Department of Computer Science

Computer Journal

Completion of experimental work

CERTIFICATE

This is to certify that

Name: Meenaz Riyaz Momin

Roll No.: 9008 Seat No:

Subject: CS358 Lab Course III Data Analytics

Class: T.Y.B.SC. (COMPUTERSCIENCE) SEMESTER-VI

Division: A

For the Academic year 2021-22.

The same has been examined by the Lecture in-charge and head of the computer Science department.

Lecture in Charge: ----------

Lecture in Charge: ----------

Internal Examiner: ---------- Head

External Examiner: ---------- Department of Computer Science

Assignment 1.

Q1. Create 'sales' Data set having 5 columns namely: ID, TV, Radio, Newspaper and Sales.(random 500 entries) Build a linear regression model by identifying independent and target variable. Split the variables into training and testing sets. then divide the training and testing sets into a 7:3 ratio, respectively and print them. Build a simple linear regression model.

In [33]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import
train_test_split
from sklearn.linear_model import
LogisticRegression
from sklearn.metrics import r2_score,
mean_squared_error
%matplotlib inline
```

In [34]:

```
df=pd.DataFrame(np.random.randint(0,1000
,size=(500,5)),columns=['ID','TV','Radio'
df
```

Out[34]:

	ID	TV	Radio	Newspaper	Sale
0	802	570	796	960	91
1	817	629	318	352	15
2	397	360	538	103	35
3	25	512	928	544	24
4	454	833	460	481	62
...	...	...	...	...	.
495	133	956	99	432	51
496	963	793	77	761	59
497	180	284	499	519	98
498	613	554	987	180	97
499	364	228	231	622	15

500 rows × 5 columns

```
df.sample(5)
```

Out[35]:

	ID	TV	Radio	Newspaper	Sale
232	74	740	245	57	23
267	412	176	692	996	10
96	245	732	514	582	66
151	520	24	359	115	8
449	783	191	518	782	79

In [29]:

```
new=df[['Sales', 'TV']]
new
```

Out[29]:

	Sales	TV
0	817	425
1	19	125
2	880	930

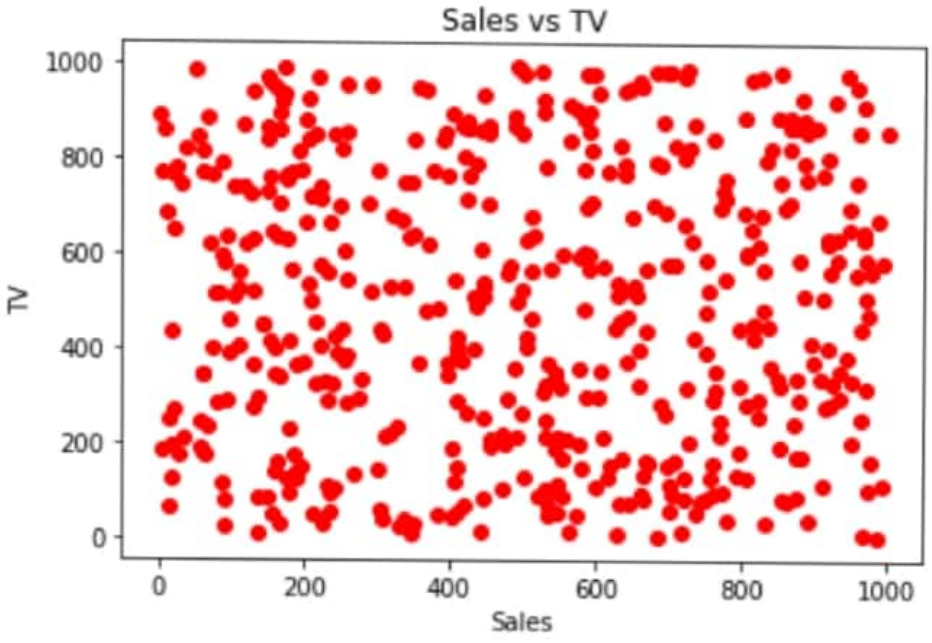
In [36]:

```
x=np.array(new[['Sales']])
y=np.array(new[['TV']])
print(x.shape)
print(y.shape)
```

```
(500, 1)
(500, 1)
```

In [32]:

```
plt.scatter(x, y, color="red")
plt.title('Sales vs TV')
plt.xlabel('Sales')
plt.ylabel('TV')
plt.show()
```



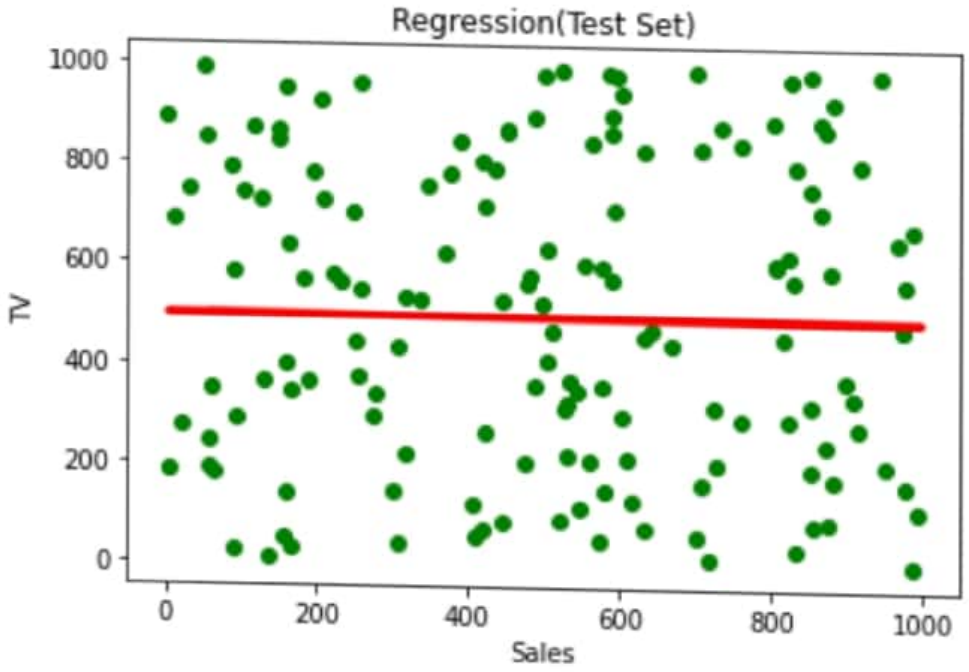
In [51]:

```
x_train,x_test,y_train,y_test=train_
test_split(x,y, test_size=0.3,
           random_state=15)
regressor = LinearRegression() # Creating
regressor.fit(x_train,y_train)
print (x_train)
```

```
[816]
[823]
[925]
[  4]
[177]
[421]
[562]
[325]
[957]
[658]
[847]
[889]
[726]
[167]
[455]
[704]
[397]
[166]
[365]
[865]
```

In [45]:

```
plt.scatter(x_test,y_test,color="green")
plt.plot(x_train,regressor.predict
(x_train),color="red",linewidth=3) # Regr
plt.title('Regression(Test Set)')
plt.xlabel('Sales')
plt.ylabel('TV')
plt.show()
```

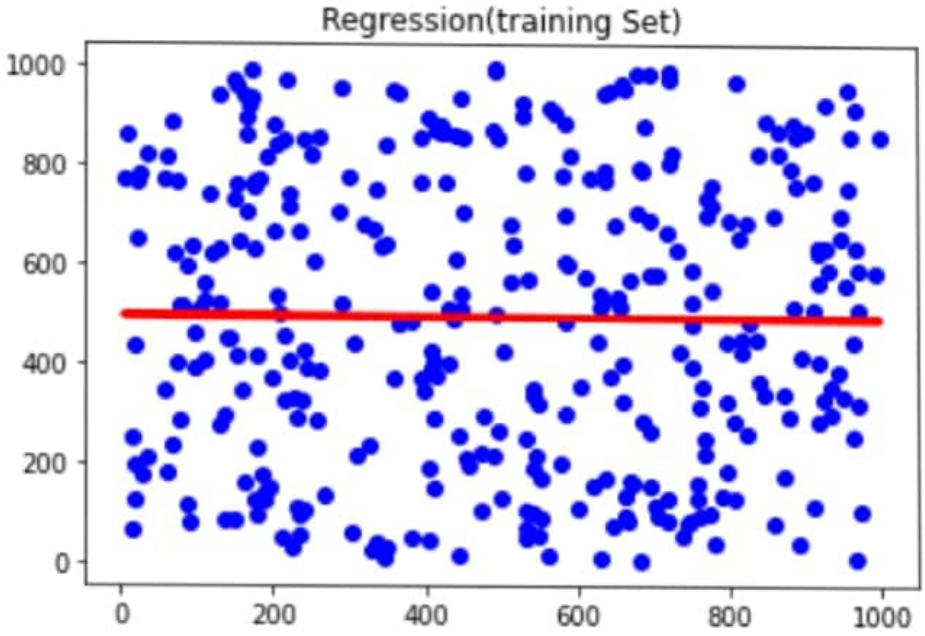


In [46]:

```
plt.scatter(x_train,y_train,color="blue")
plt.plot(x_train,regressor.predict(x_trai
plt.title('Regression(training Set)')
```

Out[46]:

```
Text(0.5, 1.0, 'Regression(t
raining Set)')
```



Q2. Create 'realestate' Data set having 4 columns namely: ID,flat, houses and purchases (random 500 entries). Build a linear regression model by identifying independent and target variable. Split the variables into training and testing sets and print them. Build a simple linear regression model for predicting purchases.

In [47]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train
from sklearn.linear_model import Logistic
from sklearn.metrics import r2_score,mean
```

```
from sklearn.linear_model import Logistic
from sklearn.metrics import r2_score,mean
%matplotlib inline
```

In [50]:

```
df=pd.DataFrame(np.random.randint(0,1000,
df #realstate data
```

Out[50]:

	ID	flat	house	purches
0	286	102	504	515
1	797	979	249	962
2	873	932	216	285
3	430	9	764	761
4	310	132	44	865
...	...	...	...	...
495	250	154	137	484
496	817	962	120	342
497	838	380	363	544
498	674	444	691	319

```
new=df[['house', 'purches']]
new
```

Out[52]:

	house	purches
0	504	515
1	249	962
2	216	285
3	764	761
4	44	865
...	...	...
495	137	484
496	120	342
497	363	544
498	691	319
499	334	899

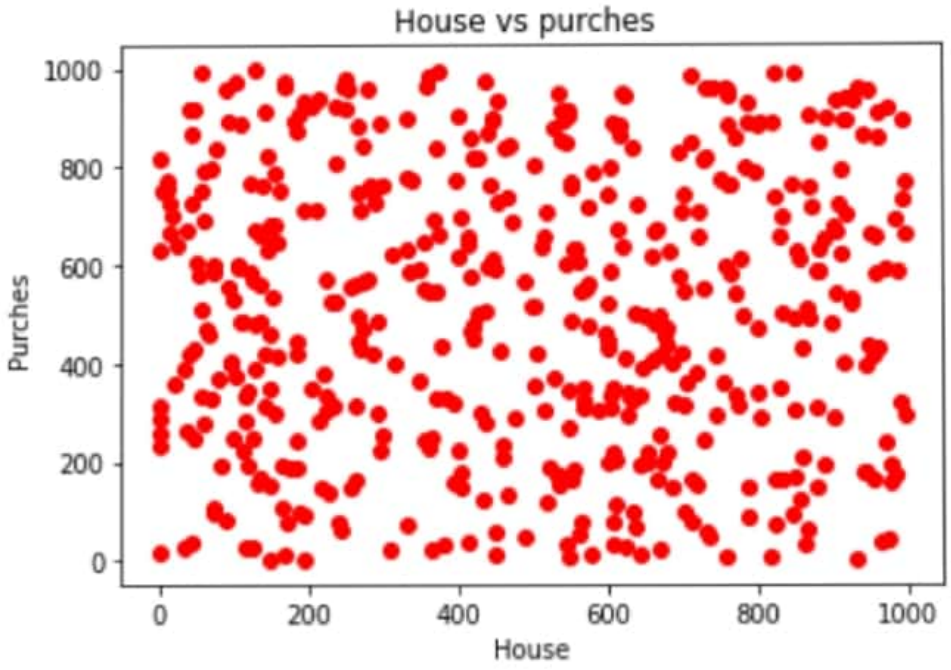
500 rows × 2 columns

```
x=np.array(new[['house']])
y=np.array(new[['purches']])
print(x.shape)
print(y.shape)
```

```
(500, 1)
(500, 1)
```

In [56]:

```
plt.scatter(x, y, color="red")
plt.title('House vs purches ')
plt.xlabel('House')
plt.ylabel('Purches')
plt.show()
```



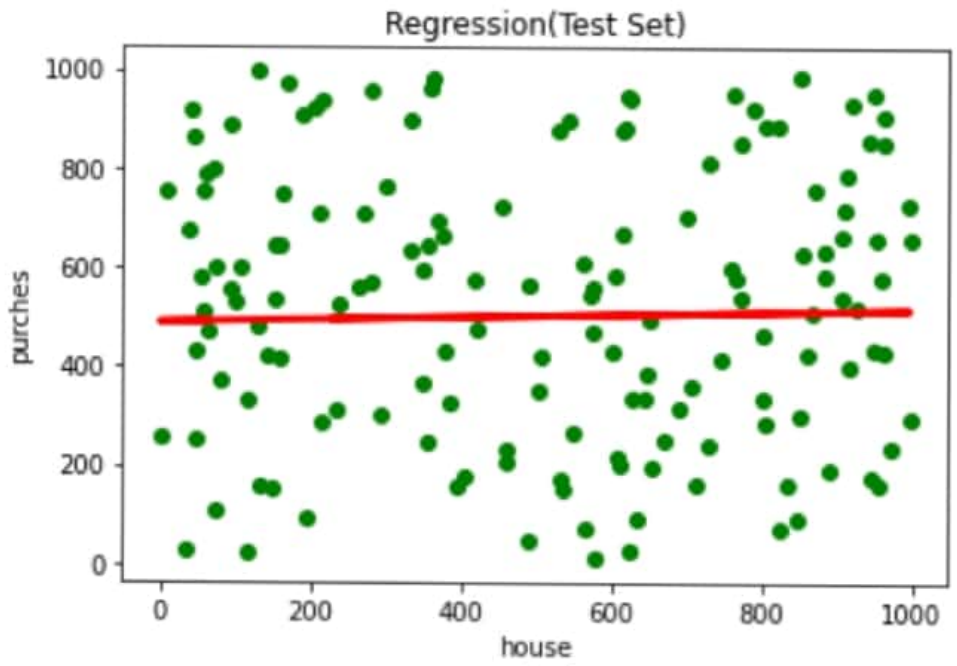
```
x_train,x_test,y_train,y_test=train_test_
regressor = LinearRegression() # Creating
regressor.fit(x_train,y_train)
print (x_train)
```

```
[419]
[291]
[148]
[  0]
[633]
[563]
[195]
[918]
[724]
[131]
[601]
[560]
[714]
[937]
[363]
[137]
[196]
[552]
[185]
```

In [58]:

```
plt.scatter(x_test,y_test,color="green")
plt.plot(x_train,regressor.predict(x_trai
plt.title('Regression(Test Set)')
```

```
plt.scatter(x_test,y_test,color="green")
plt.plot(x_train,regressor.predict(x_trai
plt.title('Regression(Test Set)')
plt.xlabel('house')
plt.ylabel('purches')
plt.show()
```



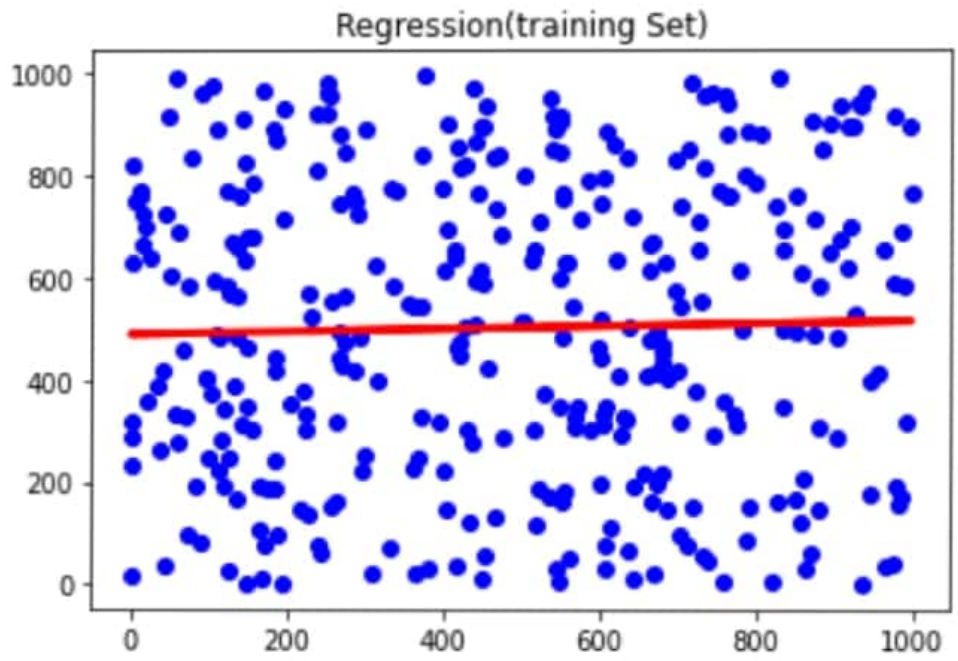
In [59]:

```
plt.scatter(x_train,y_train,color="blue")
plt.plot(x_train,regressor.predict(x_trai
plt.title('Regression(training Set)')
```

Out[59]:

```
Text(0.5, 1.0, 'Regression(t
```

```
raining Set)')
```



In [62]:

```
y_pred = regressor.predict(x_test)
print('R2 score: %.2f' % r2_score(y_test,
print('Mean Error :',mean_squared_error(y
```

```
R2 score: -0.01
Mean Error : 75457.361960543
19
```

```
#def purches(hp): # A function to predict
    #result = regressor.predict(np.array(
    #return(result[0,0])
#input = int(input('Enter price of your h
#print('The purches will be : ',int(purch
```

In above example the r2 mean value is negative that means the data is not related to each other and the linear regression line is not best fit

Q3. Create 'User' Data set having 5 columns namely: User ID, Gender, Age, EstimatedSalary and Purchased. Build a logistic regression model that can predict whether on the given parameter a person will buy a car or not.

In [ ]:

```
!pip install seaborn
```

```
Collecting seaborn
  WARNING: Retrying (Retry(t
otal=4, connect=None, read=N
one, redirect=None, status=N
one)) after connection broke
```

In [12]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train
from sklearn.linear_model import Logistic
from sklearn.metrics import r2_score,mean
%matplotlib inline
import seaborn as sn
from sklearn import metrics
```

In [13]:

```
df=pd.read_excel("User.xlsx")
df
```

Out[13]:

	User ID	Gender	Age	SalaryEstimate	I
0	101	female	20	25000	
1	102	male	22	32000	
2	103	male	25	50000	
3	104	male	19	45000	
4	105	female	21	28000	

In [14]:

```
x = df.iloc[:, [2,3]].values
y = df.iloc[:, 4].values
```

In [15]:

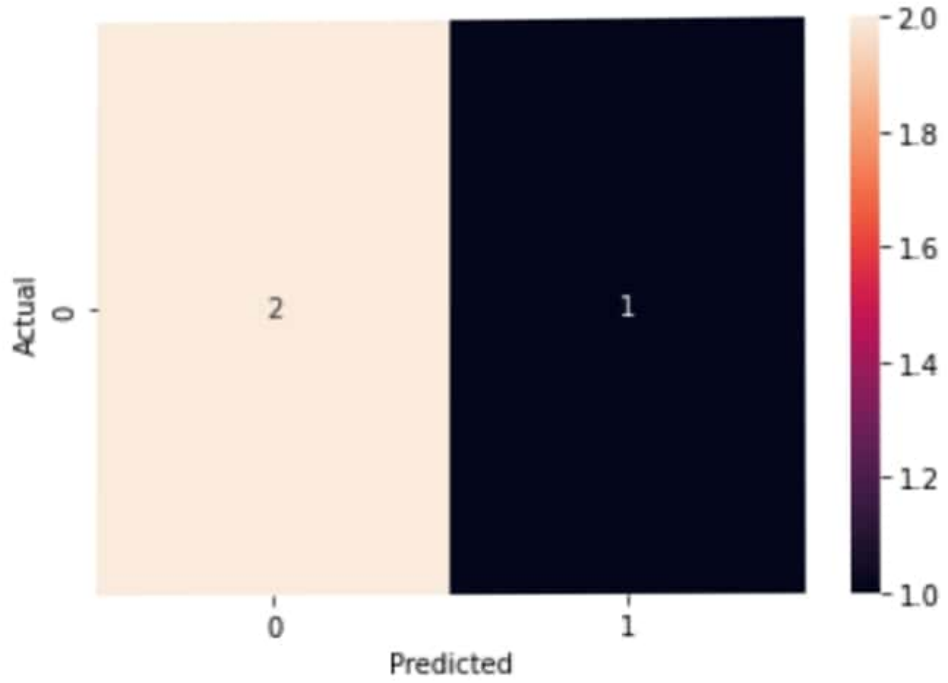
```
x_train, x_test, y_train, y_test = train_

logistic_regression= LogisticRegression()
logistic_regression.fit(x_train,y_train)
y_pred=logistic_regression.predict(x_test
```

In [16]:

```
confusion_matrix = pd.crosstab(y_test, y_
sn.heatmap(confusion_matrix, annot=True)
print('Accuracy: ',metrics.accuracy_score
plt.show()
```

```
Accuracy:  0.6666666666666666
```



In [17]:

```
print (x_test)
print (y_pred)
```

```
[[   25 50000]
 [   24 20000]
 [   21 28000]]
```

Assignment 2

Set A

Q1. Create the following dataset in python Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules. Repeat the process with different min_sup values.

In [7]:

```
import pandas as pd
from mlxtend.frequent_patterns import
apriori,association_rules
from mlxtend.preprocessing import
TransactionEncoder
```

In [9]:

```
transaction=[['Bread', 'Milk'],
     ['Bread', 'Diaper', 'Beer','Eggs'],
     ['Milk', 'Diaper', 'Beer','Coke'],
     ['Bread', 'Milk', 'Diaper', 'Beer'],
            ['Bread', 'Milk', 'Diaper', '
```

In [10]:

```
te=TransactionEncoder()
te_array=te.fit(transaction).
transform(transaction)
df=pd.DataFrame(te_array, columns=
```

```
transform(transaction)
df=pd.DataFrame(te_array, columns=
                te.columns_)
df
```

Out[10]:

	Beer	Bread	Coke	Diaper	Eggs	
0	False	True	False	False	False	
1	True	True	False	True	True	F
2	True	False	True	True	False	
3	True	True	False	True	False	
4	False	True	True	True	False	

In [11]:

```
freq_items=apriori(df, min_support=0.5,
                   use_colnames=True)
freq_items
```

Out[11]:

	support	itemsets
0	0.6	(Beer)
1	0.8	(Bread)
2	0.8	(Diaper)
3	0.8	(Milk)
4	0.6	(Diaper, Beer)
5	0.6	(Diaper, Bread)
6	0.6	(Milk, Bread)
7	0.6	(Diaper, Milk)

In [12]:

```
rules = association_rules(freq_items,
metric ='support', min_threshold=0.05)
rules = rules.sort_values(['support',
'confidence'], ascending =[False,False])
print(rules)
```

```
  antecedents consequents  a
ntecedent support  consequen
t support  support  \
1      (Beer)    (Diaper)
0.6                   0.8
0.6
0    (Diaper)      (Beer)
0.8                   0.6
0.6
```

```
2    (Diaper)     (Bread)
0.8                 0.8
0.6
3     (Bread)    (Diaper)
0.8                 0.8
0.6
4      (Milk)     (Bread)
0.8                 0.8
0.6
5     (Bread)      (Milk)
0.8                 0.8
0.6
6    (Diaper)      (Milk)
0.8                 0.8
0.6
7      (Milk)    (Diaper)
0.8                 0.8
0.6

   confidence    lift  lever
age  conviction
1         1.00  1.2500
0.12         inf
0         0.75  1.2500
0.12          1.6
2         0.75  0.9375     -
0.04          0.8
3         0.75  0.9375     -
0.04          0.8
4         0.75  0.9375     -
0.04          0.8
5         0.75  0.9375     -
0.04          0.8
```

```
0          0.75  1.2500
0.12          1.6
2          0.75  0.9375     -
0.04          0.8
3          0.75  0.9375     -
0.04          0.8
4          0.75  0.9375     -
0.04          0.8
5          0.75  0.9375     -
0.04          0.8
6          0.75  0.9375     -
0.04          0.8
7          0.75  0.9375     -
0.04          0.8
```

Q2.Create your own transactions dataset and apply the above process on your dataset.

In [15]:

```
import pandas as pd
from mlxtend.frequent_patterns import
apriori, association_rules
from mlxtend.preprocessing import
TransactionEncoder
```

In [16]:

```
transaction=[['Black', 'Red', 'White'],
             ['White', 'Red', 'Blue','Gree
             ['Black', 'Blue','Green'],
             ['White', 'Black'],
```

In [17]:

```
te=TransactionEncoder()
te_array=te.fit(transaction).transform(tr
df=pd.DataFrame(te_array, columns=te.colu
df
```

Out[17]:

	Black	Blue	Green	Red	White
0	True	False	False	True	True
1	False	True	True	True	True
2	True	True	True	False	False
3	True	False	False	False	True
4	False	True	True	True	True

In [18]:

```
freq_items=apriori(df, min_support=0.5, u
freq_items
```

Out[18]:

	support	itemsets
0	0.6	(Black)
1	0.6	(Blue)
2	0.6	(Green)
3	0.6	(Red)
4	0.8	(White)
5	0.6	(Green, Blue)
6	0.6	(Red, White)

In [19]:

```
rules = association_rules(freq_items, met
)
rules = rules.sort_values(['support', 'co
print(rules)
```

```
  antecedents consequents  a
ntecedent support  consequen
t support  support  \
0     (Green)      (Blue)
0.6                 0.6
0.6
1      (Blue)     (Green)
0.6                 0.6
```

```
0.6
3     (White)        (Red)
0.8                  0.6
0.6

   confidence      lift  lev
erage  conviction
0         1.00  1.666667
0.24          inf
1         1.00  1.666667
0.24          inf
2         1.00  1.250000
0.12          inf
3         0.75  1.250000
0.12           1.6
```

Set B:

Q1.Download the Market basket dataset. Write a python program to read thedataset and display its information. Preprocess the data (drop null valuesetc.)Convert the categorical values into numeric format.Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules.

```
import pandas as pd
import numpy as np
from mlxtend.preprocessing import Transac
from mlxtend.frequent_patterns import ass
```

In [25]:

```
df=pd.read_csv('archive.zip',header=None)
df
```

Out[25]:

	0	1	2	
0	shrimp	almonds	avocado	ve
1	burgers	meatballs	eggs	
2	chutney	NaN	NaN	
3	turkey	avocado	NaN	
4	mineral water	milk	energy bar	w
...	...	...	...	
7496	butter	light mayo	fresh bread	
7497	burgers	frozen vegetables	eggs	

7496	butter	light mayo	fresh bread
7497	burgers	frozen vegetables	eggs
7498	chicken	NaN	NaN
7499	escalope	green tea	NaN
7500	eggs	frozen smoothie	yogurt cake

7501 rows × 20 columns

In [26]:

```
df.describe ()
```

Out[26]:

	0	1	2	
count	7501	5747	4389	334
unique	115	117	115	11
top	mineral water	mineral water	mineral water	minera wate
freq	577	484	375	20

In [27]:

```
df.isnull().sum()
```

```
df.isnull().sum()
```

Out[27]:

```
0         0
1      1754
2      3112
3      4156
4      4972
5      5637
6      6132
7      6520
8      6847
9      7106
10     7245
11     7347
12     7414
13     7454
14     7476
15     7493
16     7497
17     7497
18     7498
19     7500
dtype: int64
```

In [28]:

```
df=df.dropna()
df
```

Out[28]:

	0	1	2	
0	shrimp	almonds	avocado	vegetable mi

In [29]:

```
df.describe()
```

Out[29]:

	0	1	2
count	1	1	1
unique	1	1	1
top	shrimp	almonds	avocado	vege
freq	1	1	1

In [34]:

```
transactions = []
for i in range(0,len(df) ):
    transactions.append([str(df.values[i,
                        for j in range(0
```

```
transactions
```

Out[35]:

```
[['shrimp', 'almonds', 'avoc
ado', 'vegetables mix', 'gre
en grapes']]
```

In [36]:

```
te=TransactionEncoder()
te_array=te.fit(transactions).transform(t
df=pd.DataFrame(te_array, columns=te.colu
df
```

Out[36]:

	almonds	avocado	green grapes	shrimp
0	True	True	True	True

In [37]:

```
freq_items=apriori(df, min_support=0.5, u
freq_items
```

Out[37]:

	support	itemsets
0	1.0	(almonds)
1	1.0	(avocado)
2	1.0	(green grapes)
3	1.0	(shrimp)
4	1.0	(vegetables mix)
5	1.0	(avocado, almonds)
6	1.0	(green grapes, almonds)
7	1.0	(shrimp, almonds)
8	1.0	(vegetables mix, almonds)
9	1.0	(green grapes, avocado)
10	1.0	(shrimp, avocado)
11	1.0	(vegetables mix, avocado)
12	1.0	(green grapes, shrimp)
13	1.0	(green grapes, vegetables mix)
14	1.0	(shrimp, vegetables mix)
		(green grapes, avocado

```
rules = association_rules(freq_items, met
)
rules = rules.sort_values(['support', 'co
print(rules)
```

```
          antecedents
consequents  \
0             (avocado)
(almonds)
1             (almonds)
(avocado)
2        (green grapes)
(almonds)
3             (almonds)
(green grapes)
4              (shrimp)
(almonds)
..                   ...
...
175      (green grapes)
(shrimp, vegetables mix, avo
cado, almonds)
176           (shrimp)  (gree
n grapes, vegetables mix, av
```

jupyter Assignment 3 Logout

Menu Not Trusted Python 3 (ipykernel)

Run Markdown Voilà

Assignment 3

Q1. Consider any text paragraph. Preprocess the text to remove any special characters and digits. Generate the summary using extractive summarization process.

In [3]:

```
import nltk
nltk.download("all")
```

```
[nltk_data] Downloading coll
ection 'all'
[nltk_data]    |
[nltk_data]    | Downloading
package abc to
[nltk_data]    |     /data/u
ser/0/ru.iiec.pydroid3/app_H
OME/nltk_data.
[nltk_data]    |     ..
[nltk_data]    |   Package a
bc is already up-to-date!
[nltk_data]    | Downloading
package alpino to
[nltk_data]    |     /data/u
ser/0/ru.iiec.pydroid3/app_H
OME/nltk_data.
```

```
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
%matplotlib inline
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import
WordNetLemmatizer
```

In [4]:

```
txt=""" Hello all, welcome to Python Prog
Acedamy. Python Programming Academy is a
nice Programming skills. It is difficult
enrolled in this Academy.. """
```

In [5]:

```
token_sentence=sent_tokenize(txt)
token_word=word_tokenize(txt)
print("Tokenize sentences:\n",token_sente
print("Tokenize Words:\n", token_word)
```

```
Tokenize sentences:
 [' Hello all, welcome to Py
thon Programming\nAcedamy.',
'Python Programming Academy
is a\nnice Programming skill
s.', 'It is difficult to get
\nenrolled in this Academ
y..']
Tokenize Words:
 ['Hello', 'all', ',', 'welc
ome', 'to', 'Python', 'Progr
amming', 'Acedamy', '.', 'Py
thon', 'Programming', 'Acade
my', 'is', 'a', 'nice', 'Pro
gramming', 'skills', '.', 'I
t', 'is', 'difficult', 'to',
'get', 'enrolled', 'in', 'th
is', 'Academy', '..']
```

In [8]:

```
freq=FreqDist(token_word)
print(freq)
```

```
<FreqDist with 21 samples an
d 28 outcomes>
```

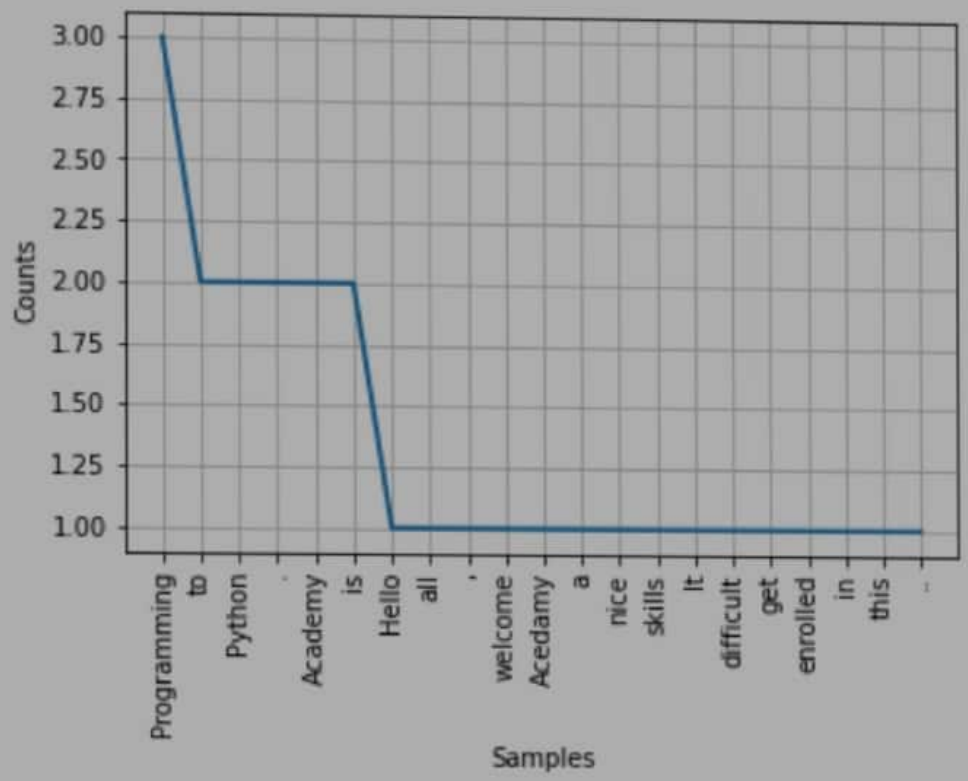
In [10]:

```
print(freq.most_common(2))
```

```
[('Programming', 3), ('to',
2)]
```

In [14]:

```
freq.plot(28, cumulative=False)
plt.show()
```



```
stop_words=set(stopwords.words("english")
print(stop_words)
```

```
{'here', "shan't", 'yoursel
f', 'whom', 'up', 'hers', 'h
e', 'again', 'most', 'hersel
f', 'from', 'same', 'your',
'hasn', 'for', 'into', 'abou
t', 'aren', "it's", 'just',
'their', 'through', 'more',
"won't", 're', 'did', 'doin
g', 'my', 'has', 'me', 'hims
elf', "aren't", 'ma', 'had',
'wasn', 'each', 'while', 'b
y', 'him', 't', "you're", 'w
on', "haven't", "you'd", 'hi
s', 'd', 'be', 'a', 'until',
'the', 'before', 'these', 's
o', "should've", 'you', 'al
l', "mightn't", 'with', "sho
uldn't", 'no', 'off', 'whe
n', 'against', 'they', 'onc
e', 'own', 'too', 'some', 't
hem', 'then', 'very', 'don',
'weren', 'being', 'ourselve
s', 'are', 'during', "sh
e's", "weren't", 'shan', "wa
sn't", 'were', 'yours', "do
n't", 'how', "couldn't",
's', 'o', 'needn', 'isn', "y
ou'll", 'and', 'themselves',
'do', 'myself', 'between',
```

```
filter_list=[]
for words in token_word:
    if words not in stop_words:
        filter_list.append(words)

print("Tokenized words:\n",token_word)
print("\nAfter removing stop words :\n",
      filter_list)
```

```
Tokenized words:
 ['Hello', 'all', ',', 'welc
ome', 'to', 'Python', 'Progr
amming', 'Acedamy', '.', 'Py
thon', 'Programming', 'Acade
my', 'is', 'a', 'nice', 'Pro
gramming', 'skills', '.', 'I
t', 'is', 'difficult', 'to',
'get', 'enrolled', 'in', 'th
is', 'Academy', '..']

After removing stop words :
 ['Hello', ',', 'welcome',
'Python', 'Programming', 'Ac
edamy', '.', 'Python', 'Prog
ramming', 'Academy', 'nice',
'Programming', 'skills',
'.', 'It', 'difficult', 'ge
t', 'enrolled', 'Academy',
'..']
```

```
stemmer=PorterStemmer ()
stemmed_words=[]
for words in filter_list:
    stemmed_words.append(stemmer.
                         stem(words))

print("Stemmed :\n", stemmed_words)
```

```
Stemmed :
 ['hello', ',', 'welcom', 'p
ython', 'program', 'acedam
i', '.', 'python', 'progra
m', 'academi', 'nice', 'prog
ram', 'skill', '.', 'it', 'd
ifficult', 'get', 'enrol',
'academi', '..']
```

Q3. Consider the following review messages. Perform sentiment analysis on the messages.

**i. I purchased headphones online. I am very happy with the product. ii. I saw the movie yesterday. The animation was really good but the script was ok. iii. I enjoy listening to music iv. I take a walk in the park everyday

```
from nltk.sentiment.vader import
SentimentIntensityAnalyzer
```

In [27]:

```
vader=SentimentIntensityAnalyzer()
text1=""" I purchased headphones online.
I am very happy with the product."""

print(vader.polarity_scores(text1))
```

```
{'neg': 0.0, 'neu': 0.667,
'pos': 0.333, 'compound': 0.
6115}
```

In [28]:

```
text2=""" I saw the movie yesterday.
The animation was really good but the
script was ok."""

print(vader.polarity_scores(text2))
```

```
{'neg': 0.0, 'neu': 0.71, 'p
os': 0.29, 'compound': 0.598
9}
```

```
text3=""" I enjoy listening to music """

print(vader.polarity_scores(text3))
```

```
{'neg': 0.0, 'neu': 0.484,
'pos': 0.516, 'compound': 0.
4939}
```

In [31]:

```
text4=" I take a walk in the park everyda

print(vader.polarity_scores(text4))
```

```
{'neg': 0.0, 'neu': 1.0, 'po
s': 0.0, 'compound': 0.0}
```