Recommender System Case Study

Problem Statement: Sam’s next exam would be to build a “Recommender System” using the Singular Value Decomposition (SVD) algorithm. Questions would be asked on the basis of what you’ve learnt in the respective module. Questions:

1. Implementing User-Based Recommender System using SVD (Singular Value Decomposition) method: a. Load the ‘ratings’ and ‘movies’ datasets which is a part of ‘MovieLense’ b. Find the unique number of users and movies in the ‘ratings’ dataset c. Create a rating matrix for the ‘ratings’ dataset and store it in ‘Ratings’ d. Load the ‘ratings’ dataset as SVD’s Dataset object and compute 3-fold cross-validation using the SVD object e. Find all the movies rated as 5 stars by user id ‘5’ and store it in ‘ratings_1’ data frame f. Create a shallow copy of the ‘movies’ dataset and store the result in ‘user_5’ g. Train a recommender system using the SVD object and predict the ratings for user id ‘5’ h. Print the top10 movie recommendations for the user id ‘5

In [1]:

```
import os
os.chdir('C:\\Users\\veena\\OneDrive\\Desktop\\intellipaat assignment pdf s')
```

In [2]:

```
import pandas as pd
ratings=pd.read_csv('ratings.csv')
ratings.head()
```

Out[2]:

	userId	movieId	rating	timestamp
0	1	2	3.5	1112486027
1	1	29	3.5	1112484676
2	1	32	3.5	1112484819
3	1	47	3.5	1112484727
4	1	50	3.5	1112484580

In [3]:

```
movies=pd.read_csv("movies.csv")
movies.head()
```

Out[3]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [4]:

```
n_users=ratings.userId.unique()
n_users
```

Out[4]:

```
array([   1,    2,    3, ..., 7118, 7119, 7120], dtype=int64)
```

In [5]:

```
n_movies=ratings.movieId.unique()
n_movies
```

Out[5]:

```
array([     2,     29,     32, ...,  32013, 102596,  65651], dtype=int64)
```

In [6]:

```
print(f'number of users={n_users}and Number of movies={n_movies}')
```

```
number of users=[   1    2    3 ... 7118 7119 7120]and Number of movies=[     2
29     32 ...  32013 102596  65651]
```

In [7]:

```
Ratings=ratings.pivot(index='userId',columns='movieId',values='rating').fillna(0)
Ratings.head()
```

Out[7]:

movieId	1	2	3	4	5	6	7	8	9	10	...	129350	129354	129428	129707	130
userId																
1	0.0	3.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
2	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
3	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	4.0	...	0.0	0.0	0.0	0.0	
5	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	

5 rows × 14026 columns

In [8]:

```
# conda install -c conda-forge scikit-surprise
```

In [9]:

```
from surprise import Reader,Dataset,SVD
from surprise.model_selection import cross_validate
```

In [10]:

```
reader=Reader()
reader
```

Out[10]:

```
<surprise.reader.Reader at 0x2475d117fa0>
```

In [11]:

```
data=Dataset.load_from_df(ratings[['userId','movieId','rating']],reader)
```

In [12]:

```
svd=SVD()
```

In [13]:

```
cross_validate(svd,data,measures=['RMSE','MAE'],cv=3,verbose=True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.8441  0.8459  0.8450  0.8450  0.0007
MAE (testset)     0.6469  0.6474  0.6479  0.6474  0.0004
Fit time          38.27   35.54   35.24   36.35   1.36
Test time         3.75    3.50    3.20    3.48    0.22
```

Out[13]:

```
{'test_rmse': array([0.84413101, 0.8458596 , 0.84496193]),
 'test_mae': array([0.64686998, 0.64735633, 0.64785583]),
 'fit_time': (38.265305519104004, 35.5358145236969, 35.23546385765076),
 'test_time': (3.7485849857330322, 3.4955708980560303, 3.2033848762512207)}
```

In [14]:

```
ratings.head()
```

Out[14]:

	userId	movieId	rating	timestamp
0	1	2	3.5	1112486027
1	1	29	3.5	1112484676
2	1	32	3.5	1112484819
3	1	47	3.5	1112484727
4	1	50	3.5	1112484580

In [15]:

```
ratings_1=ratings[(ratings['userId']==1)&(ratings['rating']==5)]
ratings_1=ratings_1.set_index('movieId')
ratings_1=ratings_1.join(movies)['title']
ratings_1.head()
```

Out[15]:

```
movieId
4993           Honky Tonk Freeway (1981)
5952        Harder They Come, The (1973)
7153                Dreamers, The (2003)
8507    Barkleys of Broadway, The (1949)
Name: title, dtype: object
```

In [16]:

```
user_1=movies.copy()
user_1=user_1.reset_index()
```

In [17]:

```
data=Dataset.load_from_df(ratings[['userId','movieId','rating']],reader)
trainset=data.build_full_trainset()
svd.fit(trainset)
```

Out[17]:

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x2475cf15b50>
```

In [18]:

```
user_1['Estimate_score']=user_1['movieId'].apply(lambda x:svd.predict(1,x).est)
user_1=user_1.drop(['movieId','genres','index'],axis=1)
user_1=user_1.sort_values('Estimate_score',ascending=False)
print(user_1.head(10))
```

```
                                                 title  Estimate_score
5853        Lord of the Rings: The Two Towers, The (2002)        4.717112
7041    Lord of the Rings: The Return of the King, The...        4.704398
4897    Lord of the Rings: The Fellowship of the Ring,...        4.692378
10081               Taste of Tea, The (Cha no aji) (2004)        4.517837
657                                    Underground (1995)        4.501614
16897                                    Jane Eyre (2011)        4.499993
6666             Lagaan: Once Upon a Time in India (2001)        4.473225
15208                                       Cosmos (1980)        4.469709
7356                              Band of Brothers (2001)        4.467460
7664                                     Gladiator (1992)        4.449940
```

In [ ]:

Case Study 2 - K-means

Problem Statement: Consider yourself to be Sam who is a data scientist. He has been approached by a retail car showroom to help them segregate the cars into different clusters Tasks to be performed:

1. Building the k-means clustering algorithm: a. Start off by extracting the 'mpg', 'disp' & 'hp' columns from the 'mtcars' data.frame. Store the result in 'car_features' b. Build the kmeans algorithm on top of 'car_features'. Here, the number of clusters should be 3 c. Bind the clustering vector to 'car_features'. d. Extract observations belonging to individual clusters
2. On the same 'car_features'dataset build a k-means algorithm, where the number of clusters is 5 a. Bind the clustering vector to 'car_features' b. Extract observations belonging to individual clusters

In [ ]:

```
import os
os.chdir('C:\\Users\\veena\\OneDrive\\Desktop')
```

In [20]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
data=pd.read_csv('cars-3.csv')
data.head()
```

Out[20]:

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

In [21]:

```
car_features=data[['mpg','disp','hp']]
car_features.head()
```

Out[21]:

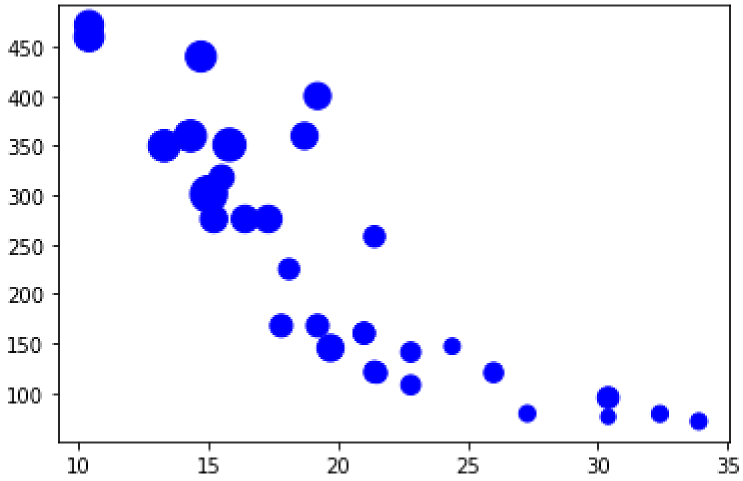
	mpg	disp	hp
0	21.0	160.0	110
1	21.0	160.0	110
2	22.8	108.0	93
3	21.4	258.0	110
4	18.7	360.0	175

In [22]:

```
f1=car_features['mpg'].values
f2=car_features['disp'].values
f3=car_features['hp'].values
x=np.array(list(zip(f1,f2,f3)))
plt.scatter(f1,f2,f3,c='blue')
```

Out[22]:

```
<matplotlib.collections.PathCollection at 0x247a71a2700>
```



In [27]:
```
import sklearn
```

In [29]:
```
from sklearn.cluster import KMeans
```

In [33]:
```
kmeans = KMeans(n_clusters=3)
kmeans =kmeans.fit(x)
labels=kmeans.predict(x)
centroids=kmeans.cluster_centers_
#print(C)
print(centroids)
```

```
C:\Users\veena\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set th
e value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\veena\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are l
ess chunks than available threads. You can avoid it by setting the environment var
iable OMP_NUM_THREADS=1.
  warnings.warn(
[[ 14.64444444 388.22222222 232.11111111]
 [ 24.5        122.29375     96.875     ]
 [ 17.01428571 276.05714286 150.71428571]]
```

In [ ]:

KMeans Clustering Assignment

Problem Statement: You work in XYZ Company as a Python. The company officials want you to write code for a clustering problem. Dataset: customers.csv Tasks to be performed:

1. K-Means Clustering:

- Load customer data.
- Check the number of cells in each column with null values.
- Create a scatter plot with Age as X and Spending Score as Y.
- Draw a scatter plot displaying data points colored on the basis of clusters.

In [ ]:

In [1]:

```
import os
os.chdir('C:\\Users\\veena\\OneDrive\\Desktop\\intellipaat assighnment pdf s')
```

In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [5]:

```
data=pd.read_csv('customers.csv')
data.head()
```

Out[5]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [10]:

```
data.isnull().sum()
```

Out[10]:

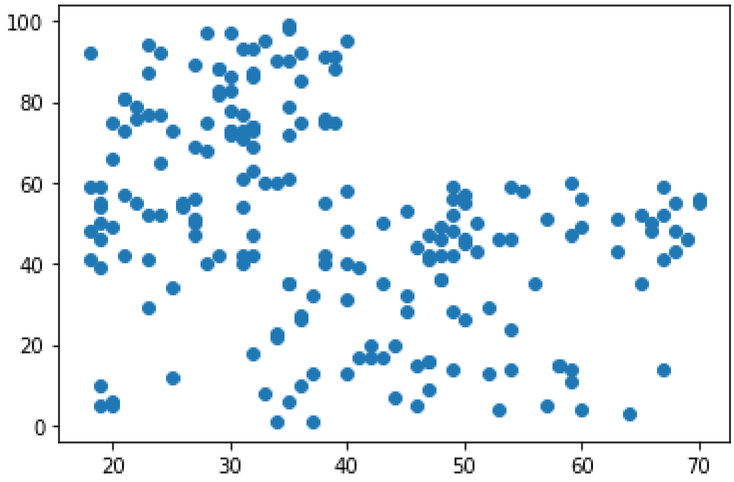
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [17]:

```
x=data['Age']
y=data['Spending Score (1-100)']
plt.scatter(x,y)
```

Out[17]:

```
<matplotlib.collections.PathCollection at 0x23899d6c370>
```



In [ ]:

KMeans Clustering Assignment

Problem Statement: You work in XYZ Company as a Python Developer. The company officials want you to write code for a clustering problem. Dataset: customers.csv Tasks to be performed:

1. K-Means Clustering:

- Load customer data.
- Check the number of cells in each column with null values.
- Create a scatter plot with Age as X and Spending Score as Y.
- Find out the best number for clusters between 1 and 10 (inclusive) using the elbowmethod.
- Draw a scatter plot displaying data points colored on the basis of clusters.

In [ ]:

```
import os
os.chdir('C:\\Users\\veena\\OneDrive\\Desktop\\intellipaat assighnment pdf s')
```

In [18]:

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [19]:

```
data=pd.read_csv('customers.csv')
data.head()
```

Out[19]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [20]:

```
data.isnull().sum()
```

Out[20]:

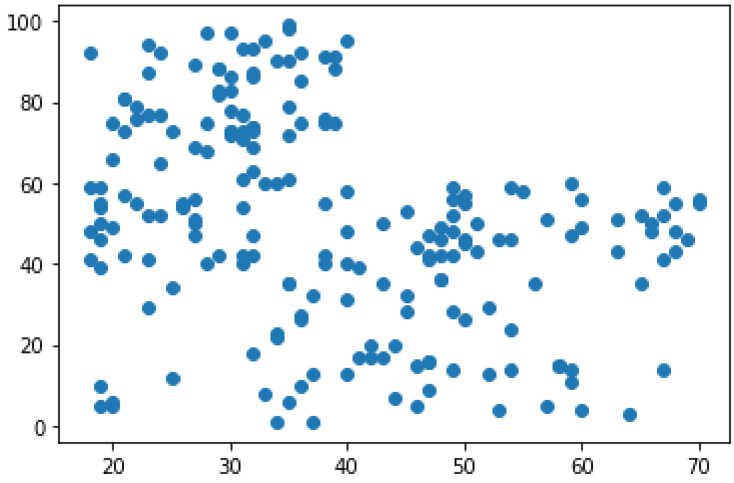
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [23]:

```
x=data['Age']
y=data['Spending Score (1-100)']
plt.scatter(x,y)
```

Out[23]:

```
<matplotlib.collections.PathCollection at 0x2389b4bd340>
```



In [24]:
```
from sklearn.cluster import KMeans
```

In [47]:
```
km=KMeans(n_clusters=3)
km
```

Out[47]:
```
▾ KMeans
KMeans(n_clusters=3)
```

In [48]:
```
y_predicted=km.fit_predict(data[['Age','Spending Score (1-100)']])
y_predicted
```

```
C:\Users\veena\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set th
e value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\veena\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWa
rning: KMeans is known to have a memory leak on Windows with MKL, when there are l
ess chunks than available threads. You can avoid it by setting the environment var
iable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[48]:
```
array([2, 1, 0, 1, 2, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 2, 1, 0, 1, 2, 1,
       0, 1, 0, 1, 2, 2, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 2, 1, 2, 2,
       0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1,
       2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 0, 1, 2, 1, 0, 1, 0, 1,
       2, 1, 0, 1, 0, 1, 0, 1, 0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 2, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1])
```

In [49]:
```
data['cluster']=y_predicted
data.head()
```

Out[49]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	cluster
0	1	Male	19	15	39	2
1	2	Male	21	15	81	1
2	3	Female	20	16	6	0
3	4	Female	23	16	77	1
4	5	Female	31	17	40	2

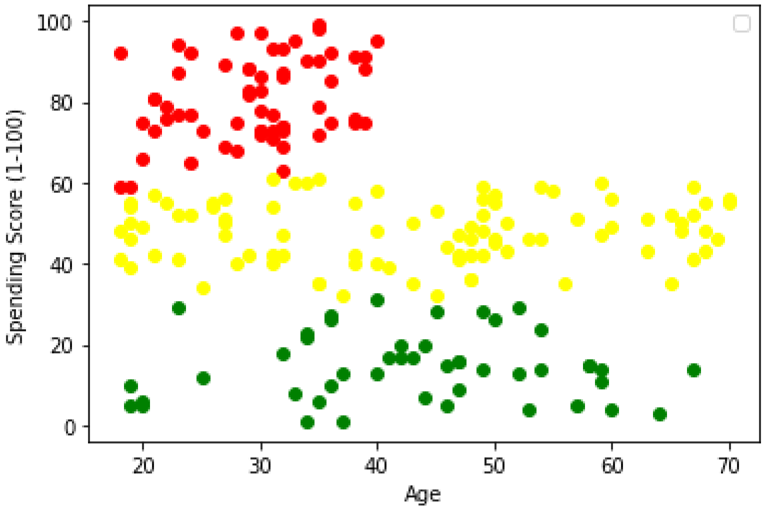
In [50]:

```
df1 = data[data.cluster==0]
df2 = data[data.cluster==1]
df3 = data[data.cluster==2]
plt.scatter(df1.Age,df1['Spending Score (1-100)'],color='green')
plt.scatter(df2.Age,df2['Spending Score (1-100)'],color='red')
plt.scatter(df3.Age,df3['Spending Score (1-100)'],color='yellow')
plt.xlabel('Age')
plt.ylabel('Spending Score (1-100)')
plt.legend()
```

```
No artists with labels found to put in legend.  Note that artists whose label star
t with an underscore are ignored when legend() is called with no argument.
```

Out[50]:

```
<matplotlib.legend.Legend at 0x2389c504550>
```



In [ ]:

Agglomerative Clustering Assignment

Problem Statement: You work in XYZ Company as a Python Developer. The company officials want you to write code for an Agglomerative Clustering Problem. Tasks to be performed:

- Load iris data from load_iris function from sklearn.datasets package.
- From the dataset extract the data property.
- Train an AgglomerativeClustring model based on the data.
- Plot dendrogram to visualize the clustering linkage

In [ ]:

In [ ]:

```
import os
os.chdir('C:\\Users\\veena\\OneDrive\\Desktop\\intellipaat assighnment pdf s')
```

In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import datasets
from sklearn.datasets import load_iris
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram,linkage
```

In [4]:

```
iris=datasets.load_iris()
```

In [5]:

```
x_features=iris.data
y_labels=iris.target
```

In [6]:

```
model= AgglomerativeClustering(linkage='ward', n_clusters=3)
```

In [7]:

```
model.fit(x_features)
predicted_labels=model.labels_
```

In [8]:

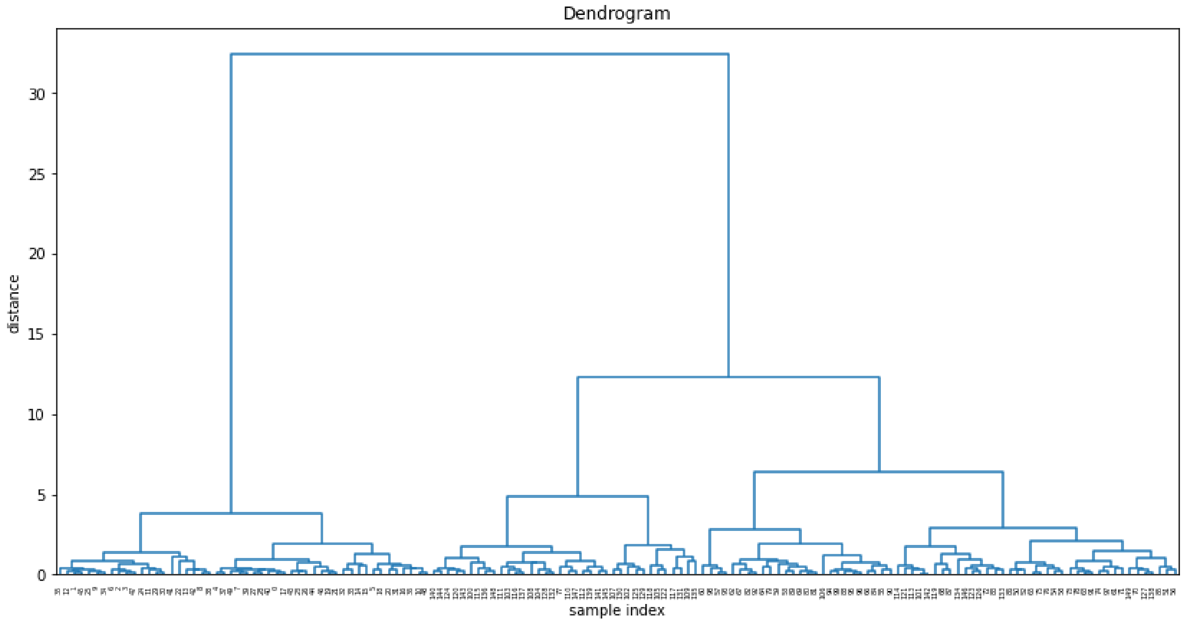
```
predicted_labels
```

Out[8]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2,
       2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2,
       2, 0, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0], dtype=int64)
```

In [10]:

```
linkage_matrix = linkage(x_features, 'ward')
plot = plt.figure(figsize=(14,7))
dendrogram(linkage_matrix, color_threshold=0,)
plt.title('Dendrogram')
plt.xlabel('sample index')
plt.ylabel('distance')
plt.show()
```



In [ ]:

Dendrogram Assignment

Problem Statement: You work in XYZ Company as a Python. The company officials want you to write code for a Agglomerative Clustering Problem. Data:[[5,3], [10,15], [15,12], [24,10], [30,30], [85,70], [71,80], [60,78], [70,55], [80,91],] Tasks to be performed: Using the np.array function create an np array from the data given above. Generate a scatter plot for the data. Plot dendrogram to visualize the clustering linkage

In [11]:

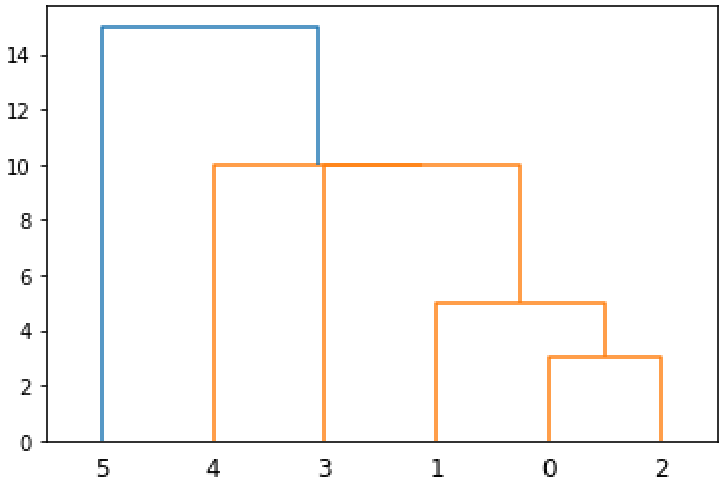
```
import numpy as np
from scipy.cluster import hierarchy
import matplotlib.pyplot as plt
```

In [15]:

```
x = np.array([5.,3.,10.,15.,15.,12.,24.,10.,30.,
               30.,85.,70.,71.,80.,60.])
```

In [16]:

```
Z = hierarchy.linkage(x, 'single')
plt.figure()
dn = hierarchy.dendrogram(Z)
```



Association Rule Mining Assignment

Problem Statement: You work in XYZ Company as a Python. The company officials want you to write code for a Association Rule Mining Dataset: retail_dataset.csv Tasks to be performed:

- Using pandas import the dataset as dataframe
- Install the mixtend library to use apriory and association rule mining
- Using the apriori algorithm generate a list of item frequently brought together.
- Generate the association rules for the given items from apriori algorithm

In [1]:

```
import os
os.chdir('C:\\Users\\veena\\OneDrive\\Desktop\\intellipaat assignment pdf s')
```

In [2]:

```
pip install apyori
```

```
Requirement already satisfied: apyori in c:\users\veena\anaconda3\lib\site-package
s (1.1.2)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from apyori import apriori
```

In [6]:

```
retail=pd.read_csv("C:\Users\veena\OneDrive\Desktop\intellipaat assignment pdf s\
retail.head()
```

```
  Input In [6]
    retail=pd.read_csv("C:\Users\veena\OneDrive\Desktop\intellipaat assighnment pd
f s\retail.csv", sep='[,|:_}]',encoding='unicode_escape',engine='python')

^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position
2-3: truncated \UXXXXXXXX escape
```

```
In [ ]:
```