Lab 5- Suyash Srivastava - 22MCS108

September 23, 2022

0.1 1. Data Description: Some new variations of an old disease have come which depends on gender and blood of the patients.

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("file.csv")
df.head()
```

[ ]:
```
   S.no Gender  Sugar_Level  Blood_component_A  Blood_component_B  \
0     0   Male          150           6.987503           3.125571
1     1   Male          150          -2.400811           5.886076
2     2   Male          150           7.225148           6.240692
3     3   Male          150           4.653442           1.291124
4     4   Male          150           6.383007           6.388507

   Blood_component_C  Height  Blood_component_D
0           8.515168     190         -10.179726
1           1.445175     190           9.303857
2           8.707427     190         -10.026728
3          -0.540128     190           1.554792
4           7.101920     190         -10.369927
```

0.2 2. Use data cleaning and transformation on the given dataset.

[ ]:
```
# Checking na in dataset
df.isna().sum()
```

[ ]:
```
S.no                 0
Gender               0
Sugar_Level          0
Blood_component_A    2
Blood_component_B    1
Blood_component_C    1
Height               0
Blood_component_D    1
```

```
dtype: int64
```

```
[ ]: df=df.fillna(0)
```

```
[ ]: df.isna().sum()
```

```
[ ]: S.no                 0
     Gender               0
     Sugar_Level          0
     Blood_component_A    0
     Blood_component_B    0
     Blood_component_C    0
     Height               0
     Blood_component_D    0
     dtype: int64
```

```
[ ]: # Removing Duplicates
     df.duplicated().sum()
```

```
[ ]: 0
```

```
[ ]: # Drop column that's not required
     df.drop(["S.no","Height","Sugar_Level"],axis=1,inplace=True)
     df['Gender']=df['Gender'].replace("Male",1).replace("Female",0)
     df.head()
     # df.tail()
```

```
[ ]:    Gender  Blood_component_A  Blood_component_B  Blood_component_C  \
     0       1           6.987503           3.125571           8.515168
     1       1          -2.400811           5.886076           1.445175
     2       1           7.225148           6.240692           8.707427
     3       1           4.653442           1.291124          -0.540128
     4       1           6.383007           6.388507           7.101920

        Blood_component_D
     0         -10.179726
     1           9.303857
     2         -10.026728
     3           1.554792
     4         -10.369927
```

```
[ ]: df.tail()
```

```
[ ]:      Gender  Blood_component_A  Blood_component_B  Blood_component_C  \
     495       0           8.607937          -8.624392          -8.178292
     496       0          -2.129690           6.375274          -1.240179
     497       0           7.120702           5.675873           8.057511
```

```
498      0          8.266646          6.571732          7.927840
499      0          6.615513          6.832900          6.690389

     Blood_component_D
495           1.212125
496           9.501309
497          -7.442512
498          -9.368286
499          -9.565085
```

[ ]:
```
# apply MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
import warnings
warnings.filterwarnings("ignore")

df_n = df.copy()
min_max_scaler = MinMaxScaler()
df_n=x = df.iloc[:, [0,1,2,3,4]] #selected only numerical parameters
df_n.iloc[:, [1,2,3,4]] = min_max_scaler.fit_transform(df_n.iloc[:, [1,2,3,4]])␣
 ↪# min-max scaling
# df_n.head()
df_n.tail()
```

[ ]:
```
     Gender  Blood_component_A  Blood_component_B  Blood_component_C  \
495      0           0.855446           0.143323           0.126862
496      0           0.148134           0.872192           0.462765
497      0           0.757478           0.838207           0.912903
498      0           0.832964           0.881739           0.906626
499      0           0.724200           0.894429           0.846716

     Blood_component_D
495           0.565798
496           0.926937
497           0.188737
498           0.104836
499           0.096261
```

0.3 3. Show the data head.

[ ]:
```
df_n.head()
```

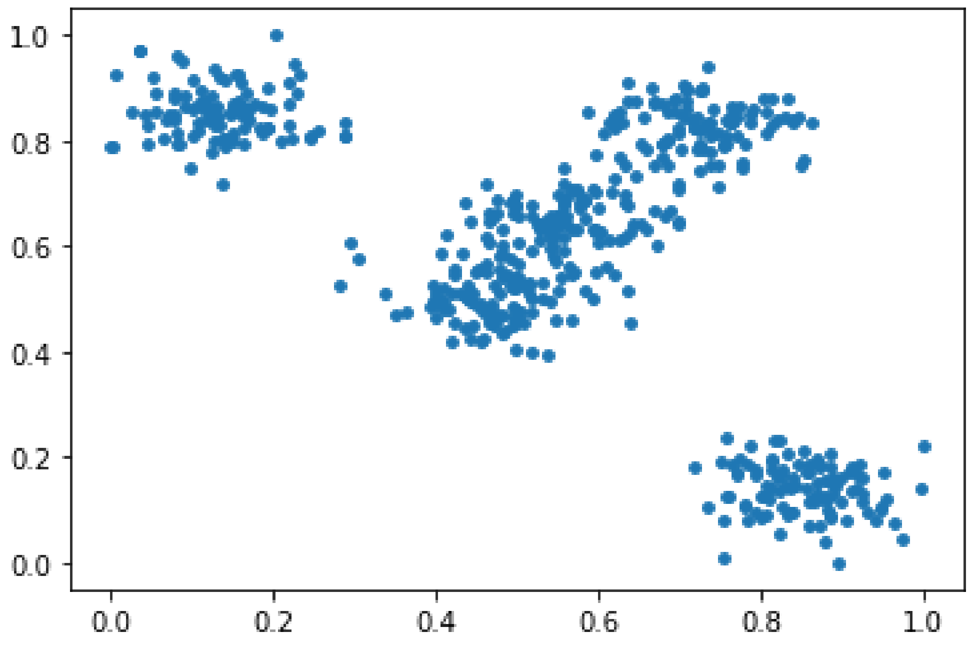
[ ]:
```
   Gender  Blood_component_A  Blood_component_B  Blood_component_C  \
0       1           0.748704           0.714282           0.935060
1       1           0.130274           0.848421           0.592773
2       1           0.764358           0.865653           0.944369
3       1           0.594954           0.625141           0.496657
4       1           0.708885           0.872835           0.866639
```

```
   Blood_component_D
0           0.069483
1           0.918334
2           0.076149
3           0.580727
4           0.061197
```

0.4 4. Implement following using sk-learn:

0.4.1 a. DBSCAN

[ ]:
```
plt.scatter(df_n['Blood_component_A'],df_n['Blood_component_B'],s=15)
plt.show()
```



[ ]:
```
from sklearn.cluster import DBSCAN
db_default = DBSCAN(eps = 0.4, min_samples = 2).fit(df_n) #fit dataset
labels = db_default.labels_
```

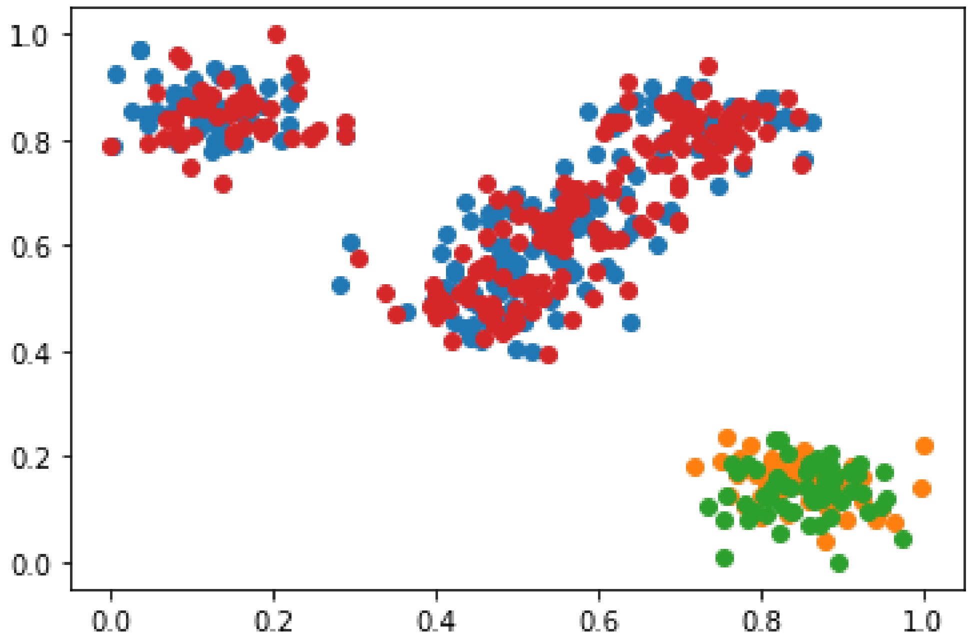
[ ]:
```
set(labels)
```

[ ]:
```
{0, 1, 2, 3}
```

[ ]:
```
labels
```

[ ]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 2, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       2, 2, 3, 3, 3, 2, 3, 2, 3, 2, 3, 2, 2, 3, 2, 3, 3, 3, 3, 3, 2, 3,
       2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 2, 3, 3, 3, 2, 2, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 3, 2, 3, 3, 3, 3,
       3, 2, 3, 3, 3, 2, 3, 2, 3, 2, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3, 3, 2,
       3, 3, 3, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 2, 3, 3, 3, 3, 2, 2, 2, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3,
       3, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 2, 3, 3, 3, 3, 3, 3,
       3, 2, 3, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3,
       2, 3, 3, 3, 3, 2, 3, 3, 3, 2, 3, 3, 2, 3, 3, 3, 3, 2, 3, 2, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3])
```

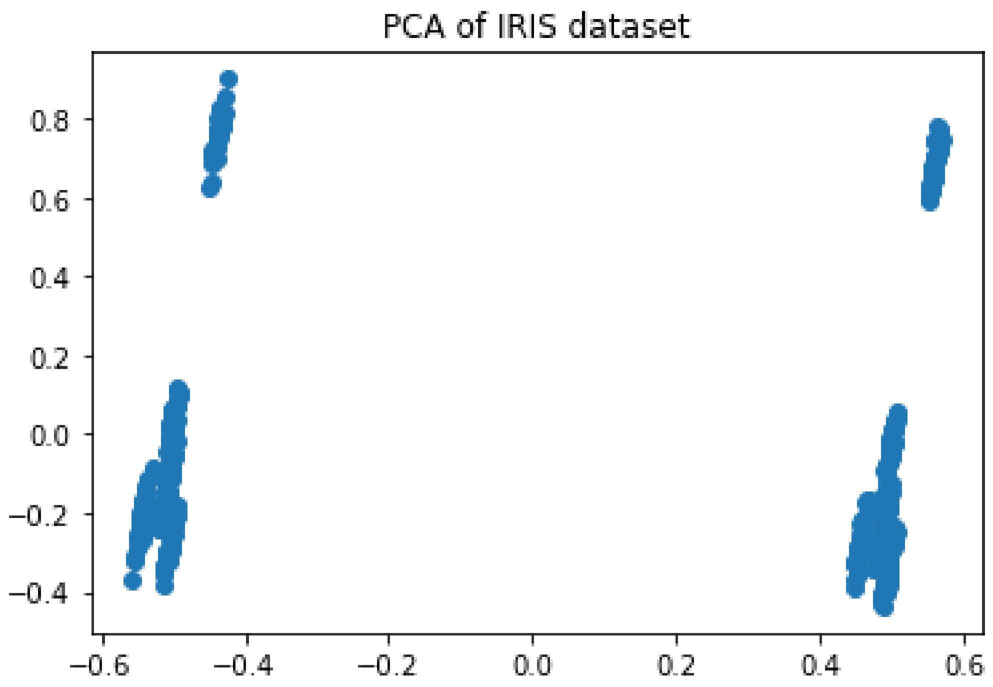
[ ]:
```
y_predict=labels
plt.scatter(df_n.Blood_component_A[y_predict == 0],df_n.
  ↪Blood_component_B[y_predict == 0])
plt.scatter(df_n.Blood_component_A[y_predict == 1],df_n.
  ↪Blood_component_B[y_predict == 1])
plt.scatter(df_n.Blood_component_A[y_predict == 2],df_n.
  ↪Blood_component_B[y_predict == 2])
plt.scatter(df_n.Blood_component_A[y_predict == 3],df_n.
  ↪Blood_component_B[y_predict == 3])
plt.show()
```



```
[ ]: from sklearn.decomposition import PCA
     pca = PCA(n_components=2)
     X_r = pca.fit(df_n).transform(df_n)
     # y=df_n
     plt.figure()
     # colors = ["navy", "turquoise", "darkorange"]
     # lw = 2
     # target_names=df_n

     # for color, i, target_name in zip(colors, [0, 1, 2], target_names):
     #     plt.scatter(
     #         X_r[ == i, 0], X_r[y == i, 1], color=color, alpha=0.8, lw=lw,␣
      ↪label=target_name
     #     )
     plt.scatter(X_r[:,0],X_r[:,1])
     plt.title("PCA of dataset")
     plt.show()

     # X_r
```



0.4.2 b. OPTICS

0.5 5. Find total no of variation of the disease

0.6 6. Find total no of noise points in data using DBSCAN

0.7 7. Create 2D-plot for all clusters using all required features.

0.8 8. Implement following from scratch:

```
a. DENCLUE
b. STING
c. CLIQUE
```

[ ]:

```

```