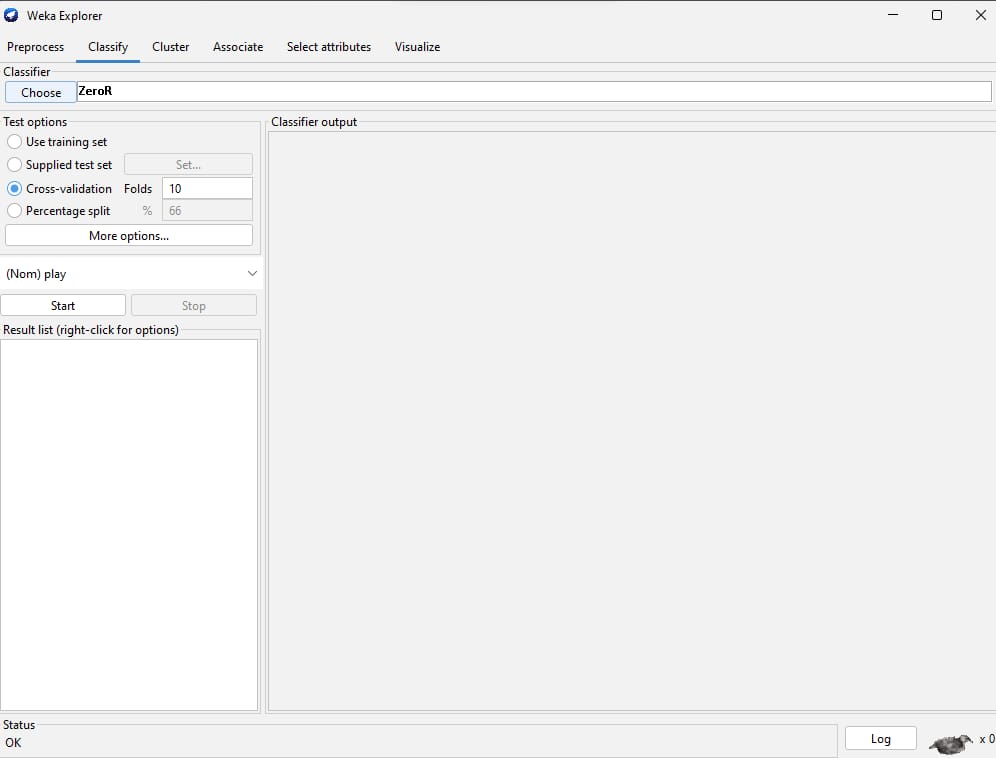
1. **Open the weka tool and click on the explorer**

****

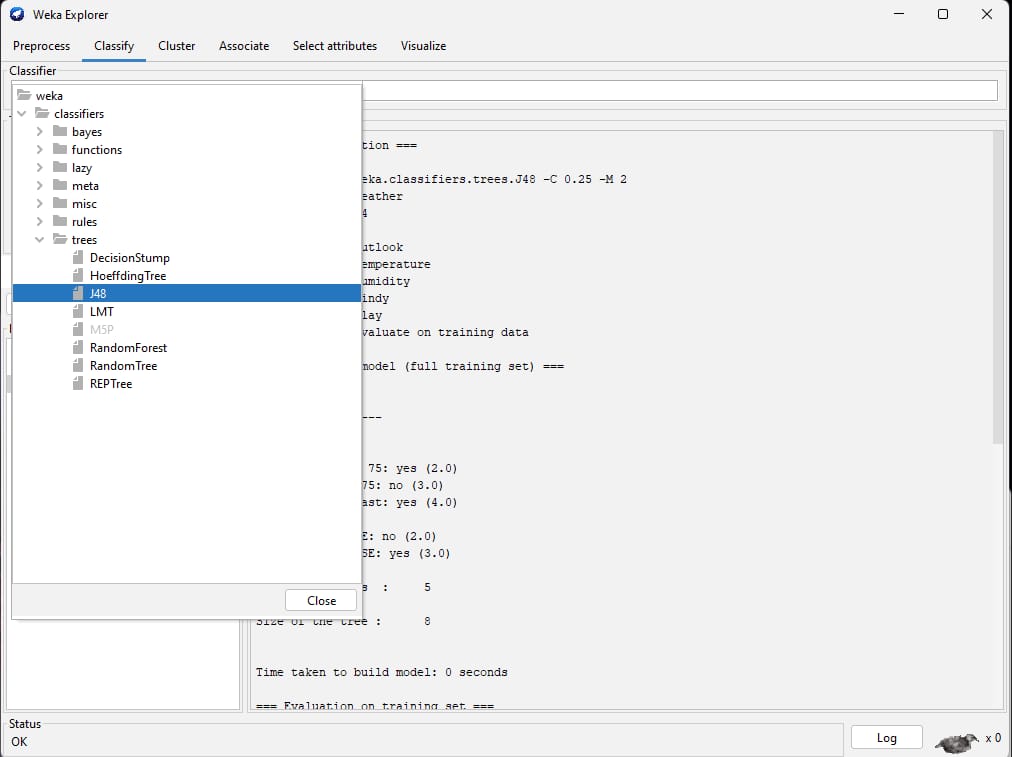
1. **Now click the open file, and locate the file from program files of local disc**

****

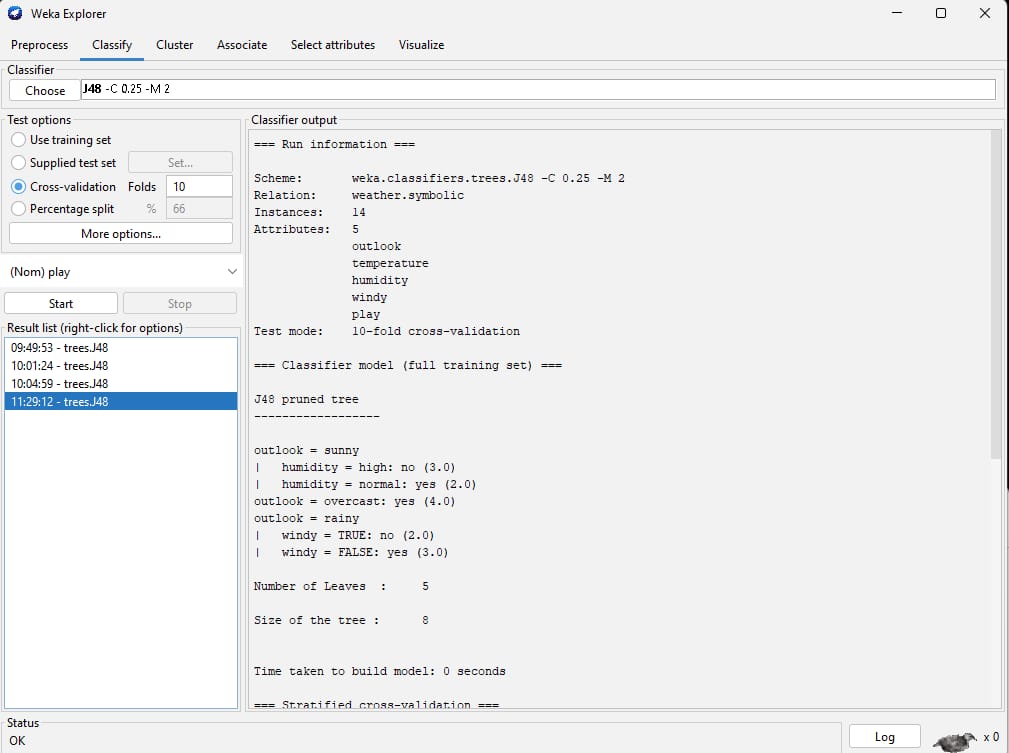
1. **Now click the classify section of WEKA tool and, and select the option: Use Training Set**



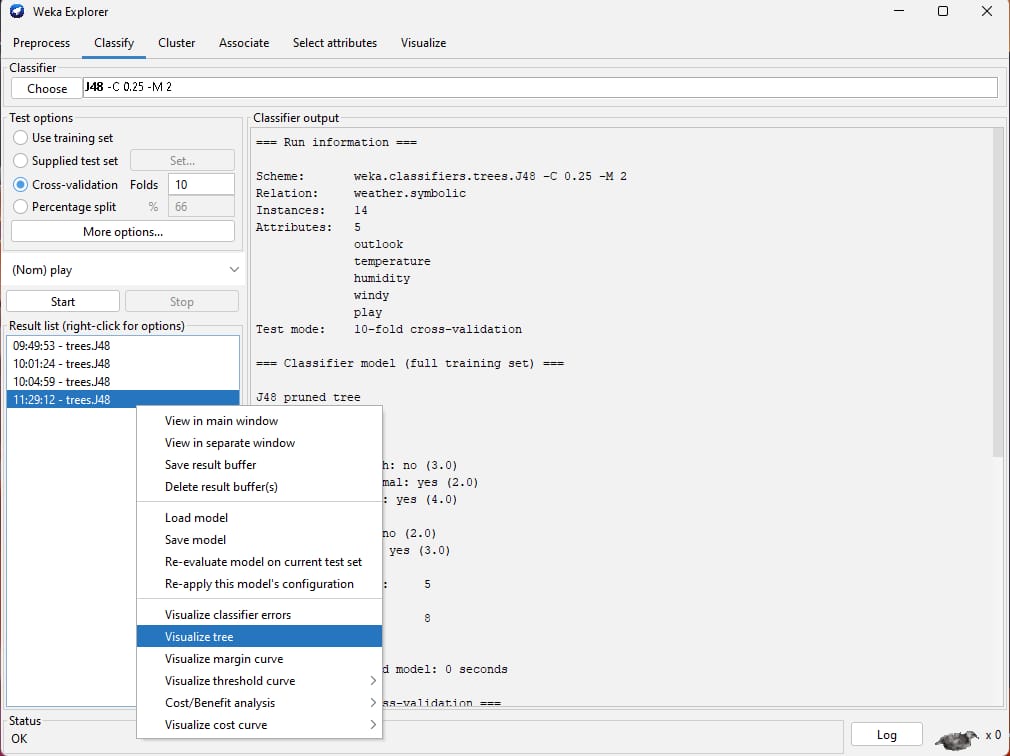
1. **Now choose the *J48* option from the tree section in the choose option, which gives us the C 4.5 Algorithm which ideally known as decision tree algorithm. (*J48 is based on the top-down strategy, a recursive divide and conquer strategy)* and select the start function.**

****

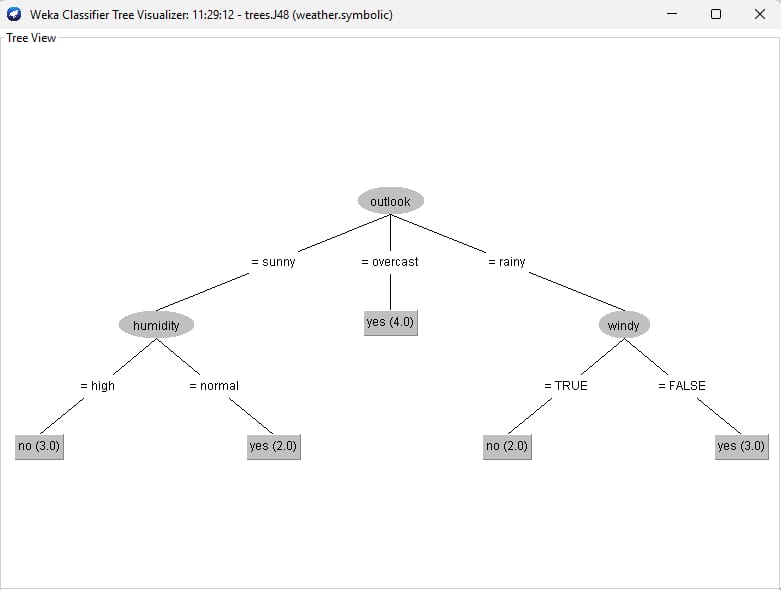
1. **We will get the confusion matrix and a result list (*at the left side of the window)***

****

1. **Now right click on the result produced in the result list, and select visualize tree to produce the decision tree.**

****

1. **Now the decision tree is produced.**

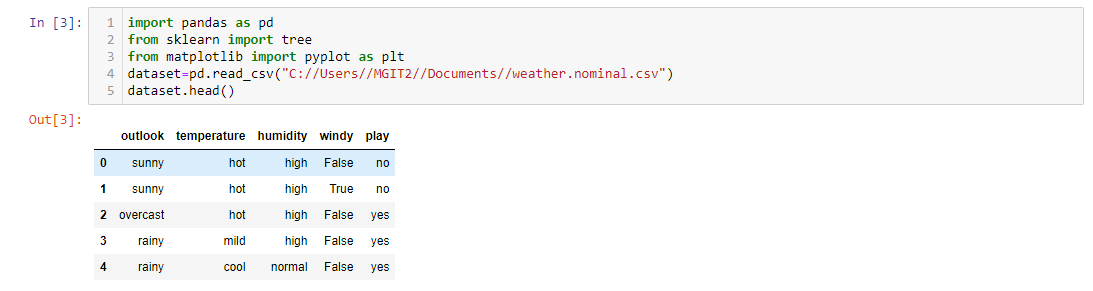
****

**VISUALIZATION OF DECISION TREE (PYTHON)**

**AIM:** To visualize the decision tree using python.

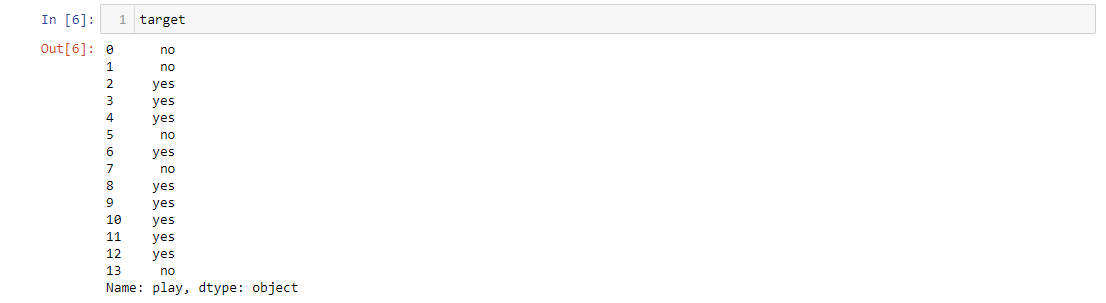
**PROCEDURE:**

1. **Selecting the dataset**

****

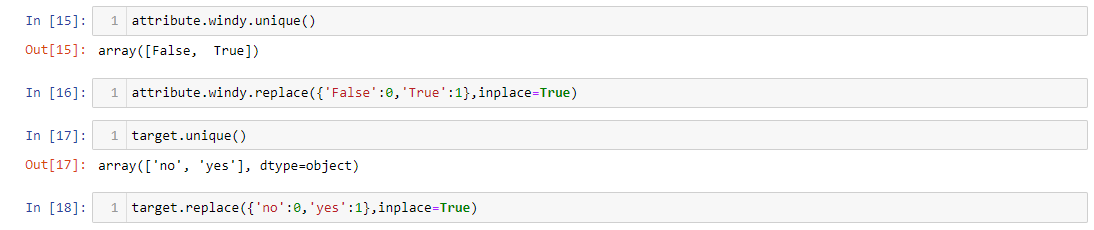
1. **Separating independent variable and target variable**



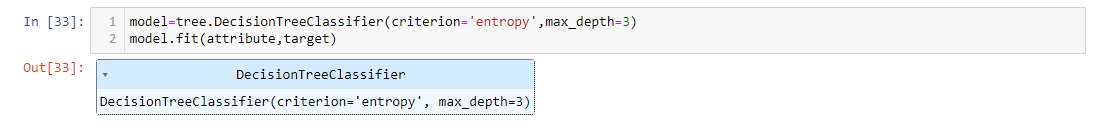


1. **Converting nominal data to numeric data**

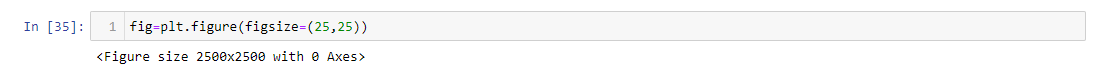


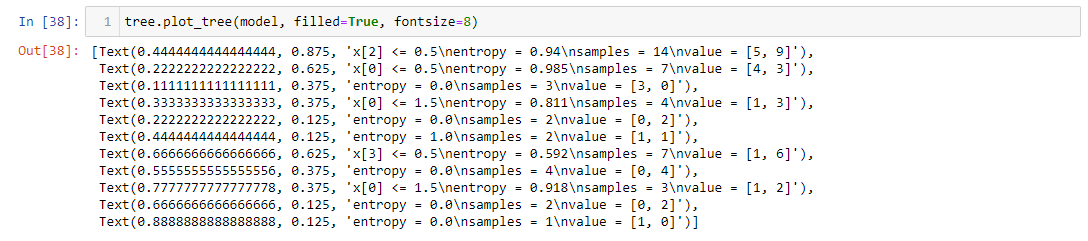


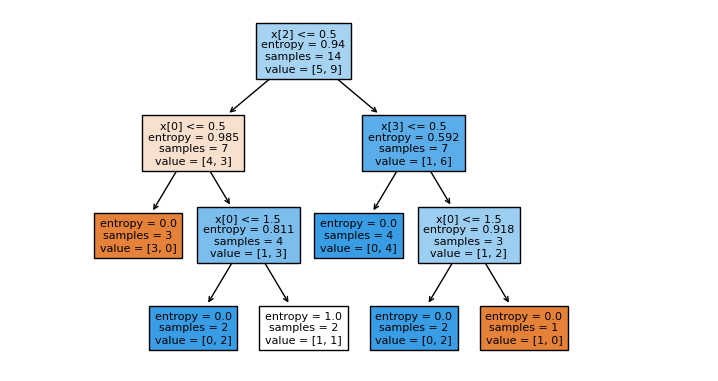
1. **Decision tree model building**

****

1. **Decision tree plotting**

****





**Gain Ratio Attribute**

**AIM:** To implement Gain Ratio Attribute.

**PROCEDURE:**

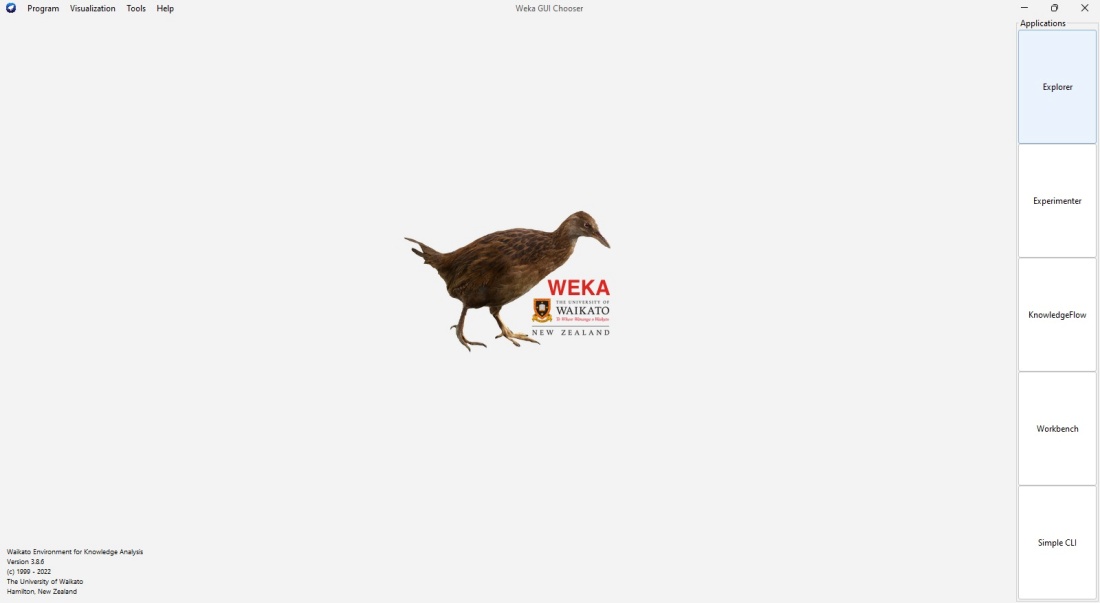
1.open weka explorer

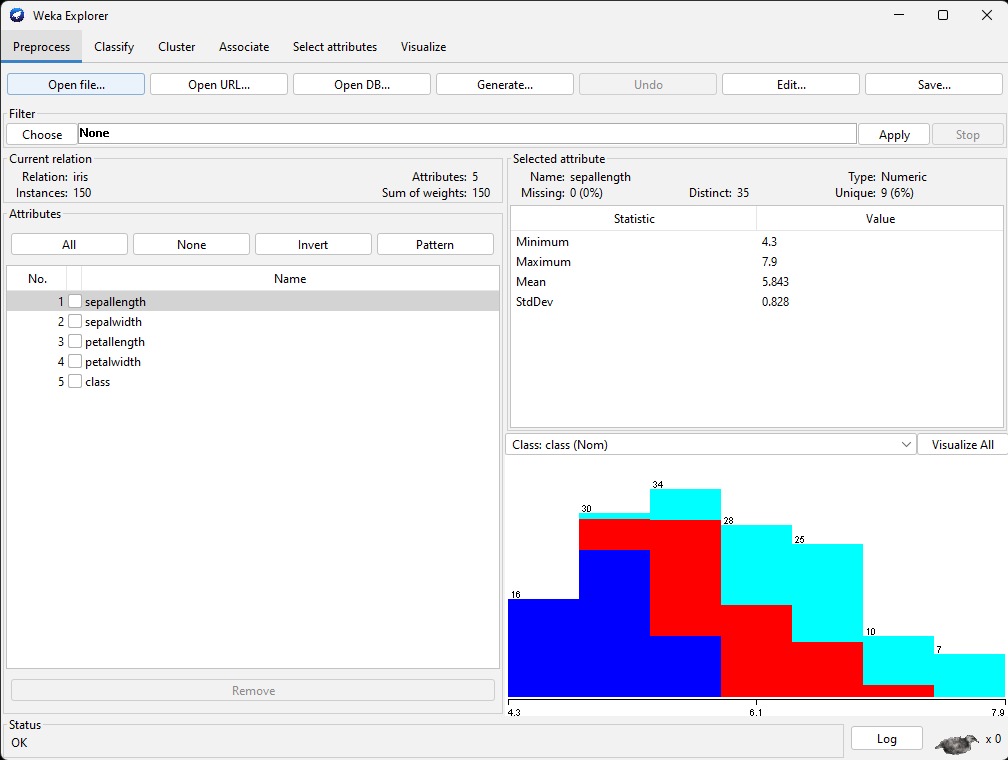
2.Goto open file->C Drive->program files->weka 3.8.6->Data->iris.

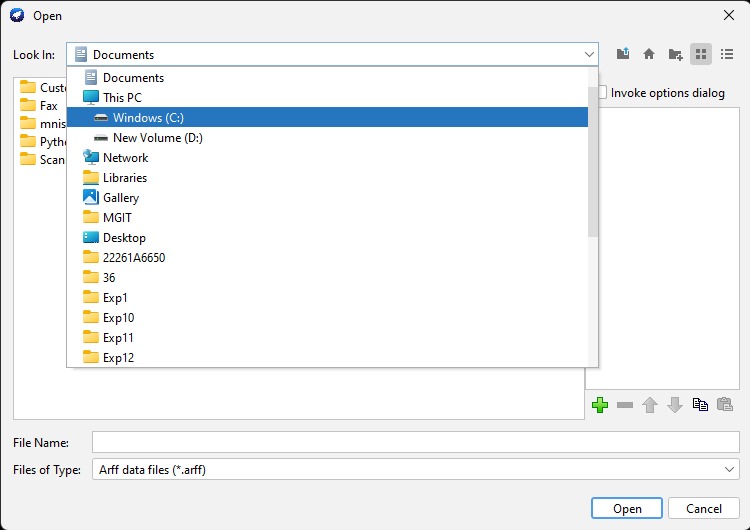
3.Goto classify,then choose Gain Ratio Attribute in classifier->Meta->Attribute selected classifier.

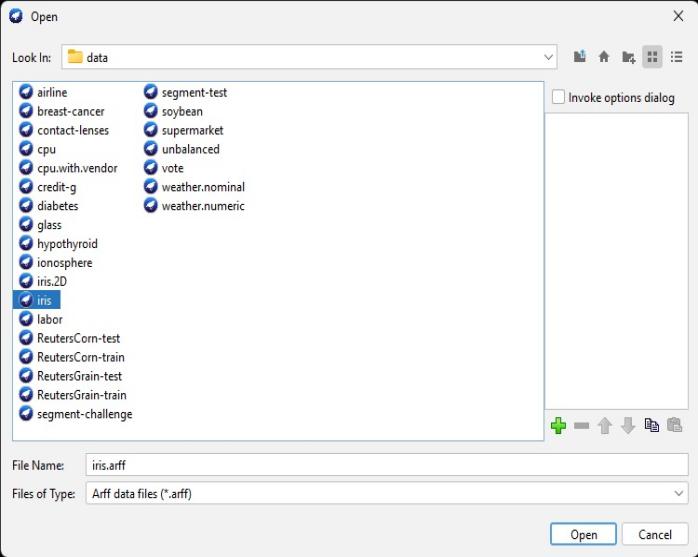
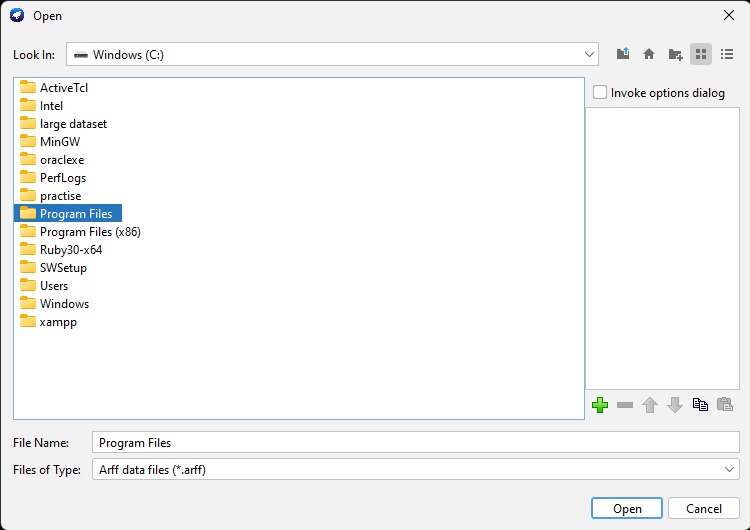
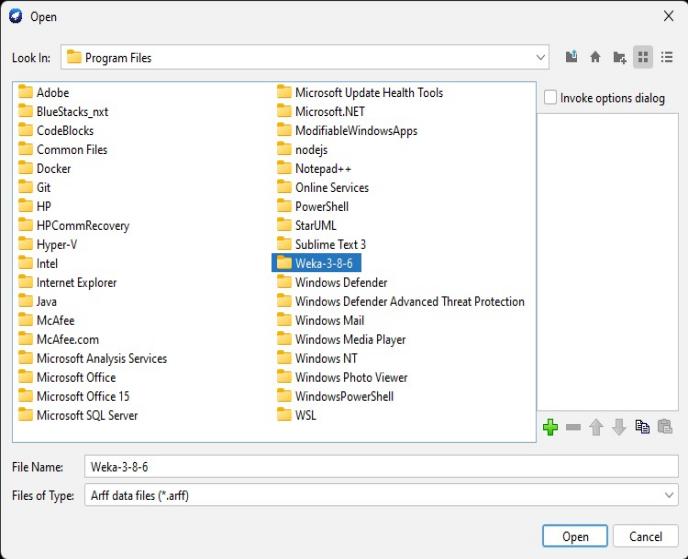
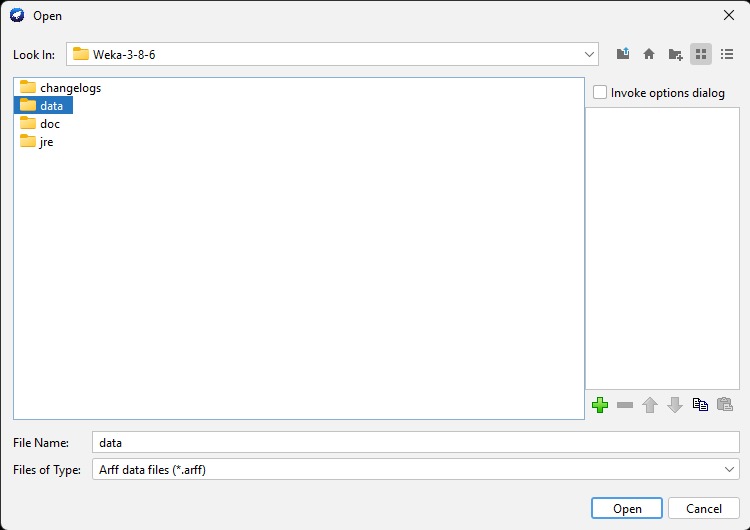
4.choose Ranker Algorithmfor Gain Ratio.

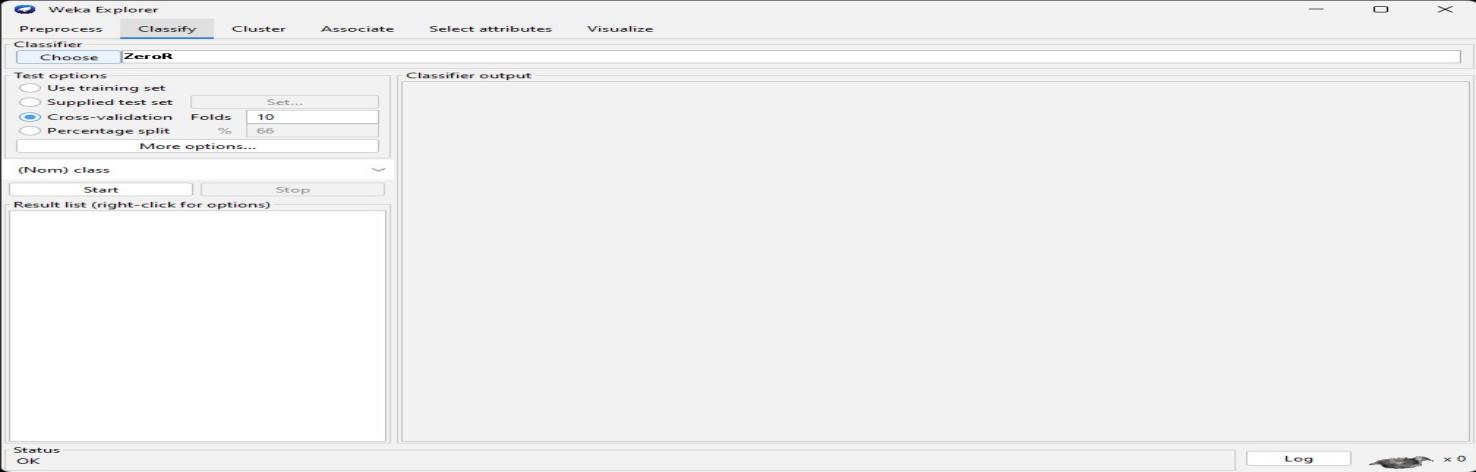
5.Then click on use training set and then start to view the output.

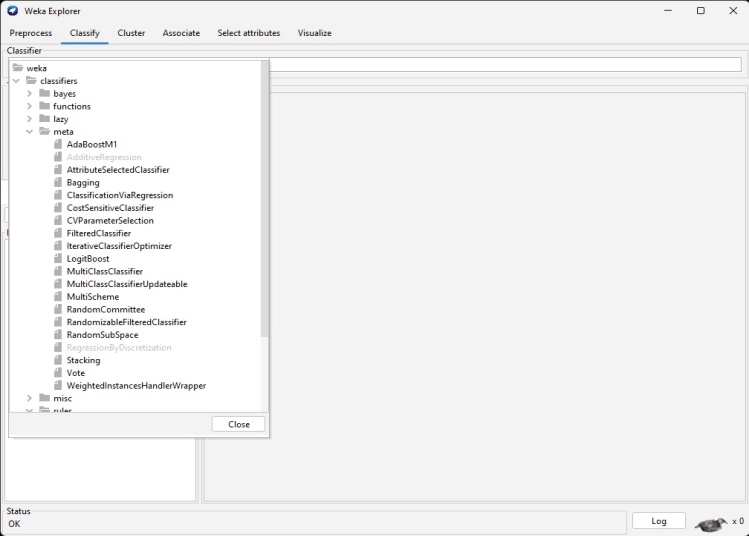
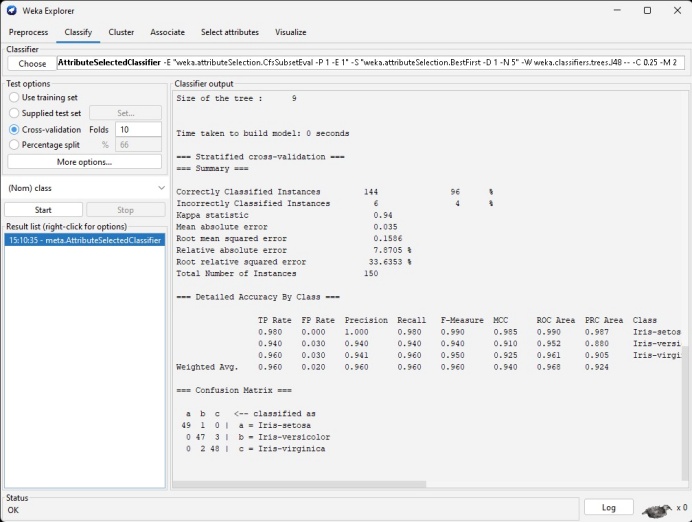


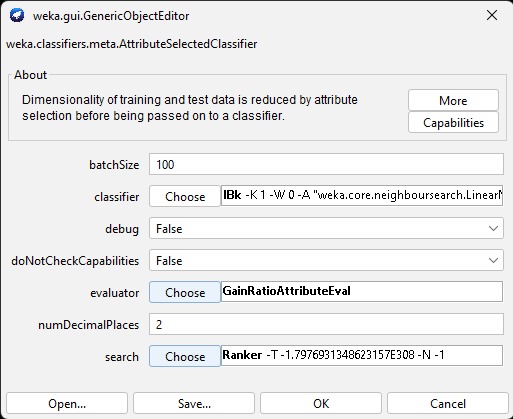
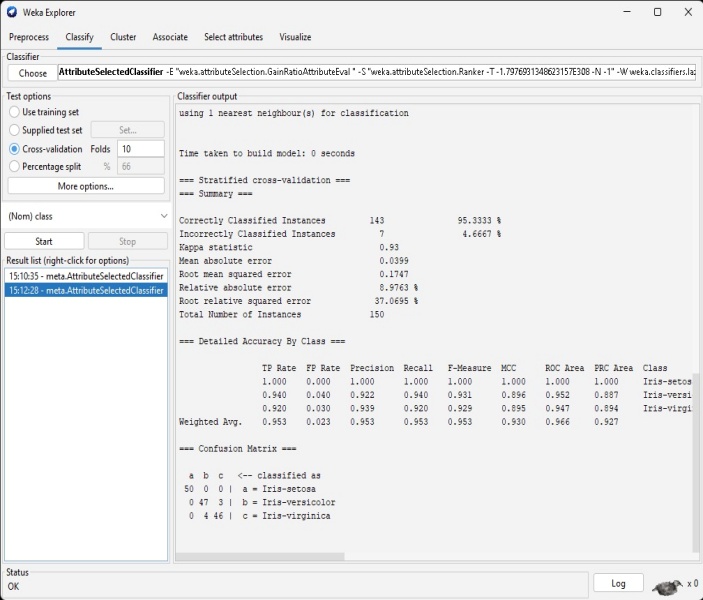
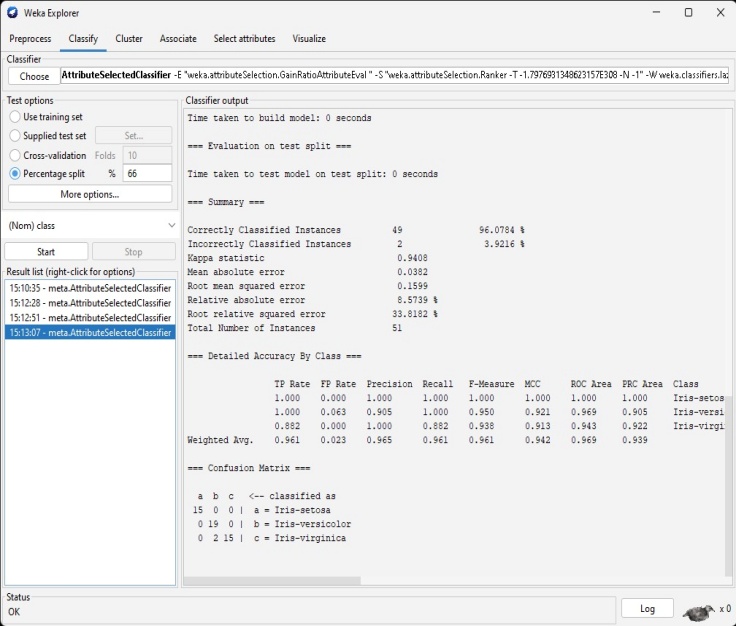
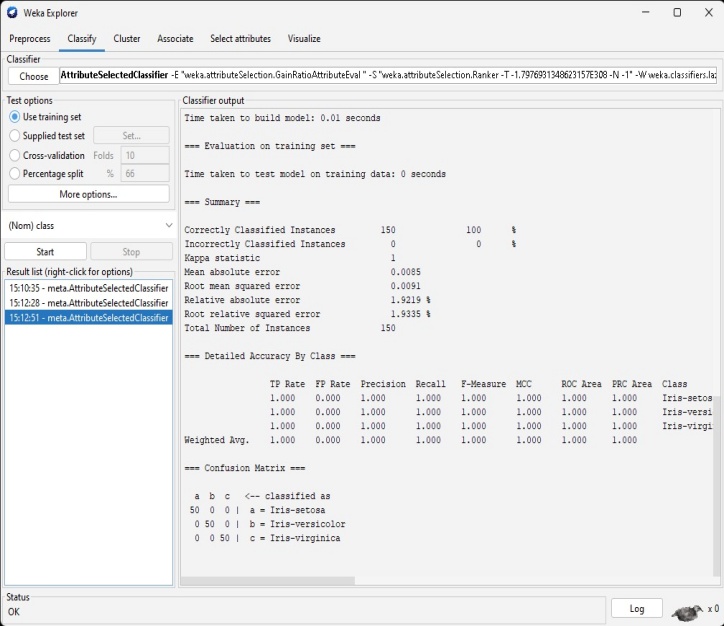








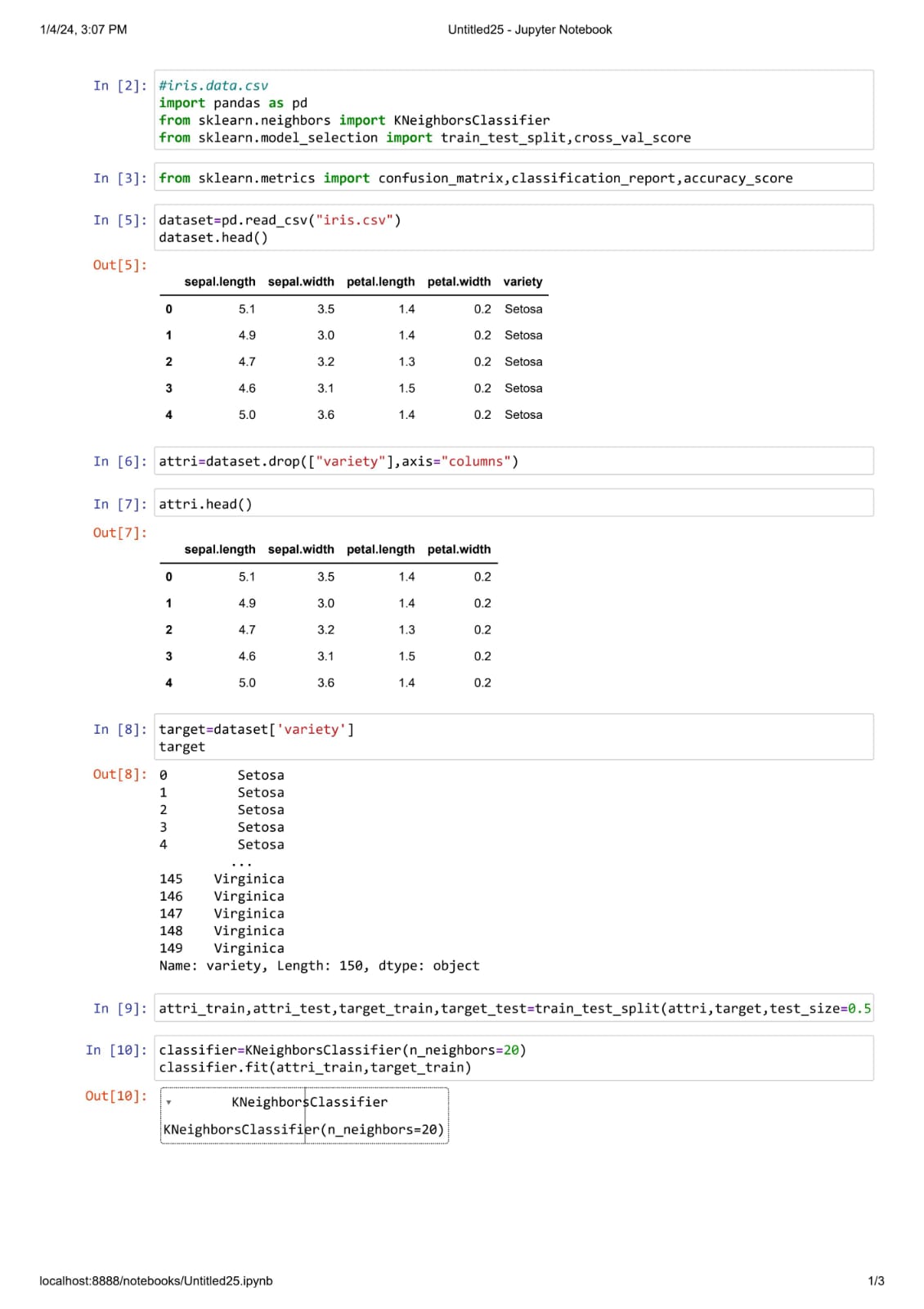


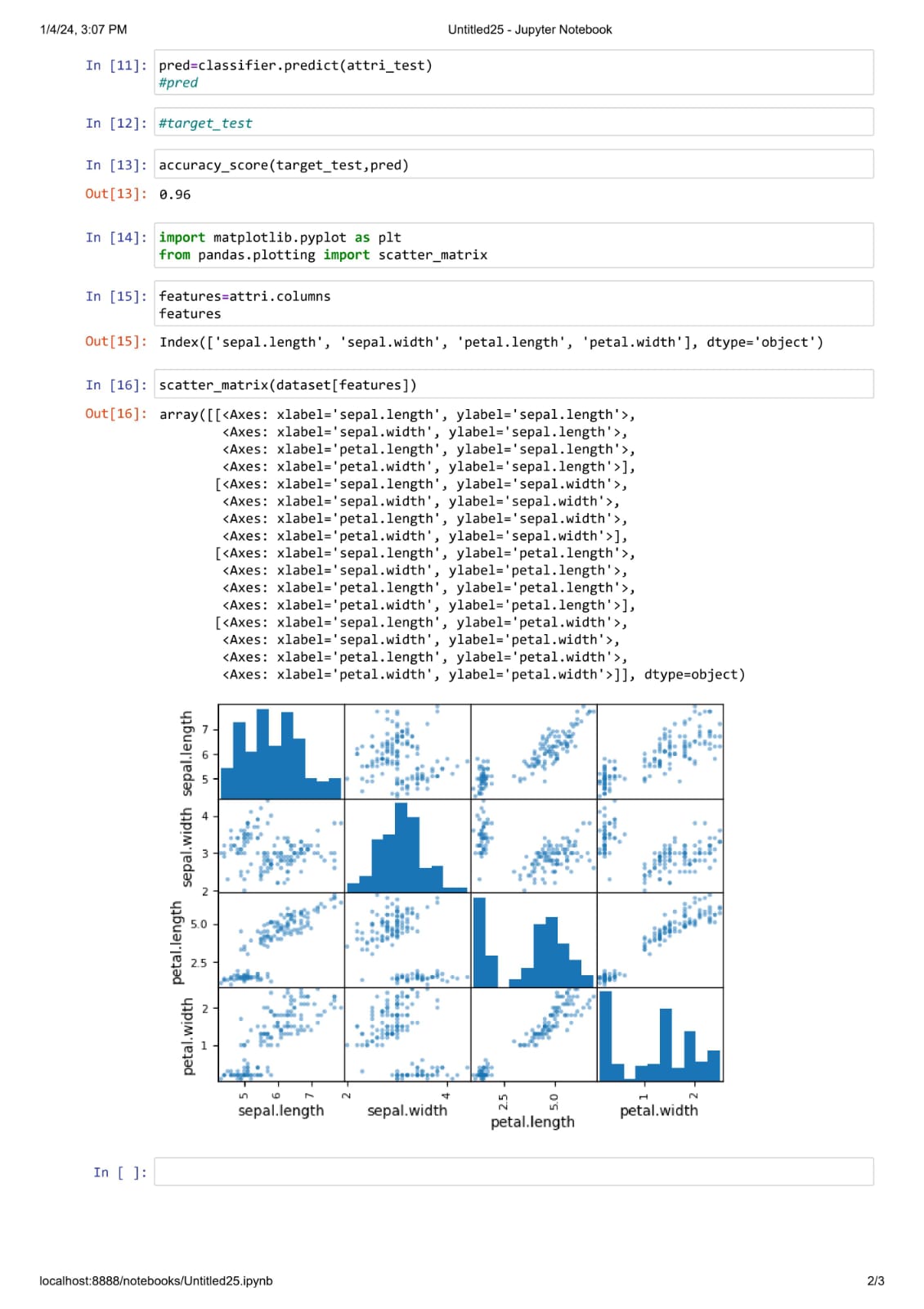
  


Gain Ratio Attribute(Python)

Aim: To implement Gain Ratio Attribute using python program.

Program:





Bayesian Classification(Python)

Aim: Write a python code for Bayesian Classification using iris Dataset

Program:

In [1]: 

**import** pandas **as** pd

**from** sklearn.naive\_bayes **import** GaussianNB

In [2]: 

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.model\_selection **import** cross\_val\_score

In [3]: 

**from** sklearn.preprocessing **import** LabelEncoder

In [4]: 

In [5]: 

**from** sklearn.metrics **import** confusion\_matrix

In [6]: 

**from** sklearn.metrics **import** accuracy\_score

In [16]: 

dataset**=**pd.read\_csv("C:\\Users\\prani\\OneDrive\\Documents\\iris.data.csv")

In [17]: 

Attribute**=**dataset.drop(['Id','species'],axis**=**'columns')

target**=**dataset['species']

In [18]: 

In [19]: 

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(Attribute,target,test\_size**=**0.4

classifier**=**GaussianNB()

classifier.fit(x\_train,y\_train)

In [20]:

Out[20]:

▾ GaussianNB

GaussianNB()

In [22]: 

y\_pred**=**classifier.predict(x\_test) accuracy\_score(y\_test,y\_pred)

Out[22]: 0.9666666666666667

In [23]: 

confusion\_matrix(y\_test,y\_pred)

Out[23]: array([[17, 0, 0],

[ 0, 20, 0],

[ 0, 2, 21]], dtype=int64)

cross\_val\_score(classifier,x\_train,y\_train,cv**=**5)

In [24]: 

Out[24]: array([0.88888889, 0.94444444, 1. , 1. , 0.94444444])

cross\_val\_score(classifier,x\_train,y\_train,cv**=**3)

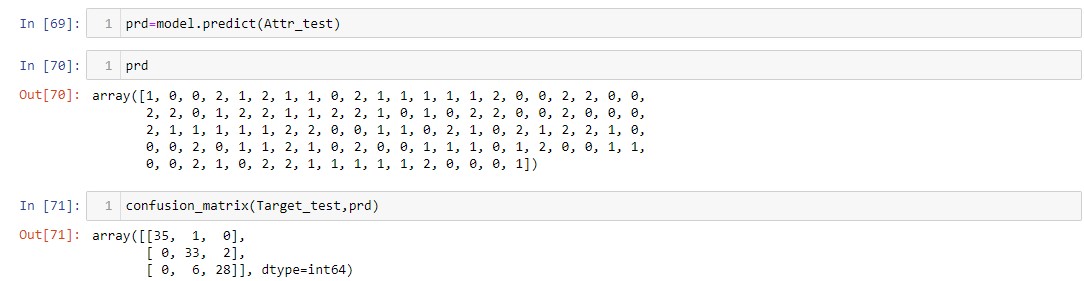
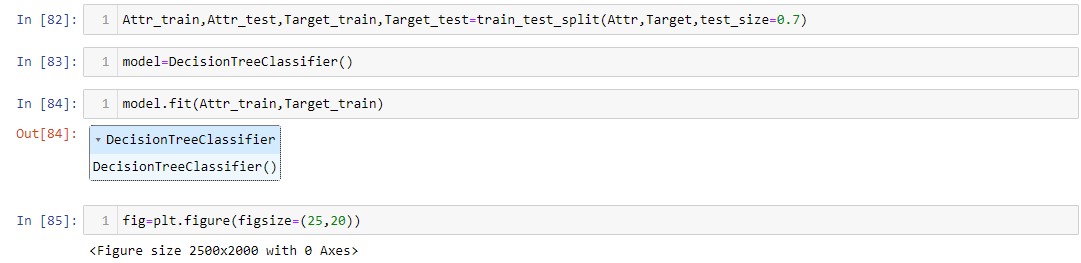
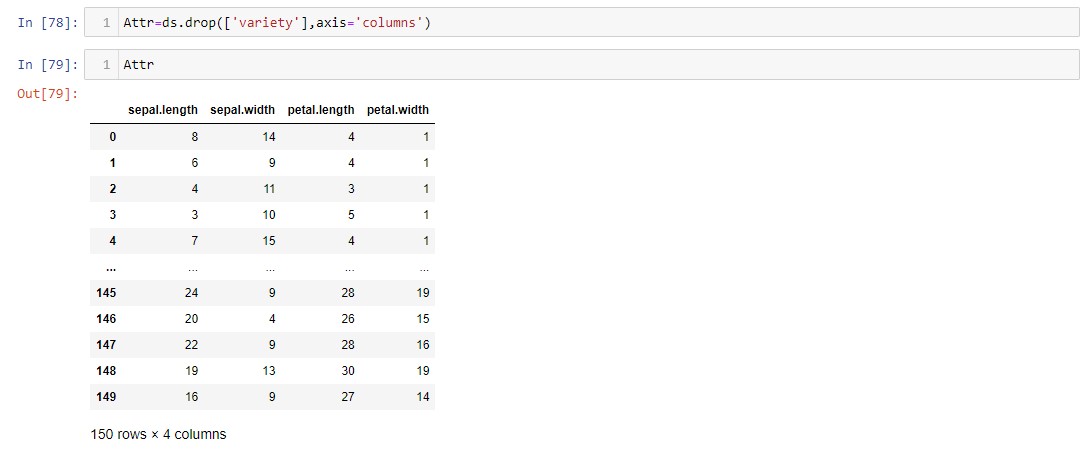
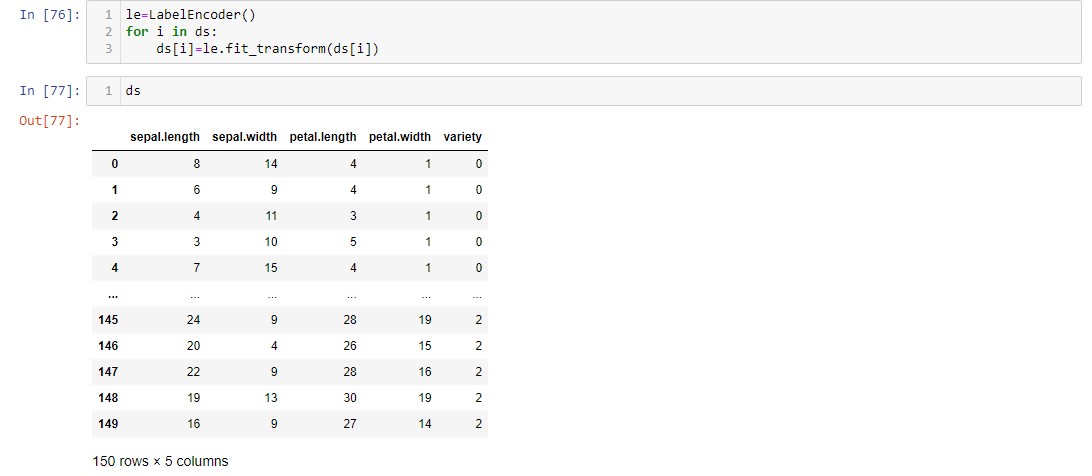
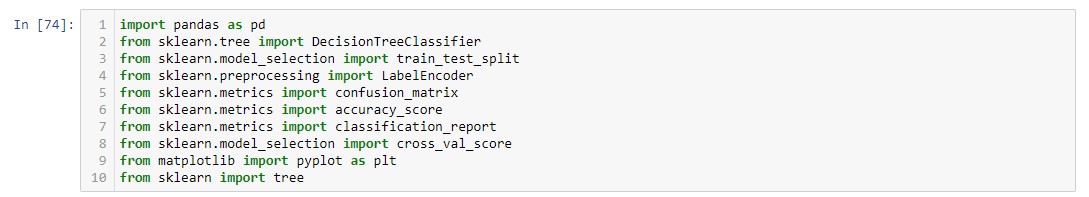
In [25]: 

Out[25]: array([0.9 , 0.96666667, 0.96666667])

In [ ]: 

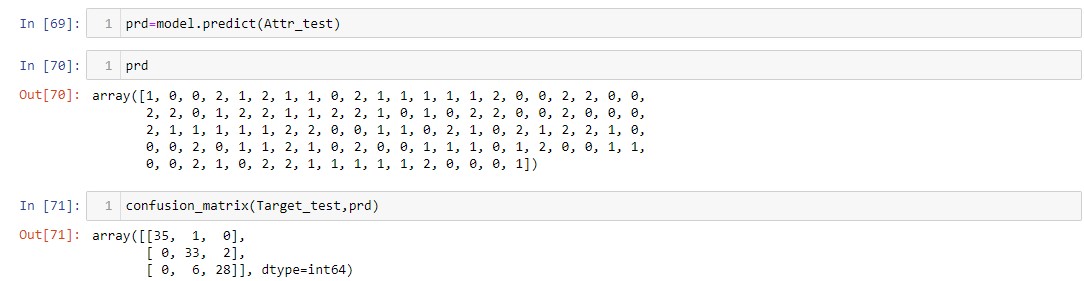
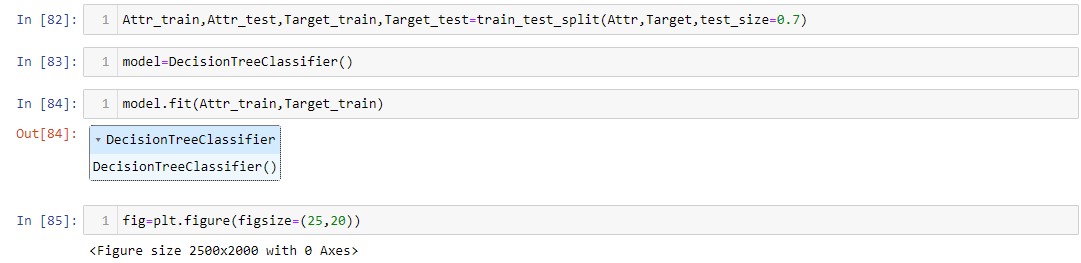
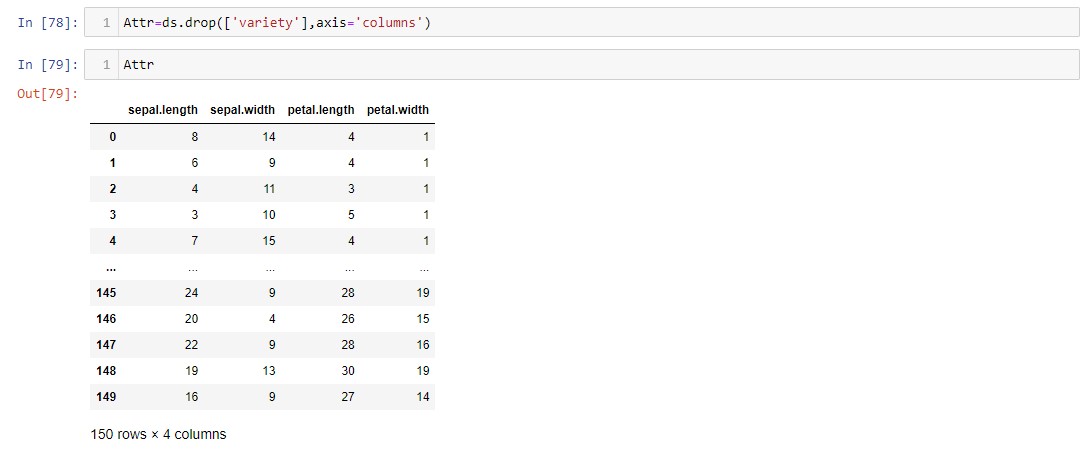
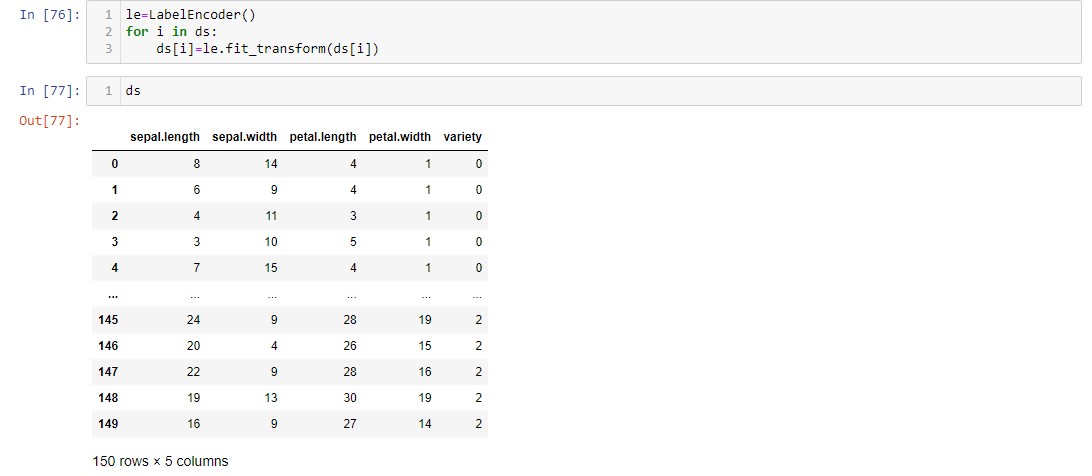
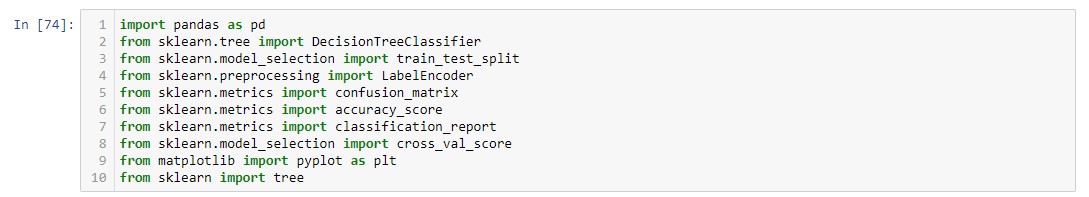
**MODEL EVALUATION AND SELECTION**

**AIM: To write a python program for model evaluation and selection PROGRAM:**



**MODEL EVALUATION AND SELECTION**

**AIM: To write a python program for model evaluation and selection PROGRAM:**

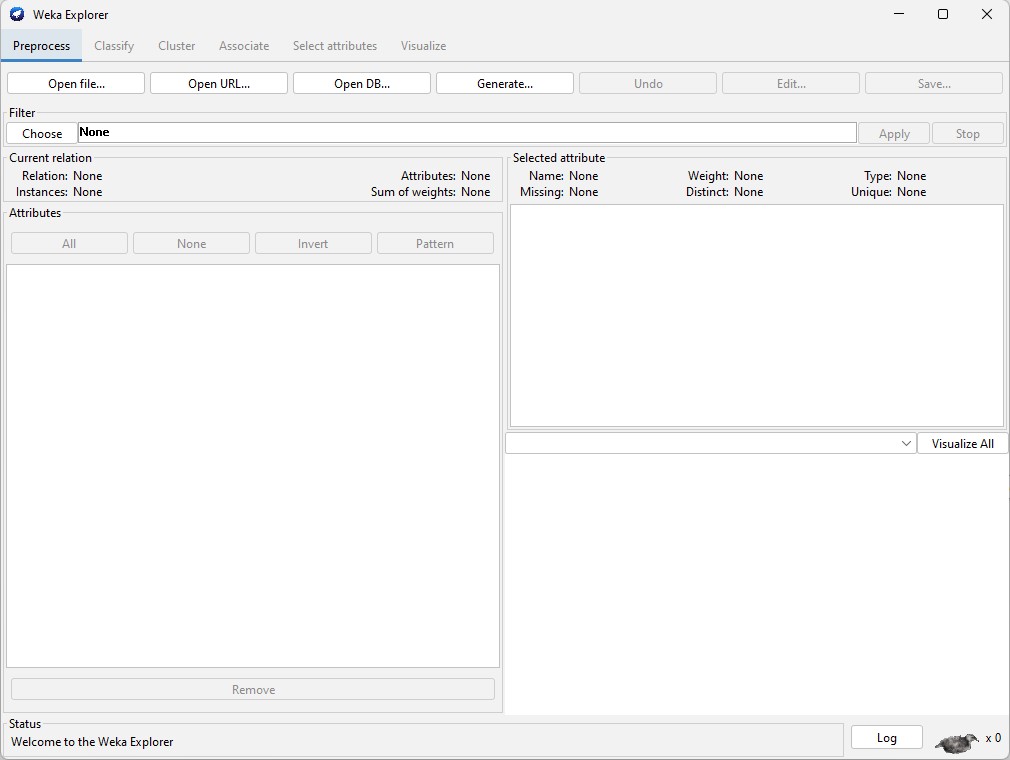


**NAIVE BAYE’S CLASSIFICATION**

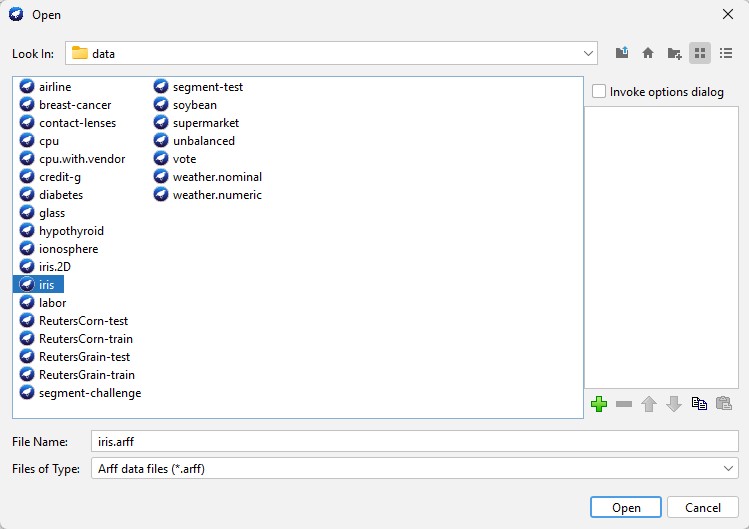
**AIM: To classify the dataset using Naive Baye’s Theorem algorithm**

**PROCEDURE:**

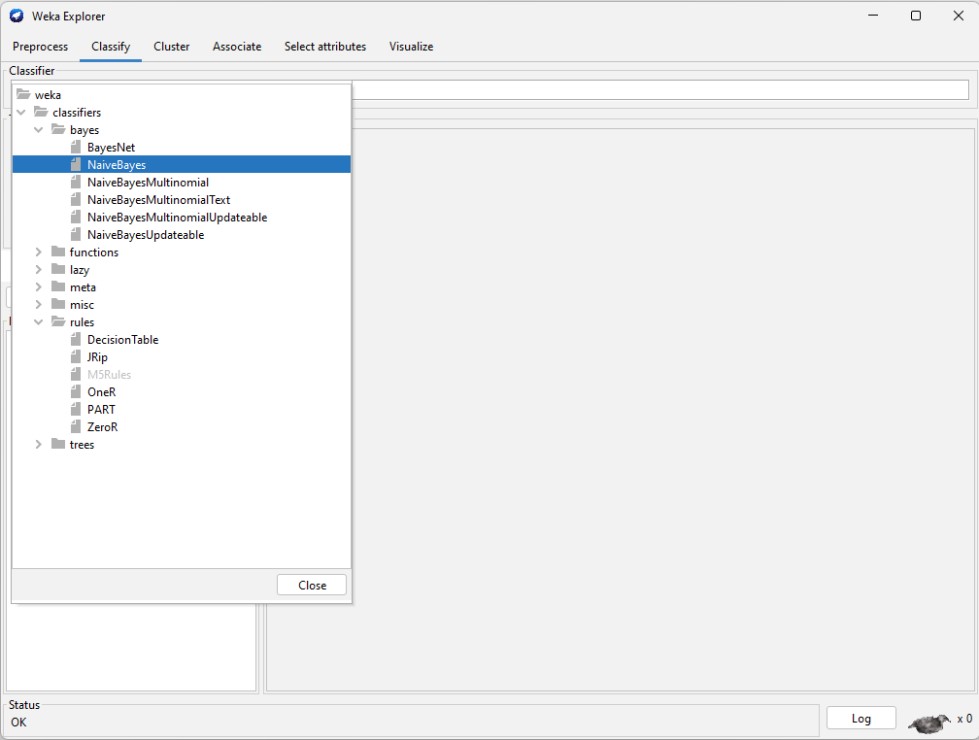
1. Open WEKA tool and open explorer.



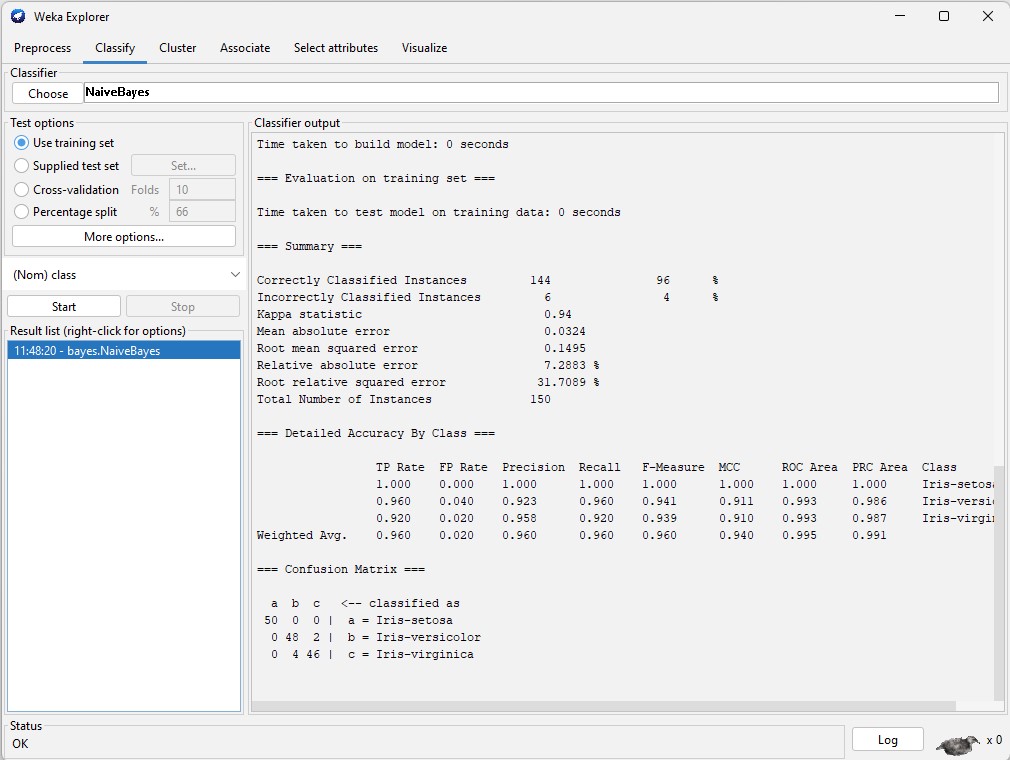
1. Now click “**Open File”,** and locate the dataset from program file.



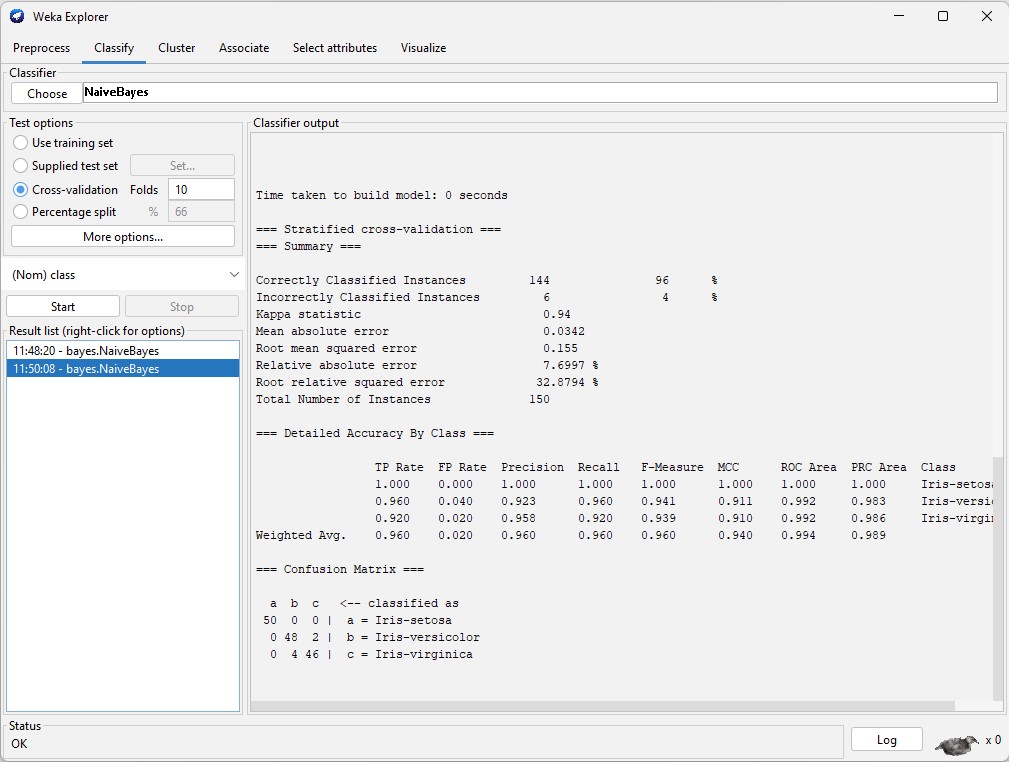
1. Now click on the **“Classify”** and choose **“NaiveBayes”** from **“classifiers”.**



1. From **“Test options”** choose **“Use training set”** and run the classifier.



1. Now choose **“Cross-validation”** and run the classifier to obtain the confusion matrix and a result list.



**Decision Tree (iris)**



**AIM:**  To implement decision tree using python program

