Experiment NO.1

Dimension Product

```
create table product(
 product_id int primary key,
 product_name varchar(50),
 product_category varchar(50),
 unit_price int
)
insert into product values(1010, 'LG', 'TV',2000)
insert into product values(1011, 'Sony', 'TV',35000)
insert into product values(1012, 'Redmi', 'Mobile',20000)
insert into product values(1013, 'Samsung', 'TV',50000)
insert into product values(1014, 'Apple', 'Mobile',75000)
insert into product values(1015, 'Nokia', 'Mobile',22000)
select * from product
```

Dimension Employee

```
create table employee(
 employee_id int primary key,
 employee_name varchar(50),
 title varchar(50),
 department varchar(50),
 region varchar(50)
)
insert into employee values(191041, 'rohit','rd','sales','kharghar')
insert into employee values(191042, 'mohit','rd','sales','kharghar')
insert into employee values(191043, 'shubham','rd','sales','kharghar')
insert into employee values(191044, 'dipak','rd','sales','kharghar')
insert into employee values(191045, 'aditi','rd','sales','kharghar')
insert into employee values(191046, 'kaneshka','rd','sales','kharghar')
select * from employee
```

Dimension Customer

```
create table customer(
 customer_id int primary key,
 customer_name varchar(50),
 customer_address varchar(50),
 city varchar(10),
 zip int
)
insert into customer values(191056, 'rohit','kharghar','navimumbai',410210)
insert into customer values(191057, 'mohit','kharghar','navimumbai',410210)
insert into customer values(191058, 'shubham','kharghar','navimumbai',410210)
insert into customer values(191059, 'kanishka','kharghar','navimumbai',410210)
```

```
insert into customer values(191060, 'dipak','kharghar','navimumbai',410210)
insert into customer values(191061, 'aditi','kharghar','navimumbai',410210)
insert into customer values(191062, 'rohan','kharghar','navimumbai',410210)
select * from customer
```

Dimension time

```
create table time(
 order_id int primary key,
 order_date int,
 year int,
 quater int,
 month int,
)
insert into time values(202101,25,2021,1,9)
insert into time values(202102,26,2021,1,10)
insert into time values(202103,24,2021,1,4)
insert into time values(202104,27,2021,1,7)
insert into time values(202105,29,2021,1,8)
insert into time values(202106,30,2021,1,10)
select * from time
```

Dimension Sales

```
create table sales(
       product_id int FOREIGN KEY REFERENCES product(product_id),
       order_id int FOREIGN KEY REFERENCES time(order_id),
       customer_id int FOREIGN KEY REFERENCES customer(customer_id),
       employee_id int FOREIGN KEY REFERENCES employee(employee_id),
       total int,
       quantity int,
       discount int
)
insert into sales values(1010,202101,191056,191041,20000,10,2000)
insert into sales values(1012,202102,191057,191042,20000,1,2000)
insert into sales values(1013,202103,191058,191043,50000,1,5000)
insert into sales values(1014,202104,191059,191044,75000,1,7500)
insert into sales values(1011,202105,191060,191045,35000,1,3500)
insert into sales values(1015,202106,191061,191046,22000,1,2200)
select * from sales

-- All
Select p.product_name,p.product_category,p.unit_price,t.order_id,c.customer_name,
e.employee_name,s.quantity,s.total,s.discount
from sales s
inner join product p on s.product_id=p.product_id
```

```
inner join time t on s.order_id=t.order_id
inner join employee e on s.employee_id=e.employee_id
inner join customer c on s.customer_id=c.customer_id
```

Experiment No.2

```
-- SLICE
Select p.product_name,p.product_category,p.unit_price,t.order_id,c.customer_name,
e.employee_name,s.quantity,s.total,s.discount
from sales s
inner join product p on s.product_id=p.product_id
inner join time t on s.order_id=t.order_id
inner join employee e on s.employee_id=e.employee_id
inner join customer c on s.customer_id=c.customer_id
where p.product_name = 'LG'
-- DICE
Select p.product_name,p.product_category,p.unit_price,t.order_id,c.customer_name,
e.employee_name,s.quantity,s.total,s.discount
from sales s
inner join product p on s.product_id=p.product_id
inner join time t on s.order_id=t.order_id
inner join employee e on s.employee_id=e.employee_id
inner join customer c on s.customer_id=c.customer_id
where p.product_name = 'LG' and quantity = 10
-- ROLLUP
Select product_name, SUM(s.total)
from sales s
inner join product p on s.product_id=p.product_id
inner join time t on s.order_id=t.order_id
inner join employee e on s.employee_id=e.employee_id
inner join customer c on s.customer_id=c.customer_id
where p.product_name = 'LG'
group by p.product_name
-- Drill Down
Select s.quantity, SUM(s.total) sum
from sales s
inner join product p on s.product_id=p.product_id
inner join time t on s.order_id=t.order_id
inner join employee e on s.employee_id=e.employee_id
inner join customer c on s.customer_id=c.customer_id
where p.product_name = 'LG'
group by s.quantity
```

Experiment No.3

BINNING Method-

```
import pandas as pd
import numpy as np
import math
unsort_data = [34, 8, 9, 15, 26, 24, 21, 25, 21, 28, 29, 4]
data = np.sort(unsort_data)
print(data)
bin1 = np.zeros((1, 4))
for i in range(0, 3, 4):
    k = int(1/4)
    mean = (data[i] + data[i+1]+data[i+2] + data[i+3])/4
    for j in range(4):
        bin1[k, j] = mean
print("\n----Mean Bin-1:----\n", bin1)

bin2 = np.zeros((1, 4))
for i in range(0, 3, 4):
    k = int(1/4)
    mean = (data[i+4]+data[i+5]+data[i+6] + data[i+7])/4
    for j in range(4):
        bin2[k, j] = mean
print("\n----Mean Bin-2:----\n", bin2)

bin3 = np.zeros((1, 4))
for i in range(0, 3, 4):
    k = int(1/4)
    mean = (data[i+8]+data[i+9]+data[i+10] + data[i+11])/4
    for j in range(4):
        bin3[k, j] = mean
print("\n----Mean Bin-3:----\n", bin3)

for i in range(0, 3, 4):
    k = int(i/4)
    for j in range(4):
        bin1[k, j] = data[i+2]
print("\n----Median Bin-1:----\n", bin1)

for i in range(0, 3, 4):
    k = int(1/4)
    for j in range(4):
        bin2[k, j] = data[i+6]
print("\n----Median Bin-2----\n", bin2)

for i in range(0, 3, 4):
    k = int(1/4)
    for j in range(4):
        bin3[k, j] = data[i+10]
print("\n----Median Bin-3----\n", bin3)

for i in range(0, 3, 4):
    k = int(1/4)
    for j in range(4):
```

```
        if (data[i+j]-data[i+1]) < (data[i+3]-data[i+j]):
            bin1[k, j] = data[i]
        else:
            bin1[k, j] = data[i+3]
print("\n----Boundary Bin-1----\n", bin1)

for i in range(0, 3, 4):
    k = int(i/4)
    for j in range(4):
        if (data[i+j]-data[i+1]) < (data[i+3]-data[i+j]):
            bin2[k, j] = data[i+4]
        else:
            bin2[k, j] = data[i+7]
print("\n----Boundary Bin-2----\n", bin2)

for i in range(0, 3, 4):
    k = int(1/4)
    for j in range(4):
        if (data[i+j]-data[i+1]) < (data[i+3]-data[i+j]):
            bin3[k, j] = data[i+8]
        else:
            bin3[k, j]-data[i+11]
print("\n----Boundary Bin-3:----\n", bin3)
```

OUTPUT:

[4 8 9 15 21 21 24 25 26 28 29 34]

----Mean Bin-1:----

[[9. 9. 9. 9.]]

----Mean Bin-2:----

[[22.75 22.75 22.75 22.75]]

----Mean Bin-3:----

[[29.25 29.25 29.25 29.25]]

----Median Bin-1:----

[[9. 9. 9. 9.]]

------Median Bin-2----

[[24. 24. 24. 24.]]

----Median Bin-3----

[[29. 29. 29. 29.]]

----Boundary Bin-1----

[[4. 4. 4. 15.]]

----Boundary Bin-2----

[[21. 21. 21. 25.]]

----Boundary Bin-3:----

[[26. 26. 26. 29.]]

Experiment No.4

```
# Naive Bayes Algorithm
import pandas as pd
import csv
def pci(data):
    class_count = [0, 0]
    for i in range(len(data)):
        if data.iloc[i, -1] == 'Yes':
            class_count[0] += 1
        else:
            class_count[1] += 1
            return class_count[0], class_count[1]
def pcix(data, x, c1, c2):
    p1, p2 = 1, 1
    count_yes = [0 for i in range(len(data.columns) - 1)]
    count_no = [0 for i in range(len(data.columns) - 1)]
    for i in range(len(data)):
        for j in range(len(data.columns) - 1):
            if data.iloc[i, j] == x[j]:
                if data.iloc[i, -1] == 'Yes':
                    count_yes[j] += 1
            else:
                count_no[j] += 1
    for i in range(len(count_yes)):
        p1 = p1 * count_yes[i] / c1
        p2 = p2 * count_no[i] / c2
    return p1, p2
data = pd.read_csv('NB.csv')
data = data.iloc[:, 1:]
X = input('Enter tuple to classify: ')
X = X.split(',')
c1, c2 = pci(data)
p1, p2 = pcix(data, X, c1, c2)
if (p1 * c1 / len(data)) > (p2 * c2 / len(data)):
    print('Buys')
else:
    print('No Buy')
```

OUTPUT:

Enter tuple to classify: Youth, Medium, Yes, Excellent

No buy

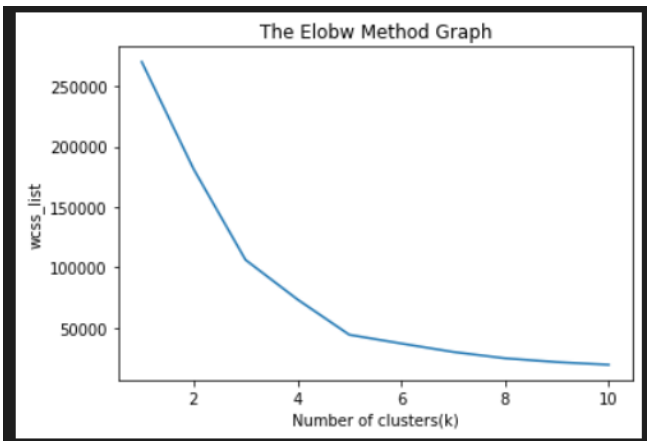
NB.csv file for naïve bayes

```
id,age,income,student,credit_rating,buys_computer
1,youth,low,no,fair,yes
2,youth,high,yes,excellent,no
3,youth,high,no,fair,yes
4,middle,medium,yes,fair,yes
5,senior,low,yes,fair,yes
6,senior,low,yes,excellent,yes
7,middle,high,yes,excellent,yes
8,youth,medium,no,fair,no
9,middle,low,no,excellent,yes
10,senior,medium,yes,fair,yes
```

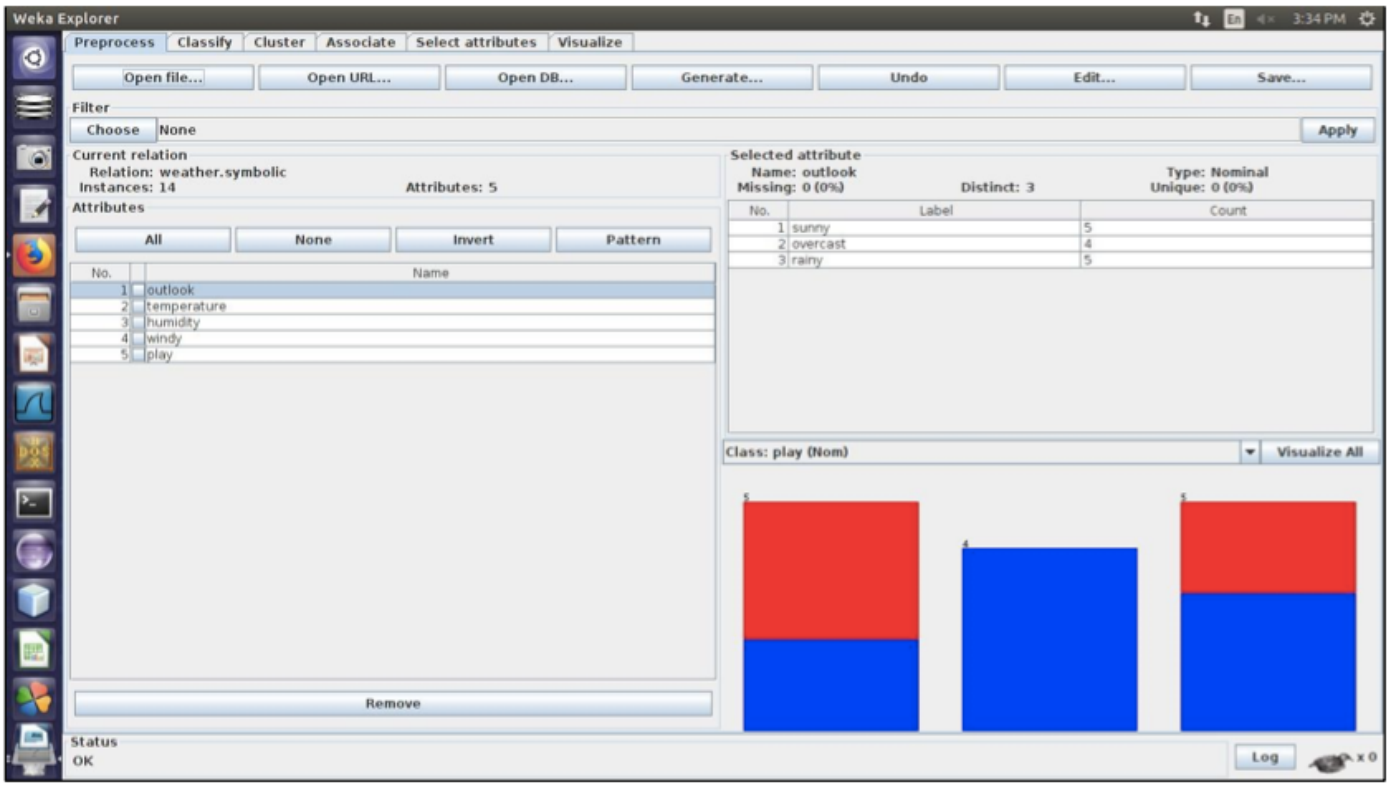
Experiment No. 5

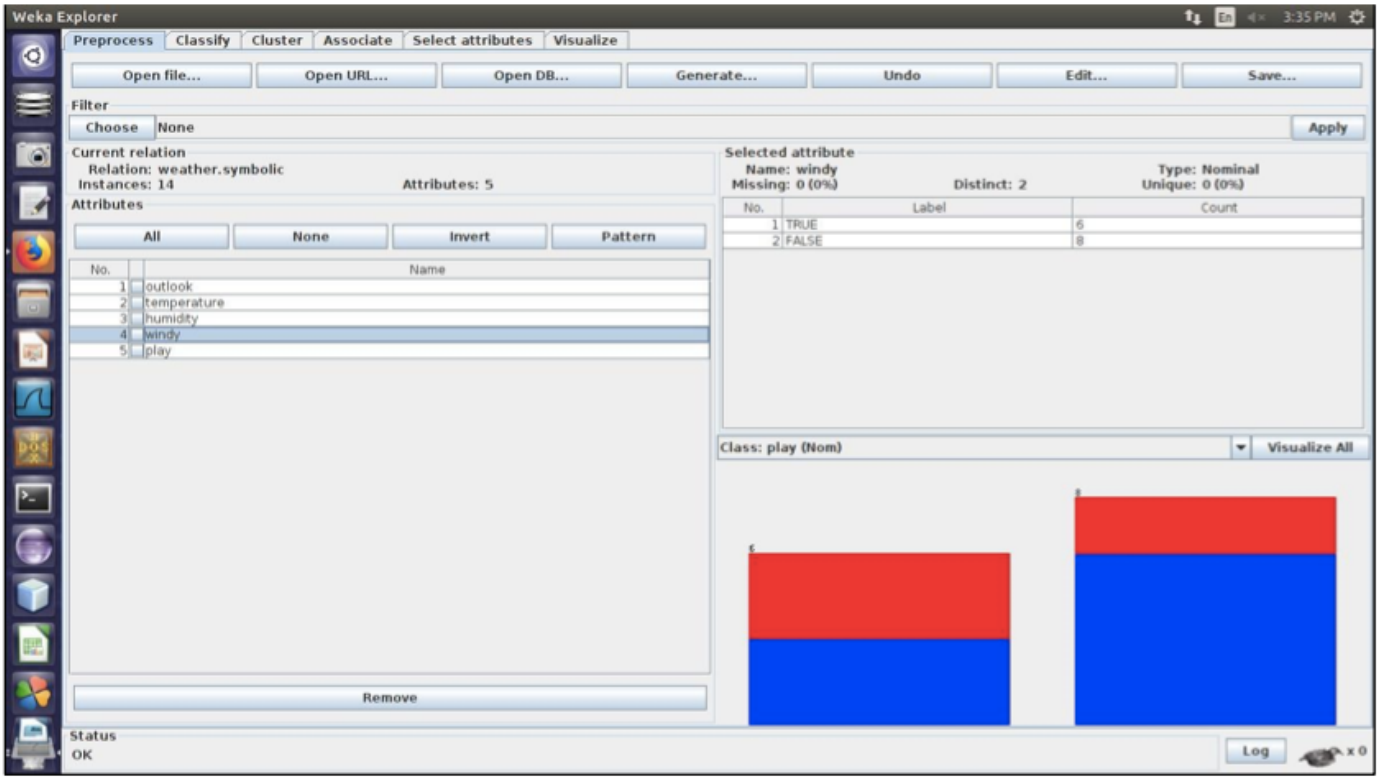
```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Mall_Customers_data.csv')

x = dataset.iloc[:, [3, 4]].values

#finding optimal number of clusters using the elbow method
from sklearn.cluster import KMeans
wcss_list= []  #Initializing the list for the values of WCSS

#Using for loop for iterations from 1 to 10.
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
mtp.plot(range(1, 11), wcss_list)
mtp.title('The Elobw Method Graph')
mtp.xlabel('Number of clusters(k)')
mtp.ylabel('wcss_list')
mtp.show()

#training the K-means model on a dataset
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)

#visulaizing the clusters
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster
1') #for first cluster
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster
2') #for second cluster
mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3')
#for third cluster
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster
4') #for fourth cluster
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster
5') #for fifth cluster
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow',
label = 'Centroid')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```

OUTPUT:

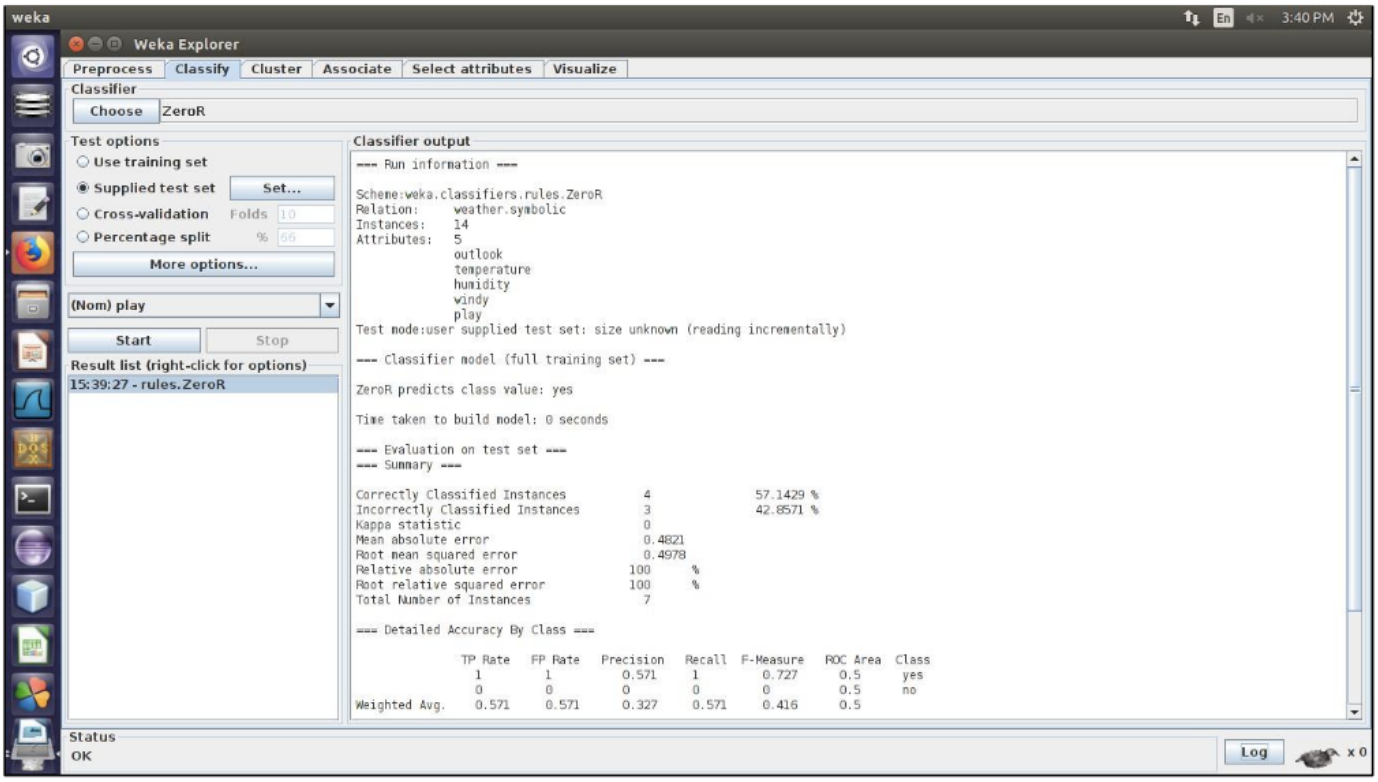


Experiment No.6:

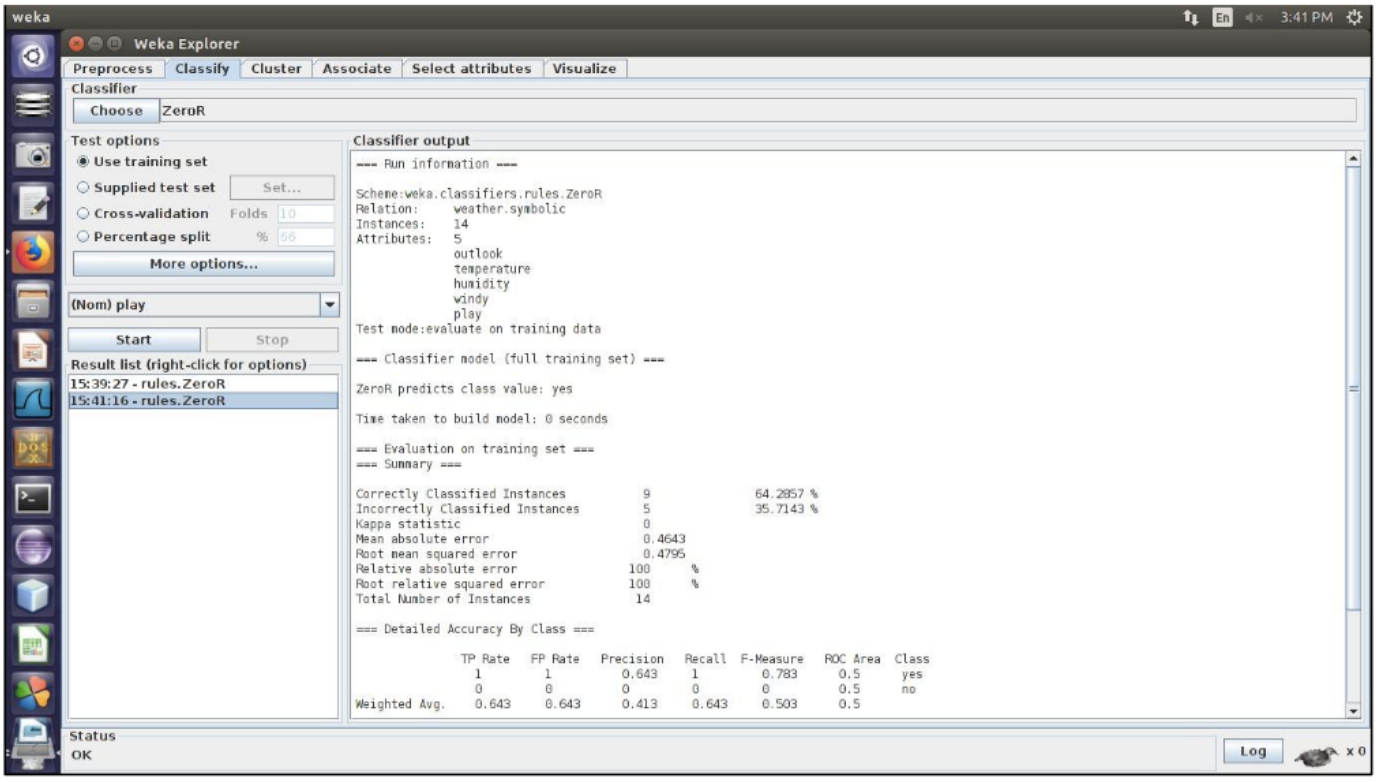



Weka Explorer
3:35 PM
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Current relation
Relation: weather.symbolic
Instances: 14
Attributes: 5
Attributes
All
None
Invert
Pattern
No.
Name
1 outlook
2 temperature
3 humidity
4 windy
5 play
Remove
Selected attribute
Name: windy
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)
No.
Label
Count
1 TRUE 6
2 FALSE 8
Class: play (Nom)
Visualize All
Status
OK
Log
x 0

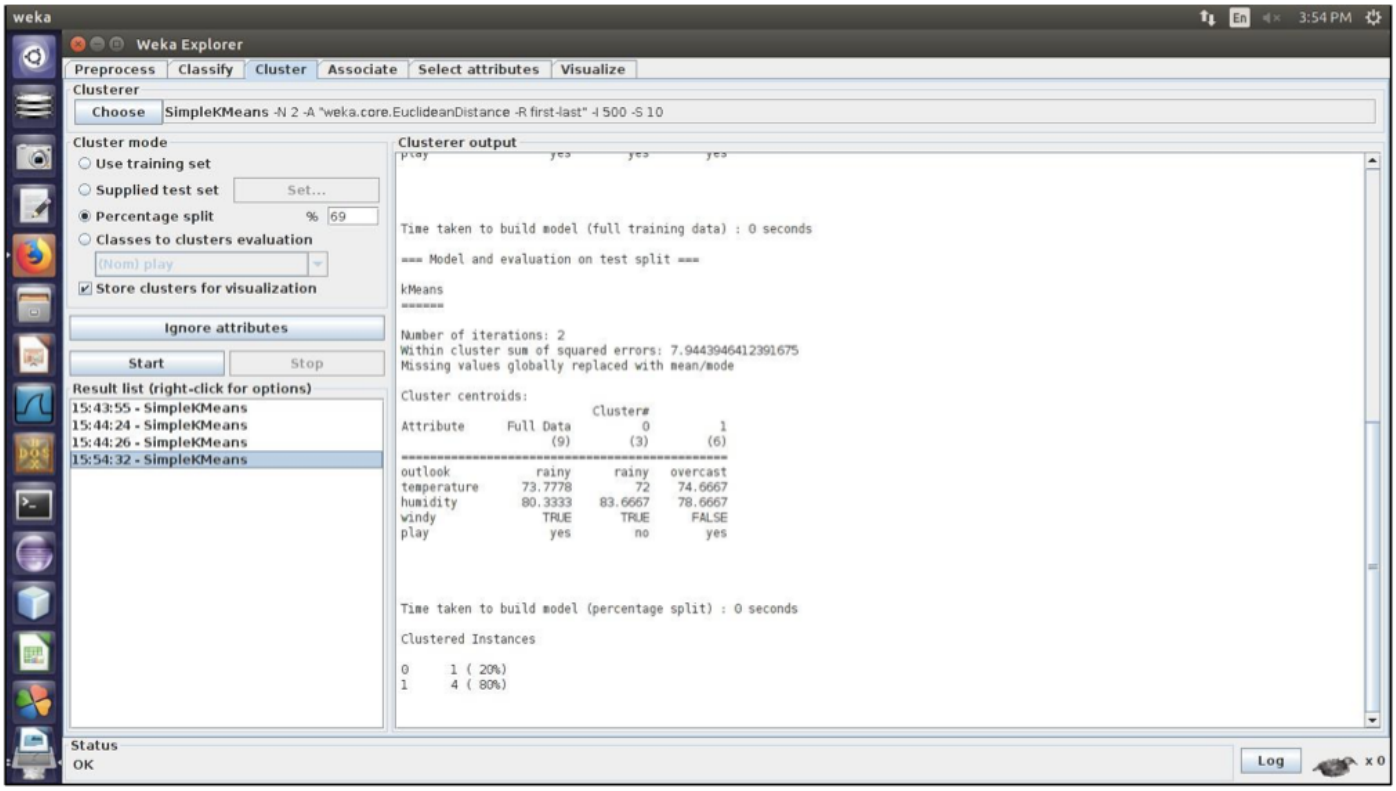
Classify: Supplies test



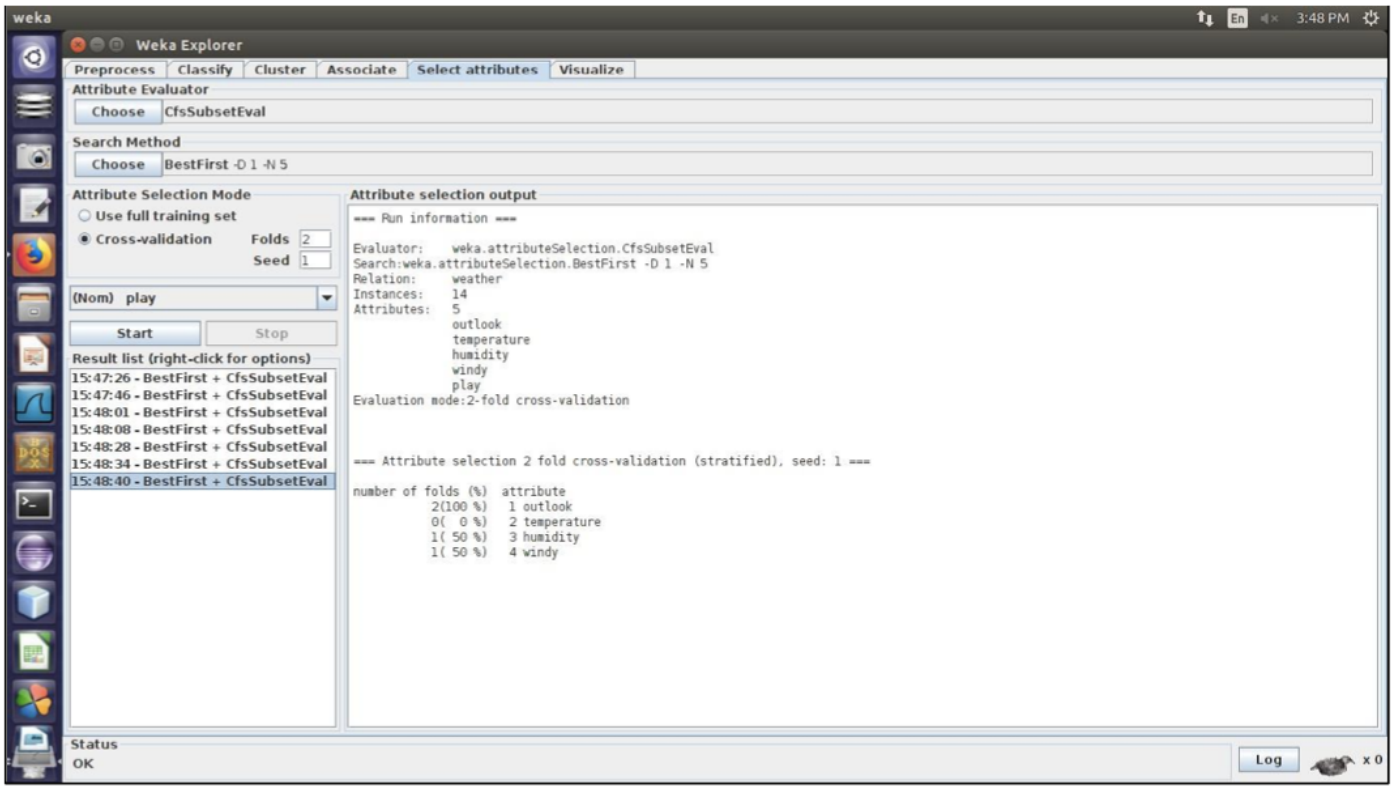
Use training set



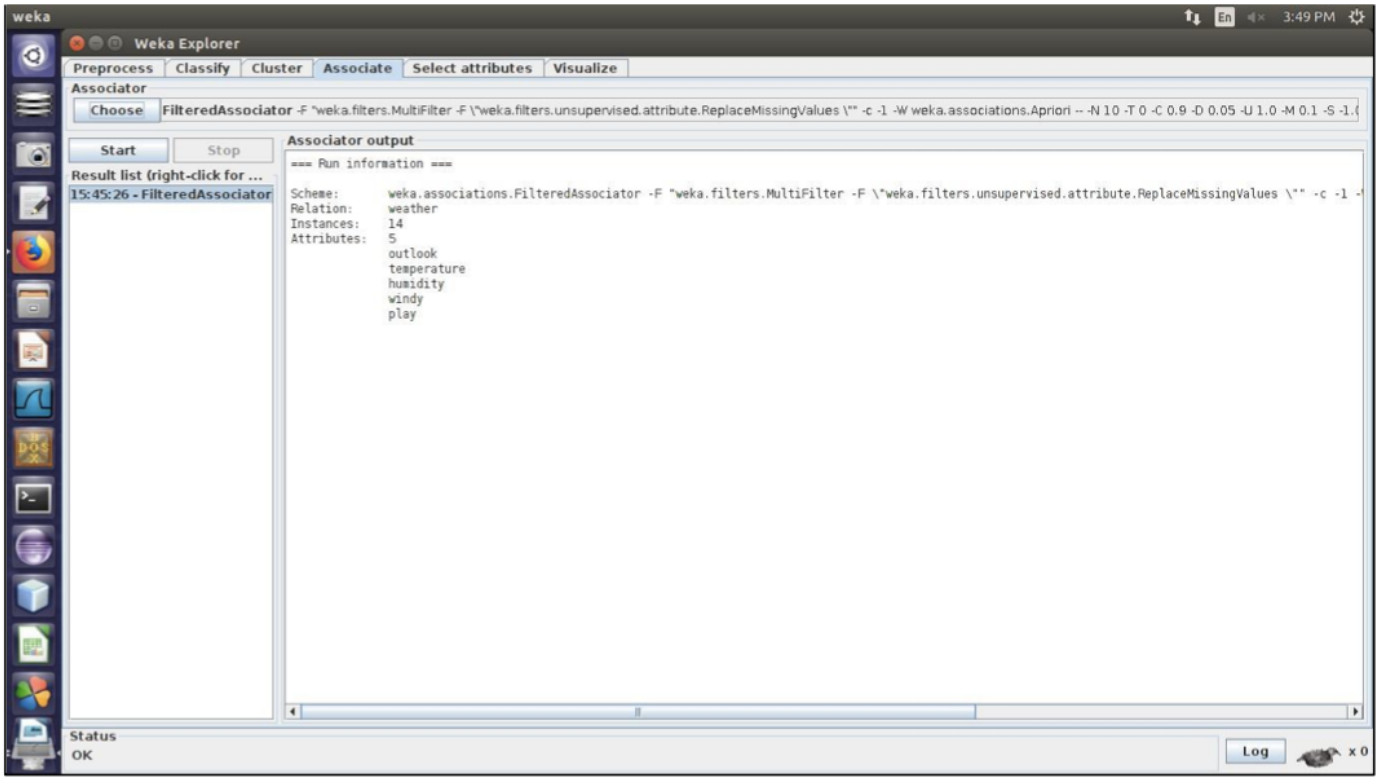
Cluster kmean: using numerical file



Select Attributes



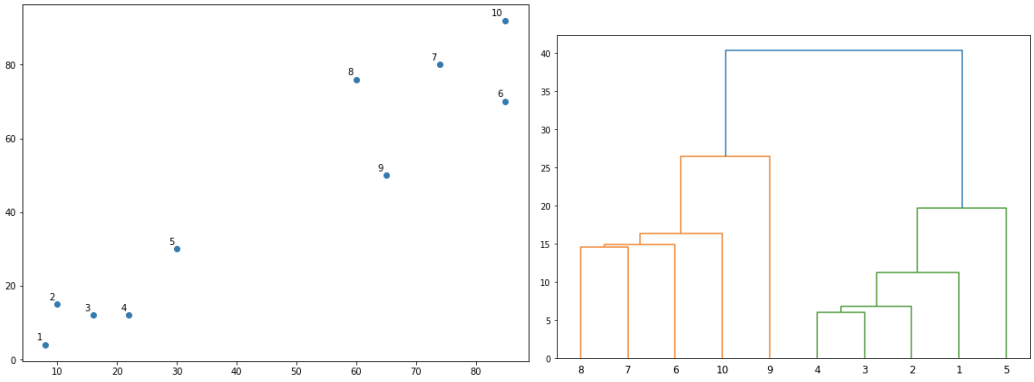
Associate:

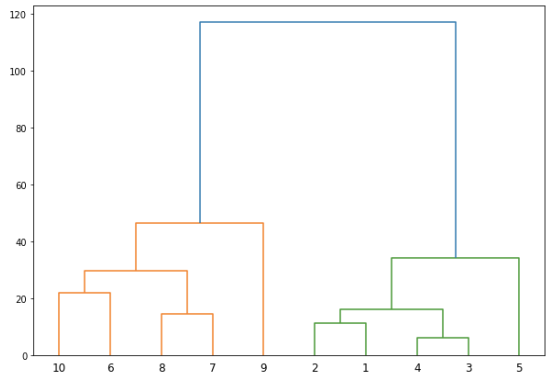

weka
3:49 PM
Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
FilteredAssociator -F "weka.filters.MultiFilter -F \"weka.filters.unsupervised.attribute.ReplaceMissingValues \"" -c -1 -W weka.associations.Apriori -- -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.(
Start
Stop
Result list (right-click for ...
15:45:26 - FilteredAssociator
Associator output
=== Run information ===
Scheme: weka.associations.FilteredAssociator -F "weka.filters.MultiFilter -F \"weka.filters.unsupervised.attribute.ReplaceMissingValues \"" -c -1 -
Relation: weather
Instances: 14
Attributes: 5
outlook
temperature
humidity
windy
play
Status
OK
Log
x 0

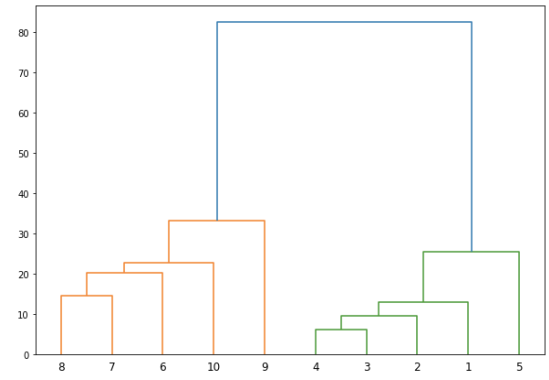
Experiment No.7:

```
import numpy as np
X = np.array([[8,4],[10,15],[16,12],[22,12],[30,30],[85,70],[74,80],[60,76],[65,50],[85,92]])
import matplotlib.pyplot as plt
labels = range(1,11)
plt.figure(figsize=(10,7))
plt.subplots_adjust(bottom=0.1)
plt.scatter(X[:,0],X[:,1],label='True Position')
for label,x,y in zip(labels,X[:,0],X[:,1]):
    plt.annotate(label,xy=(x,y),xytext=(-3,3),textcoords='offset points',ha='right',va='bottom')
plt.show()

# Single Linkage
from scipy.cluster.hierarchy import dendrogram,linkage
from matplotlib import pyplot as plt
linked = linkage(X,'single')
labelList = range(1,11)
plt.figure(figsize=(10,7))
dendrogram(linked,orientation='top',labels=labelList,
distance_sort='descending',show_leaf_counts=True)
plt.show()

#Complete Linkage
from scipy.cluster.hierarchy import dendrogram,linkage
from matplotlib import pyplot as plt
linked = linkage(X,'complete')
labelList = range(1,11)
plt.figure(figsize=(10,7))
dendrogram(linked,orientation='top',labels=labelList,
distance_sort='descending',show_leaf_counts=True)
plt.show()

#Average Linkage
from scipy.cluster.hierarchy import dendrogram,linkage
from matplotlib import pyplot as plt
linked = linkage(X,'average')
labelList = range(1,11)
plt.figure(figsize=(10,7))
dendrogram(linked,orientation='top',labels=labelList,
distance_sort='descending',show_leaf_counts=True)
plt.show()
```

OUTPUT:







Experiment No.8

```
data = [
        ['T100',['I1','I2','I5']],
        ['T200',['I2','I4']],
        ['T300',['I2','I3']],
        ['T400',['I1','I2','I4']],
        ['T500',['I1','I3']],
        ['T600',['I2','I3']],
        ['T700',['I1','I3']],
        ['T800',['I1','I2','I3','I5']],
        ['T900',['I1','I2','I3']]
        ]

init = []
for i in data:
    for q in i[1]:
        if(q not in init):
            init.append(q)
init = sorted(init)
print(init)
sp = 0.4
s = int(sp*len(init))
s
from collections import Counter
c = Counter()
for i in init:
    for d in data:
        if(i in d[1]):
            c[i]+=1
print("C1:")
for i in c:
    print(str([i])+": "+str(c[i]))
print()
l = Counter()
for i in c:
    if(c[i] >= s):
        l[frozenset([i])]+=c[i]
print("L1:")
for i in l:
    print(str(list(i))+": "+str(l[i]))
print()
pl = l
```

```
pos = 1
for count in range (2,1000):
    nc = set()
    temp = list(l)
    for i in range(0,len(temp)):
        for j in range(i+1,len(temp)):
            t = temp[i].union(temp[j])
            if(len(t) == count):
                nc.add(temp[i].union(temp[j]))
    nc = list(nc)
    c = Counter()
    for i in nc:
        c[i] = 0
        for q in data:
            temp = set(q[1])
            if(i.issubset(temp)):
                c[i]+=1
    print("C"+str(count)+":")
    for i in c:
        print(str(list(i))+": "+str(c[i]))
    print()
    l = Counter()
    for i in c:
        if(c[i] >= s):
            l[i]+=c[i]
    print("L"+str(count)+":")
    for i in l:
        print(str(list(i))+": "+str(l[i]))
    print()
    if(len(l) == 0):
        break
    pl = l
    pos = count
print("Result: ")
print("L"+str(pos)+":")
for i in pl:
    print(str(list(i))+": "+str(pl[i]))
print()
```

OUTPUT:

```
['I1', 'I2', 'I3', 'I4', 'I5']
C1:
['I1']: 6
['I2']: 7
['I3']: 6
['I4']: 2
['I5']: 2

L1:
['I1']: 6
['I2']: 7
['I3']: 6
['I4']: 2
['I5']: 2
```

```
C2:
['I1', 'I3']: 4
['I2', 'I4']: 2
['I4', 'I5']: 0
['I1', 'I4']: 1
['I3', 'I5']: 1
['I1', 'I5']: 2
['I1', 'I2']: 4
['I3', 'I4']: 0
['I2', 'I3']: 4
['I2', 'I5']: 2

L2:
['I1', 'I3']: 4
['I2', 'I4']: 2
['I1', 'I5']: 2
['I1', 'I2']: 4
['I2', 'I3']: 4
['I2', 'I5']: 2

C3:
['I2', 'I4', 'I5']: 0
['I1', 'I3', 'I2']: 2
['I1', 'I3', 'I5']: 1
['I2', 'I3', 'I4']: 0
['I1', 'I2', 'I5']: 2
['I2', 'I3', 'I5']: 1
['I2', 'I1', 'I4']: 1

L3:
['I1', 'I3', 'I2']: 2
['I1', 'I2', 'I5']: 2

C4:
['I5', 'I1', 'I3', 'I2']: 1

L4:

Result:
L3:
['I1', 'I3', 'I2']: 2
['I1', 'I2', 'I5']: 2
```

Experiment No.9

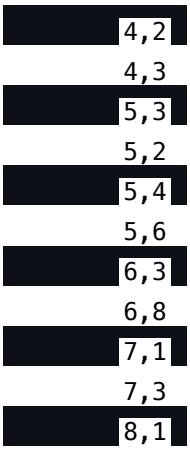
```
import networkx as nx
import numpy as np
from numpy import array
import matplotlib.pyplot as plt

with open('HITS.txt') as f:
    lines = f.readlines()

G = nx.DiGraph()
for line in lines:
    t = tuple(line.strip().split(','))
    G.add_edge(*t)

h, a = nx.hits(G, max_iter=100)
h = dict(sorted(h.items(), key=lambda x: x[0]))
a = dict(sorted(a.items(), key=lambda x: x[0]))

print(np.round(list(a.values()), 3))
print(np.round(list(h.values()), 3))

pr = nx.pagerank(G)
pr = dict(sorted(pr.items(), key=lambda x: x[0]))
print(np.round(list(pr.values()), 3))


sim = nx.simrank_similarity(G)
lol = [[sim[u][v] for v in sorted(sim[u])] for u in sorted(sim)]
sim_array = np.round(array(lol), 3)
print(sim_array)

nx.draw(G, with_labels=True, node_size=2000, edge_color='#eb4034', width=3, font_size=16,
font_weight=500, arrowsize=20, alpha=0.8)
plt.savefig("graph.png")
```

OUTPUT:

```
[0.088 0.187 0.369 0.128 0.059 0.11  0.    0.059]
[0.043 0.144 0.03  0.187 0.268 0.144 0.154 0.03 ]
[0.241 0.137 0.218 0.24  0.077 0.035 0.019 0.034]
[[1.    0.207 0.221 0.193 0.217 0.269 0.    0.171]
 [0.207 1.    0.355 0.369 0.302 0.553 0.    0.369]
 [0.221 0.355 1.    0.242 0.4   0.324 0.    0.427]
 [0.193 0.369 0.242 1.    0.229 0.548 0.    0.243]
 [0.217 0.302 0.4   0.229 1.    0.271 0.    0.498]
 [0.269 0.553 0.324 0.548 0.271 1.    0.    0.244]
 [0.    0.    0.    0.    0.    0.    1.    0.   ]
 [0.171 0.369 0.427 0.243 0.498 0.244 0.    1.   ]]
```

//HITS.txt file

```
1,4
2,3
2,5
3,1
```


4,2
4,3
5,3
5,2
5,4
5,6
6,3
6,8
7,1
7,3
8,1

Experiment NO.10

```
import networkx as nx
import matplotlib.pyplot as plt
G = nx.DiGraph()
G.add_edges_from([('A', 'D'), ('B', 'C'), ('B', 'E'), ('C', 'A'),
('D', 'C'), ('E', 'D'), ('E', 'B'), ('E', 'F'),
('E', 'C'), ('F', 'C'), ('F', 'H'), ('G', 'A'), ('G', 'C'), ('H', 'A')])
plt.figure(figsize =(10, 10))
nx.draw_networkx(G, with_labels = True)
hubs,authorities = nx.hits(G, max_iter = 50, normalized = True)
print("Hub Scores:", hubs)
print("Authority Scores:", authorities)
```

OUTPUT:

```
Hub Scores: {'A': 0.04642540403219994, 'D': 0.13366037526115382, 'B': 0.15763599442967322, 'C':
0.03738913224642654, 'E': 0.25881445984686646, 'F': 0.1576359944296732, 'H':
0.03738913224642654, 'G': 0.17104950750758036} Authority Scores: {'A': 0.10864044011724346, 'D':
0.13489685434358, 'B': 0.11437974073336447, 'C': 0.3883728003876181, 'E': 0.06966521184241478,
'F': 0.11437974073336447, 'H': 0.06966521184241475, 'G': 0.0}
```