REPLACING MISSING VALUES

AIM: To fill missing values of dataset using python.

PROGRAM:

1. Save data in excel sheet in .csv format.
2. Open Jupyter through Anaconda.
3. Import .csv file to Jupyter notebook.
4. Fill missing values using mean, median, standard deviation, minimum and maximum values.
5. Print data.

item_id	item	quantity	price	bought	forenoon	afternoon
1	milk	2	67	672	456	
2	sugar	1		453	234	
3	chips		45	456	322	
4	coffee	2	45	672	564	
5	meat	4	56	786	221	
6	chocos	3		345		213
7	juice	1	78	765		344
8	jam		65	665		333
9	bread	3				567
10	butter	4				322

```
In [1]: 1 import pandas as pd
        2 data=pd.read_csv("C:\\Users\\MGIT2\\Documents\\dm1 - Sheet1.csv")
        3 print(data)
```

```
   item_id    item  quantity  price  bought  forenoon  afternoon
0        1    milk       2.0    1.0   672.0     456.0        NaN
1        2   sugar       1.0    NaN   453.0     234.0        NaN
2        3   chips       NaN   45.0   456.0     322.0        NaN
3        4  coffee       2.0   45.0   672.0     564.0        NaN
4        5    meat       4.0   56.0   786.0     221.0        NaN
5        6  chocos       3.0    NaN   345.0       NaN      213.0
6        7   juice       1.0   78.0   765.0       NaN      344.0
7        8     jam       NaN   65.0   665.0       NaN      333.0
8        9   bread       3.0    NaN     NaN       NaN      567.0
9       10  butter       4.0    NaN     NaN       NaN      322.0
```

In [2]:
```
data['quantity']=data['quantity'].fillna(data['quantity'].mean())
print(data)
```

	item_id	item	quantity	price	bought	forenoon	afternoon
0	1	milk	2.0	1.0	672.0	456.0	NaN
1	2	sugar	1.0	NaN	453.0	234.0	NaN
2	3	chips	2.5	45.0	456.0	322.0	NaN
3	4	coffee	2.0	45.0	672.0	564.0	NaN
4	5	meat	4.0	56.0	786.0	221.0	NaN
5	6	chocos	3.0	NaN	345.0	NaN	213.0
6	7	juice	1.0	78.0	765.0	NaN	344.0
7	8	jam	2.5	65.0	665.0	NaN	333.0
8	9	bread	3.0	NaN	NaN	NaN	567.0
9	10	butter	4.0	NaN	NaN	NaN	322.0

In [4]:
```
data['price ']=data['price '].fillna(data['price '].median())
print(data)
```

	item_id	item	quantity	price	bought	forenoon	afternoon
0	1	milk	2.0	1.0	672.0	456.0	NaN
1	2	sugar	1.0	50.5	453.0	234.0	NaN
2	3	chips	2.5	45.0	456.0	322.0	NaN
3	4	coffee	2.0	45.0	672.0	564.0	NaN
4	5	meat	4.0	56.0	786.0	221.0	NaN
5	6	chocos	3.0	50.5	345.0	NaN	213.0
6	7	juice	1.0	78.0	765.0	NaN	344.0
7	8	jam	2.5	65.0	665.0	NaN	333.0
8	9	bread	3.0	50.5	NaN	NaN	567.0
9	10	butter	4.0	50.5	NaN	NaN	322.0

In [5]:
```
data['bought']=data['bought'].fillna(data['bought'].std())
print(data)
```

	item_id	item	quantity	price	bought	forenoon	afternoon
0	1	milk	2.0	1.0	672.000000	456.0	NaN
1	2	sugar	1.0	50.5	453.000000	234.0	NaN
2	3	chips	2.5	45.0	456.000000	322.0	NaN
3	4	coffee	2.0	45.0	672.000000	564.0	NaN
4	5	meat	4.0	56.0	786.000000	221.0	NaN
5	6	chocos	3.0	50.5	345.000000	NaN	213.0
6	7	juice	1.0	78.0	765.000000	NaN	344.0
7	8	jam	2.5	65.0	665.000000	NaN	333.0
8	9	bread	3.0	50.5	162.022706	NaN	567.0
9	10	butter	4.0	50.5	162.022706	NaN	322.0

In [6]:
```
data['forenoon']=data['forenoon'].fillna(data['forenoon'].min())
print(data)
```

	item_id	item	quantity	price	bought	forenoon	afternoon
0	1	milk	2.0	1.0	672.000000	456.0	NaN
1	2	sugar	1.0	50.5	453.000000	234.0	NaN
2	3	chips	2.5	45.0	456.000000	322.0	NaN
3	4	coffee	2.0	45.0	672.000000	564.0	NaN
4	5	meat	4.0	56.0	786.000000	221.0	NaN
5	6	chocos	3.0	50.5	345.000000	221.0	213.0
6	7	juice	1.0	78.0	765.000000	221.0	344.0
7	8	jam	2.5	65.0	665.000000	221.0	333.0
8	9	bread	3.0	50.5	162.022706	221.0	567.0
9	10	butter	4.0	50.5	162.022706	221.0	322.0

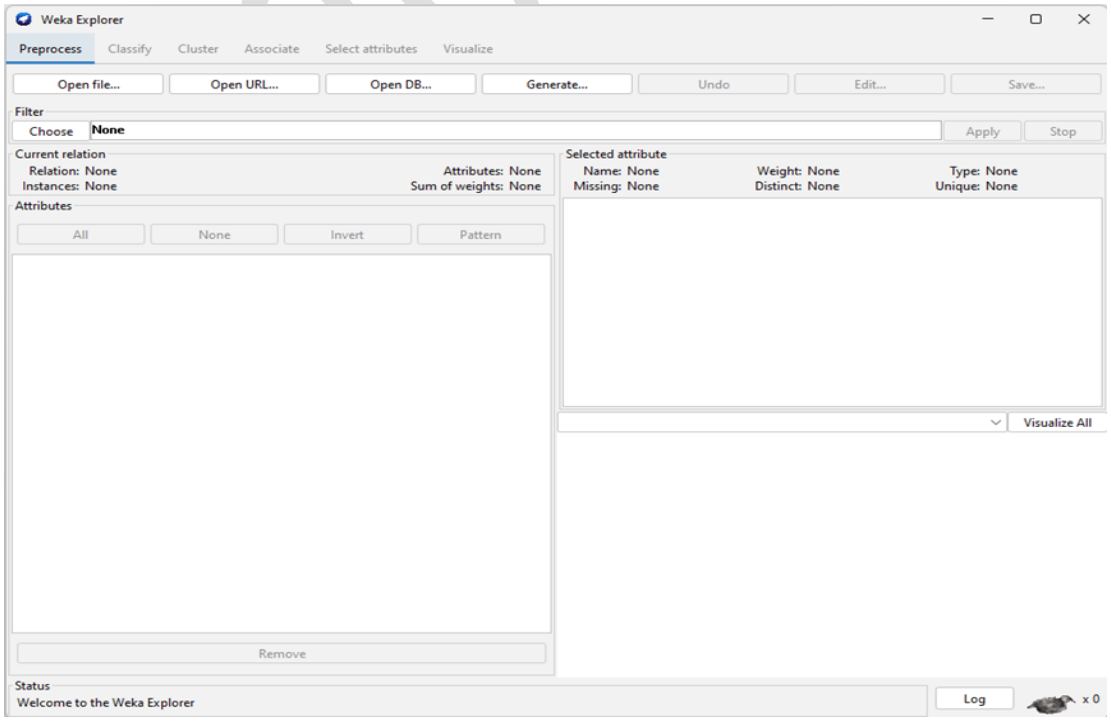
In [7]:
```
1 data['afternoon']=data['afternoon'].fillna(data['afternoon'].max())
2 print(data)
```

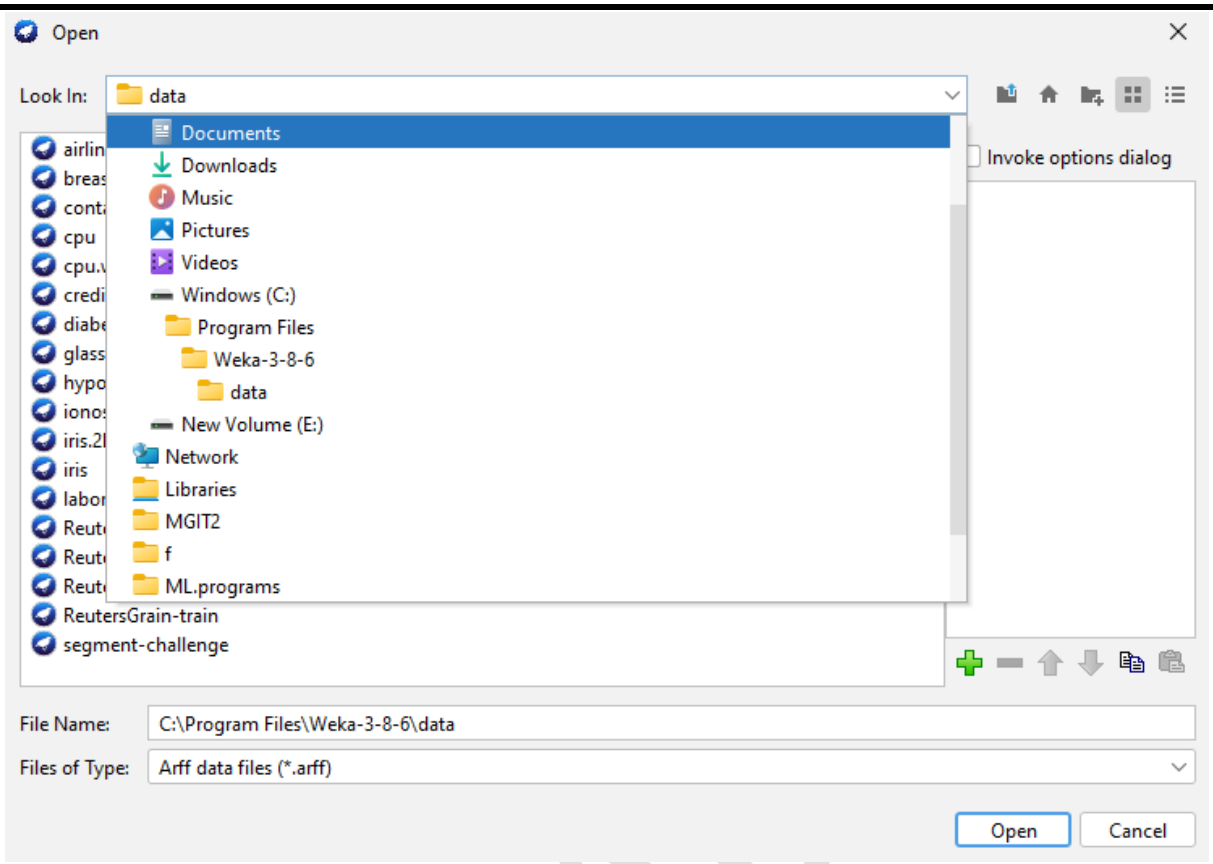
	item_id	item	quantity	price	bought	forenoon	afternoon
0	1	milk	2.0	1.0	672.000000	456.0	567.0
1	2	sugar	1.0	50.5	453.000000	234.0	567.0
2	3	chips	2.5	45.0	456.000000	322.0	567.0
3	4	coffee	2.0	45.0	672.000000	564.0	567.0
4	5	meat	4.0	56.0	786.000000	221.0	567.0
5	6	chocos	3.0	50.5	345.000000	221.0	213.0
6	7	juice	1.0	78.0	765.000000	221.0	344.0
7	8	jam	2.5	65.0	665.000000	221.0	333.0
8	9	bread	3.0	50.5	162.022706	221.0	567.0
9	10	butter	4.0	50.5	162.022706	221.0	322.0

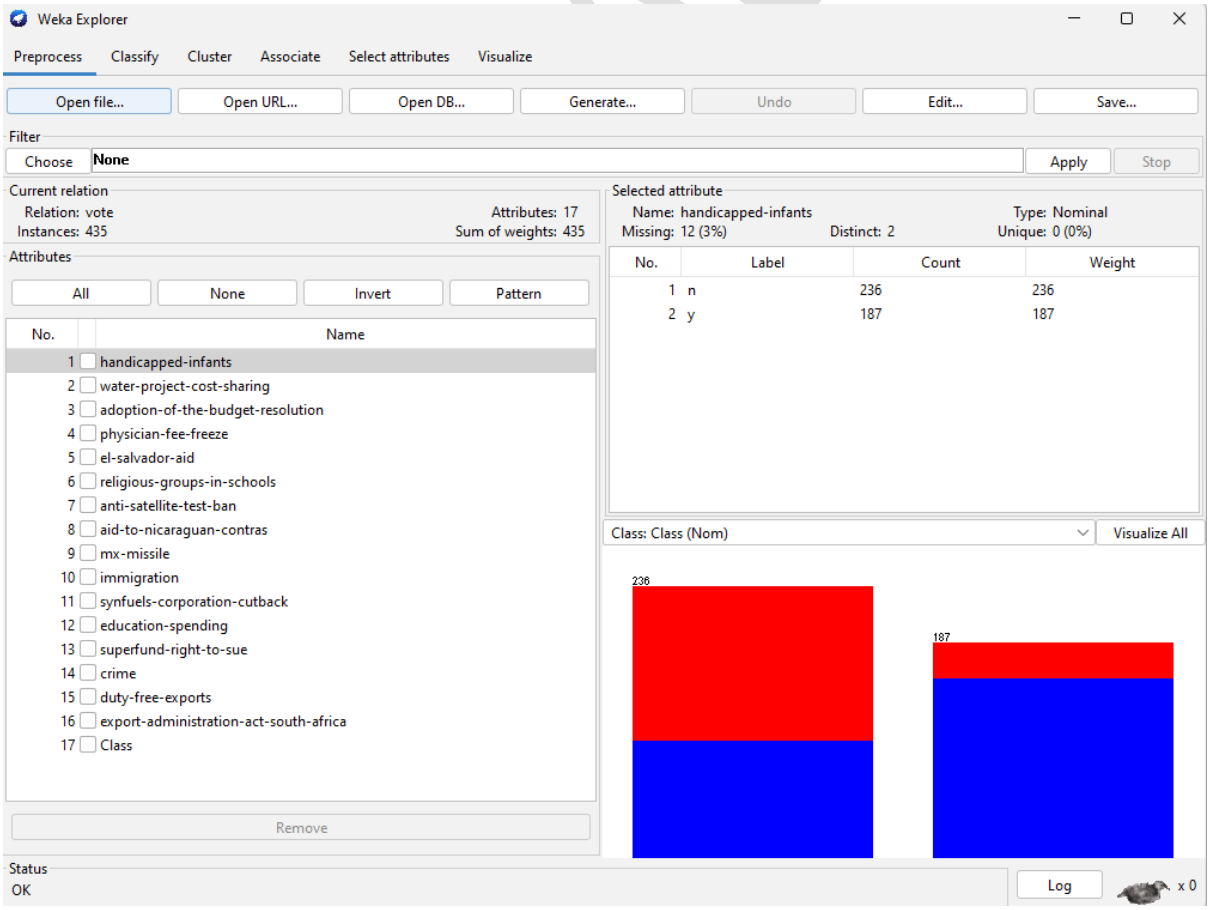
REPLACING MISSING VALUES

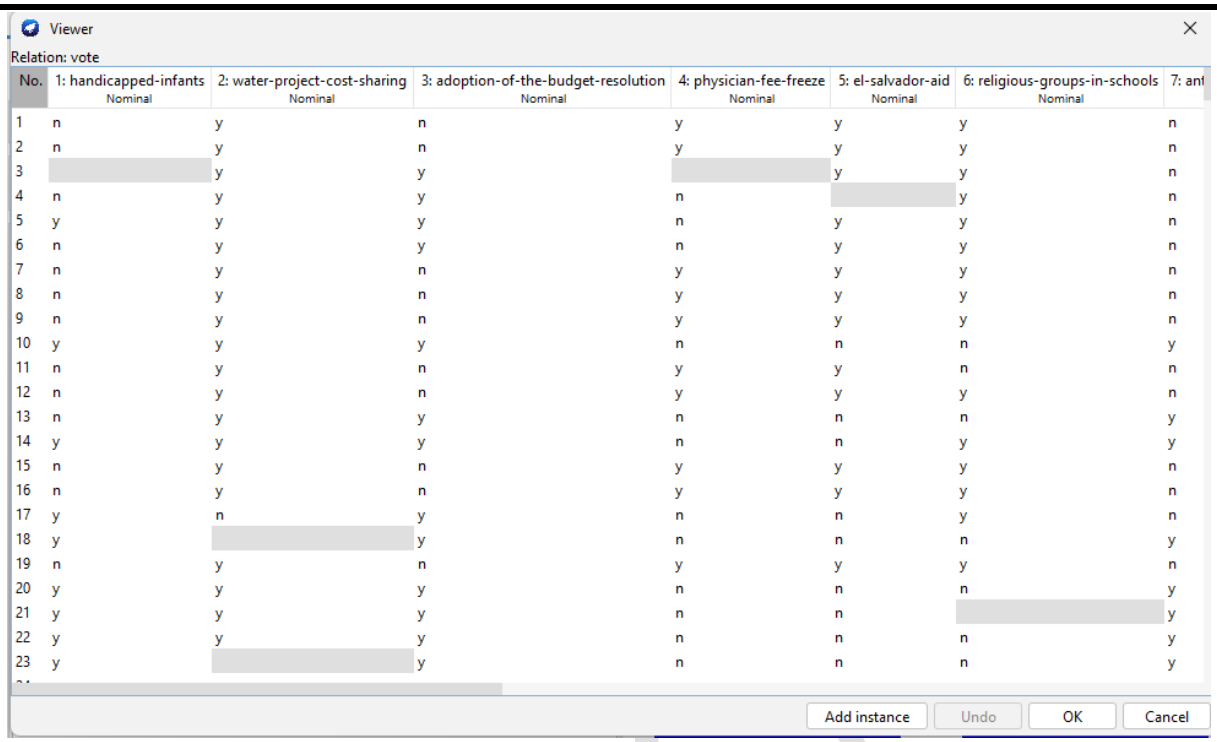
AIM: Replacing missing values using WEKA tool.

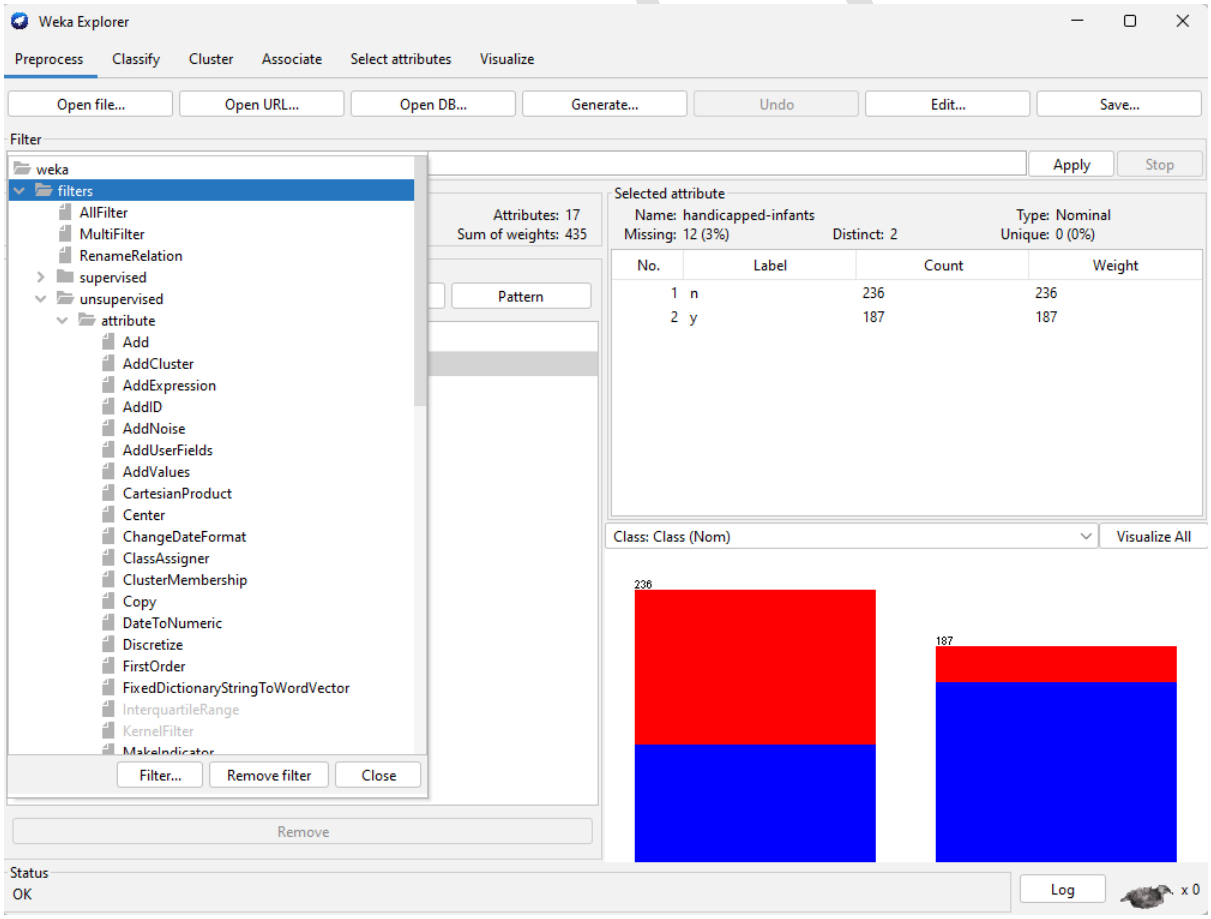
PROGRAM:

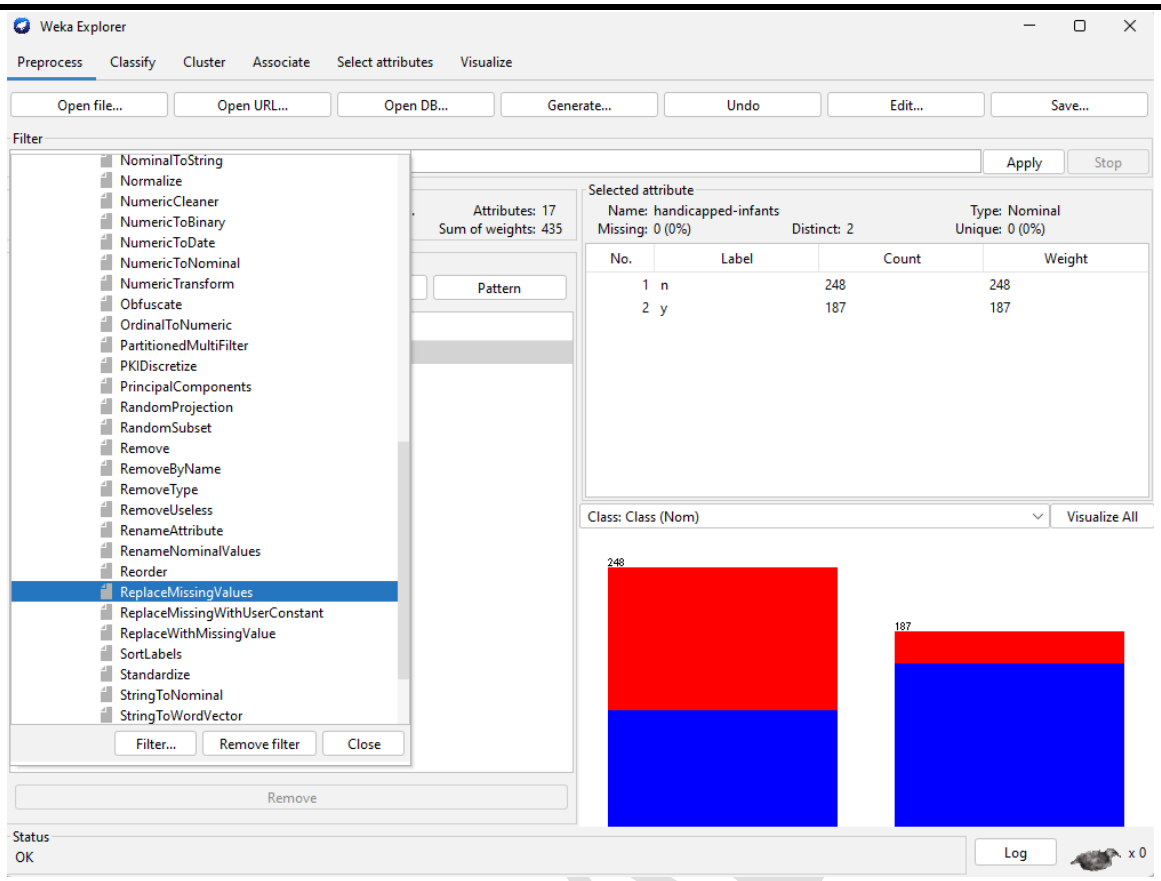


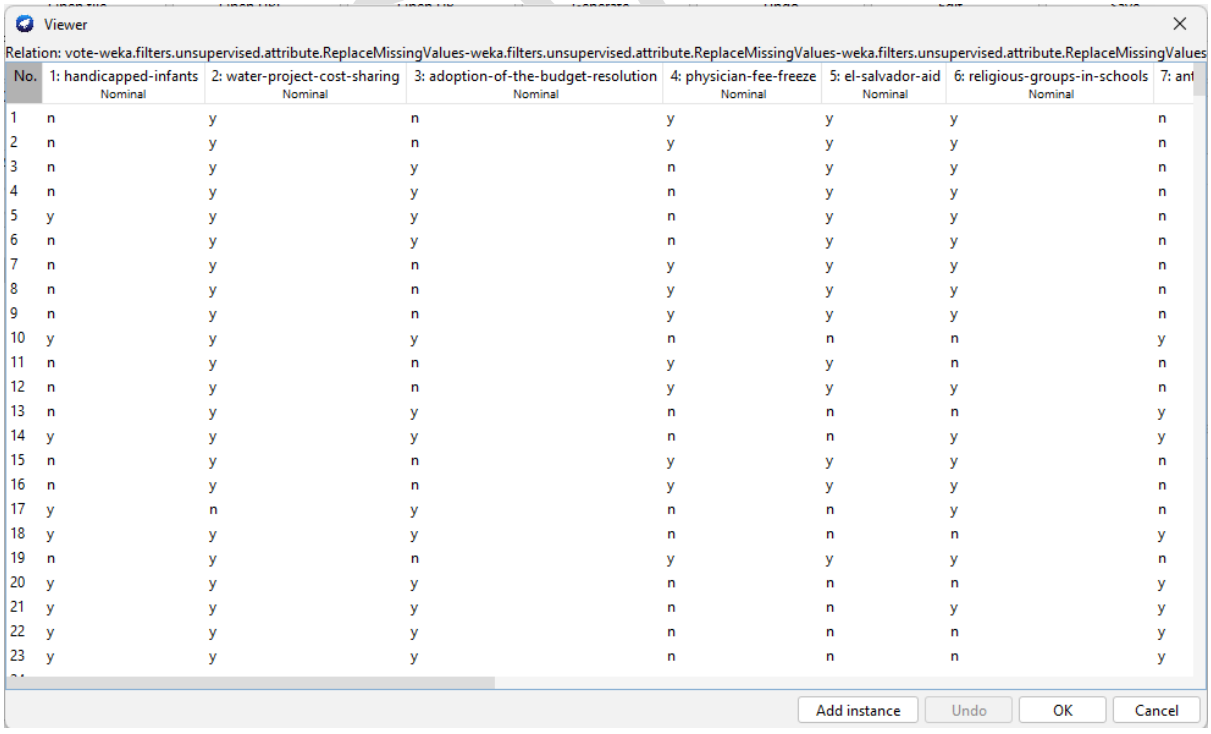

Open
Look In:
data
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Program Files
Weka-3-8-6
data
New Volume (E:)
Network
Libraries
MGIT2
f
ML.programs
Invoke options dialog
ReutersGrain-train
segment-challenge
File Name:
C:\Program Files\Weka-3-8-6\data
Files of Type:
Arff data files (*.arff)
Open
Cancel


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: vote
Instances: 435
Attributes: 17
Sum of weights: 435
Attributes
All
None
Invert
Pattern
No.
Name
1 handicapped-infants
2 water-project-cost-sharing
3 adoption-of-the-budget-resolution
4 physician-fee-freeze
5 el-salvador-aid
6 religious-groups-in-schools
7 anti-satellite-test-ban
8 aid-to-nicaraguan-contras
9 mx-missile
10 immigration
11 synfuels-corporation-cutback
12 education-spending
13 superfund-right-to-sue
14 crime
15 duty-free-exports
16 export-administration-act-south-africa
17 Class
Remove
Selected attribute
Name: handicapped-infants
Missing: 12 (3%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)
No.
Label
Count
Weight
1 n 236 236
2 y 187 187
Class: Class (Nom)
Visualize All
236
187
Status
OK
Log
x 0


Viewer
Relation: vote
No. 1: handicapped-infants Nominal 2: water-project-cost-sharing Nominal 3: adoption-of-the-budget-resolution Nominal 4: physician-fee-freeze Nominal 5: el-salvador-aid Nominal 6: religious-groups-in-schools Nominal 7: ant
1 n y n y y y n
2 n y n y y y n
3 y y y y n
4 n y y n y n
5 y y y n y y n
6 n y y n y y n
7 n y n y y y n
8 n y n y y y n
9 n y n y y y n
10 y y y n n n y
11 n y n y y n n
12 n y n y y y n
13 n y y n n n y
14 y y y n n y y
15 n y n y y y n
16 n y n y y y n
17 y n y n n y n
18 y y n n n y
19 n y n y y y n
20 y y y n n n y
21 y y y n n y
22 y y y n n n y
23 y y n n n y
Add instance
Undo
OK
Cancel


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Apply
Stop
weka
filters
AllFilter
MultiFilter
RenameRelation
supervised
unsupervised
attribute
Add
AddCluster
AddExpression
AddID
AddNoise
AddUserFields
AddValues
CartesianProduct
Center
ChangeDateFormat
ClassAssigner
ClusterMembership
Copy
DateToNumeric
Discretize
FirstOrder
FixedDictionaryStringToWordVector
InterquartileRange
KernelFilter
MakeIndicator
Filter...
Remove filter
Close
Attributes: 17
Sum of weights: 435
Pattern
Remove
Selected attribute
Name: handicapped-infants
Missing: 12 (3%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)
No. Label Count Weight
1 n 236 236
2 y 187 187
Class: Class (Nom)
Visualize All
236
187
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Apply
Stop
NominalToString
Normalize
NumericCleaner
NumericToBinary
NumericToDate
NumericToNominal
NumericTransform
Obfuscate
OrdinalToNumeric
PartitionedMultiFilter
PKIDiscretize
PrincipalComponents
RandomProjection
RandomSubset
Remove
RemoveByName
RemoveType
RemoveUseless
RenameAttribute
RenameNominalValues
Reorder
ReplaceMissingValues
ReplaceMissingWithUserConstant
ReplaceWithMissingValue
SortLabels
Standardize
StringToNominal
StringToWordVector
Filter...
Remove filter
Close
Attributes: 17
Sum of weights: 435
Pattern
Remove
Selected attribute
Name: handicapped-infants
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)
No. Label Count Weight
1 n 248 248
2 y 187 187
Class: Class (Nom)
Visualize All
248
187
Status
OK
Log
x 0


Viewer
Relation: vote-weka.filters.unsupervised.attribute.ReplaceMissingValues-weka.filters.unsupervised.attribute.ReplaceMissingValues-weka.filters.unsupervised.attribute.ReplaceMissingValues
No. 1: handicapped-infants Nominal 2: water-project-cost-sharing Nominal 3: adoption-of-the-budget-resolution Nominal 4: physician-fee-freeze Nominal 5: el-salvador-aid Nominal 6: religious-groups-in-schools Nominal 7: ant
1 n y n y y y n
2 n y n y y y n
3 n y y n y y n
4 n y y n y y n
5 y y y n y y n
6 n y y n y y n
7 n y n y y y n
8 n y n y y y n
9 n y n y y y n
10 y y y n n n y
11 n y n y y n n
12 n y n y y y n
13 n y y n n n y
14 y y y n n y y
15 n y n y y y n
16 n y n y y y n
17 y n y n n y n
18 y y y n n n y
19 n y n y y y n
20 y y y n n n y
21 y y y n n y y
22 y y y n n n y
23 y y y n n n y
Add instance
Undo
OK
Cancel

DATA SMOOTHING

AIM: Write a python program to smooth the given set of data using binning method.

PROGRAM:

[1]:
```
def create_bins(nums):
    bins=[]
    n=len(nums)
    i=0
    while i<n:
        if(n-1)<binsize:
            bins.append(nums[i:])
            i+=n-i
        else:
            bins.append(nums[i:i+binsize])
            i+=binsize
    return bins
```

[2]:
```
def mean_smooth(bins):
    for b in bins:
        avg=0
        n=len(b)
        for num in b:
            avg+=num
        avg=avg/n
        for i in range(n):
            b[i]=avg
    print(bins)
```

[3]:
```
def median_smooth(bins):
    for b in bins:
        n=len(b)
        if n%2==0:
            median=(b[n//2]+b[n//2-1])/2
        else:
            median=b[n//2]
        for i in range(n):
            b[i]=median
    print(bins)
```

[4]:
```
def bound_smooth(bins):
    for b in bins:
        n=len(b)
        l=b[0]
        r=b[n-1]
        for i in range(n):
            if(b[i]-1)<=(r-b[i]):
                b[i]=1
            else:
                b[i]=r
    print(bins)
```

In [12]:
```
nums=[int(x) for x in input("Enter numbers: ").split()]
nums.sort()
binsize=int(input("Enter bin size: "))
n=len(nums)
bins=create_bins(nums)
print(bins)
```

```
Enter numbers: 22 33 21 42 12 35 43 58 56 45 65 32
Enter bin size: 4
[[12, 21, 22, 32], [33, 35, 42, 43], [45, 56, 58, 65]]
```

In [14]:
```
mean_smooth(bins)
bins=create_bins(nums)
print(bins)
median_smooth(bins)
bins=create_bins(nums)
print(bins)
boun_smooth(bins)
bins=create_bins(nums)
print(bins)
```

```
[[21.75, 21.75, 21.75, 21.75], [38.25, 38.25, 38.25, 38.25], [56.0, 56.0, 56.0, 56.0]]
[[12, 21, 22, 32], [33, 35, 42, 43], [45, 56, 58, 65]]
[[21.5, 21.5, 21.5, 21.5], [38.5, 38.5, 38.5, 38.5], [57.0, 57.0, 57.0, 57.0]]
[[12, 21, 22, 32], [33, 35, 42, 43], [45, 56, 58, 65]]
[[12, 12, 12, 32], [33, 33, 43, 43], [45, 65, 65, 65]]
[[12, 21, 22, 32], [33, 35, 42, 43], [45, 56, 58, 65]]
```

COVARIANCE TEST

AIM: To find the type of relationship between two random variables by performing co-variance test on the dataset loaded using python.

PROGRAM:

In [1]:
```
import pandas as pd
df=pd.DataFrame({"X":[1.8,1.5,2.1,2.4,0.2],"Y":[2.5,4.3,4.5,4.1,2.2]})
print(df)
```

```
     X    Y
0  1.8  2.5
1  1.5  4.3
2  2.1  4.5
3  2.4  4.1
4  0.2  2.2
```

In [2]:
```
df.cov()
df["X"].cov(df["Y"])
```

Out[2]: 0.6299999999999999

CORELATION TEST

AIM: To write a python program to find correlation for given dataset**.**

PROGRAM:

In [1]:
```
import pandas as pd
df=pd.DataFrame({"X":[40,21,25,31,38,47],"Y":[78,70,60,55,80,66]})
print(df)
```

```
    X   Y
0  40  78
1  21  70
2  25  60
3  31  55
4  38  80
5  47  66
```

In [2]:
```
df.corr()
df["X"].corr(df["Y"])
```

Out[2]: 0.3471102283886791

CHI-SQUARE TEST

AIM: To write a python program to implement Chi^2 test.

PROGRAM:

In [4]:
```
import pandas as pd
from scipy.stats import chi2_contingency
from scipy.stats import chi2
```

In [5]:
```
df=pd.read_csv("tips.csv")
```

In [8]:
```
df.head(10)
```

Out[8]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4
6	8.77	2.00	Male	No	Sun	Dinner	2
7	26.88	3.12	Male	No	Sun	Dinner	4
8	15.04	1.96	Male	No	Sun	Dinner	2
9	14.78	3.23	Male	No	Sun	Dinner	2

In [14]:
```
freq_table=pd.crosstab(df['sex'],df['smoker'])
print(freq_table)
```

```
smoker  No  Yes
sex
Female  54   33
Male    97   60
```

In [15]:
```
observed_value=freq_table.values
print(observed_value)
```

```
[[54 33]
 [97 60]]
```

In [21]:
```
res=chi2_contingency(observed_value,correction=False)
print(res)
```

```
Chi2ContingencyResult(statistic=0.001934818536627623, pvalue=0.964915107315
732, dof=1, expected_freq=array([[53.84016393, 33.15983607],
       [97.15983607, 59.84016393]]))
```

In [17]:
```
p=0.05
doff=res.dof
print(doff)
```

```
1
```

In [25]:
```
critical_value=chi2.ppf(1-p,doff)
print(critical_value)
```

```
3.841458820694124
```

In [23]:
```
if res.statistic > critical_value :
    print("Reject Null Hypothesis")
else :
    print("Accept Null Hypothesis")
```

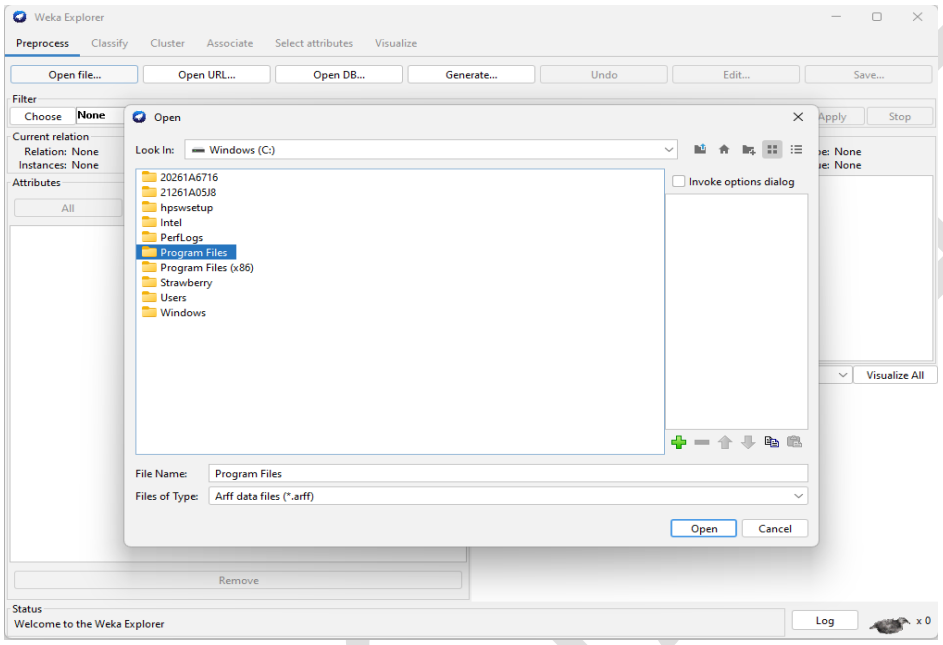
```
Accept Null Hypothesis
```

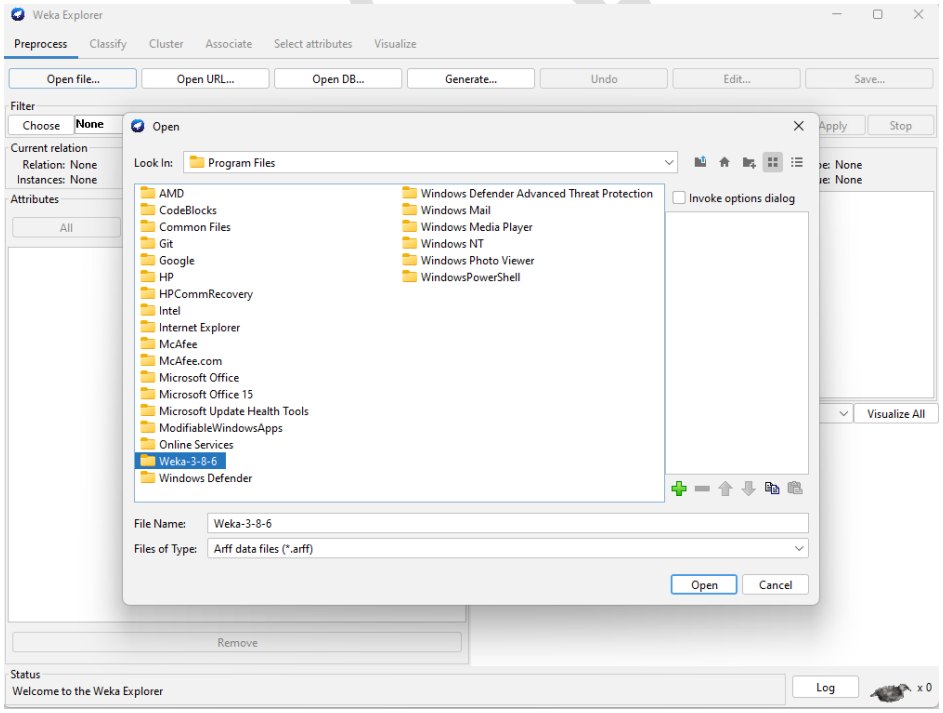
APRIORI ALGORITHM

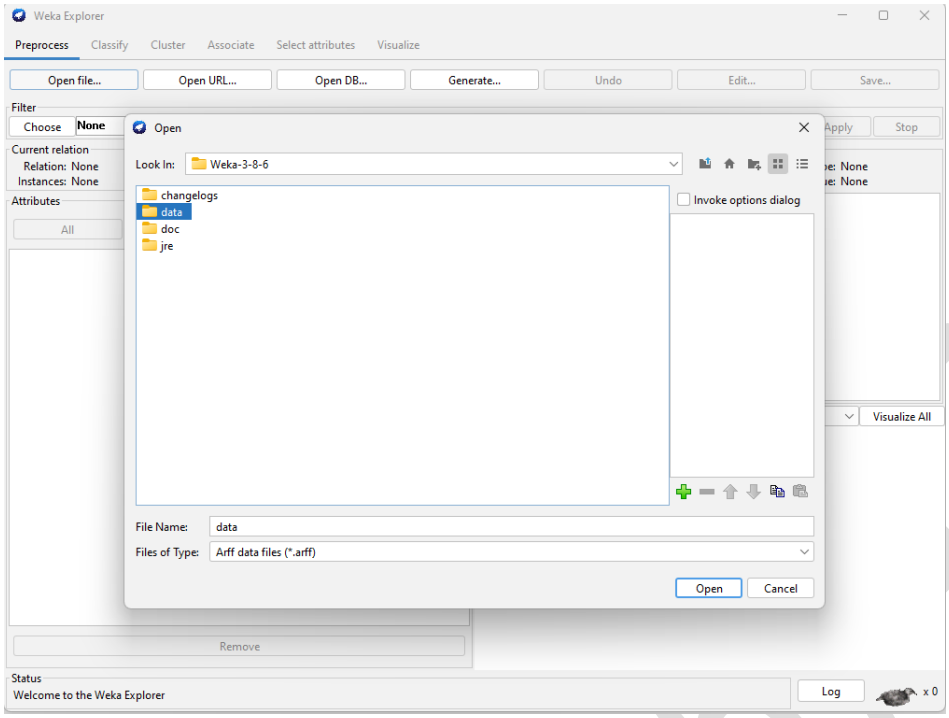
AIM: WEKA tool to implement Apriori algorithm for association rule mining.

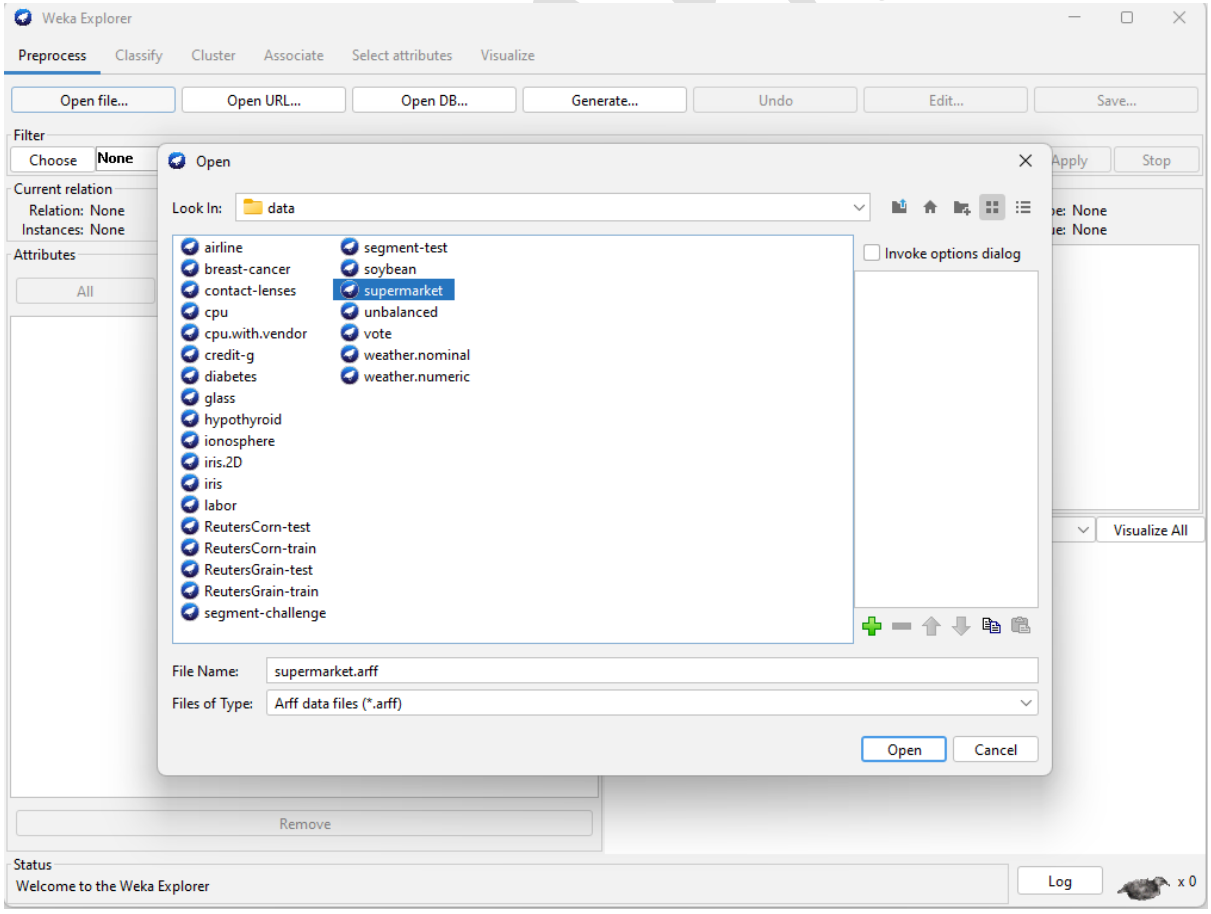
PROGRAM:

Step 1: Loading data



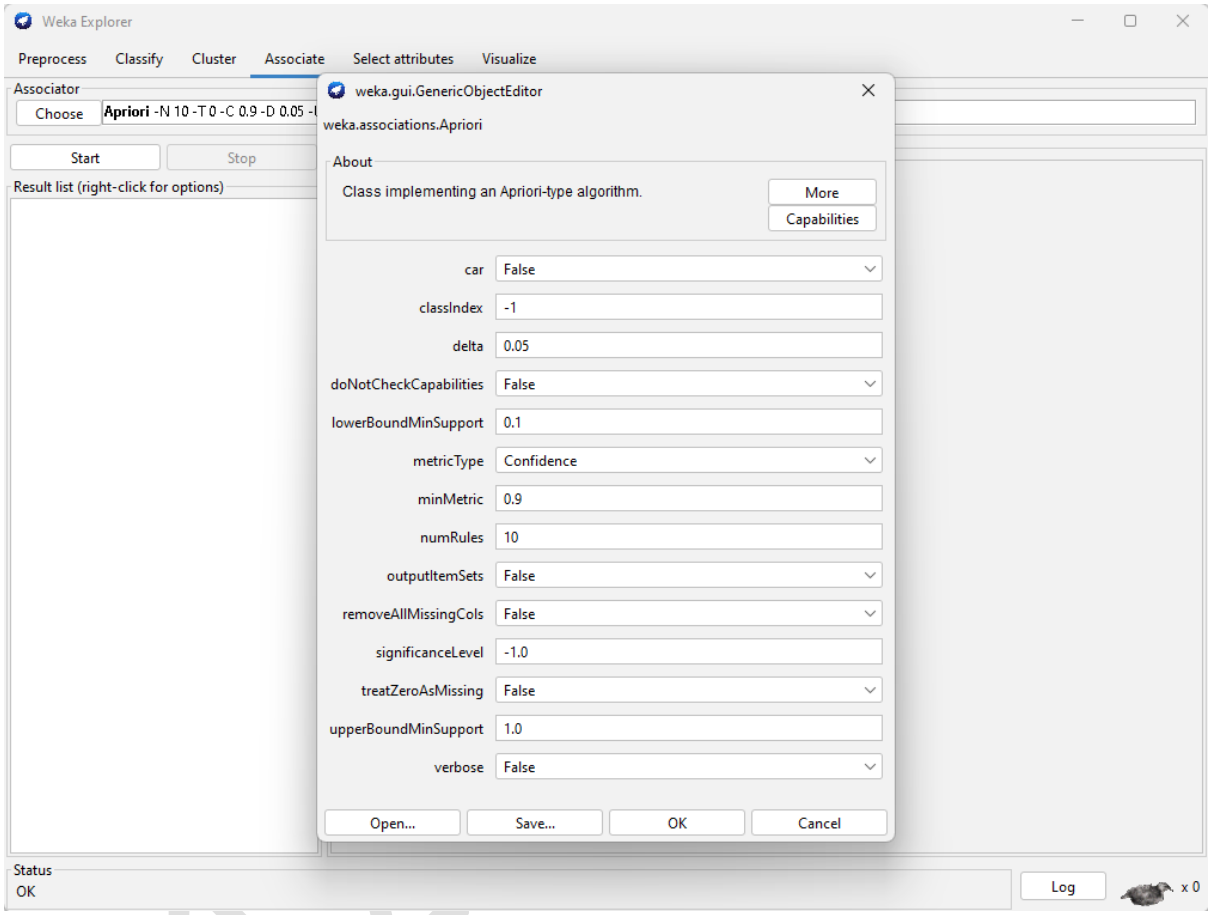



Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: None
Instances: None
Attributes
All
Open
Look In:
Weka-3-8-6
changelogs
data
doc
jre
Invoke options dialog
File Name:
data
Files of Type:
Arff data files (*.arff)
Open
Cancel
Visualize All
Remove
Status
Welcome to the Weka Explorer
Log
x 0


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: None
Instances: None
Attributes
All
Open
Look In:
data
airline
breast-cancer
contact-lenses
cpu
cpu.with.vendor
credit-g
diabetes
glass
hypothyroid
ionosphere
iris.2D
iris
labor
ReutersCorn-test
ReutersCorn-train
ReutersGrain-test
ReutersGrain-train
segment-challenge
segment-test
soybean
supermarket
unbalanced
vote
weather.nominal
weather.numeric
Invoke options dialog
File Name:
supermarket.arff
Files of Type:
Arff data files (*.arff)
Open
Cancel
Visualize All
Remove
Status
Welcome to the Weka Explorer
Log
x 0

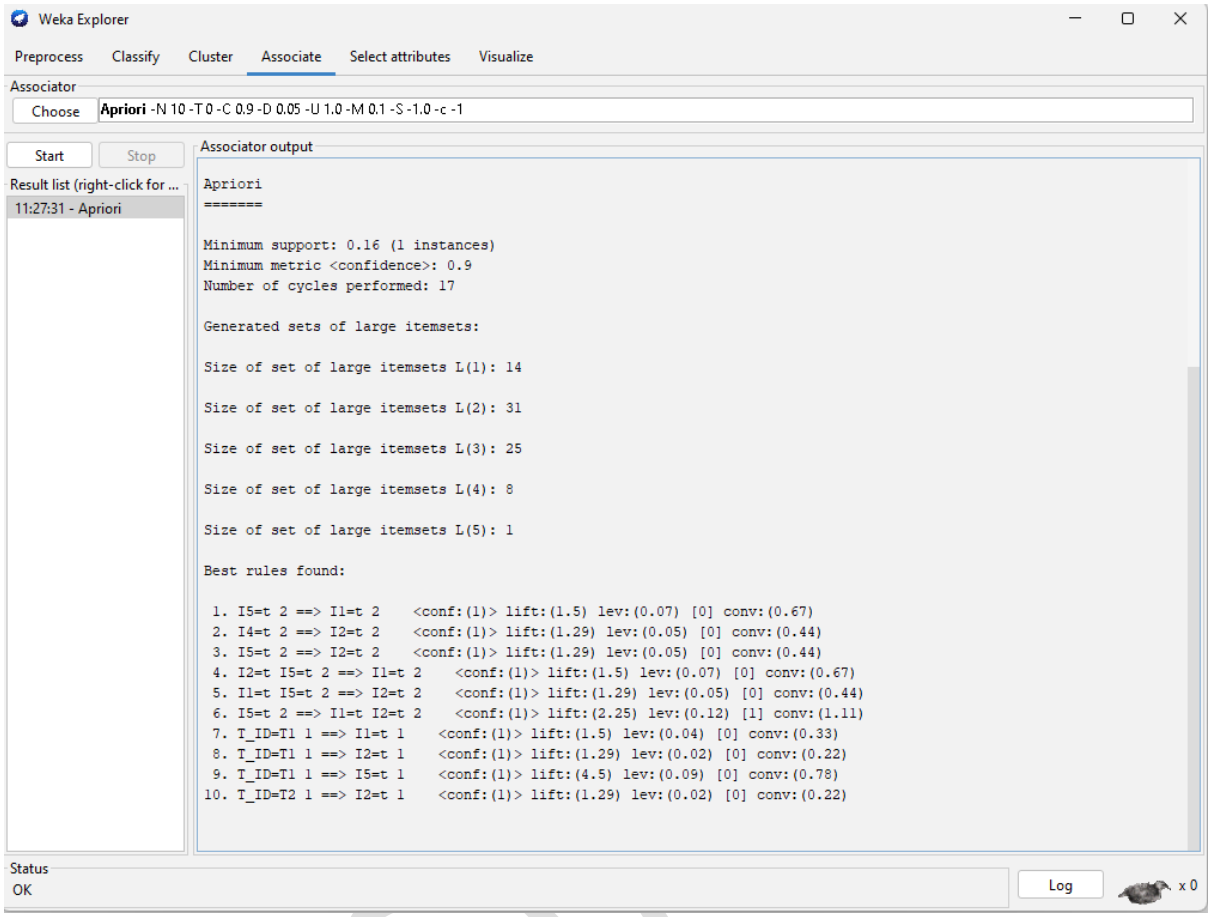
Step 2: Select Apriori algorithm from Associate tab.

Step 3: Setting test options.



Step 4: Apply the algorithm.

Step 5: Analysing the results.



APRIORI ALGORITHM

AIM: Python program to implement Apriori algorithm to extract frequent item sets for association rule mining.

PROGRAM:

In [2]:
```
import pandas as pd
import numpy as np
from mlxtend.frequent_patterns import apriori
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import association_rules
```

In [3]:
```
dataset=[['Milk','Onion','Nutmeg','Kidney Beans','Eggs','Yogurt'],
         ['Dill','Onion','Nutmeg','Kidney Beans','Eggs','Yogurt'],
         ['Milk','Apple','Kidney Beans','Eggs'],
         ['Milk','Unicorn','Corn','Kidney Beans','Yogurt'],
         ['Corn','Onion','Onion','Kidney Beans','Ice cream','Eggs']]
```

In [7]:
```
te=TransactionEncoder()
te_ary=te.fit(dataset).transform(dataset)
df=pd.DataFrame(te_ary,columns=te.columns_)
df
```

Out[7]:

	Apple	Corn	Dill	Eggs	Ice cream	Kidney Beans	Milk	Nutmeg	Onion	Unicorn	Yogurt
0	False	False	False	True	False	True	True	True	True	False	True
1	False	False	True	True	False	True	False	True	True	False	True
2	True	False	False	True	False	True	True	False	False	False	False
3	False	True	False	False	False	True	True	False	False	True	True
4	False	True	False	True	True	True	False	False	True	False	False

```
In [8]: apriori(df,min_support=0.6,use_colnames=True)
```

Out[8]:

	support	itemsets
0	0.8	(Eggs)
1	1.0	(Kidney Beans)
2	0.6	(Milk)
3	0.6	(Onion)
4	0.6	(Yogurt)
5	0.8	(Kidney Beans, Eggs)
6	0.6	(Onion, Eggs)
7	0.6	(Kidney Beans, Milk)
8	0.6	(Kidney Beans, Onion)
9	0.6	(Yogurt, Kidney Beans)
10	0.6	(Kidney Beans, Onion, Eggs)

```
In [9]: frequent_itemsets=apriori(df,min_support=0.6,use_colnames=True)
```

```
In [10]: association_rules(frequent_itemsets,metric="confidence",min_threshold=0.7)
```

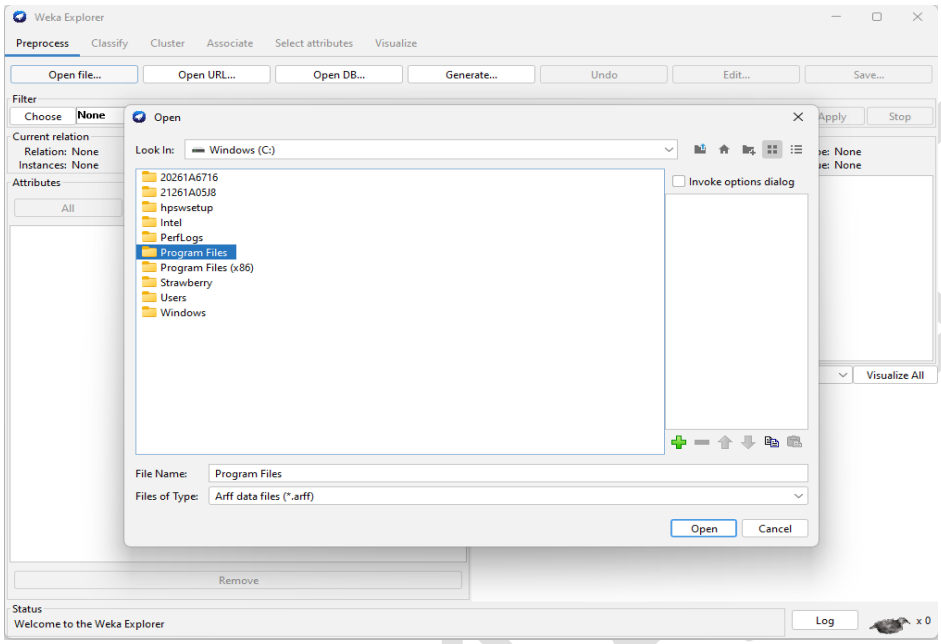
Out[10]:

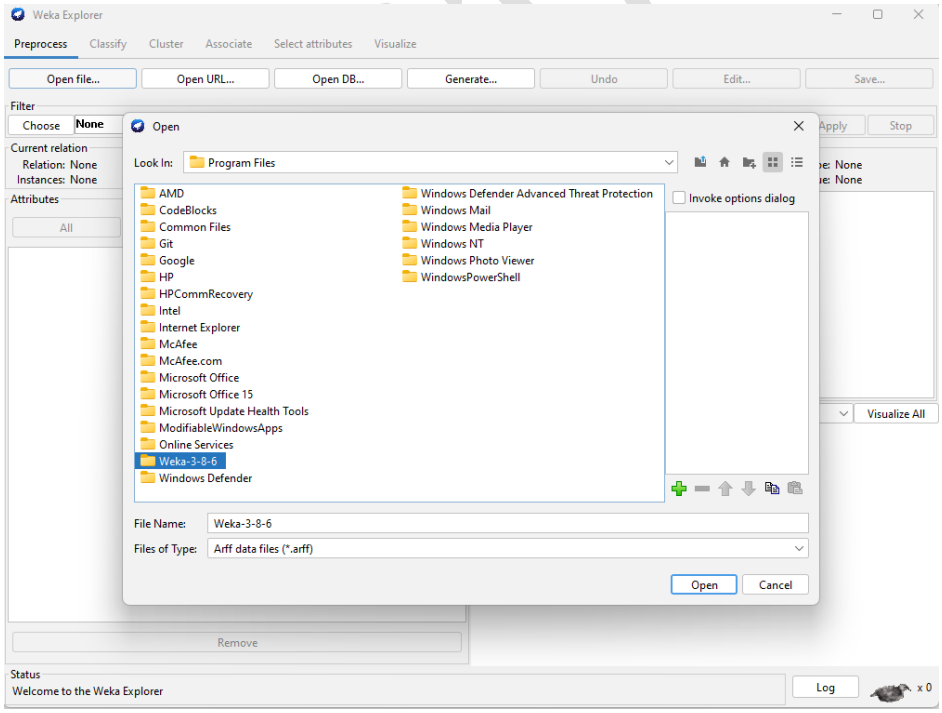
	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(Kidney Beans)	(Eggs)	1.0	0.8	0.8	0.80	1.00	0.00
1	(Eggs)	(Kidney Beans)	0.8	1.0	0.8	1.00	1.00	0.00
2	(Onion)	(Eggs)	0.6	0.8	0.6	1.00	1.25	0.12
3	(Eggs)	(Onion)	0.8	0.6	0.6	0.75	1.25	0.12
4	(Milk)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00
5	(Onion)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00
6	(Yogurt)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00
7	(Kidney Beans, Onion)	(Eggs)	0.6	0.8	0.6	1.00	1.25	0.12
8	(Kidney Beans, Eggs)	(Onion)	0.8	0.6	0.6	0.75	1.25	0.12
9	(Onion, Eggs)	(Kidney Beans)	0.6	1.0	0.6	1.00	1.00	0.00
10	(Onion)	(Kidney Beans, Eggs)	0.6	0.8	0.6	1.00	1.25	0.12
11	(Eggs)	(Kidney Beans, Onion)	0.8	0.6	0.6	0.75	1.25	0.12

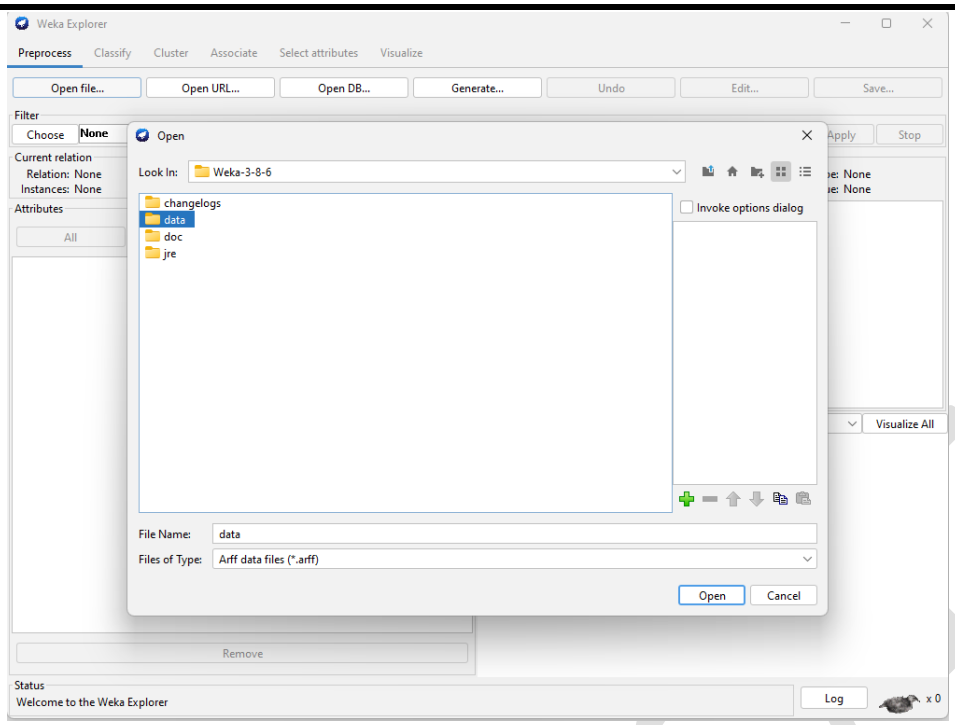
FP GROWTH ALGORITHM

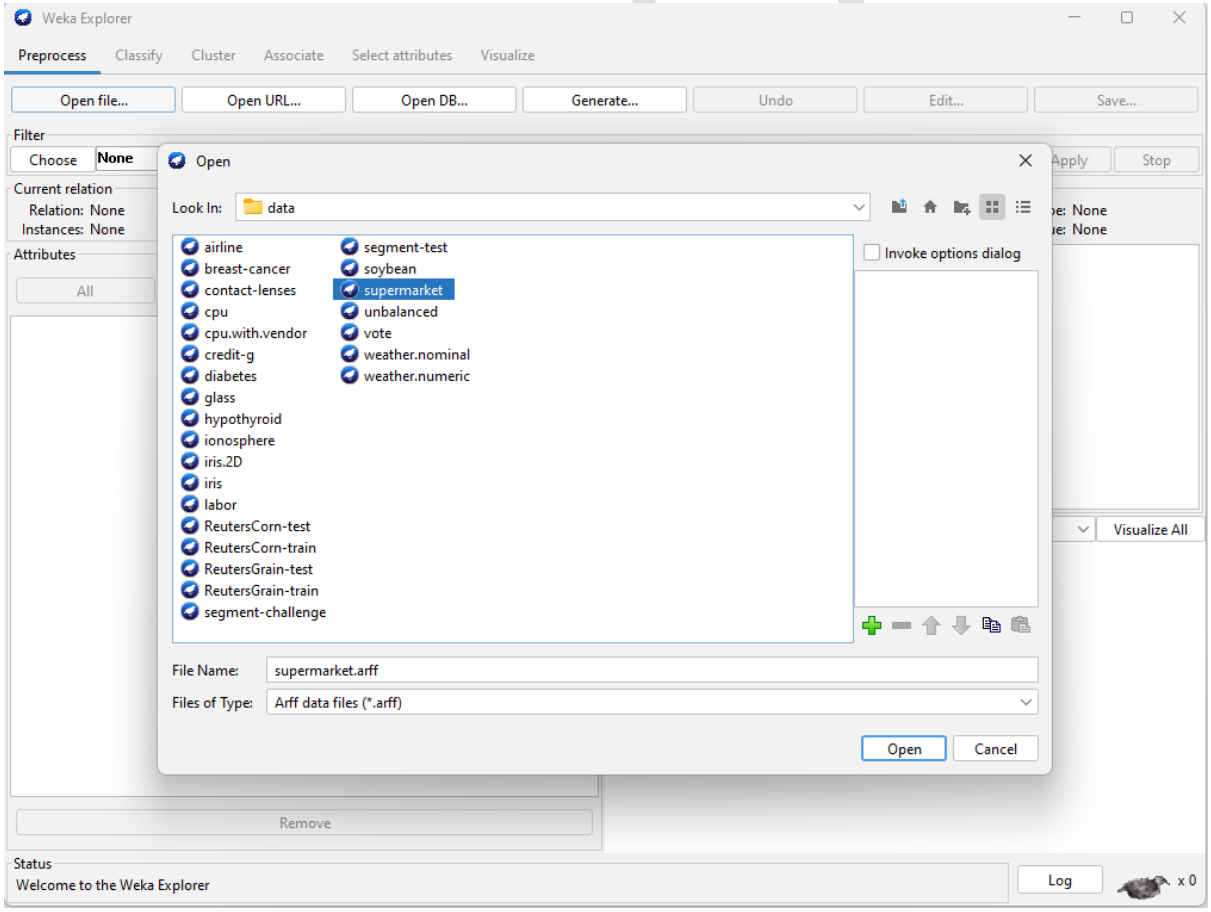
AIM: WEKA tool to implement FP Growth algorithm for association rule mining.

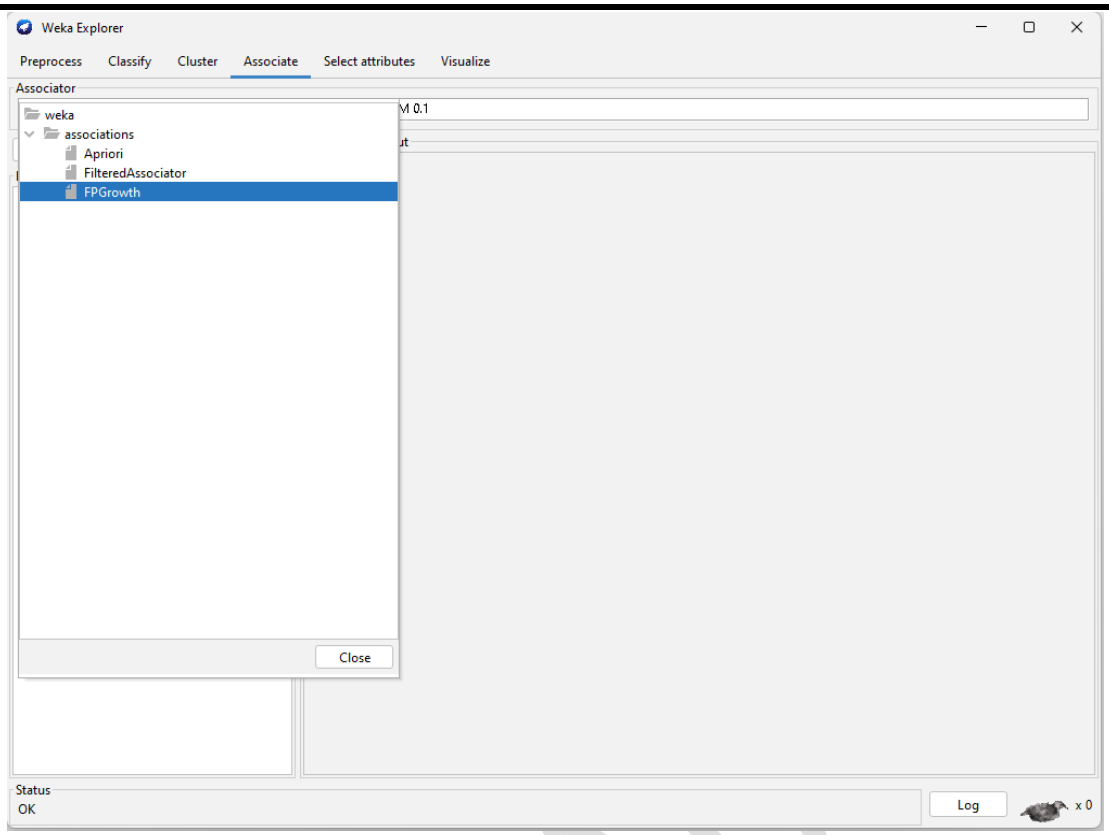
PROGRAM:

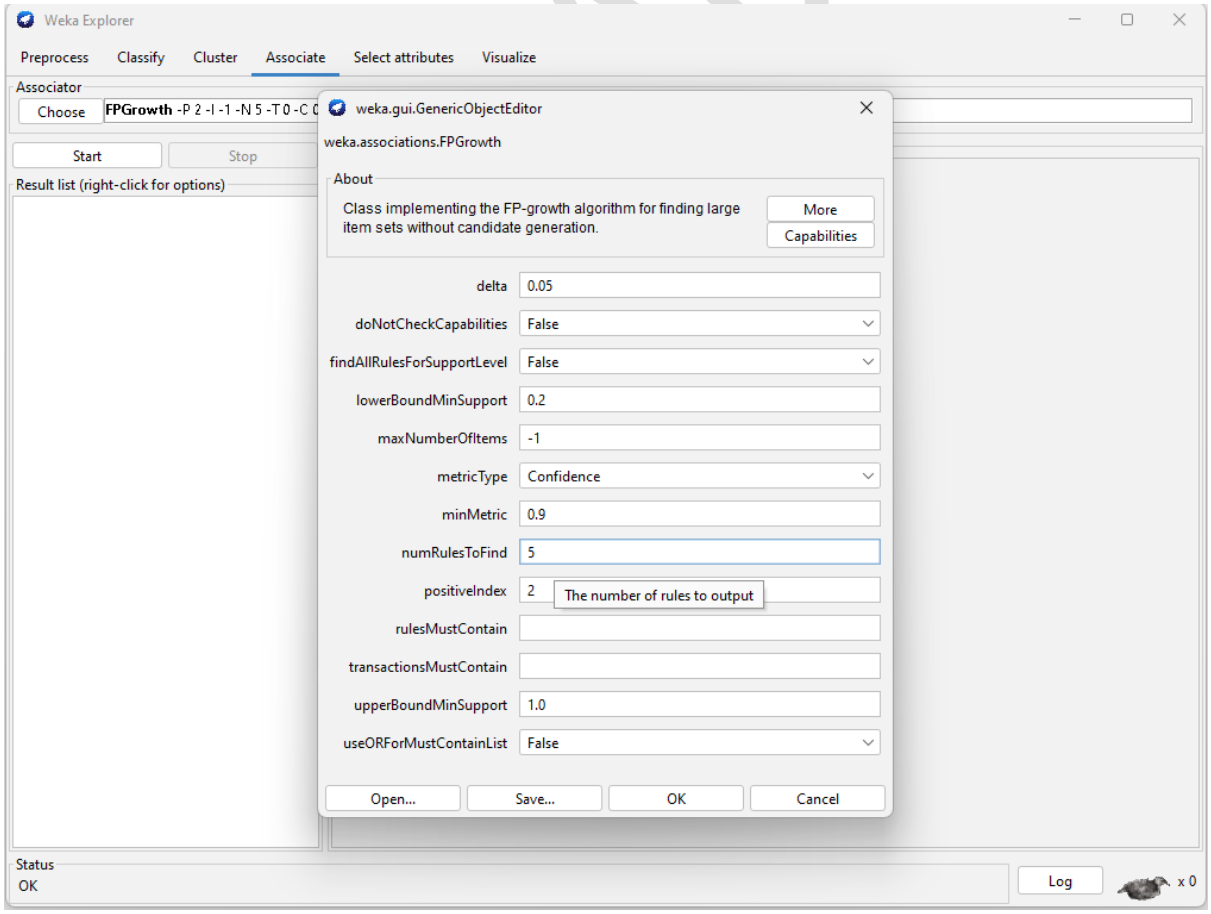


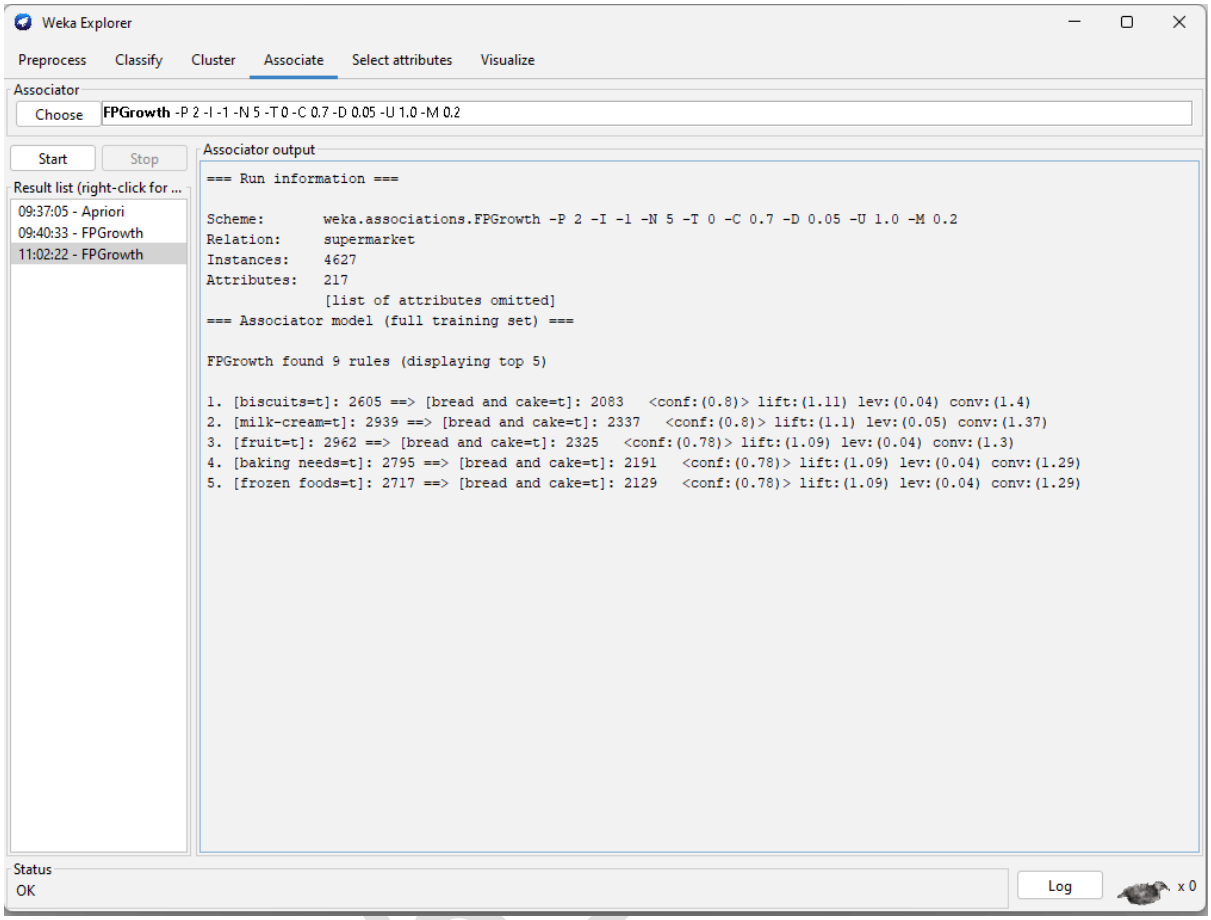



Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: None
Instances: None
Attributes
All
Open
Look In:
Weka-3-8-6
changelogs
data
doc
jre
Invoke options dialog
File Name:
data
Files of Type:
Arff data files (*.arff)
Open
Cancel
Visualize All
Remove
Status
Welcome to the Weka Explorer
Log
x 0


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: None
Instances: None
Attributes
All
Open
Look In:
data
airline
breast-cancer
contact-lenses
cpu
cpu.with.vendor
credit-g
diabetes
glass
hypothyroid
ionosphere
iris.2D
iris
labor
ReutersCorn-test
ReutersCorn-train
ReutersGrain-test
ReutersGrain-train
segment-challenge
segment-test
soybean
supermarket
unbalanced
vote
weather.nominal
weather.numeric
Invoke options dialog
File Name:
supermarket.arff
Files of Type:
Arff data files (*.arff)
Open
Cancel
Visualize All
Remove
Status
Welcome to the Weka Explorer
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
weka
associations
Apriori
FilteredAssociator
FPGrowth
Close
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
FPGrowth -P 2 -I -1 -N 5 -T 0 -C
Start
Stop
Result list (right-click for options)
weka.gui.GenericObjectEditor
weka.associations.FPGrowth
About
Class implementing the FP-growth algorithm for finding large item sets without candidate generation.
More
Capabilities
delta 0.05
doNotCheckCapabilities False
findAllRulesForSupportLevel False
lowerBoundMinSupport 0.2
maxNumberOfItems -1
metricType Confidence
minMetric 0.9
numRulesToFind 5
positiveIndex 2
The number of rules to output
rulesMustContain
transactionsMustContain
upperBoundMinSupport 1.0
useORForMustContainList False
Open...
Save...
OK
Cancel
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
FPGrowth -P 2 -I -1 -N 5 -T 0 -C 0.7 -D 0.05 -U 1.0 -M 0.2
Start
Stop
Result list (right-click for ...
09:37:05 - Apriori
09:40:33 - FPGrowth
11:02:22 - FPGrowth
Associator output
=== Run information ===
Scheme: weka.associations.FPGrowth -P 2 -I -1 -N 5 -T 0 -C 0.7 -D 0.05 -U 1.0 -M 0.2
Relation: supermarket
Instances: 4627
Attributes: 217
[list of attributes omitted]
=== Associator model (full training set) ===
FPGrowth found 9 rules (displaying top 5)
1. [biscuits=t]: 2605 ==> [bread and cake=t]: 2083 <conf:(0.8)> lift:(1.11) lev:(0.04) conv:(1.4)
2. [milk-cream=t]: 2939 ==> [bread and cake=t]: 2337 <conf:(0.8)> lift:(1.1) lev:(0.05) conv:(1.37)
3. [fruit=t]: 2962 ==> [bread and cake=t]: 2325 <conf:(0.78)> lift:(1.09) lev:(0.04) conv:(1.3)
4. [baking needs=t]: 2795 ==> [bread and cake=t]: 2191 <conf:(0.78)> lift:(1.09) lev:(0.04) conv:(1.29)
5. [frozen foods=t]: 2717 ==> [bread and cake=t]: 2129 <conf:(0.78)> lift:(1.09) lev:(0.04) conv:(1.29)
Status
OK
Log
x 0

FP GROWTH ALGORITHM

AIM: Python program to implement FP Growth algorithm to extract frequent item sets for association rule mining.

PROGRAM:

In [1]:
```
import pandas as pd
import numpy as np
from mlxtend.frequent_patterns import fpgrowth
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import association_rules
```

In [2]:
```
dataset=[['I1','I2','I5'],
         ['I2','I4'],
         ['I2','I3'],
         ['I1','I2','I4'],
         ['I1','I3'],
         ['I2','I3'],
         ['I1','I3'],
         ['I1','I2','I3','I5'],
         ['I1','I2','I3']]
```

In [3]:
```
te=TransactionEncoder()
te_ary=te.fit(dataset).transform(dataset)
df=pd.DataFrame(te_ary,columns=te.columns_)
df
```

Out[3]:

	I1	I2	I3	I4	I5
0	True	True	False	False	True
1	False	True	False	True	False
2	False	True	True	False	False
3	True	True	False	True	False
4	True	False	True	False	False
5	False	True	True	False	False
6	True	False	True	False	False
7	True	True	True	False	True
8	True	True	True	False	False

In [4]:
```
fpgrowth(df,min_support=0.2,use_colnames=True)
```

Out[4]:

	support	itemsets
0	0.777778	(I2)
1	0.666667	(I1)
2	0.222222	(I5)
3	0.222222	(I4)
4	0.666667	(I3)
5	0.444444	(I1, I2)
6	0.444444	(I3, I1)
7	0.222222	(I3, I1, I2)
8	0.222222	(I5, I1)
9	0.222222	(I5, I2)
10	0.222222	(I5, I1, I2)
11	0.222222	(I4, I2)
12	0.444444	(I3, I2)

In [5]:
```
from mlxtend.frequent_patterns import apriori
%timeit -n 100 -r 10 apriori(df,min_support=0.2, use_colnames=True)
```

```
1.69 ms ± 57.9 µs per loop (mean ± std. dev. of 10 runs, 100 loops each)
```

In [6]:
```
%timeit -n 100 -r 10 fpgrowth(df,min_support=0.2, use_colnames=True)
```

```
652 µs ± 39.9 µs per loop (mean ± std. dev. of 10 runs, 100 loops each)
```

CATEGORICAL TO NUMERIC DATA CONVERSION

AIM: To convert categorical variables in a dataset in a csv file into numerical data.

PROCEDURE:

- Open the jupyter notebook and load file.
- Check for categorical values in given data set.
- Find unique values of the categorical attributes.
- Count each value present in each categorical attribute.
- Replace the categorical values with numeric values.
- Store in another data set.

PROGRAM:

```
In [1]: import pandas as pd
        import numpy as np

In [31]: df=pd.read_csv("C:\\Users\\MGIT\\Data mining\\tips.csv")

In [32]: print(df)
```

```
     total_bill   tip     sex smoker   day    time  size
0         16.99  1.01  Female     No   Sun  Dinner     2
1         10.34  1.66    Male     No   Sun  Dinner     3
2         21.01  3.50    Male     No   Sun  Dinner     3
3         23.68  3.31    Male     No   Sun  Dinner     2
4         24.59  3.61  Female     No   Sun  Dinner     4
..          ...   ...     ...    ...   ...     ...   ...
239       29.03  5.92    Male     No   Sat  Dinner     3
240       27.18  2.00  Female    Yes   Sat  Dinner     2
241       22.67  2.00    Male    Yes   Sat  Dinner     2
242       17.82  1.75    Male     No   Sat  Dinner     2
243       18.78  3.00  Female     No  Thur  Dinner     2

[244 rows x 7 columns]
```

```
In [14]: df_categorical=df.select_dtypes(exclude=[np.number])

In [15]: df_categorical['sex'].unique()

Out[15]: array(['Female', 'Male'], dtype=object)

In [16]: df_categorical.sex.value_counts()

Out[16]: sex
         Male      157
         Female     87
         Name: count, dtype: int64

In [17]: df_categorical.sex.replace({"Male":0,"Female":1},inplace=True)
```

```
In [18]: df_categorical
```

Out[18]:

	sex	smoker	day	time
0	1	No	Sun	Dinner
1	0	No	Sun	Dinner
2	0	No	Sun	Dinner
3	0	No	Sun	Dinner
4	1	No	Sun	Dinner
...	...	...	...	...
239	0	No	Sat	Dinner
240	1	Yes	Sat	Dinner
241	0	Yes	Sat	Dinner
242	0	No	Sat	Dinner
243	1	No	Thur	Dinner

244 rows × 4 columns

```
In [21]: df_categorical['smoker'].unique()
Out[21]: array(['No', 'Yes'], dtype=object)

In [22]: df_categorical.smoker.replace({"No":0,"Yes":1},inplace=True)

In [23]: df_categorical
```

Out[23]:

	sex	smoker	day	time
0	1	0	Sun	Dinner
1	0	0	Sun	Dinner
2	0	0	Sun	Dinner
3	0	0	Sun	Dinner
4	1	0	Sun	Dinner
...	...	...	...	...
239	0	0	Sat	Dinner
240	1	1	Sat	Dinner
241	0	1	Sat	Dinner
242	0	0	Sat	Dinner
243	1	0	Thur	Dinner

244 rows × 4 columns

```
In [24]: df_categorical['day'].unique()
Out[24]: array(['Sun', 'Sat', 'Thur', 'Fri'], dtype=object)
```



```
In [25]: df_categorical.day.replace({"Sun":7,"Sat":6,"Fri":5,"Thur":4},inplace=True)
```

In [26]:
```
df_categorical
```

Out[26]:

	sex	smoker	day	time
0	1	0	7	Dinner
1	0	0	7	Dinner
2	0	0	7	Dinner
3	0	0	7	Dinner
4	1	0	7	Dinner
...	...	...	...	...
239	0	0	6	Dinner
240	1	1	6	Dinner
241	0	1	6	Dinner
242	0	0	6	Dinner
243	1	0	4	Dinner

244 rows × 4 columns

In [27]:
```
df_categorical['time'].unique()
```

Out[27]:
```
array(['Dinner', 'Lunch'], dtype=object)
```

In [29]:
```
df_categorical.time.replace({"Lunch":0,"Dinner":1},inplace=True)
```

In [34]:
```
df_categorical
```

Out[34]:

	sex	smoker	day	time
0	1	0	7	1
1	0	0	7	1
2	0	0	7	1
3	0	0	7	1
4	1	0	7	1
...	...	...	...	...
239	0	0	6	1
240	1	1	6	1
241	0	1	6	1
242	0	0	6	1
243	1	0	4	1

244 rows × 4 columns

VISUALIZATION OF DECISION TREE (PYTHON)

AIM: To visualize the decision tree using python.

PROCEDURE:

1. Selecting the dataset

In [3]:
```
import pandas as pd
from sklearn import tree
from matplotlib import pyplot as plt
dataset=pd.read_csv("C://Users//MGIT2//Documents//weather.nominal.csv")
dataset.head()
```

Out[3]:

	outlook	temperature	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes

2. Separating independent variable and target variable

In [4]:
```
attribute=dataset.drop(['play'],axis='columns')
target=dataset['play']
```

In [5]:
```
attribute
```

Out[5]:

	outlook	temperature	humidity	windy
0	sunny	hot	high	False
1	sunny	hot	high	True
2	overcast	hot	high	False
3	rainy	mild	high	False
4	rainy	cool	normal	False
5	rainy	cool	normal	True
6	overcast	cool	normal	True
7	sunny	mild	high	False
8	sunny	cool	normal	False
9	rainy	mild	normal	False
10	sunny	mild	normal	True
11	overcast	mild	high	True
12	overcast	hot	normal	False
13	rainy	mild	high	True

In [6]:
```
target
```

Out[6]:
```
0      no
1      no
2     yes
3     yes
4     yes
5      no
6     yes
7      no
8     yes
9     yes
10    yes
11    yes
12    yes
13     no
Name: play, dtype: object
```

3. Converting nominal data to numeric data

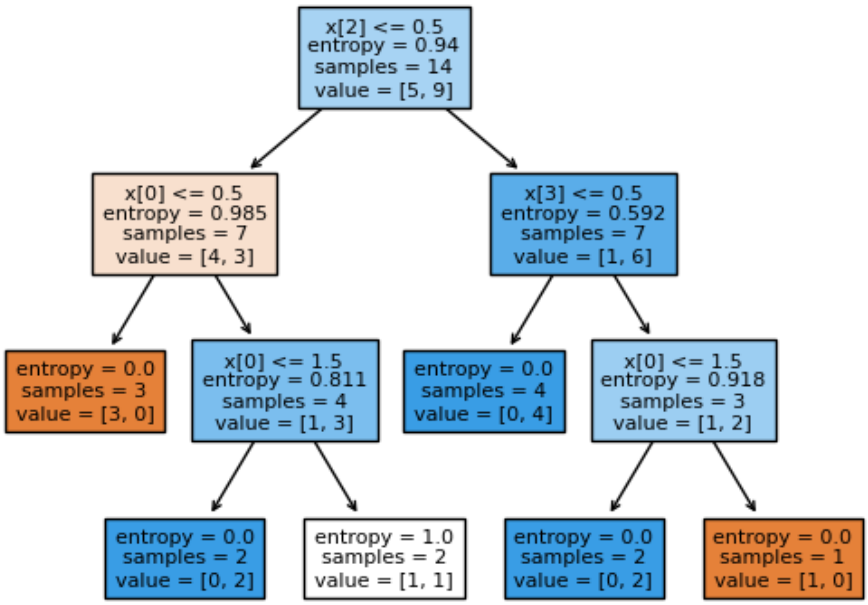
```
In [7]: attribute.outlook.unique()
Out[7]: array(['sunny', 'overcast', 'rainy'], dtype=object)

In [9]: attribute.outlook.value_counts()
Out[9]: outlook
        sunny       5
        rainy       5
        overcast    4
        Name: count, dtype: int64

In [32]: attribute.outlook.replace({'sunny':0,'overcast':1,'rainy':2},inplace=True)

In [11]: attribute.temperature.unique()
Out[11]: array(['hot', 'mild', 'cool'], dtype=object)

In [12]: attribute.temperature.replace({'hot':0,'mild':1,'cool':2},inplace=True)

In [13]: attribute.humidity.unique()
Out[13]: array(['high', 'normal'], dtype=object)

In [14]: attribute.humidity.replace({'high':0,'normal':1},inplace=True)

In [15]: attribute.windy.unique()
Out[15]: array([False,  True])

In [16]: attribute.windy.replace({'False':0,'True':1},inplace=True)

In [17]: target.unique()
Out[17]: array(['no', 'yes'], dtype=object)

In [18]: target.replace({'no':0,'yes':1},inplace=True)
```

4. Decision tree model building

```
In [33]: model=tree.DecisionTreeClassifier(criterion='entropy',max_depth=3)
         model.fit(attribute,target)
Out[33]: DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

5. Decision tree plotting

```
In [35]: fig=plt.figure(figsize=(25,25))
         <Figure size 2500x2500 with 0 Axes>

In [38]: tree.plot_tree(model, filled=True, fontsize=8)
Out[38]: [Text(0.4444444444444444, 0.875, 'x[2] <= 0.5\nentropy = 0.94\nsamples = 14\nvalue = [5, 9]'),
          Text(0.2222222222222222, 0.625, 'x[0] <= 0.5\nentropy = 0.985\nsamples = 7\nvalue = [4, 3]'),
          Text(0.1111111111111111, 0.375, 'entropy = 0.0\nsamples = 3\nvalue = [3, 0]'),
          Text(0.3333333333333333, 0.375, 'x[0] <= 1.5\nentropy = 0.811\nsamples = 4\nvalue = [1, 3]'),
          Text(0.2222222222222222, 0.125, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
          Text(0.4444444444444444, 0.125, 'entropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
          Text(0.6666666666666666, 0.625, 'x[3] <= 0.5\nentropy = 0.592\nsamples = 7\nvalue = [1, 6]'),
          Text(0.5555555555555556, 0.375, 'entropy = 0.0\nsamples = 4\nvalue = [0, 4]'),
          Text(0.7777777777777778, 0.375, 'x[0] <= 1.5\nentropy = 0.918\nsamples = 3\nvalue = [1, 2]'),
          Text(0.6666666666666666, 0.125, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
          Text(0.8888888888888888, 0.125, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]')]
```


x[2] <= 0.5
entropy = 0.94
samples = 14
value = [5, 9]
x[0] <= 0.5
entropy = 0.985
samples = 7
value = [4, 3]
x[3] <= 0.5
entropy = 0.592
samples = 7
value = [1, 6]
entropy = 0.0
samples = 3
value = [3, 0]
x[0] <= 1.5
entropy = 0.811
samples = 4
value = [1, 3]
entropy = 0.0
samples = 4
value = [0, 4]
x[0] <= 1.5
entropy = 0.918
samples = 3
value = [1, 2]
entropy = 0.0
samples = 2
value = [0, 2]
entropy = 1.0
samples = 2
value = [1, 1]
entropy = 0.0
samples = 2
value = [0, 2]
entropy = 0.0
samples = 1
value = [1, 0]

MODEL EVALUATION AND SELECTION

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

PROGRAM:

In [74]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from matplotlib import pyplot as plt
from sklearn import tree
```

In [75]:
```
ds=pd.read_csv("C://Users//MGIT2//Documents//iris.csv")
ds.head()
```

Out[75]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

In [76]:
```
le=LabelEncoder()
for i in ds:
    ds[i]=le.fit_transform(ds[i])
```

In [77]:
```
ds
```

Out[77]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	8	14	4	1	0
1	6	9	4	1	0
2	4	11	3	1	0
3	3	10	5	1	0
4	7	15	4	1	0
...	...	...	...	...	...
145	24	9	28	19	2
146	20	4	26	15	2
147	22	9	28	16	2
148	19	13	30	19	2
149	16	9	27	14	2

150 rows × 5 columns

In [78]:
```
Attr=ds.drop(['variety'],axis='columns')
```

In [79]:
```
Attr
```

Out[79]:

	sepal.length	sepal.width	petal.length	petal.width
0	8	14	4	1
1	6	9	4	1
2	4	11	3	1
3	3	10	5	1
4	7	15	4	1
...	...	...	...	...
145	24	9	28	19
146	20	4	26	15
147	22	9	28	16
148	19	13	30	19
149	16	9	27	14

150 rows × 4 columns

In [80]:
```
Target=ds['variety']
```

In [81]:
```
Target
```

Out[81]:
```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: variety, Length: 150, dtype: int32
```

In [82]:
```
Attr_train,Attr_test,Target_train,Target_test=train_test_split(Attr,Target,test_size=0.7)
```

In [83]:
```
model=DecisionTreeClassifier()
```

In [84]:
```
model.fit(Attr_train,Target_train)
```

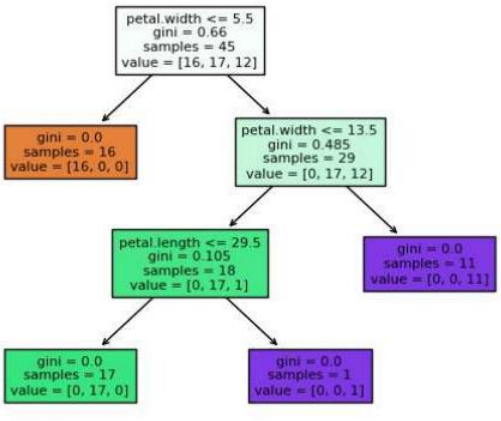
Out[84]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [85]:
```
fig=plt.figure(figsize=(25,20))
```

```
<Figure size 2500x2000 with 0 Axes>
```

In [89]:
```
tree.plot_tree(model, filled=True, feature_names=['sepal.length','sepal.width','petal.length','petal.width'], fontsize=
```

Out[89]:
```
[Text(0.4, 0.875, 'petal.width <= 5.5\ngini = 0.66\nsamples = 45\nvalue = [16, 17, 12]'),
 Text(0.2, 0.625, 'gini = 0.0\nsamples = 16\nvalue = [16, 0, 0]'),
 Text(0.6, 0.625, 'petal.width <= 13.5\ngini = 0.485\nsamples = 29\nvalue = [0, 17, 12]'),
 Text(0.4, 0.375, 'petal.length <= 29.5\ngini = 0.105\nsamples = 18\nvalue = [0, 17, 1]'),
 Text(0.2, 0.125, 'gini = 0.0\nsamples = 17\nvalue = [0, 17, 0]'),
 Text(0.6, 0.125, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.8, 0.375, 'gini = 0.0\nsamples = 11\nvalue = [0, 0, 11]')]
```



In [69]:
```
prd=model.predict(Attr_test)
```

In [70]:
```
prd
```

Out[70]:
```
array([1, 0, 0, 2, 1, 2, 1, 1, 0, 2, 1, 1, 1, 1, 1, 2, 0, 0, 2, 2, 0, 0,
       2, 2, 0, 1, 2, 2, 1, 1, 2, 2, 1, 0, 1, 0, 2, 2, 0, 0, 2, 0, 0, 0,
       2, 1, 1, 1, 1, 1, 2, 2, 0, 0, 1, 1, 0, 2, 1, 0, 2, 1, 2, 2, 1, 0,
       0, 0, 2, 0, 1, 1, 2, 1, 0, 2, 0, 0, 1, 1, 1, 0, 1, 2, 0, 0, 1, 1,
       0, 0, 2, 1, 0, 2, 2, 1, 1, 1, 1, 1, 2, 0, 0, 0, 1])
```

In [71]:
```
confusion_matrix(Target_test,prd)
```

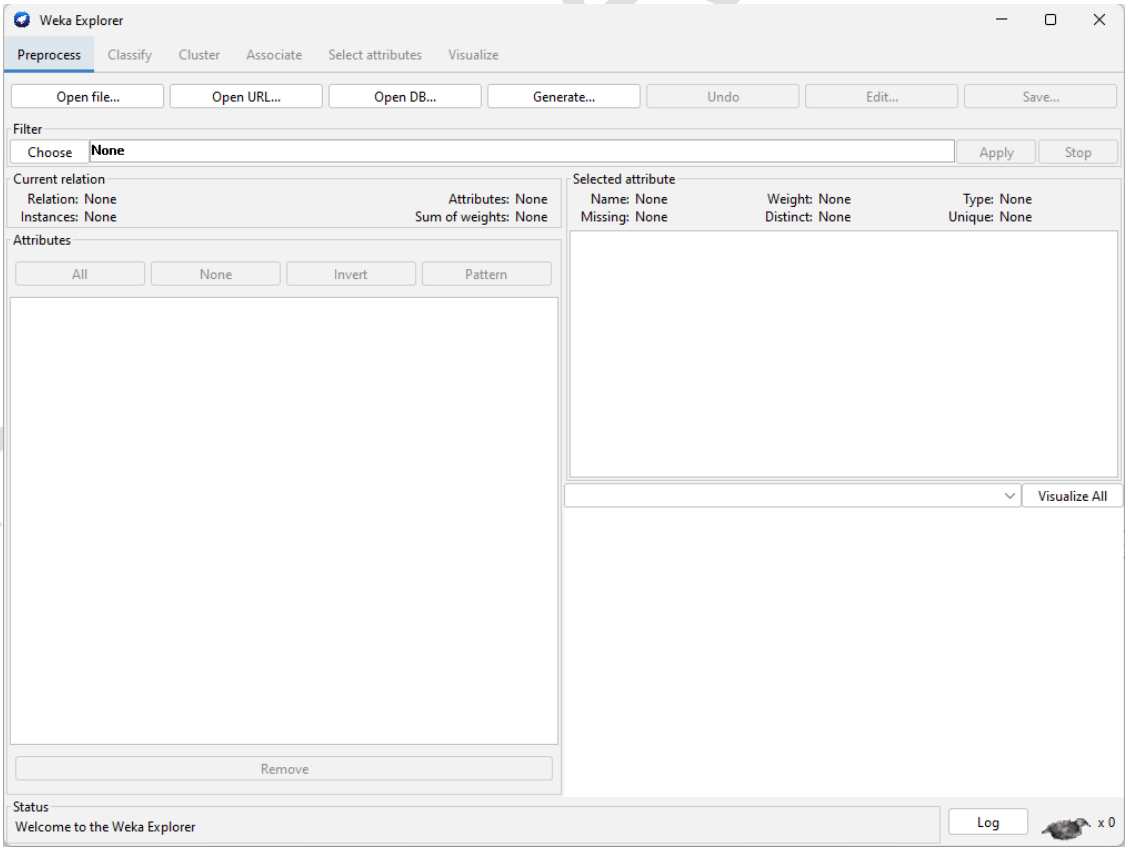
Out[71]:
```
array([[35,  1,  0],
       [ 0, 33,  2],
       [ 0,  6, 28]], dtype=int64)
```

NAIVE BAYE'S CLASSIFICATION

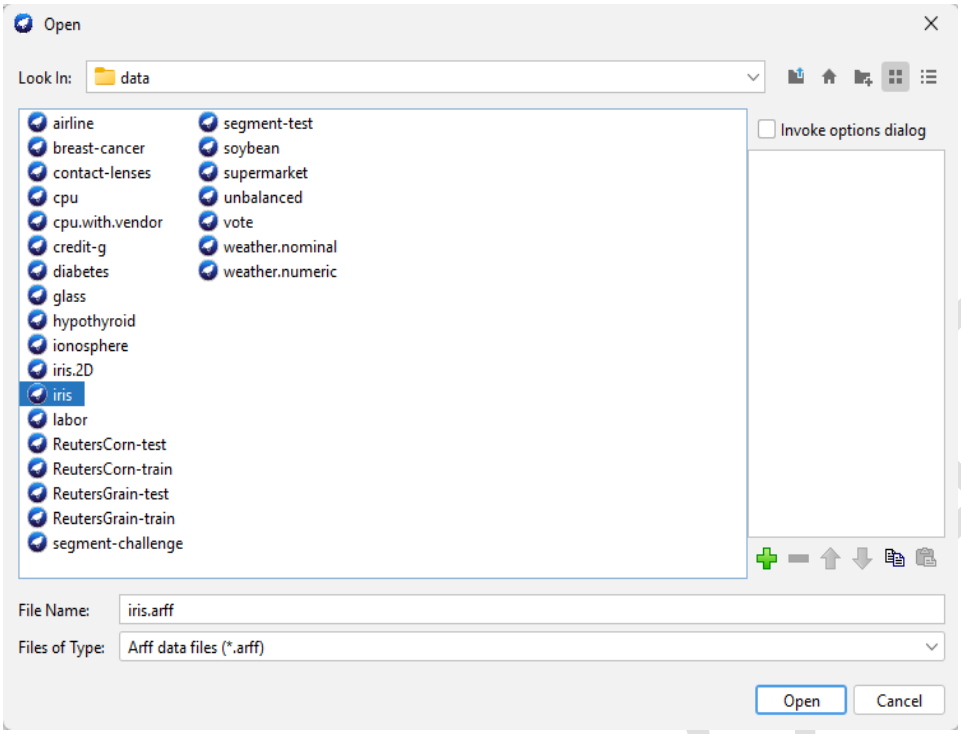
AIM: To classify the dataset using Naive Baye's Theorem algorithm

PROCEDURE:

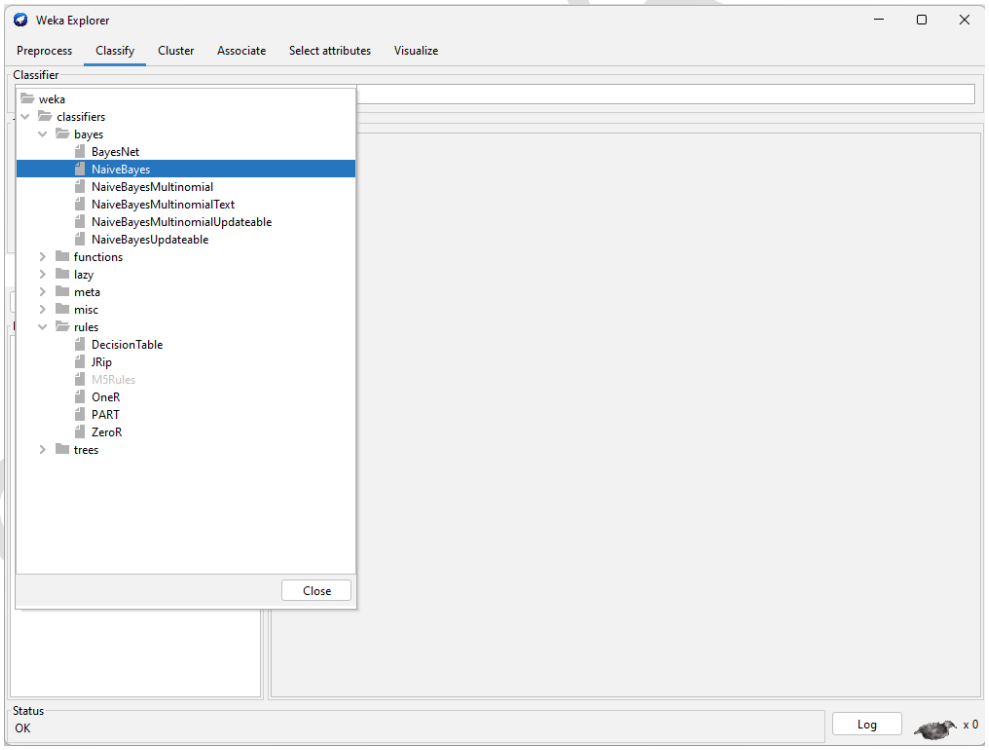
1. Open WEKA tool and open explorer.



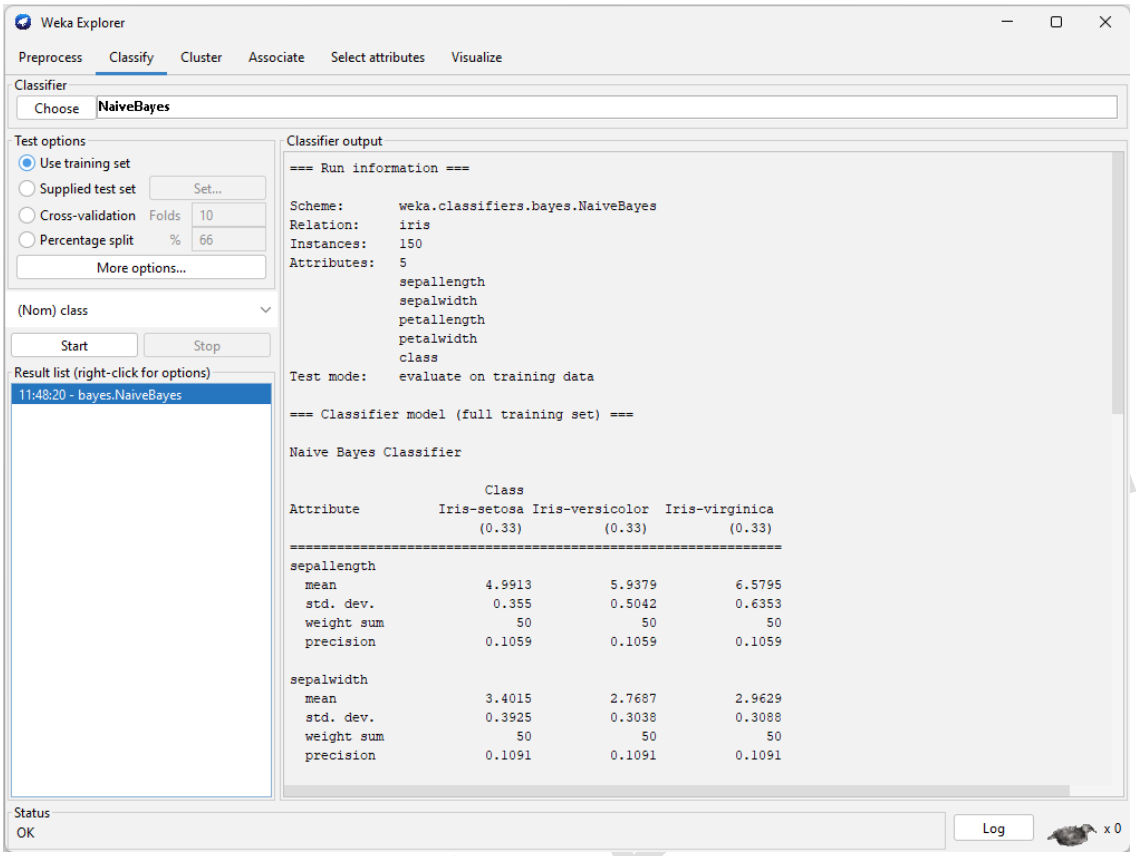
2. Now click "**Open File",** and locate the dataset from program file.

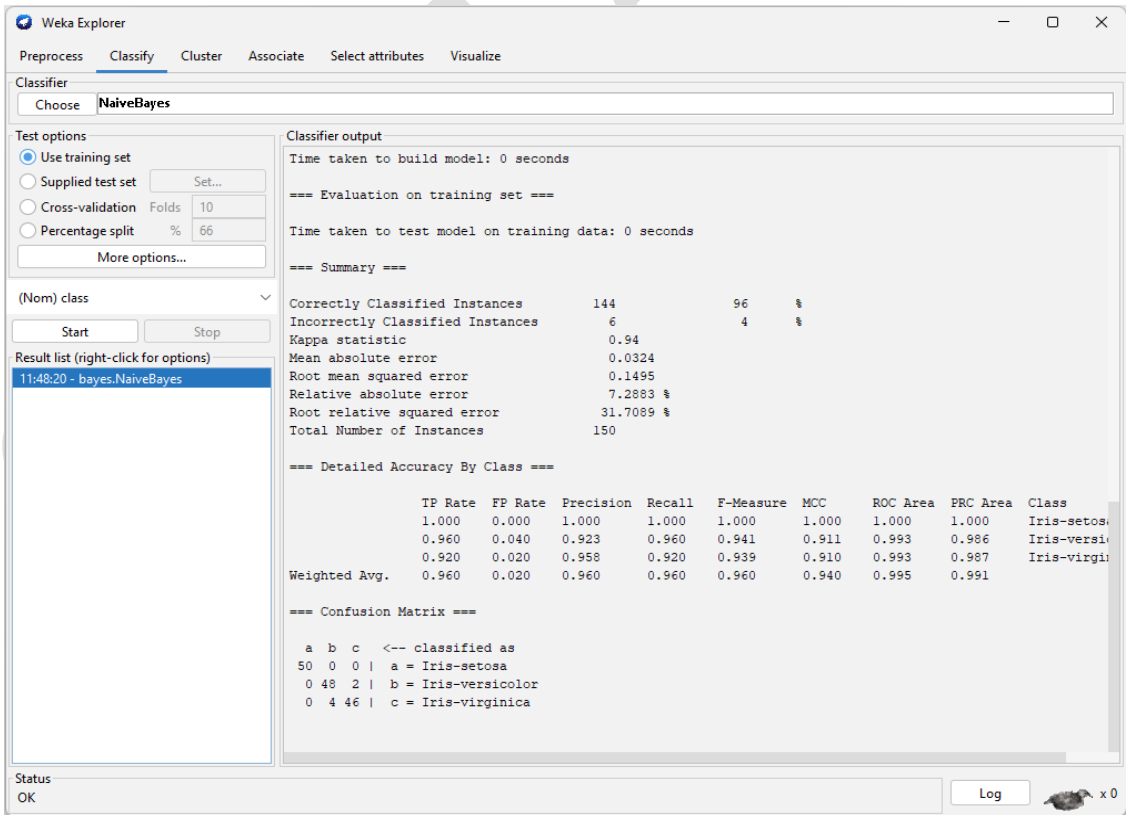


3. Now click on the **"Classify"** and choose **"NaiveBayes"** from **"classifiers".**

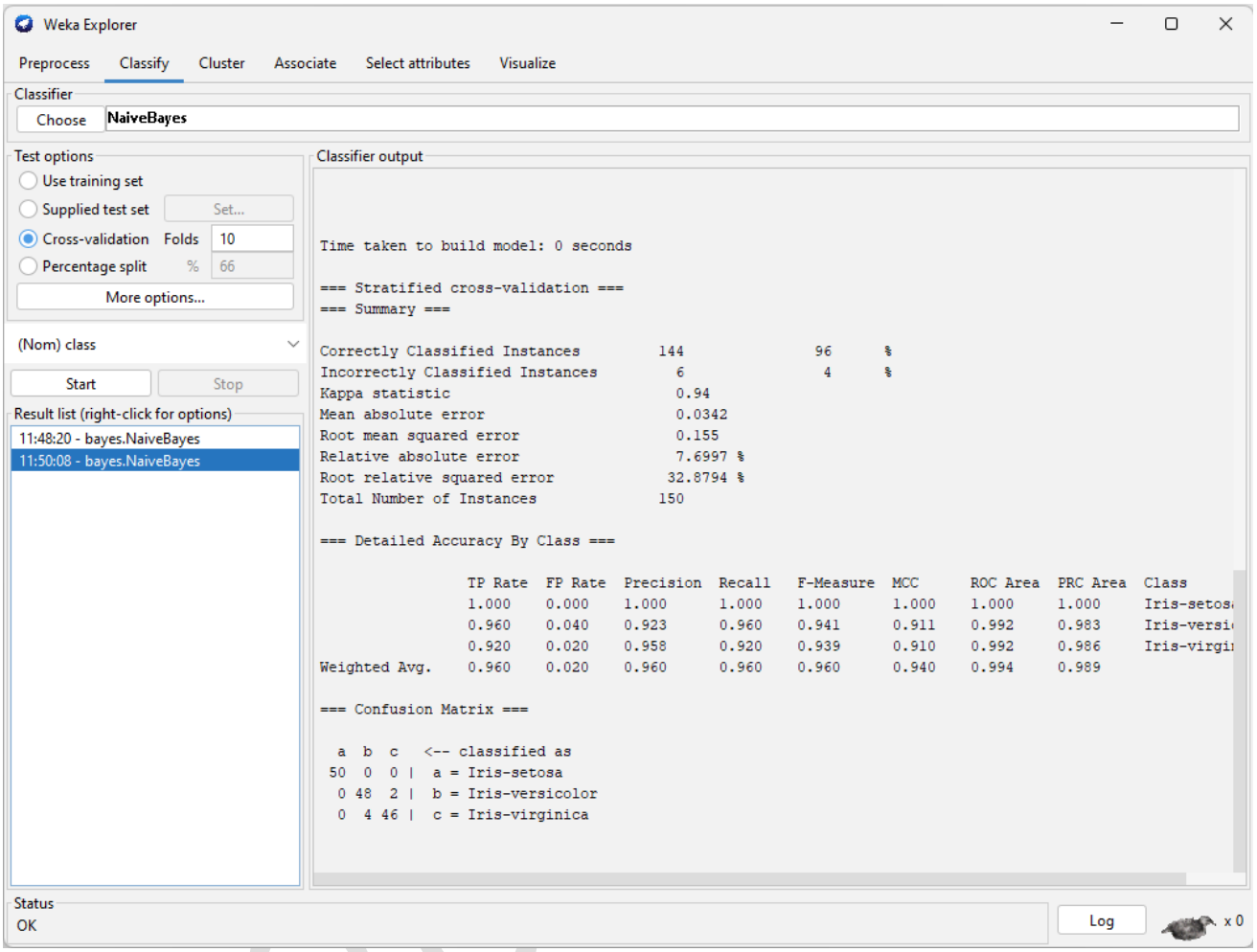


4. From **"Test options"** choose **"Use training set"** and run the classifier.





5. Now choose **"Cross-validation"** and run the classifier to obtain the confusion matrix and a result list.



Gain Ratio Attribute

AIM: To implement Gain Ratio Attribute.

PROCEDURE:

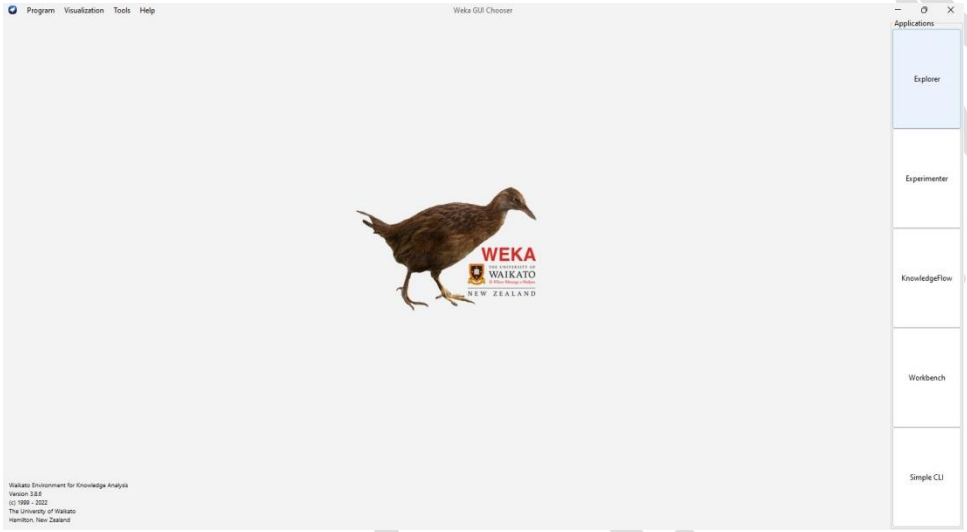
1.open weka explorer

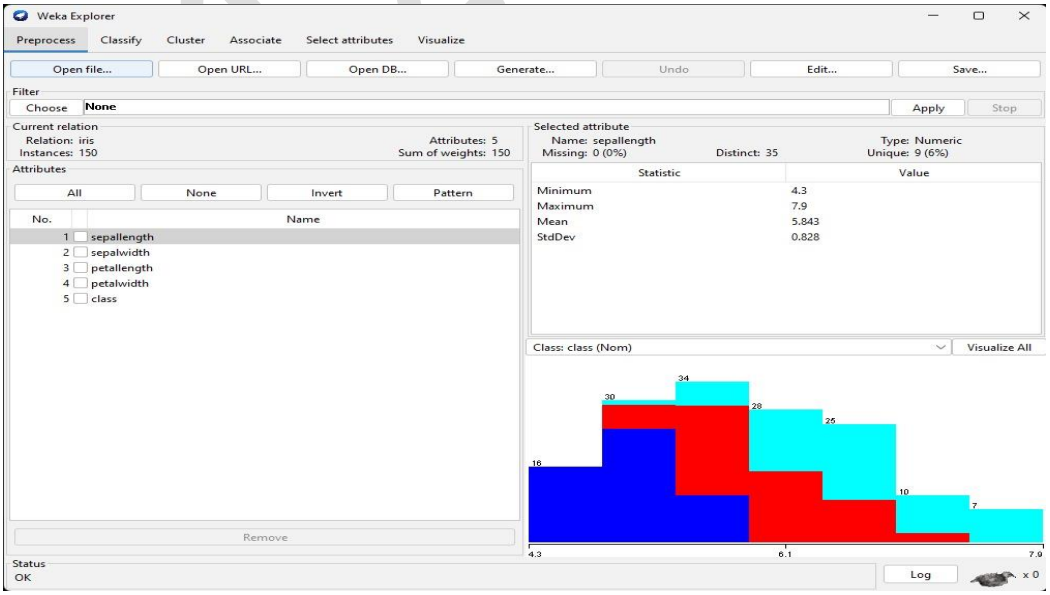
2.Goto open file->C Drive->program files->weka 3.8.6->Data->iris.

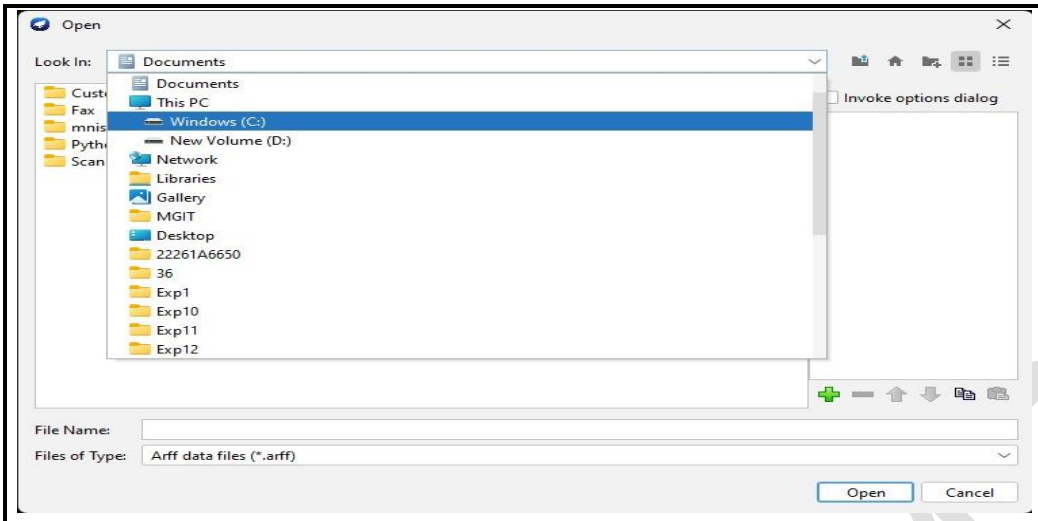
3.Goto classify,then choose Gain Ratio Attribute in classifier->Meta->Attribute selected classifier.

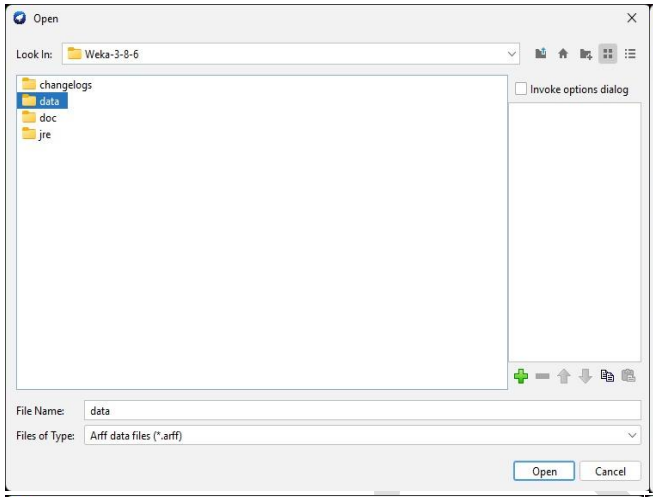
4.choose Ranker Algorithmfor Gain Ratio.

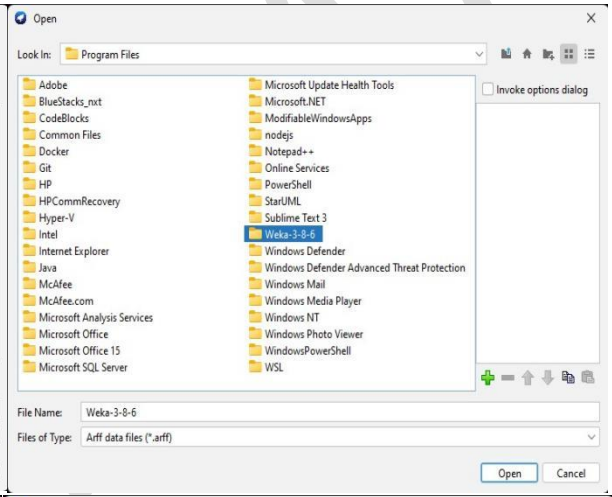
5.Then click on use training set and then start to view the output.

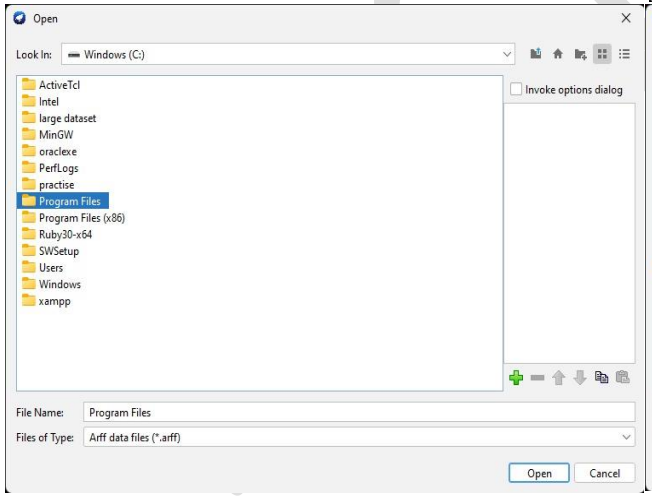


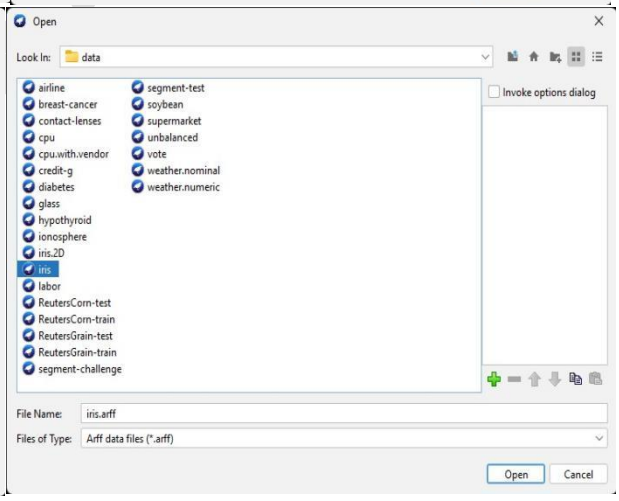


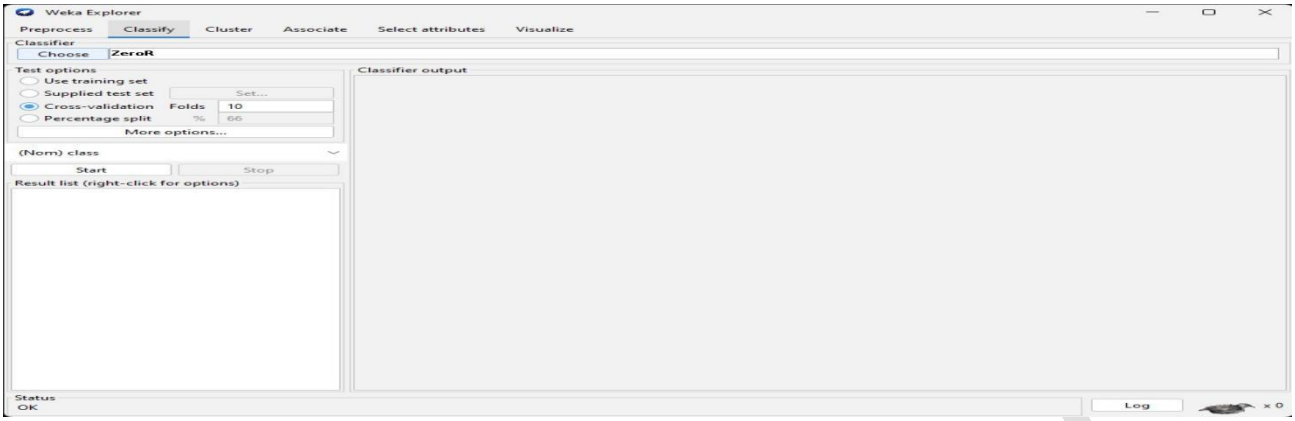

Open
Look In: Documents
Documents
This PC
Windows (C:)
New Volume (D:)
Network
Libraries
Gallery
MGIT
Desktop
22261A6650
36
Exp1
Exp10
Exp11
Exp12
Invoke options dialog
File Name:
Files of Type: Arff data files (*.arff)
Open
Cancel

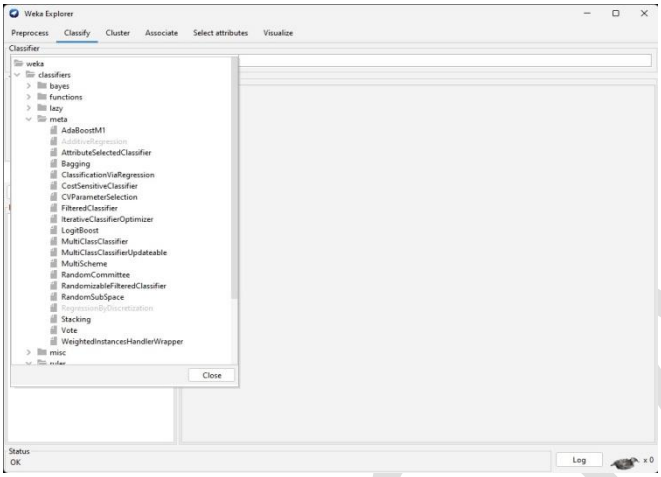

Open
Look In: Weka-3-8-6
changelogs
data
doc
jre
Invoke options dialog
File Name: data
Files of Type: Arff data files (*.arff)
Open
Cancel

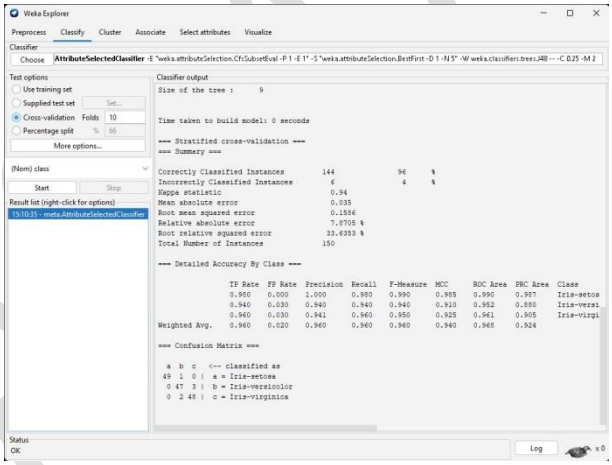

Open
Look In: Program Files
Adobe
BlueStacks_nxt
CodeBlocks
Common Files
Docker
Git
HP
HPCommRecovery
Hyper-V
Intel
Internet Explorer
Java
McAfee
McAfee.com
Microsoft Analysis Services
Microsoft Office
Microsoft Office 15
Microsoft SQL Server
Microsoft Update Health Tools
Microsoft.NET
ModifiableWindowsApps
nodejs
Notepad++
Online Services
PowerShell
StarUML
Sublime Text 3
Weka-3-8-6
Windows Defender
Windows Defender Advanced Threat Protection
Windows Mail
Windows Media Player
Windows NT
Windows Photo Viewer
WindowsPowerShell
WSL
Invoke options dialog
File Name: Weka-3-8-6
Files of Type: Arff data files (*.arff)
Open
Cancel

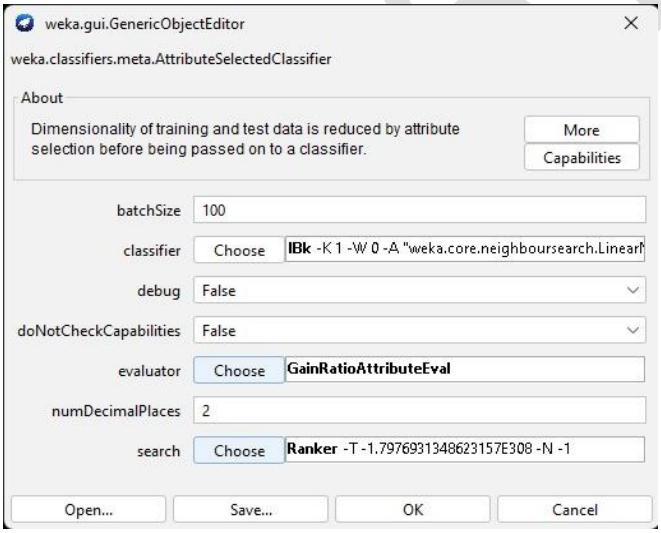

Open
Look In: Windows (C:)
ActiveTcl
Intel
large dataset
MinGW
oraclexe
PerfLogs
practise
Program Files
Program Files (x86)
Ruby30-x64
SWSetup
Users
Windows
xampp
Invoke options dialog
File Name: Program Files
Files of Type: Arff data files (*.arff)
Open
Cancel

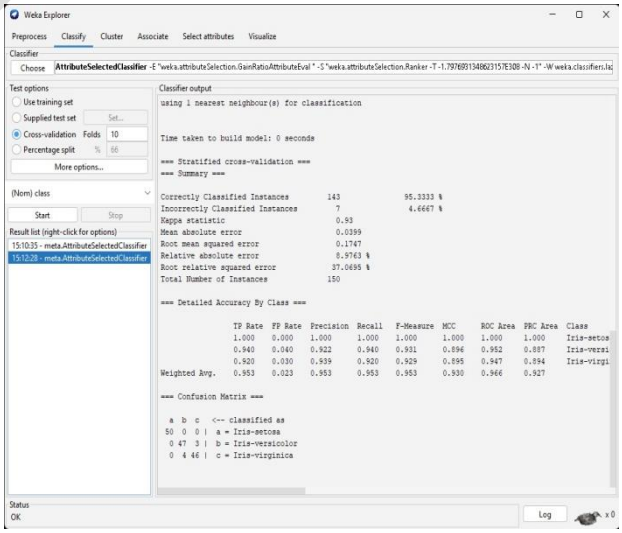

Open
Look In: data
airline
breast-cancer
contact-lenses
cpu
cpu.with.vendor
credit-g
diabetes
glass
hypothyroid
ionosphere
iris.2D
iris
labor
ReutersCorn-test
ReutersCorn-train
ReutersGrain-test
ReutersGrain-train
segment-challenge
segment-test
soybean
supermarket
unbalanced
vote
weather.nominal
weather.numeric
Invoke options dialog
File Name: iris.arff
Files of Type: Arff data files (*.arff)
Open
Cancel

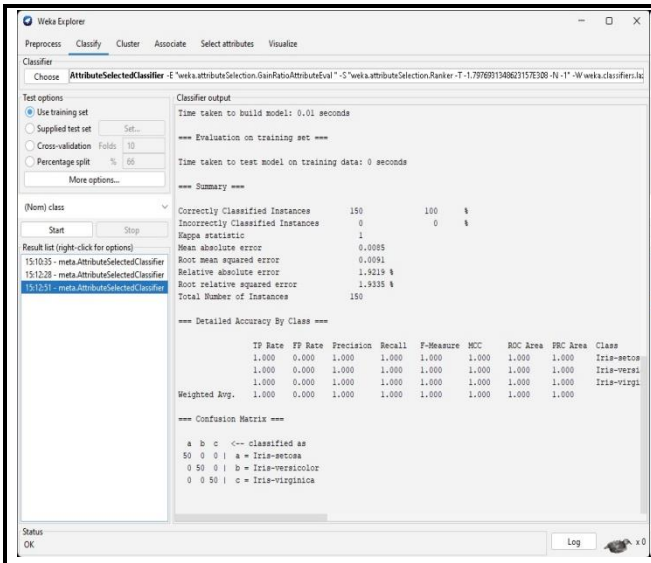

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
ZeroR
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
Classifier output
(Nom) class
Start
Stop
Result list (right-click for options)
Status
OK
Log
x 0

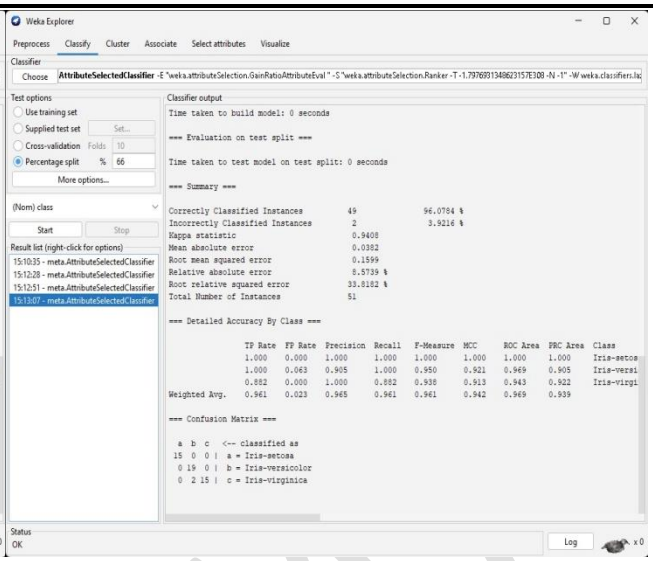

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
weka
classifiers
bayes
functions
lazy
meta
AdaBoostM1
AttributeSelectedClassifier
Bagging
ClassificationViaRegression
CostSensitiveClassifier
CVParameterSelection
FilteredClassifier
IterativeClassifierOptimizer
LogitBoost
MultiClassClassifier
MultiClassClassifierUpdateable
MultiScheme
RandomCommittee
RandomizableFilteredClassifier
RandomSubSpace
RegressionByDiscretization
Stacking
Vote
WeightedInstancesHandlerWrapper
misc
Close
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
AttributeSelectedClassifier -E "weka.attributeSelection.CfsSubsetEval -P 1 -E 1" -S "weka.attributeSelection.BestFirst -D 1 -N 5" -W weka.classifiers.trees.J48 -- -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
Classifier output
Size of the tree : 9
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 144 96 %
Incorrectly Classified Instances 6 4 %
Kappa statistic 0.94
Mean absolute error 0.035
Root mean squared error 0.1586
Relative absolute error 7.8705 %
Root relative squared error 33.6353 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.980 0.000 1.000 0.980 0.990 0.985 0.990 0.987 Iris-setos
0.940 0.030 0.940 0.940 0.940 0.910 0.952 0.880 Iris-versi
0.960 0.030 0.941 0.960 0.950 0.925 0.961 0.905 Iris-virgi
Weighted Avg. 0.960 0.020 0.960 0.960 0.960 0.940 0.968 0.924
=== Confusion Matrix ===
a b c <-- classified as
49 1 0 | a = Iris-setosa
0 47 3 | b = Iris-versicolor
0 2 48 | c = Iris-virginica
Status
OK
Log
x 0


weka.gui.GenericObjectEditor
weka.classifiers.meta.AttributeSelectedClassifier
About
Dimensionality of training and test data is reduced by attribute selection before being passed on to a classifier.
More
Capabilities
batchSize
100
classifier
Choose
IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearN
debug
False
doNotCheckCapabilities
False
evaluator
Choose
GainRatioAttributeEval
numDecimalPlaces
2
search
Choose
Ranker -T -1.7976931348623157E308 -N -1
Open...
Save...
OK
Cancel


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
AttributeSelectedClassifier -E "weka.attributeSelection.GainRatioAttributeEval " -S "weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1" -W weka.classifiers.la
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
Classifier output
using 1 nearest neighbour(s) for classification
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 143 95.3333 %
Incorrectly Classified Instances 7 4.6667 %
Kappa statistic 0.93
Mean absolute error 0.0399
Root mean squared error 0.1747
Relative absolute error 8.9763 %
Root relative squared error 37.0695 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Iris-setos
0.940 0.040 0.922 0.940 0.931 0.896 0.952 0.887 Iris-versi
0.920 0.030 0.939 0.920 0.929 0.895 0.947 0.894 Iris-virgi
Weighted Avg. 0.953 0.023 0.953 0.953 0.953 0.930 0.966 0.927
=== Confusion Matrix ===
a b c <-- classified as
50 0 0 | a = Iris-setosa
0 47 3 | b = Iris-versicolor
0 4 46 | c = Iris-virginica
Status
OK
Log
x 0


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose AttributeSelectedClassifier -E "weka.attributeSelection.GainRatioAttributeEval " -S "weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1" -W weka.classifiers.la
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)
15:10:35 - meta.AttributeSelectedClassifier
15:12:28 - meta.AttributeSelectedClassifier
15:12:51 - meta.AttributeSelectedClassifier
Classifier output
Time taken to build model: 0.01 seconds
=== Evaluation on training set ===
Time taken to test model on training data: 0 seconds
=== Summary ===
Correctly Classified Instances 150 100 %
Incorrectly Classified Instances 0 0 %
Kappa statistic 1
Mean absolute error 0.0085
Root mean squared error 0.0091
Relative absolute error 1.9219 %
Root relative squared error 1.9335 %
Total Number of Instances 150
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Iris-setos
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Iris-versi
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Iris-virgi
Weighted Avg. 1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000
=== Confusion Matrix ===
a b c <-- classified as
50 0 0 | a = Iris-setosa
0 50 0 | b = Iris-versicolor
0 0 50 | c = Iris-virginica
Status
OK
Log
x 0


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose AttributeSelectedClassifier -E "weka.attributeSelection.GainRatioAttributeEval " -S "weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1" -W weka.classifiers.la
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)
15:10:35 - meta.AttributeSelectedClassifier
15:12:28 - meta.AttributeSelectedClassifier
15:12:51 - meta.AttributeSelectedClassifier
15:13:07 - meta.AttributeSelectedClassifier
Classifier output
Time taken to build model: 0 seconds
=== Evaluation on test split ===
Time taken to test model on test split: 0 seconds
=== Summary ===
Correctly Classified Instances 49 96.0784 %
Incorrectly Classified Instances 2 3.9216 %
Kappa statistic 0.9408
Mean absolute error 0.0382
Root mean squared error 0.1599
Relative absolute error 8.5739 %
Root relative squared error 33.8182 %
Total Number of Instances 51
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Iris-setos
1.000 0.063 0.905 1.000 0.950 0.921 0.969 0.905 Iris-versi
0.882 0.000 1.000 0.882 0.938 0.913 0.943 0.922 Iris-virgi
Weighted Avg. 0.961 0.023 0.965 0.961 0.961 0.942 0.969 0.939
=== Confusion Matrix ===
a b c <-- classified as
15 0 0 | a = Iris-setosa
0 19 0 | b = Iris-versicolor
0 2 15 | c = Iris-virginica
Status
OK
Log
x 0

GAIN RATIO ATTRIBUTE (*PYTHON*)

AIM: To implement Gain Ratio Attribute using python.

PROGRAM:

In [1]:
```
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score
```

In [2]:
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
```

In [5]:
```
dataset=pd.read_csv("C:/Users/MPIT/Documents/irisdata.csv")
dataset.head()
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [7]:
```
attri=dataset.drop(["species"],axis="columns")
```

In [8]:
```
attri.head()
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width
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

In [9]:
```
target=dataset["species"]
target
```

Out[9]:
```
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
          ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

In [14]:
```
attri_train,attri_test,target_train,target_test=train_test_split(attri,target,test_size=0.5)
```

In [15]:
```
classifier=KNeighborsClassifier(n_neighbors=20)
classifier.fit(attri_train,target_train)
```

Out[15]:
```
KNeighborsClassifier(n_neighbors=20)
```

In [16]:
```
pred=classifier.predict(attri_test)
```

In [17]:
```
accuracy_score(target_test,pred)
```

Out[17]:
```
0.9333333333333333
```

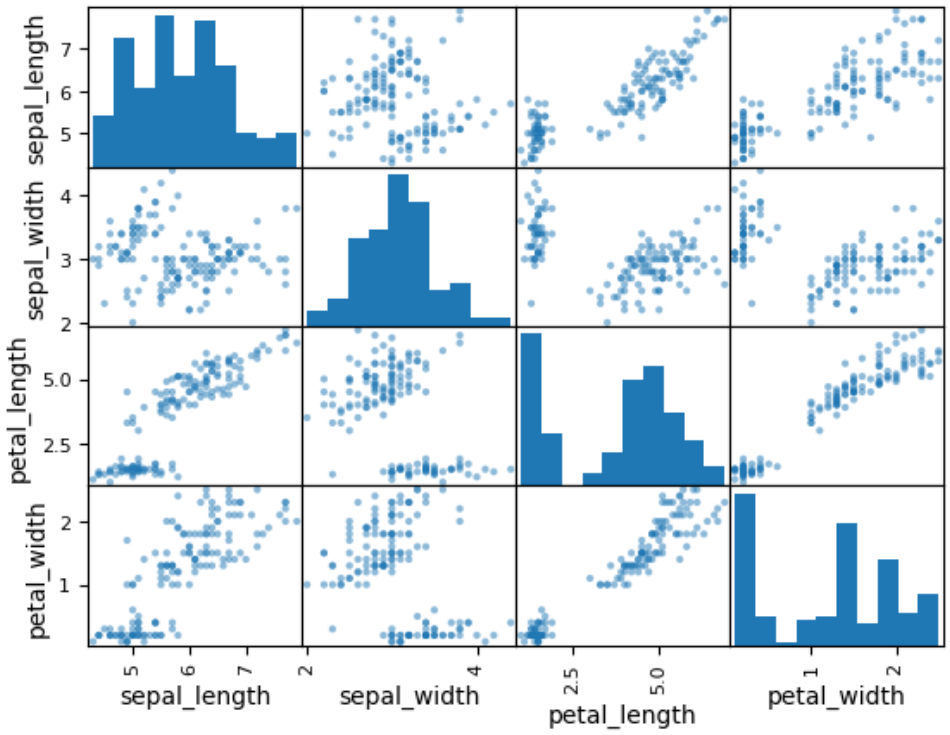
In [18]:
```
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
```

In [19]:
```
features=attri.columns
features
```

Out[19]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width'], dtype='object')
```

In [20]:
```
scatter_matrix(dataset[features])
```

Out[20]:
```
array([[<Axes: xlabel='sepal_length', ylabel='sepal_length'>,
        <Axes: xlabel='sepal_width', ylabel='sepal_length'>,
        <Axes: xlabel='petal_length', ylabel='sepal_length'>,
        <Axes: xlabel='petal_width', ylabel='sepal_length'>],
       [<Axes: xlabel='sepal_length', ylabel='sepal_width'>,
        <Axes: xlabel='sepal_width', ylabel='sepal_width'>,
        <Axes: xlabel='petal_length', ylabel='sepal_width'>,
        <Axes: xlabel='petal_width', ylabel='sepal_width'>],
       [<Axes: xlabel='sepal_length', ylabel='petal_length'>,
        <Axes: xlabel='sepal_width', ylabel='petal_length'>,
        <Axes: xlabel='petal_length', ylabel='petal_length'>,
        <Axes: xlabel='petal_width', ylabel='petal_length'>],
       [<Axes: xlabel='sepal_length', ylabel='petal_width'>,
        <Axes: xlabel='sepal_width', ylabel='petal_width'>,
        <Axes: xlabel='petal_length', ylabel='petal_width'>,
        <Axes: xlabel='petal_width', ylabel='petal_width'>]], dtype=object)
```

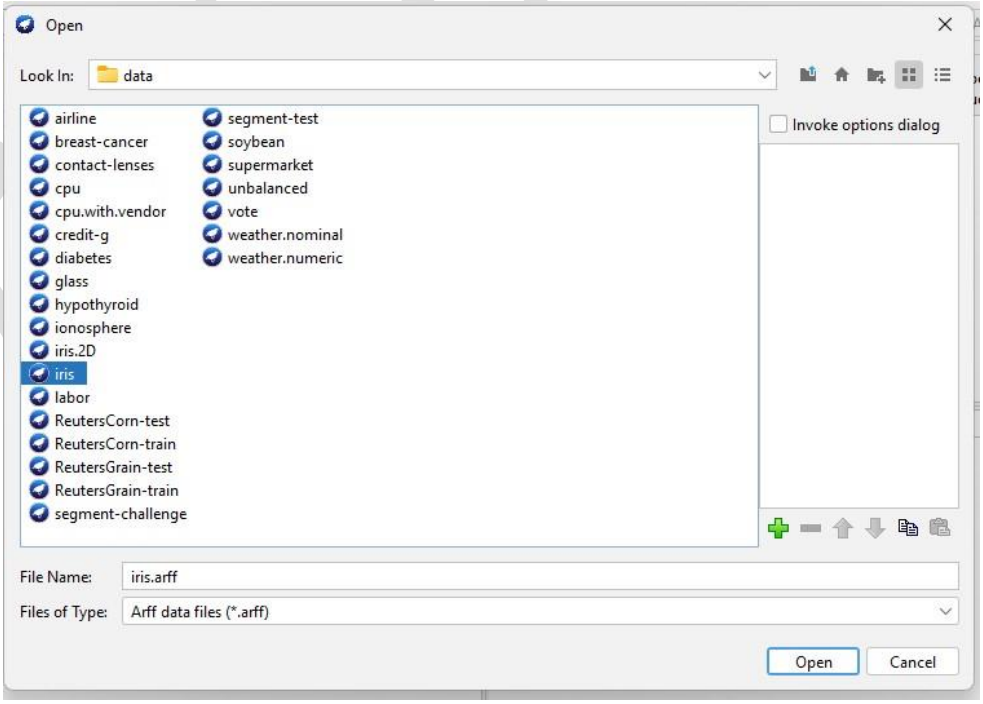


K-MEANS CLUSTERING ALGORITHM (WEKA)

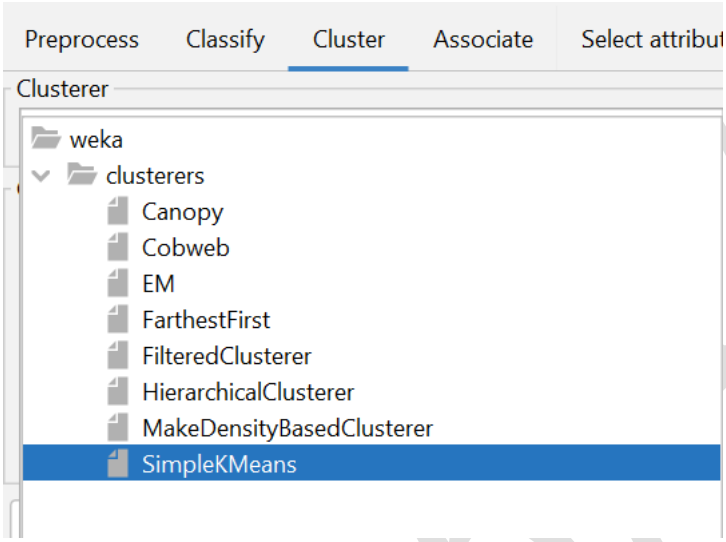
AIM: Run the K-means clustering algorithm for the below data set, and study the clusters formed in WEKA.

PROCEDURE:

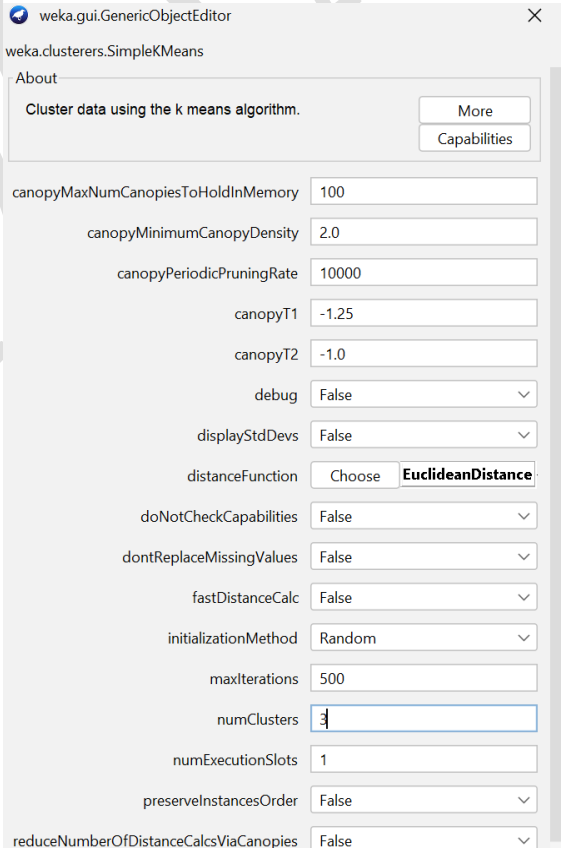
Step 1: In the preprocessing interface, open the Weka Explorer and load the required dataset, and we are taking the iris.arff dataset.



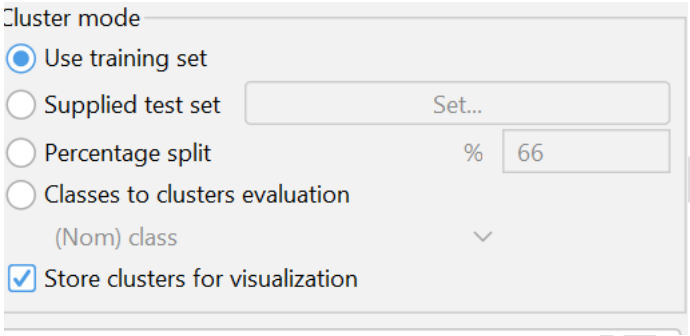
Step 2: Find the 'cluster' tab in the explorer and press the choose button to execute clustering. A dropdown list of available clustering algorithms appears as a result of this step and selects the simple-k means algorithm.



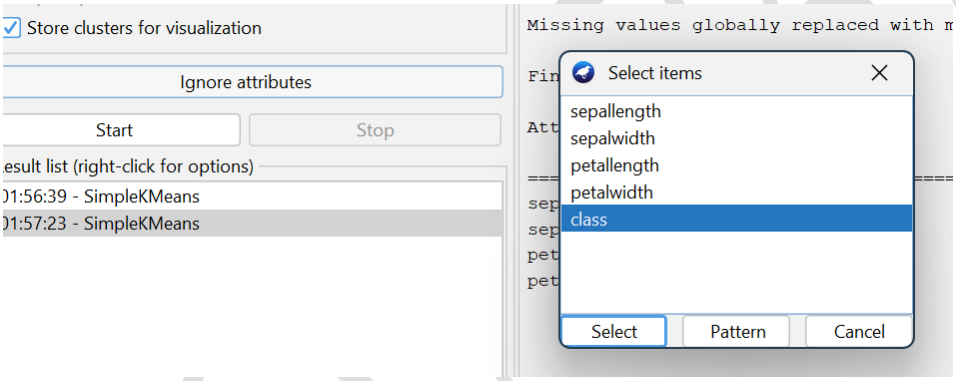
Step 3: Then, to the right of the choose icon, press the text button to bring up the popup window shown in the screenshots. We enter three for the number of clusters in this window and leave the seed value alone. The seed value is used to generate a random number that is used to make internal assignments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



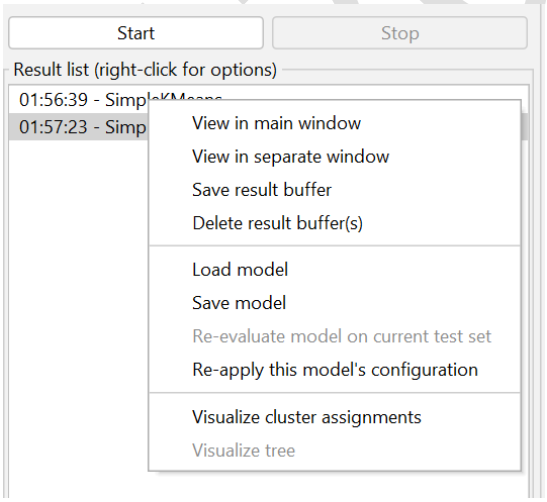
Step 4: One of the choices has been chosen. We must ensure that they are in the 'cluster mode' panel before running the clustering algorithm

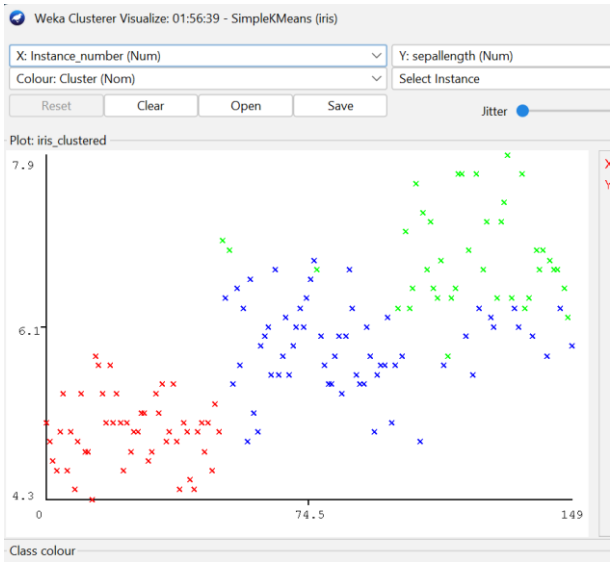


Step 5: Click on the ignore attribute and select class and select the start button



Step 6: Right click on Simple K Means and select Visualize Cluster assignments to visualize the cluster.





Following results are shown. Which show final cluster centroid and number of iteration.

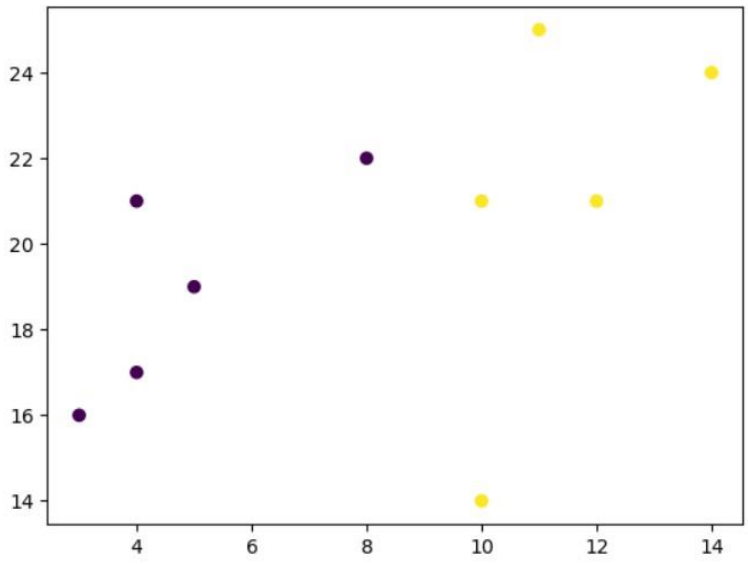
```
Number of iterations: 6
Within cluster sum of squared errors: 6.998114004826762

Initial starting points (random):

Cluster 0: 6.1,2.9,4.7,1.4
Cluster 1: 6.2,2.9,4.3,1.3
Cluster 2: 6.9,3.1,5.1,2.3

Missing values globally replaced with mean/mode

Final cluster centroids:
                           Cluster#
Attribute      Full Data          0          1          2
                 (150.0)     (61.0)     (50.0)     (39.0)
=============================================================
sepallength       5.8433     5.8885      5.006     6.8462
sepalwidth         3.054     2.7377      3.418     3.0821
petallength       3.7587     4.3967      1.464     5.7026
petalwidth        1.1987      1.418      0.244     2.0795
```

K-MEANS CLUSTERING ALGORITHM (PYTHON)

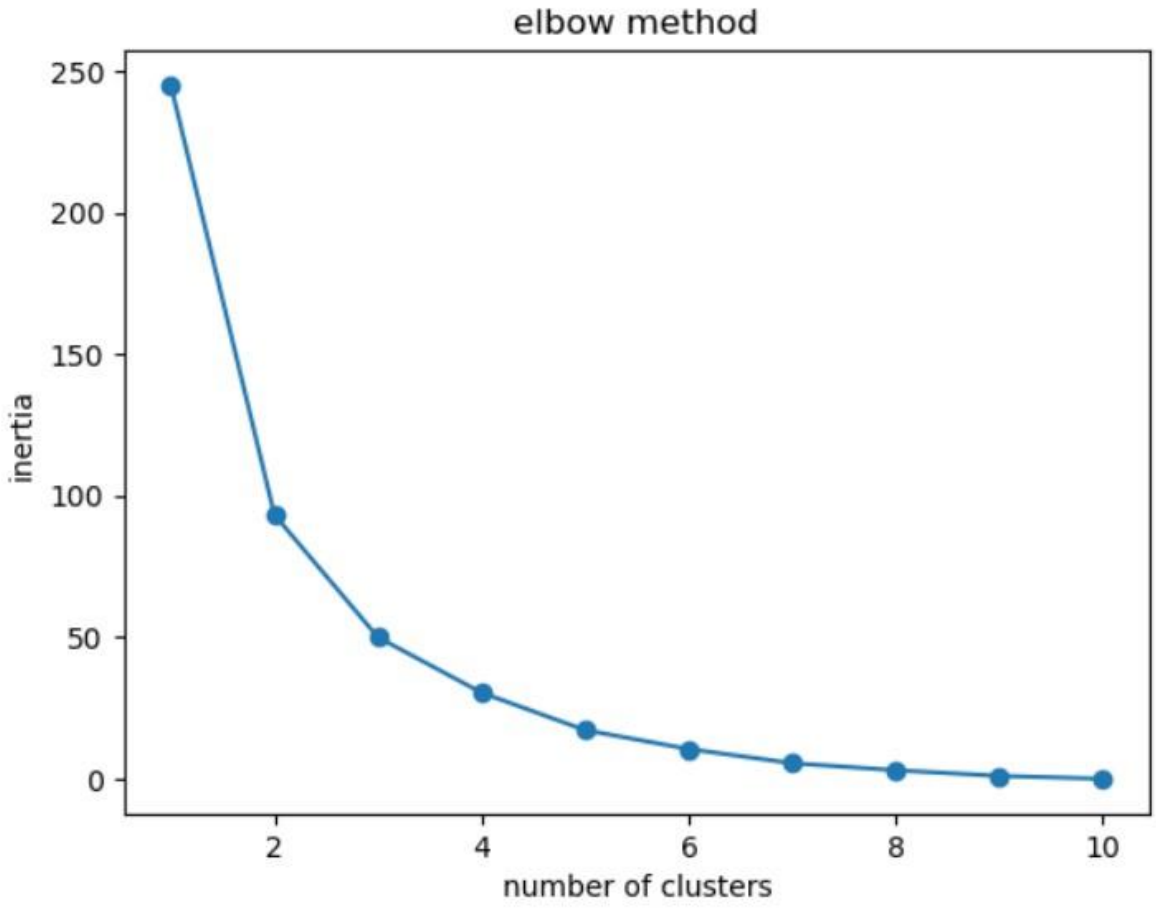
AIM: To implement K-Means Clustering algorithm using python.

PROGRAM:

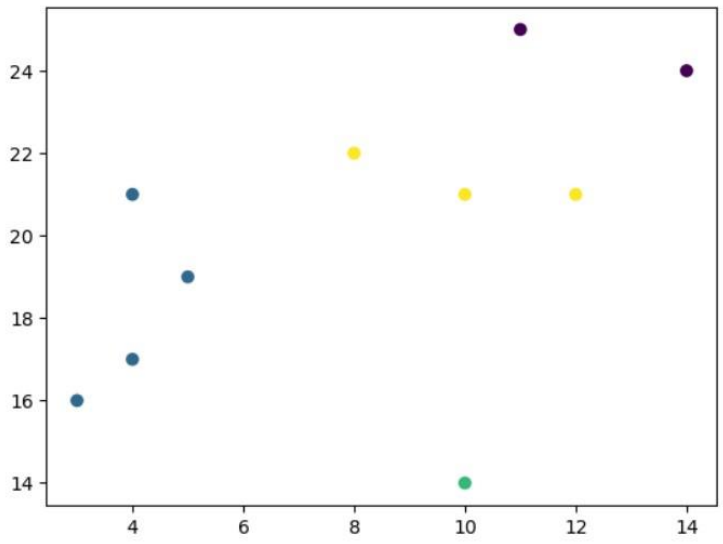
```
import numpy as np
import matplotlib.pyplot as plt

x = [4, 5, 10, 4, 3, 11, 14, 8, 10, 12]
y = [21, 19, 14, 17, 16, 25, 24, 22, 21, 21]

classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]

plt.scatter(x, y, c=classes)
plt.show()
```




```
from sklearn.cluster import KMeans

data = list(zip(x, y))
inertias = []

for i in range(1,11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(data)
    inertias.append(kmeans.inertia_)

plt.plot(range(1,11), inertias, marker='o')
plt.title('Elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```




```
kmeans = KMeans(n_clusters=4)
kmeans.fit(data)

plt.scatter(x, y, c=kmeans.labels_)
plt.show()
```



K Means Using Python Code:

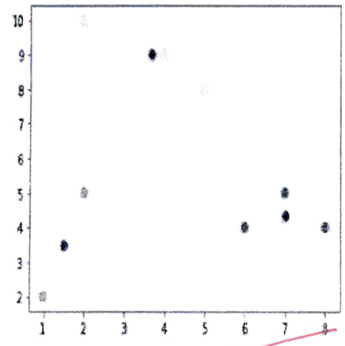
jupyter Untitled14 Last Checkpoint 12/09/2022 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [6]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
data={'x':[2,2,8,5,7,6,1,4],'y':[10,5,4,8,5,4,2,9]}
df=pd.DataFrame(data,columns=['x','y'])
kmeans=KMeans(n_clusters=3).fit(df)
centroids=kmeans.cluster_centers_
print(centroids)
plt.scatter(df['x'],df['y'],c=kmeans.labels_.astype(float),s=50,alpha=0.5)
plt.scatter(centroids[:,0],centroids[:,1],c='red',s=50)
plt.show()
```

```
[[7.         4.33333333]
 [1.5        3.5       ]
 [3.66666667 9.        ]]
```



EXPERIMENT NAME: K Medoid

AIM: To Implement K Medoid Using Python.

```
import pandas as pd
import numpy as np
from sklearn_extra.cluster import KMedoids
from sklearn import preprocessing
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('D:/iris11.csv',sep=',',header='infer')
df.head()
```

Out[2]:

	sepallength	sepalwidth	petallength	petalwidth	class
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

In [3]:
```
X=df
Y=df['class']
```

In [4]:
```
le = preprocessing.LabelEncoder()

X['class'] = le.fit_transform(X['class'])

Y = le.transform(Y)
```

In [5]:
```
cols = X.columns
```

In [6]:
```
from sklearn.preprocessing import MinMaxScaler

ms = MinMaxScaler()

X = ms.fit_transform(X)
```

ROLL NO: 20261A6743

```
X = pd.DataFrame(X, columns=[cols])
```

```
X.head()
```

Out[8]:

	sepallength	sepalwidth	petallength	petalwidth	class
0	0.222222	0.625000	0.067797	0.041667	0.0
1	0.166667	0.416667	0.067797	0.041667	0.0
2	0.111111	0.500000	0.050847	0.041667	0.0
3	0.083333	0.458333	0.084746	0.041667	0.0
4	0.194444	0.666667	0.067797	0.041667	0.0

In [9]:
```
km = KMedoids(n_clusters=4,random_state=0)
km.fit(X)
```

Out[9]:
```
KMedoids(n_clusters=4, random_state=0)
```

In [10]:
```
km.cluster_centers_
```

Out[10]:
```
array([[0.41666667, 0.29166667, 0.69491525, 0.75      , 1.        ],
       [0.38888889, 0.33333333, 0.52542373, 0.5       , 0.5       ],
       [0.19444444, 0.58333333, 0.08474576, 0.04166667, 0.        ],
       [0.69444444, 0.41666667, 0.76271186, 0.83333333, 1.        ]])
```

In [11]:
```
km.inertia_
```

Out[11]:
```
28.255889443186035
```

In [12]:
```
labels = km.labels_
correct_labels = sum(Y == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, Y.si
```

```
Result: 50 out of 150 samples were correctly labeled.
```

In [13]:
```
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(Y.size)))
```

```
Accuracy score: 0.33
```