Preprocessing the data set

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

a={'Year':pd.Series([1987,1966,2005,2010,2015,2002,2017,2020,2007]),

  'Recipient name':pd.Series(['M.S.Swami Nathan','Zubin Mehta','Latha Mangeshkar','Anish Kapoor','Amithab Bachan','Jatin Das','Rajat Sharma','NarindraSingh Kopany','Prakash Padkone']),


'Field':pd.Series(['Science','Music','Music','Arts','Film','Journalism','Art','Science&Engineering','Sports']),

  'Age':pd.Series([62,30,75,60,70,80,65,85,65]),

  'Experience':pd.Series([40,20,55,40,50,50,35,55,40]),


'salary':pd.Series([160000,290000,160000,1500000,1200000,450000,1200000,100000,100000]),

  'no of awards':pd.Series([1,1,1,1,1,1,1,1,1])}

b=pd.DataFrame(a)

print(b)

print()

print('-----info-----')

print(b.info())

print(b.description())

print()

print('--------drop duplicates--------')

print(b.drop_duplicates())

y=b.drop_duplicates()

print()

print('-------looking for null values-------')

print(y.isnull().sum())

print()

print('--------looking for not null values--------')
```

```
print(y.notnull().sum())
print()
print('--------occupying null values with random numbers--------')
print(y.fillna(method='bfill'))
print(b.description())
```

output:

```
   Year       Recipient name  ...   salary  no of awards
0  1987      M.S.Swami Nathan  ...   160000             1
1  1966          Zubin Mehta  ...   290000             1
2  2005      Latha Mangeshkar  ...   160000             1
3  2010         Anish Kapoor  ...  1500000             1
4  2015       Amithab Bachan  ...  1200000             1
5  2002           Jatin Das  ...   450000             1
6  2017         Rajat Sharma  ...  1200000             1
7  2020  NarindraSingh Kopany  ...   100000             1
8  2007       Prakash Padkone  ...   100000             1

[9 rows x 7 columns]

-----info-----
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Year            9 non-null      int64
 1   Recipient name  9 non-null      object
 2   Field           9 non-null      object
 3   Age             9 non-null      int64
```

```
 4   Experience      9 non-null      int64
 5   salary          9 non-null      int64
 6   no of awards    9 non-null      int64
dtypes: int64(5), object(2)
memory usage: 632.0+ bytes
None

--------drop duplicates--------
   Year        Recipient name  ...   salary  no of awards
0  1987      M.S.Swami Nathan  ...   160000             1
1  1966           Zubin Mehta  ...   290000             1
2  2005      Latha Mangeshkar  ...   160000             1
3  2010          Anish Kapoor  ...  1500000             1
4  2015        Amithab Bachan  ...  1200000             1
5  2002             Jatin Das  ...   450000             1
6  2017          Rajat Sharma  ...  1200000             1
7  2020  NarindraSingh Kopany  ...   100000             1
8  2007       Prakash Padkone  ...   100000             1

[9 rows x 7 columns]

-------looking for null values-------
Year             0
Recipient name   0
Field            0
Age              0
Experience       0
salary           0
no of awards     0
dtype: int64
```

--------looking for not null values--------

Year 9

Recipient name 9

Field 9

Age 9

Experience 9

salary 9

no of awards 9

dtype: int64

--------occupying null values with random numbers--------

	Year	Recipient name	...	salary	no of awards
0	1987	M.S.Swami Nathan	...	160000	1
1	1966	Zubin Mehta	...	290000	1
2	2005	Latha Mangeshkar	...	160000	1
3	2010	Anish Kapoor	...	1500000	1
4	2015	Amithab Bachan	...	1200000	1
5	2002	Jatin Das	...	450000	1
6	2017	Rajat Sharma	...	1200000	1
7	2020	NarindraSingh Kopany	...	100000	1
8	2007	Prakash Padkone	...	100000	1

[9 rows x 7 columns]

Training the data:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
a={'Year':pd.Series([1987,1966,2005,2010,2015,2002,2017,2020,2007]),
```

```
  'Recipient name':pd.Series(['M.S.Swami Nathan','Zubin Mehta','Latha Mangeshkar','Anish Kapoor','Amithab Bachan','Jatin Das','Rajat Sharma','NarindraSingh Kopany','Prakash Padkone']),

'Field':pd.Series(['Science','Music','Music','Arts','Film','Journalism','Art','Science&Engineering','Sports']),
  'Age':pd.Series([62,30,75,60,70,80,65,85,65]),
  'Experience':pd.Series([40,20,55,40,50,50,35,55,40]),

'salary':pd.Series([160000,290000,1600000,1500000,1200000,450000,1100000,185000,100000]),
  'no of awards':pd.Series([1,1,1,1,1,1,1,1,1])}
b=pd.DataFrame(a)
print(b)
print()
print("Step  1   :  Training Data 1")
print()
train1=b.sample(n=5)
print(train1)
print()
print("Step  2   :  Training Data 2")
print()
train2=b.sample(frac=.57)
print(train2)
print()
print("Step  3   :  Bias")
print()
bias=train1['Age'].subtract(train2['Experience'])
print(bias)
tbias=bias.sum()
print()
print("Step 4   : Final Bias")
```

```
print()
E=(tbias/5)
print("Total Error : " ,E)
print()
output:
   Year        Recipient name  ...   salary  no of awards
0  1987      M.S.Swami Nathan  ...   160000            1
1  1966           Zubin Mehta  ...   290000            1
2  2005      Latha Mangeshkar  ...  1600000            1
3  2010          Anish Kapoor  ...  1500000            1
4  2015        Amithab Bachan  ...  1200000            1
5  2002             Jatin Das  ...   450000            1
6  2017          Rajat Sharma  ...  1100000            1
7  2020  NarindraSingh Kopany  ...   185000            1
8  2007       Prakash Padkone  ...   100000            1

[9 rows x 7 columns]

Step  1   :  Training Data 1

   Year        Recipient name  ...   salary  no of awards
5  2002             Jatin Das  ...   450000            1
4  2015        Amithab Bachan  ...  1200000            1
0  1987      M.S.Swami Nathan  ...   160000            1
3  2010          Anish Kapoor  ...  1500000            1
7  2020  NarindraSingh Kopany  ...   185000            1

[5 rows x 7 columns]

Step  2   :  Training Data 2
```

	Year	Recipient name	Field	Age	Experience	salary	no of awards
3	2010	Anish Kapoor	Arts	60	40	1500000	1
0	1987	M.S.Swami Nathan	Science	62	40	160000	1
6	2017	Rajat Sharma	Art	65	35	1100000	1
5	2002	Jatin Das	Journalism	80	50	450000	1
8	2007	Prakash Padkone	Sports	65	40	100000	1

Step 3 : Bias

0 22.0

3 20.0

4 NaN

5 30.0

6 NaN

7 NaN

8 NaN

dtype: float64

Step 4 : Final Bias

Total Error : 14.4

Finding correlation covariance:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
a={'Year':pd.Series([1987,1966,2005,2010,2015,2002,2017,2020,2007]),
  'Recipient name':pd.Series(['M.S.Swami Nathan','Zubin Mehta','Latha Mangeshkar','Anish Kapoor','Amithab Bachan','Jatin Das','Rajat Sharma','NarindraSingh Kopany','Prakash Padkone']),
```

```
'Field':pd.Series(['Science','Music','Music','Arts','Film','Journalism','Art','Science&Engineering','Sports']),
   'Age':pd.Series([62,30,75,60,70,80,65,85,65]),
   'Experience':pd.Series([40,20,55,40,50,50,35,55,40]),

'salary':pd.Series([160000,290000,160000,1500000,1200000,450000,1200000,100000,100000]),
   'no of awards':pd.Series([1,1,1,1,1,1,1,1,1])}
b=pd.DataFrame(a)
print(b)
print('-----covarience-----')
print()
print('covarience b/t age and exp')
p1=np.cov(b['Age']*2,b['Experience']*2)
print(p1)
print()
print('covarience b/t year and salary')
p2=np.cov(b['Year']*2,b['salary']*2)
print('------correlation-----')
print()
print('correlation between age and experience')
c1=b[['Age','Experience']].corr()
print(c1)
c2=b[['Year','salary']].corr()
print('correlation between year and salary')
print(c2)
```

output:

```
   Year       Recipient name  ...   salary  no of awards
0  1987      M.S.Swami Nathan  ...   160000            1
1  1966          Zubin Mehta  ...   290000            1
2  2005      Latha Mangeshkar  ...   160000            1
```

3 2010 Anish Kapoor ... 1500000 1

4 2015 Amithab Bachan ... 1200000 1

5 2002 Jatin Das ... 450000 1

6 2017 Rajat Sharma ... 1200000 1

7 2020 NarindraSingh Kopany ... 100000 1

8 2007 Prakash Padkone ... 100000 1

[9 rows x 7 columns]

-----covarience-----

covarience b/t age and exp

[[1001.77777778 665.27777778]

[665.27777778 502.77777778]]

covarience b/t year and salary

------correlation-----

correlation between age and experience

	Age	Experience
Age	1.000000	0.937409
Experience	0.937409	1.000000

correlation between year and salary

	Year	salary
Year	1.000000	0.381323
salary	0.381323	1.000000

Supervised learning algorithms:

Classification:

```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
a={'Year':pd.Series([1987,1966,2005,2010,2015,2002,2017,2020,2007]),
  'Recipient name':pd.Series(['M.S.Swami Nathan','Zubin Mehta','Latha Mangeshkar','Anish Kapoor','Amithab Bachan','Jatin Das','Rajat Sharma','NarindraSingh Kopany','Prakash Padkone']),

'Field':pd.Series(['Science','Music','Music','Arts','Film','Journalism','Art','Science&Engineering','Sports']),
  'Age':pd.Series([62,30,75,60,70,80,65,85,65]),
  'Experience':pd.Series([40,20,55,40,50,50,35,55,40]),

'salary':pd.Series([160000,290000,160000,1500000,1200000,450000,1200000,100000,100000]),
  'no of awards':pd.Series([1,1,1,1,1,1,1,1,1])}
b=pd.DataFrame(a)
print(b)
print("--------classification--------")
print("select dataset:")
print()
p1=b['Experience']
print(p1)
print('------classify dataset--------')
print()
p2=b['Experience']<30
print(p2)
p3=b[b['Experience']<30]
print("--------resultant dataset--------")
print(p3)
j1=b['salary']<1000000
print(j1)
j2=b[b['salary']<1000000]
print(j2)
```

output:

	Year	Recipient name	...	salary	no of awards
0	1987	M.S.Swami Nathan	...	160000	1
1	1966	Zubin Mehta	...	290000	1
2	2005	Latha Mangeshkar	...	160000	1
3	2010	Anish Kapoor	...	1500000	1
4	2015	Amithab Bachan	...	1200000	1
5	2002	Jatin Das	...	450000	1
6	2017	Rajat Sharma	...	1200000	1
7	2020	NarindraSingh Kopany	...	100000	1
8	2007	Prakash Padkone	...	100000	1

[9 rows x 7 columns]

--------classification--------

select dataset:

0 40

1 20

2 55

3 40

4 50

5 50

6 35

7 55

8 40

Name: Experience, dtype: int64

------classify dataset--------

0 False

1 True

2 False

3 False

4 False

5 False

6 False

7 False

8 False

Name: Experience, dtype: bool

--------resultant dataset--------

Year Recipient name Field Age Experience salary no of awards

1 1966 Zubin Mehta Music 30 20 290000 1

0 True

1 True

2 True

3 False

4 False

5 True

6 False

7 True

8 True

Name: salary, dtype: bool

Year Recipient name ... salary no of awards

0 1987 M.S.Swami Nathan ... 160000 1

1 1966 Zubin Mehta ... 290000 1

2 2005 Latha Mangeshkar ... 160000 1

5 2002 Jatin Das ... 450000 1

7 2020 NarindraSingh Kopany ... 100000 1

8 2007 Prakash Padkone ... 100000 1

[6 rows x 7 columns]

Linear regression and logistic regression:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

a={'Year':pd.Series([1987,1966,2005,2010,2015,2002,2017,2020,2007]),

  'Recipient name':pd.Series(['M.S.Swami Nathan','Zubin Mehta','Latha Mangeshkar','Anish Kapoor','Amithab Bachan','Jatin Das','Rajat Sharma','NarindraSingh Kopany','Prakash Padkone']),


'Field':pd.Series(['Science','Music','Music','Arts','Film','Journalism','Art','Science&Engineering','Sports']),

  'Age':pd.Series([62,30,75,60,70,80,65,85,65]),

  'Experience':pd.Series([40,20,55,40,50,50,35,55,40]),


'salary':pd.Series([160000,290000,1600000,1500000,1200000,450000,1100000,185000,100000]),

  'no of awards':pd.Series([1,1,1,1,1,1,1,1,1])}

b=pd.DataFrame(a)

print(b)

print()

print("Step  1   :  Training Data 1")

print()

train1=b.sample(n=5)

print(train1)

print()

print("Step  2   :  Training Data 2")

print()

train2=b.sample(frac=.57)

print(train2)

print()

print("Step  3   :  Bias")
```

```
print()
bias=train1['Age'].subtract(train2['Experience'])
print(bias)
tbias=bias.sum()
print()
print("Step 4   : Final Bias")
print()
E=(tbias/5)
print("Total Error : " ,E)
print()
print("------- Xaxis Correlation --------")
print()
a=-2.3
c=3.9
r1=b['Age'].corr(b['Experience'])
print("Xv1   :  ",r1)
r2=b['Age'].corr(b['Year'])
print("Xv2   :  ",r2)
print()
s1=((a*r1)+c)
print(s1)
s2=((a*r2)+c)
print(s2)
print("------- Calculation X-Axis --------")
print()
rv1=(((a*r1)+c)+E)
print("Fv1  :  ",rv1)
rv2=(((a*r2)+c)+E)
print("Fv1  :  ",rv2)
xaxisvalue=[rv1,rv2]
```

```
print()
print("X-Axis   :",xaxisvalue)
print()
print("------- Yaxis Covariance --------")
print()
a=-2.3
c=3.9
r1=b['Age'].cov(b['Experience'])
print("Xv1   :  ",r1)
r2=b['Age'].cov(b['Year'])
print("Xv2   :  ",r2)
print()
s1=((a*r1)+c)
print(s1)
s2=((a*r2)+c)
print(s2)
print("------- Calculation Y-Axis --------")
print()
rv1=(((a*r1)+c)+E)
print("Fv1  :  ",rv1)
rv2=(((a*r2)+c)+E)
print("Fv1  :  ",rv2)
yaxisvalue=[rv1,rv2]
print()
print("Y-Axis   :",yaxisvalue)
print()
print(" -------- Display -------- ")
plt.scatter(xaxisvalue,yaxisvalue)
print("--------classification--------")
print("select dataset:")
```

```
print()
p1=b['Age']
print(p1)
output:
   Year        Recipient name  ...   salary  no of awards
0  1987      M.S.Swami Nathan  ...   160000            1
1  1966          Zubin Mehta  ...   290000            1
2  2005      Latha Mangeshkar  ...  1600000            1
3  2010         Anish Kapoor  ...  1500000            1
4  2015       Amithab Bachan  ...  1200000            1
5  2002           Jatin Das  ...   450000            1
6  2017         Rajat Sharma  ...  1100000            1
7  2020  NarindraSingh Kopany  ...   185000            1
8  2007       Prakash Padkone  ...   100000            1

[9 rows x 7 columns]

Step  1   :  Training Data 1

   Year        Recipient name  ...   salary  no of awards
6  2017         Rajat Sharma  ...  1100000            1
2  2005      Latha Mangeshkar  ...  1600000            1
7  2020  NarindraSingh Kopany  ...   185000            1
5  2002           Jatin Das  ...   450000            1
8  2007       Prakash Padkone  ...   100000            1

[5 rows x 7 columns]

Step  2   :  Training Data 2
```

	Year	Recipient name	...	salary	no of awards
5	2002	Jatin Das	...	450000	1
0	1987	M.S.Swami Nathan	...	160000	1
7	2020	NarindraSingh Kopany	...	185000	1
3	2010	Anish Kapoor	...	1500000	1
2	2005	Latha Mangeshkar	...	1600000	1

[5 rows x 7 columns]

Step 3 : Bias

0 NaN
2 20.0
3 NaN
5 30.0
6 NaN
7 30.0
8 NaN
dtype: float64

Step 4 : Final Bias

Total Error : 16.0

------- Xaxis Correlation --------

Xv1 : 0.9374093546292627
Xv2 : 0.7932052517066778

1.7439584843526958

2.0756279210746413

------- Calculation X-Axis --------

Fv1 : 17.743958484352696

Fv1 : 18.07562792107464

X-Axis : [17.743958484352696, 18.07562792107464]

------- Yaxis Covariance --------

Xv1 : 166.31944444444449

Xv2 : 213.93055555555554

-378.6347222222223

-488.1402777777777

------- Calculation Y-Axis --------

Fv1 : -362.6347222222223

Fv1 : -472.1402777777777

Y-Axis : [-362.6347222222223, -472.1402777777777]

-------- Display --------

--------classification--------

select dataset:

0 62

1 30

2 75

3 60

4 70

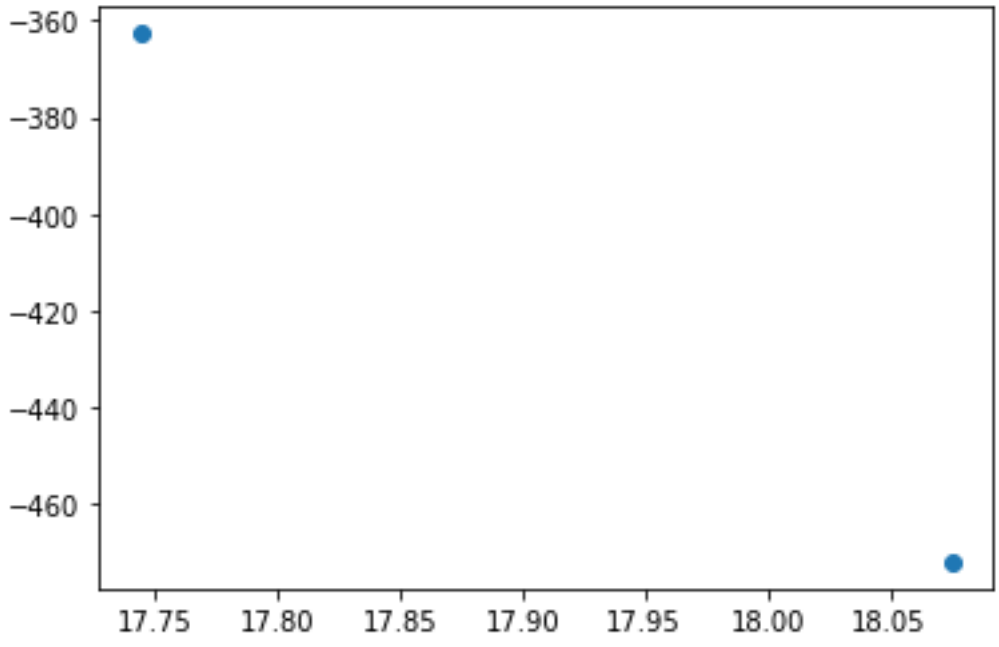
5 80

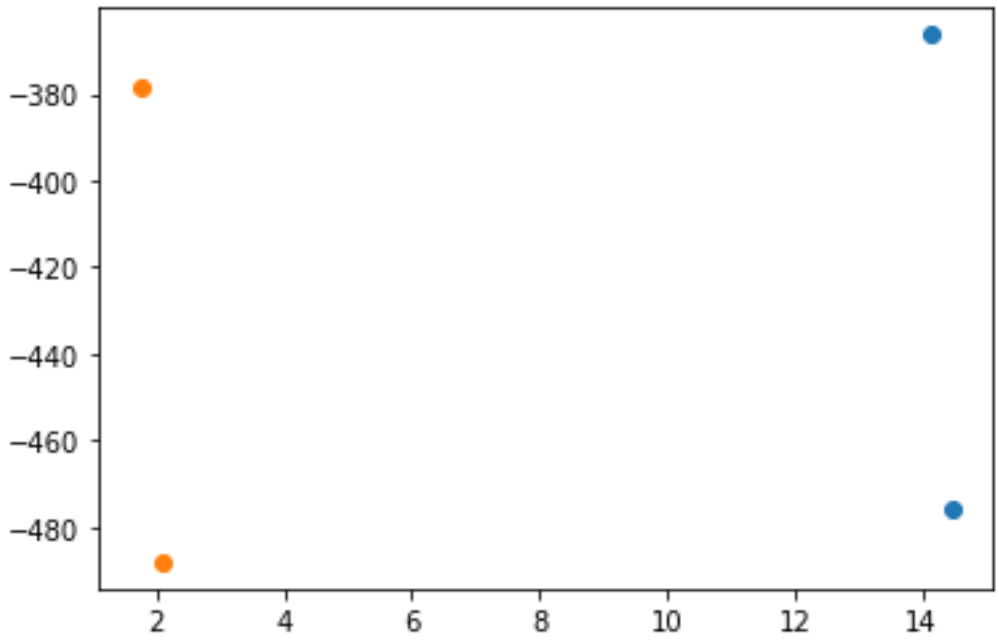
6 65

7 85

8 65

Name: Age, dtype: int64





Association:

import pandas as pd

```
from mlxtend.frequent_patterns import apriori, association_rules


# Create the dataset
data = {
    'Year': [1987, 1966, 2005, 2010, 2015, 2002, 2017, 2020, 2007],
    'Recipient Name': ['M.S. Swami Nathan', 'Zubin Mehta', 'Latha Mangeshkar', 'Anish Kapoor', 'Amitabh Bachchan', 'Jatin Das', 'Rajat Sharma', 'Narindra Singh Kopany', 'Prakash Padukone'],
    'Field': ['Science', 'Music', 'Music', 'Arts', 'Film', 'Journalism', 'Art', 'Science & Engineering', 'Sports'],
    'Age': [62, 30, 75, 60, 70, 80, 65, 85, 65],
    'Experience': [40, 20, 55, 40, 50, 50, 35, 55, 40],
    'Salary': [160000, 290000, 1600000, 1500000, 1200000, 450000, 1100000, 185000, 100000],
    'Total Awards Received': [1, 1, 1, 1, 1, 1, 1, 1, 1]
}


df = pd.DataFrame(data)


# Convert numerical columns to categorical
df['Age'] = pd.cut(df['Age'], bins=[0, 30, 50, 100], labels=['Young', 'Middle-aged', 'Old'])
df['Experience'] = pd.cut(df['Experience'], bins=[0, 30, 50, 100], labels=['Less Exp', 'Exp', 'High Exp'])
df['Salary'] = pd.cut(df['Salary'], bins=[0, 500000, 1000000, 2000000], labels=['Low', 'Medium', 'High'])


# Convert all columns to string
df = df.astype(str)


# Apply Apriori algorithm
frequent_itemsets = apriori(df, min_support=0.1, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
```

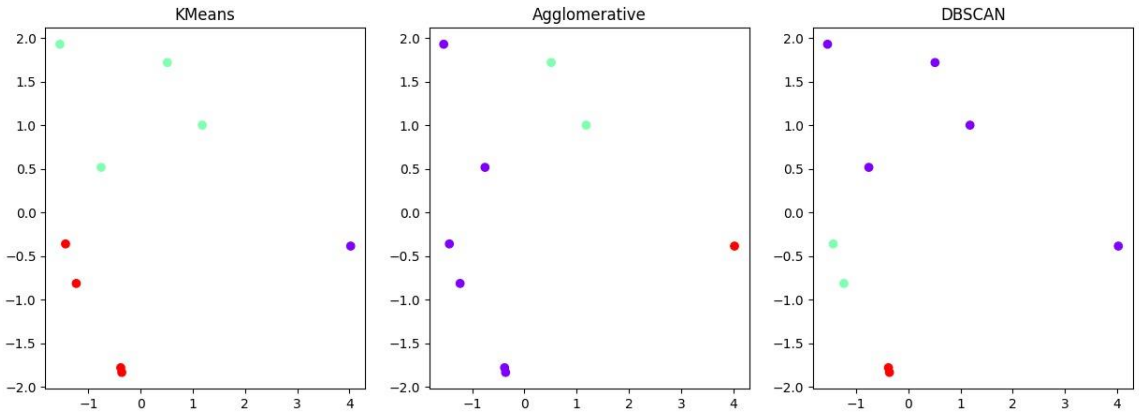
```
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

Unsupervised Learning:

K-Means algorithm

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

from sklearn.manifold import TSNE


# Define the dataset

data = {

    'Year': [1987, 1966, 2005, 2010, 2015, 2002, 2017, 2020, 2007],

    'Recipient Name': ['M.S. Swami Nathan', 'Zubin Mehta', 'Latha Mangeshkar', 'Anish Kapoor', 'Amitabh Bachchan', 'Jatin Das', 'Rajat Sharma', 'Narindra Singh Kopany', 'Prakash Padukone'],

    'Field': ['Science', 'Music', 'Music', 'Arts', 'Film', 'Journalism', 'Art', 'Science & Engineering', 'Sports'],

    'Age': [62, 30, 75, 60, 70, 80, 65, 85, 65],

    'Experience': [40, 20, 55, 40, 50, 50, 35, 55, 40],

    'Salary': [160000, 290000, 1600000, 1500000, 1200000, 450000, 1100000, 185000, 100000],

    'Total Awards Received': [1, 1, 1, 1, 1, 1, 1, 1, 1]

}


df = pd.DataFrame(data)


# Encode categorical variable 'Field'
```

```
df['Field'] = df['Field'].astype('category').cat.codes

# Standardize the dataset
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df.drop(['Recipient Name'], axis=1))

# Apply PCA for dimensionality reduction
pca = PCA(n_components=2)
principal_components = pca.fit_transform(scaled_features)

# Apply t-SNE for visualization
tsne = TSNE(n_components=2, perplexity=5, random_state=42)
tsne_results = tsne.fit_transform(scaled_features)

# Define clustering algorithms
clustering_algorithms = {
    'KMeans': KMeans(n_clusters=3),
    'Agglomerative': AgglomerativeClustering(n_clusters=3),
    'DBSCAN': DBSCAN(eps=0.5, min_samples=2)
}

# Apply clustering algorithms
plt.figure(figsize=(15, 5))
for i, (name, algorithm) in enumerate(clustering_algorithms.items(), 1):
    plt.subplot(1, 3, i)
    clusters = algorithm.fit_predict(principal_components)
    plt.scatter(principal_components[:, 0], principal_components[:, 1], c=clusters,
cmap='rainbow')
    plt.title(name)

plt.show()
```


KMeans
Agglomerative
DBSCAN