Prediction of heart failure using Unsupervised Machine Learning

Submitted by:

NAME – SHIBASIS BUGUDEI

Step 1: Data Exploration-

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming your dataset file is named 'heart_failure.csv'
df = pd.read_csv('heart_failure.csv')

# Display the first 5 rows of the dataframe
print(df.head())
```

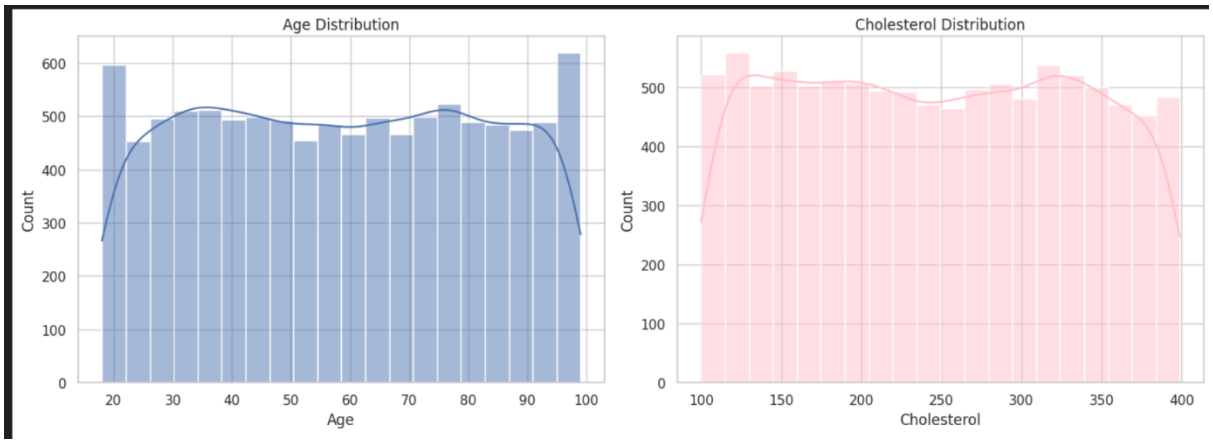
```
   Age  Gender Chest_Pain_Type  Resting_BP  Cholesterol  Fasting_Blood_Sugar  \
0   69    Male        Atypical         106          250                    1
1   32    Male     Non-anginal         124          396                    1
2   89  Female     Non-anginal         164          256                    1
3   78  Female         Typical         116          297                    1
4   38    Male     Non-anginal          88          386                    1

                    Resting_ECG  Max_Heart_Rate  Exercise_Induced_Angina  \
0         ST-T Wave Abnormality             171                        0
1  Left Ventricular Hypertrophy              73                        0
2  Left Ventricular Hypertrophy             157                        0
3                        Normal             163                        1
4         ST-T Wave Abnormality             123                        1

   Oldpeak        Slope  Num_Major_Vessels        Thalassemia  Diabetes  \
0     0.92         Flat                  2             Normal         1
1     0.92  Downsloping                  2  Reversible Defect         1
2     0.92    Upsloping                  1      Fixed Defect         1
3     0.92         Flat                  1  Reversible Defect         1
4     0.92    Upsloping                  3      Fixed Defect         0

  Smoking_History Alcohol_Consumption Physical_Activity_Level  Family_History  \
0          Former               Heavy                     Low               0
1         Current                 NaN                     Low               0
2          Former                 NaN                     Low               0
3          Former               Heavy                     Low               1
4           Never            Moderate                     Low               1

     BMI  Heart_Failure
0  36.92              1
1  36.92              1
2  36.92              0
3  36.92              0
4  36.92              1
```

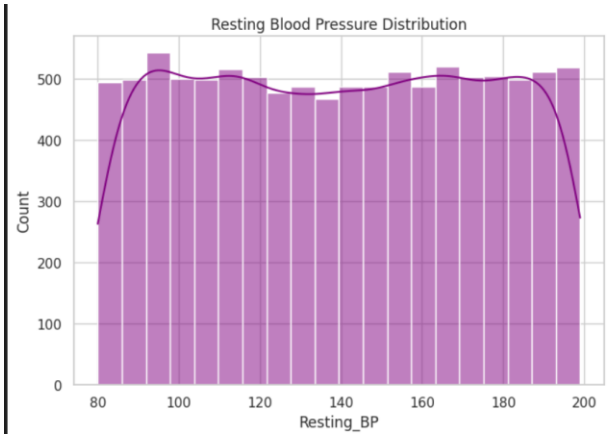
```
print(df.isnull().sum())
print(df.dtypes)
```

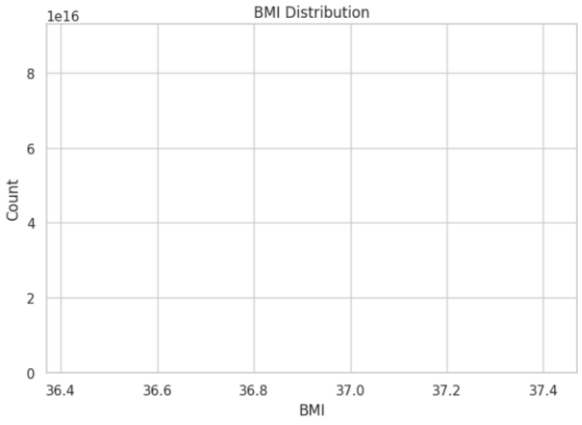
```
dtype: int64
Age                          int64
Gender                      object
Chest_Pain_Type             object
Resting_BP                   int64
Cholesterol                  int64
Fasting_Blood_Sugar          int64
Resting_ECG                 object
Max_Heart_Rate               int64
Exercise_Induced_Angina      int64
Oldpeak                    float64
Slope                       object
Num_Major_Vessels            int64
Thalassemia                 object
Diabetes                     int64
Smoking_History             object
Alcohol_Consumption         object
Physical_Activity_Level     object
Family_History               int64
BMI                        float64
Heart_Failure                int64
dtype: object
```

```
Age                        0
Gender                     0
Chest_Pain_Type            0
Resting_BP                 0
Cholesterol                0
Fasting_Blood_Sugar        0
Resting_ECG                0
Max_Heart_Rate             0
Exercise_Induced_Angina    0
Oldpeak                    0
Slope                      0
Num_Major_Vessels          0
Thalassemia                0
Diabetes                   0
Smoking_History            0
Alcohol_Consumption     3350
Physical_Activity_Level    0
Family_History             0
BMI                        0
Heart_Failure              0
```

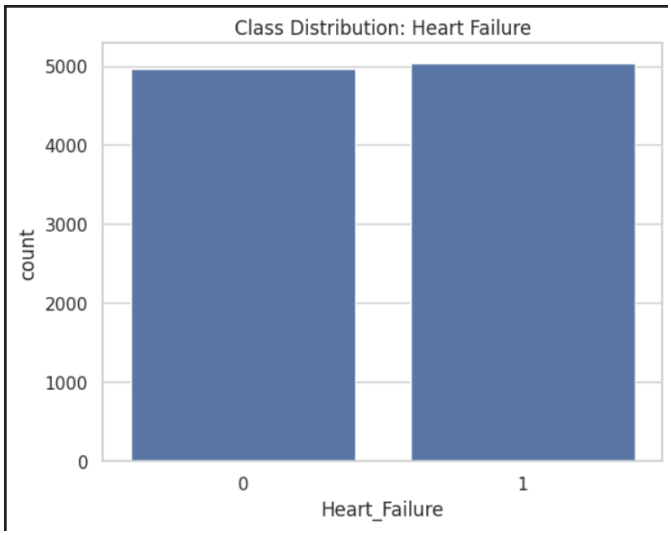
```python
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid")

plt.figure(figsize=(14, 10))

plt.subplot(2, 2, 1)
sns.histplot(df['Age'], kde=True, bins=20)
plt.title('Age Distribution')

plt.subplot(2, 2, 2)
sns.histplot(df['Cholesterol'], kde=True, bins=20, color='pink')
plt.title('Cholesterol Distribution')

plt.subplot(2, 2, 3)
sns.histplot(df['Resting_BP'], kde=True, bins=20, color='purple')
plt.title('Resting Blood Pressure Distribution')

plt.subplot(2, 2, 4)
sns.histplot(df['BMI'], kde=True, bins=20, color='salmon')
plt.title('BMI Distribution')

plt.tight_layout()
plt.show()
```







```
# Check balance of target class

print(df['Heart_Failure'].value_counts())
sns.countplot(x='Heart_Failure', data=df)
plt.title('Class Distribution: Heart Failure')
plt.show()
```



Step 2: Feature Selection and Preprocessing

- **Drop the DEATH_EVENT column (not needed for unsupervised learning).**
- **Ensure all remaining features are numerical.**
- **Standardize the features using a method like StandardScaler.**
- **Confirm the shape and scaling of the data.**

```
[15] # Drop the target column

df_unsupervised = df.drop(columns=['Heart_Failure'])

#Check non-numeric columns

non_numeric = df_unsupervised.select_dtypes (exclude=['number']).columns

print("Non-numeric columns:\n", non_numeric)
```

```
Non-numeric columns:
 Index(['Gender', 'Chest_Pain_Type', 'Resting_ECG', 'Slope', 'Thalassemia',
       'Smoking_History', 'Alcohol_Consumption', 'Physical_Activity_Level'],
      dtype='object')
```

```
df_encoded = pd.get_dummies(df_unsupervised, drop_first=True)
```

```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

x_scaled = scaler.fit_transform(df_encoded)


print("Shape of scaled data:", x_scaled.shape)
```

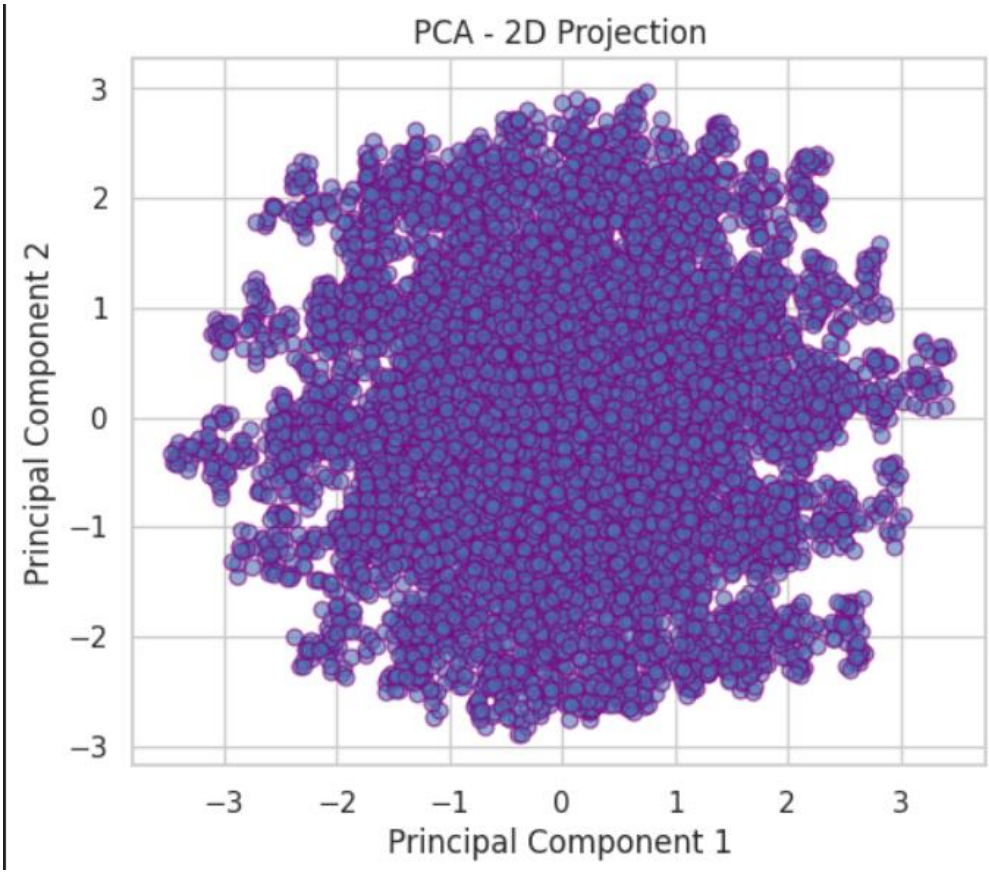
Step 3: Dimensionality Reduction with PCA

- **Apply PCA to reduce the dataset to 2 principal components.**
- **Print the explained variance ratio for each component.**
- **Create a scatter plot using the two principal components.**

```python
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

X_pca = pca.fit_transform(X_scaled)

#Explained variance

print("Explained variance ratio:", pca.explained_variance_ratio_)
```

```
Explained variance ratio: [0.06475142 0.06388744]
```

```python
plt.figure(figsize=(6, 5))
plt.scatter(X_pca[:, 0], X_pca[:, 1], s=40, alpha=0.6, edgecolors='purple')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA - 2D Projection')
plt.grid(True)
plt.show()
```



Step 4: Clustering with K-Means

- **Use the PCA-reduced data to apply K-Means clustering.**
- **Select an appropriate number of clusters (e.g., 2 or 3) and explain your choice.**
- **Assign a cluster label to each record.**
- **Visualize the clusters using the PCA plot with different colors for each cluster.**

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, random_state=42)
clusters = kmeans.fit_predict(X_pca)

#Add cluster labels to the PCA result

import numpy as np

X_pca_with_labels = np.column_stack((X_pca, clusters))
```

Why Choose 2 or 3 Clusters?

2 Clusters: Could represent low-risk vs. high-risk patients.

3 Clusters: Could potentially capture low, medium, and high risk segments or cluster based on certain shared clinical traits.

You could also evaluate optimal number of clusters using the Elbow Method for a more data-driven choice.