WEEK-6

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the Principal Components. It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
```

```
heart_data=pd.read_csv("heart.csv")
heart_data.shape
```

```
(918, 12)
```

```
heart_data.columns  ##printing the features in dataset
```

```
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol',
'FastingBS',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
```

```
heart_data.head()  ##checking top 5 datapoints
```

```
   Age Sex ChestPainType  RestingBP  Cholesterol  FastingBS RestingECG  MaxHR
\
0   40   M           ATA        140          289          0     Normal    172
1   49   F           NAP        160          180          0     Normal    156
2   37   M           ATA        130          283          0         ST     98
3   48   F           ASY        138          214          0     Normal    108
4   54   M           NAP        150          195          0     Normal    122

  ExerciseAngina  Oldpeak ST_Slope  HeartDisease
0              N      0.0       Up             0
1              N      1.0     Flat             1
2              N      0.0       Up             0
3              Y      1.5     Flat             1
4              N      0.0       Up             0
```

```
heart_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
```

```
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

```
heart_data.describe()
```

```
              Age   RestingBP  Cholesterol   FastingBS       MaxHR  \
count  918.000000  918.000000   918.000000  918.000000  918.000000
mean    53.510893  132.396514   198.799564    0.233115  136.809368
std      9.432617   18.514154   109.384145    0.423046   25.460334
min     28.000000    0.000000     0.000000    0.000000   60.000000
25%     47.000000  120.000000   173.250000    0.000000  120.000000
50%     54.000000  130.000000   223.000000    0.000000  138.000000
75%     60.000000  140.000000   267.000000    0.000000  156.000000
max     77.000000  200.000000   603.000000    1.000000  202.000000

          Oldpeak  HeartDisease
count  918.000000    918.000000
mean     0.887364      0.553377
std      1.066570      0.497414
min     -2.600000      0.000000
25%      0.000000      0.000000
50%      0.600000      1.000000
75%      1.500000      1.000000
max      6.200000      1.000000
```

```
heart_data["ChestPainType"].unique()  ## printing unique values in the
attribute
```

```
array(['ATA', 'NAP', 'ASY', 'TA'], dtype=object)
```

```
heart_data["RestingECG"].unique()    ## printing unique values in the
attribute
```

```
array(['Normal', 'ST', 'LVH'], dtype=object)
```

```
heart_data["ExerciseAngina"].unique()   ## printing unique values in the
attribute
```

```
array(['N', 'Y'], dtype=object)
```

```python
heart_data["ST_Slope"].unique()       ## printing unique values in the attribute
```

```
array(['Up', 'Flat', 'Down'], dtype=object)
```

```python
heart_data["Sex"]=heart_data["Sex"].replace({
    "M":1,
    "F":0
})
heart_data["ChestPainType"]=heart_data["ChestPainType"].replace({
    "TA":1,
    "ATA":2,
    "NAP":3,
    "ASY":4
})
heart_data["RestingECG"]=heart_data["RestingECG"].replace({
    "Normal":1,
    "ST":2,
    "LVH":3
})
heart_data["ExerciseAngina"]=heart_data["ExerciseAngina"].replace({
    "Y":1,
    "N":0
})
heart_data["ST_Slope"]=heart_data["ST_Slope"].replace({
    "Up":2,
    "Flat":0,
    "Down":1
})
```

```python
heart_data.head()
```

```
   Age  Sex  ChestPainType  RestingBP  Cholesterol  FastingBS  RestingECG  \
0   40    1              2        140          289          0           1
1   49    0              3        160          180          0           1
2   37    1              2        130          283          0           2
3   48    0              4        138          214          0           1
4   54    1              3        150          195          0           1

   MaxHR  ExerciseAngina  Oldpeak  ST_Slope  HeartDisease
0    172               0      0.0         2             0
1    156               0      1.0         0             1
2     98               0      0.0         2             0
3    108               1      1.5         0             1
4    122               0      0.0         2             0
```

```python
x=heart_data.drop("HeartDisease",axis=1)
x.head()
```

```
   Age  Sex  ChestPainType  RestingBP  Cholesterol  FastingBS  RestingECG  \
0   40    1              2        140          289          0           1
```

```
1   49    0              3        160          180          0            1
2   37    1              2        130          283          0            2
3   48    0              4        138          214          0            1
4   54    1              3        150          195          0            1

   MaxHR  ExerciseAngina  Oldpeak  ST_Slope
0    172               0      0.0         2
1    156               0      1.0         0
2     98               0      0.0         2
3    108               1      1.5         0
4    122               0      0.0         2
```

```
y=heart_data["HeartDisease"]
y.head()
```

```
0    0
1    1
2    0
3    1
4    0
Name: HeartDisease, dtype: int64
```

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_scaled=scaler.fit_transform(x)
x_scaled
```

```
array([[-1.4331398 ,  0.51595242, -1.34508565, ..., -0.8235563 ,
        -0.83243239,  1.11255416],
       [-0.47848359, -1.93816322, -0.27042192, ..., -0.8235563 ,
         0.10566353, -0.96542086],
       [-1.75135854,  0.51595242, -1.34508565, ..., -0.8235563 ,
        -0.83243239,  1.11255416],
       ...,
       [ 0.37009972,  0.51595242,  0.80424181, ...,  1.21424608,
         0.29328271, -0.96542086],
       [ 0.37009972, -1.93816322, -1.34508565, ..., -0.8235563 ,
        -0.83243239, -0.96542086],
       [-1.64528563,  0.51595242, -0.27042192, ..., -0.8235563 ,
        -0.83243239,  1.11255416]])
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x_scaled,y,test_size=0.2,rando
m_state=20)
X_train.shape
X_test.shape
```

```
(184, 11)
```

```
from sklearn.naive_bayes import GaussianNB
model=GaussianNB()
```

```
model.fit(X_train,Y_train)
model.score(X_test,Y_test)
```

```
0.875
```

Using PCA

```
from sklearn.decomposition import PCA
pca=PCA(0.95)
x_pca=pca.fit_transform(x_scaled)
x_pca
```

```
array([[ 2.71440148,  0.10360385, -0.39363476, ..., -0.23837841,
        -0.10394659,  0.09955233],
       [ 0.76496373,  1.014489  , -0.06689062, ...,  0.64583661,
        -1.33928583, -1.05203148],
       [ 1.64407243, -0.22697701, -0.16856938, ..., -1.08575885,
        -0.55981896,  0.68307491],
       ...,
       [-1.78123503, -0.65129668, -1.11386306, ...,  0.14028026,
        -0.18336652,  0.11800791],
       [ 1.63159042,  1.79674152,  1.03841409, ..., -0.31052373,
        -1.55070888, -0.22048746],
       [ 2.27307086, -0.70064563, -0.53784041, ...,  0.63593364,
        -0.2876219 , -0.28389643]])
```

```
X_train,X_test,Y_train,Y_test=train_test_split(x_pca,y,test_size=0.2,random_s
tate=20)
```

```
X_train.shape
```

```
(734, 10)
```

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train,Y_train)
model.score(X_test,Y_test)
```

```
0.8641304347826086
```