ML LAB ASSIGNMENT

SUPRATIM NAG -- CSE-AIML/22/057 -- GROUP-B

Q-13:Write a python code to implement Factorization and Generative Model.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
```

In [ ]:
```
data = pd.read_csv(r"C:\Users\SUPRATIM NAG\OneDrive\Documents\ML\Personal_Datasets\Dataset.csv")
```

In [3]:
```
data.shape
```

Out[3]: (100, 11)

In [4]:
```
data.head(1)
```

Out[4]:

	Patient ID	Age	Blood Pressure	Cholesterol Levels	Heart Rate	BMI	Diagnosis	Treatment Plan	Recovery Status	Medication Type	Follow-up Requirement
0	101	65	130	250	72	28.0	Hypertension with high cholesterol.	Medication: Lisinopril (blood pressure), Stati...	Active Recovery	Lisinopril, Statins.	Quarterly.

In [5]:
```
meddata=data[['Age','Blood Pressure','Cholesterol Levels','Heart Rate','BMI','Diagnosis']]
meddata.head(1)
```

Out[5]:

	Age	Blood Pressure	Cholesterol Levels	Heart Rate	BMI	Diagnosis
0	65	130	250	72	28.0	Hypertension with high cholesterol.

In [6]:
```
meddata['Diagnosis'] = meddata['Diagnosis'].apply(
    lambda x: 1 if any(condition in x for condition in ['Hypertension', 'Obesity', 'Overweight']) else 0
)
```

```
C:\Users\SUPRATIM NAG\AppData\Local\Temp\ipykernel_6372\2310464088.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  meddata['Diagnosis'] = meddata['Diagnosis'].apply(
```

In [7]:
```
print(meddata['Diagnosis'].value_counts())
```

```
Diagnosis
0    63
1    37
Name: count, dtype: int64
```

In [62]:
```
X = meddata.drop('Diagnosis', axis=1)
y = meddata['Diagnosis']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

In [63]:
```
model = GaussianNB()
```

In [64]:
```
kfold=model_selection.KFold(n_splits=6)
results = model_selection.cross_val_score(model,X_train, y_train, cv=kfold)
```

In [65]:
```
print("Results:", results)
print("Mean Results:", results.mean())
```

```
Results: [0.57142857 0.71428571 0.61538462 0.69230769 0.53846154 0.53846154]
Mean Results: 0.6117216117216117
```

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

In [18]:
```
#PROCESS 2
model = GaussianNB()
model.fit(X_train, y_train)
```

Out[18]: GaussianNB()

In [19]:
```
y_pred = model.predict(X_test)
```

In [23]:
```
print(y_pred)
```

```
[0 0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 1 1 0 1]
```

In [24]:
```
model.score(X_test, y_test)
```

Out[24]: 0.7

In [25]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[9 3]
 [3 5]]
              precision    recall  f1-score   support

           0       0.75      0.75      0.75        12
           1       0.62      0.62      0.62         8

    accuracy                           0.70        20
   macro avg       0.69      0.69      0.69        20
weighted avg       0.70      0.70      0.70        20
```

In [ ]: