Hypothesis_KMeans

July 19, 2023

1 Task 1: Hypothesis testing and confidence intervals

Conduct hypothesis tests and calculate confidence intervals for the Heart Disease UCI dataset.

1.1 Load Packages

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

1.2 Read CSV Dataset

[2]:
```
df = pd.read_csv("heart_disease_uci.csv")
df.head()
```

[2]:
```
   id  age     sex    dataset               cp  trestbps   chol    fbs  \
0   1   63    Male  Cleveland   typical angina     145.0  233.0   True
1   2   67    Male  Cleveland     asymptomatic     160.0  286.0  False
2   3   67    Male  Cleveland     asymptomatic     120.0  229.0  False
3   4   37    Male  Cleveland      non-anginal     130.0  250.0  False
4   5   41  Female  Cleveland  atypical angina     130.0  204.0  False

          restecg  thalch  exang  oldpeak        slope   ca  \
0  lv hypertrophy   150.0  False      2.3  downsloping  0.0
1  lv hypertrophy   108.0   True      1.5         flat  3.0
2  lv hypertrophy   129.0   True      2.6         flat  2.0
3          normal   187.0  False      3.5  downsloping  0.0
4  lv hypertrophy   172.0  False      1.4    upsloping  0.0

                thal  num
0       fixed defect    0
1             normal    2
2  reversable defect    1
3             normal    0
4             normal    0
```

1.3 Data Wrangling/Cleaning

[120]:
```
# drop empty stages (num) and cholestrol levels as we need both to be filled
df.dropna(subset=['chol', 'num'], inplace=True)
# df.num.isna().sum()
```

[121]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 890 entries, 0 to 919
Data columns (total 15 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       890 non-null    int64
 1   sex       890 non-null    object
 2   dataset   890 non-null    object
 3   cp        890 non-null    object
 4   trestbps  834 non-null    float64
 5   chol      890 non-null    float64
 6   fbs       800 non-null    object
 7   restecg   888 non-null    object
 8   thalch    838 non-null    float64
 9   exang     838 non-null    object
 10  oldpeak   831 non-null    float64
 11  slope     603 non-null    object
 12  ca        308 non-null    float64
 13  thal      432 non-null    object
 14  num       890 non-null    int64
dtypes: float64(5), int64(2), object(8)
memory usage: 143.5+ KB
```

[5]:
```
df.dataset.value_counts()
```

[5]:
```
Cleveland        304
Hungary          270
VA Long Beach    193
Switzerland      123
Name: dataset, dtype: int64
```

[6]:
```
df.num.value_counts()
```

[6]:
```
0    392
1    258
2    107
3    106
4     27
Name: num, dtype: int64
```

[7]:
```
df.isna().sum()
```

[7]:
```
id            0
age           0
sex           0
dataset       0
cp            0
trestbps     56
chol          0
fbs          90
restecg       2
thalch       52
exang        52
oldpeak      59
slope       287
ca          582
thal        458
num           0
dtype: int64
```

[8]:
```
df.describe()
```

[8]:
```
               id         age    trestbps        chol      thalch     oldpeak  \
count  890.000000  890.000000  834.000000  890.000000  838.000000  831.000000
mean   458.016854   53.580899  132.089928  199.130337  137.539379    0.889290
std    267.339571    9.389502   19.077093  110.780810   25.989709    1.095398
min      1.000000   28.000000    0.000000    0.000000   60.000000   -2.600000
25%    223.250000   47.000000  120.000000  175.000000  120.000000    0.000000
50%    461.500000   54.000000  130.000000  223.000000  140.000000    0.500000
75%    690.750000   60.000000  140.000000  268.000000  157.000000    1.500000
max    920.000000   77.000000  200.000000  603.000000  202.000000    6.200000

               ca         num
count  308.000000  890.000000
mean     0.678571    1.008989
std      0.936378    1.145210
min      0.000000    0.000000
25%      0.000000    0.000000
50%      0.000000    1.000000
75%      1.000000    2.000000
max      3.000000    4.000000
```

1.4 Find Co-Relations

[9]:
```
df.corr()
```

[9]:

```
                id       age  trestbps      chol    thalch   oldpeak  \
id        1.000000  0.230532  0.053213 -0.376936 -0.470599  0.043070
age       0.230532  1.000000  0.253467 -0.086234 -0.360682  0.252412
trestbps  0.053213  0.253467  1.000000  0.092853 -0.114104  0.170562
chol     -0.376936 -0.086234  0.092853  1.000000  0.236121  0.047734
thalch   -0.470599 -0.360682 -0.114104  0.236121  1.000000 -0.151671
oldpeak   0.043070  0.252412  0.170562  0.047734 -0.151671  1.000000
ca        0.068357  0.372018  0.094925  0.051606 -0.265275  0.280301
num       0.267503  0.329766  0.128628 -0.231547 -0.371710  0.434298

                ca       num
id        0.068357  0.267503
age       0.372018  0.329766
trestbps  0.094925  0.128628
chol      0.051606 -0.231547
thalch   -0.265275 -0.371710
oldpeak   0.280301  0.434298
ca        1.000000  0.515338
num       0.515338  1.000000
```

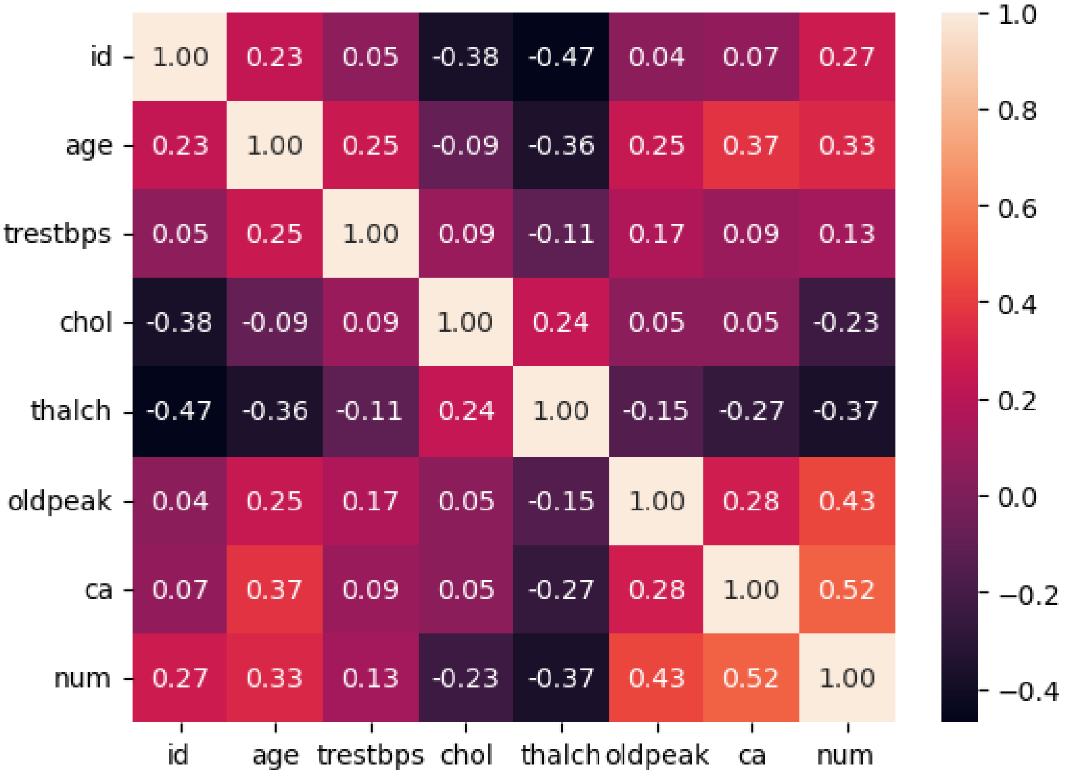
1.5 Co-Relation Heatmap

[10]:

```python
correlations = df.corr()

sns.heatmap(correlations, annot=True, fmt=".2f")
```

[10]:

```
<Axes: >
```



[11]:
```
# dropping unnecessary
df.drop(['id'], axis='columns', inplace=True)
```

[12]:
```
df
```

[12]:
```
     age     sex         dataset               cp  trestbps   chol    fbs  \
0     63    Male       Cleveland   typical angina     145.0  233.0   True
1     67    Male       Cleveland     asymptomatic     160.0  286.0  False
2     67    Male       Cleveland     asymptomatic     120.0  229.0  False
3     37    Male       Cleveland      non-anginal     130.0  250.0  False
4     41  Female       Cleveland  atypical angina     130.0  204.0  False
..   …     …             …                …        …      …    …
915   54  Female  VA Long Beach     asymptomatic     127.0  333.0   True
916   62    Male  VA Long Beach   typical angina       NaN  139.0  False
917   55    Male  VA Long Beach     asymptomatic     122.0  223.0   True
918   58    Male  VA Long Beach     asymptomatic       NaN  385.0   True
919   62    Male  VA Long Beach  atypical angina     120.0  254.0  False

              restecg  thalch  exang  oldpeak        slope   ca  \
0      lv hypertrophy   150.0  False      2.3  downsloping  0.0
1      lv hypertrophy   108.0   True      1.5         flat  3.0
2      lv hypertrophy   129.0   True      2.6         flat  2.0
```

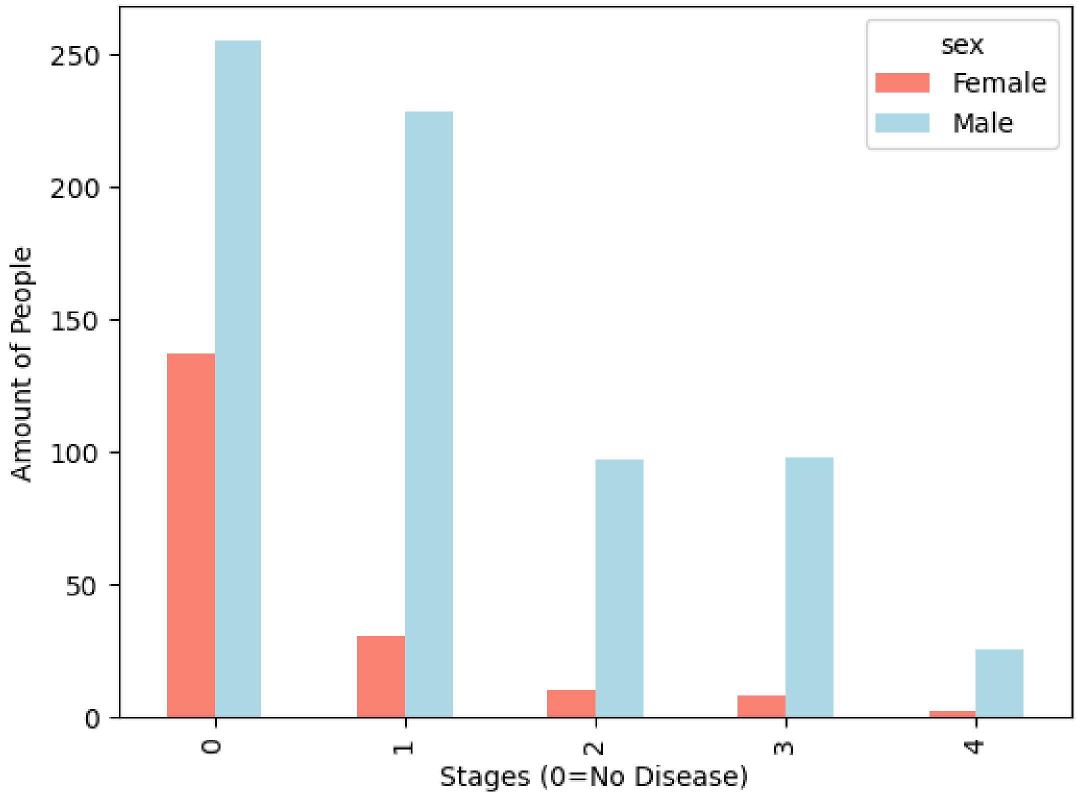
```
3              normal   187.0  False      3.5  downsloping  0.0
4      lv hypertrophy   172.0  False      1.4    upsloping  0.0
..                 …     …    …      …          …  …
915  st-t abnormality   154.0  False      0.0          NaN  NaN
916  st-t abnormality     NaN    NaN      NaN          NaN  NaN
917  st-t abnormality   100.0  False      0.0          NaN  NaN
918    lv hypertrophy     NaN    NaN      NaN          NaN  NaN
919    lv hypertrophy    93.0   True      0.0          NaN  NaN

                  thal  num
0         fixed defect    0
1               normal    2
2    reversable defect    1
3               normal    0
4               normal    0
..                 …  …
915                NaN    1
916                NaN    0
917       fixed defect    2
918                NaN    0
919                NaN    1

[890 rows x 15 columns]
```

1.6 Visualize Different Details

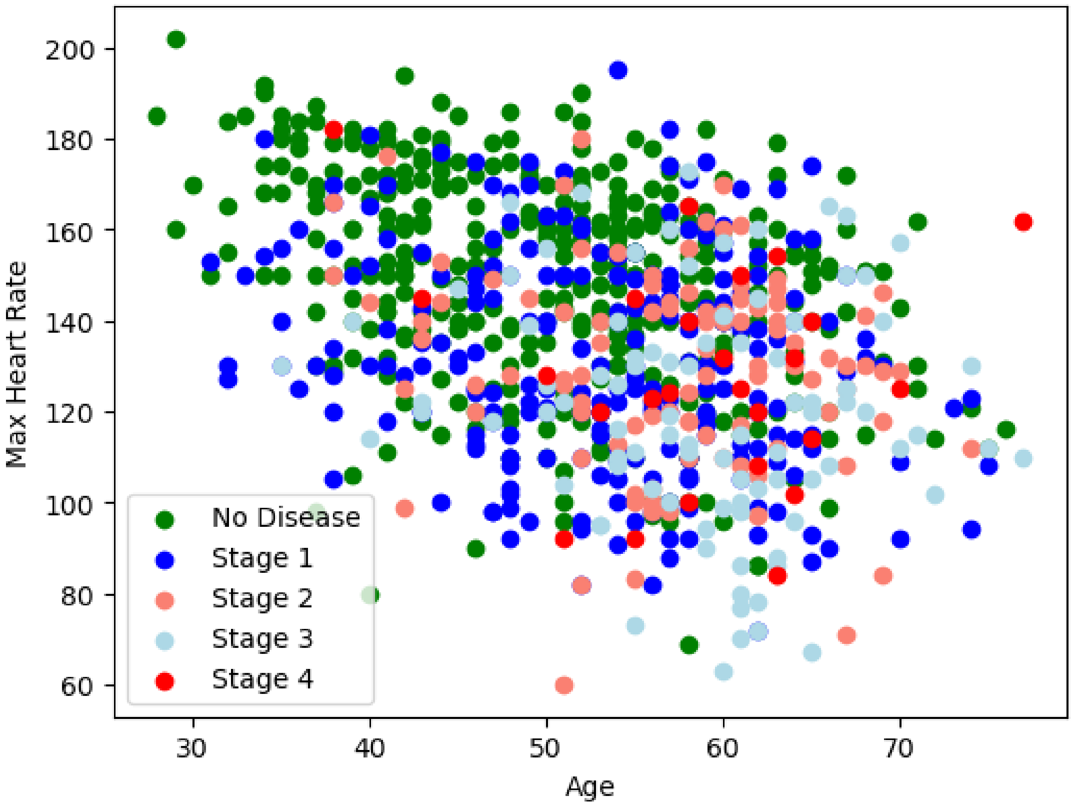
```
[13]: pd.crosstab(df.num, df.sex)
```

```
[13]: sex  Female  Male
      num
      0       137   255
      1        30   228
      2        10    97
      3         8    98
      4         2    25
```

```
[14]: pd.crosstab(df.num, df.sex).plot(kind="bar", color=['salmon', 'lightblue'])
      plt.xlabel("Stages (0=No Disease)")
      plt.ylabel("Amount of People")
      plt.show()
```



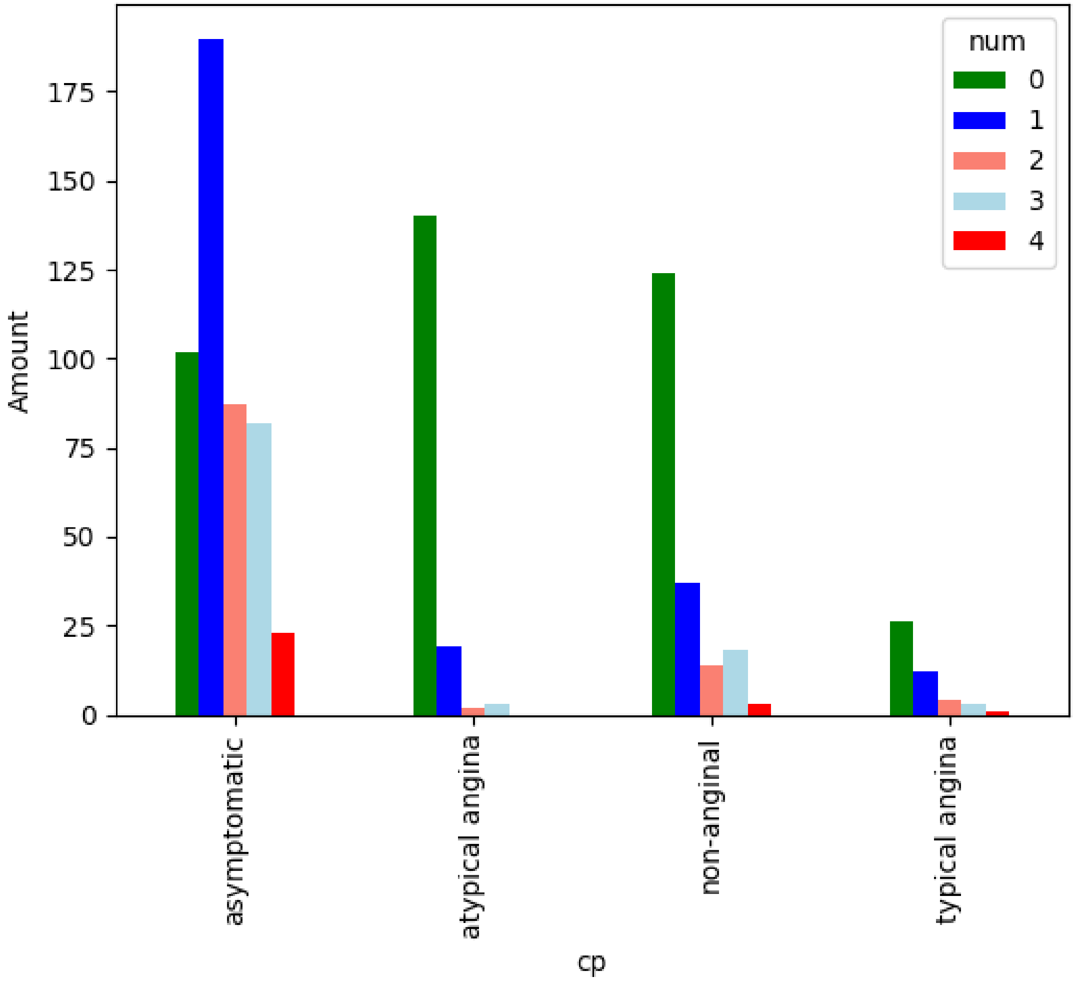
[15]:
```
# Compare Age vs max Heart Rate(thalch)
plt.scatter(df.age[df.num==0], df.thalch[df.num==0], c='green')
plt.scatter(df.age[df.num==1], df.thalch[df.num==1], c='blue')
plt.scatter(df.age[df.num==2], df.thalch[df.num==2], c='salmon')
plt.scatter(df.age[df.num==3], df.thalch[df.num==3], c='lightblue')
plt.scatter(df.age[df.num==4], df.thalch[df.num==4], c='red')

plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(['No Disease', 'Stage 1', 'Stage 2', 'Stage 3', 'Stage 4'])
plt.show()
```



[16]:
```
# Heart Disease per chest pain
pd.crosstab(df.cp, df.num)
```

[16]:
```
num                 0    1   2   3   4
cp
asymptomatic      102  190  87  82  23
atypical angina   140   19   2   3   0
non-anginal       124   37  14  18   3
typical angina     26   12   4   3   1
```

[17]:
```
pd.crosstab(df.cp, df.num).plot(kind="bar", color=['green','blue','salmon',␣
 ↪'lightblue', 'red'])
plt.ylabel("Amount")
plt.show()
```



[18]:
```
df
```

[18]:
```
     age     sex        dataset               cp  trestbps   chol    fbs  \
0     63    Male      Cleveland   typical angina     145.0  233.0   True
1     67    Male      Cleveland     asymptomatic     160.0  286.0  False
2     67    Male      Cleveland     asymptomatic     120.0  229.0  False
3     37    Male      Cleveland      non-anginal     130.0  250.0  False
4     41  Female      Cleveland  atypical angina     130.0  204.0  False
..   …     …            …              …        …    …    …
915   54  Female  VA Long Beach     asymptomatic     127.0  333.0   True
916   62    Male  VA Long Beach   typical angina       NaN  139.0  False
917   55    Male  VA Long Beach     asymptomatic     122.0  223.0   True
918   58    Male  VA Long Beach     asymptomatic       NaN  385.0   True
919   62    Male  VA Long Beach  atypical angina     120.0  254.0  False

              restecg  thalch  exang  oldpeak        slope   ca   \
0      lv hypertrophy   150.0  False      2.3  downsloping  0.0
1      lv hypertrophy   108.0   True      1.5         flat  3.0
2      lv hypertrophy   129.0   True      2.6         flat  2.0
3              normal   187.0  False      3.5  downsloping  0.0
4      lv hypertrophy   172.0  False      1.4    upsloping  0.0
..                 …     …    …      …          …  …
915  st-t abnormality   154.0  False      0.0          NaN  NaN
916  st-t abnormality     NaN    NaN      NaN          NaN  NaN
917  st-t abnormality   100.0  False      0.0          NaN  NaN
918    lv hypertrophy     NaN    NaN      NaN          NaN  NaN
919    lv hypertrophy    93.0   True      0.0          NaN  NaN

                  thal  num
0         fixed defect    0
1               normal    2
2    reversable defect    1
3               normal    0
4               normal    0
..                  …  …
915                NaN    1
916                NaN    0
917       fixed defect    2
918                NaN    0
919                NaN    1

[890 rows x 15 columns]
```

1.7 Finding Hypothesis

1.8 Null Hypothesis (H0):

There is no significant difference in cholesterol levels between patients with and without heart disease.

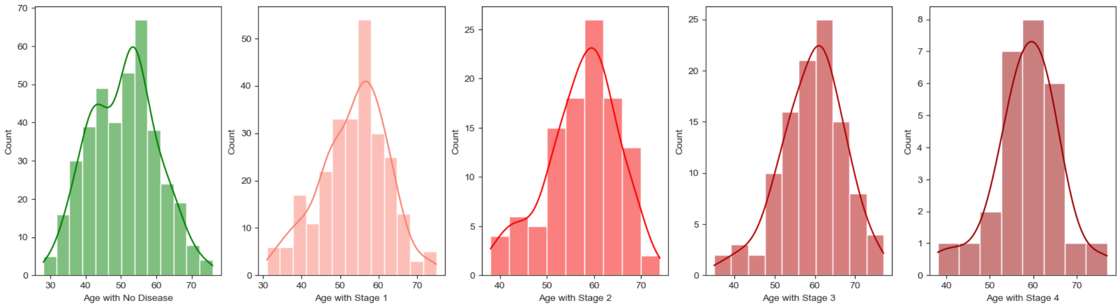
1.9 Alternative Hypothesis (H1):

There is a significant difference in cholesterol levels between patients with and without heart disease.

[79]:

```
age_heart_disease = df.groupby("num")['age']

fig, (axis1, axis2, axis3, axis4, axis5) = plt.subplots(1, 5, figsize=(20, 5))

ax = sns.histplot(age_heart_disease.get_group(0), kde=True, color='green',␣
 ↪ax=axis1)
ax.set(xlabel="Age with No Disease")
ax = sns.histplot(age_heart_disease.get_group(1), kde=True, color='salmon',␣
 ↪ax=axis2)
ax.set(xlabel="Age with Stage 1")
```

```
ax = sns.histplot(age_heart_disease.get_group(2), kde=True, color='#F70000',␣
 ↪ax=axis3)
ax.set(xlabel="Age with Stage 2")
ax = sns.histplot(age_heart_disease.get_group(3), kde=True, color='#B10000',␣
 ↪ax=axis4)
ax.set(xlabel="Age with Stage 3")
ax = sns.histplot(age_heart_disease.get_group(4), kde=True, color='#960000',␣
 ↪ax=axis5)
ax.set(xlabel="Age with Stage 4")

plt.show()
```



[80]:

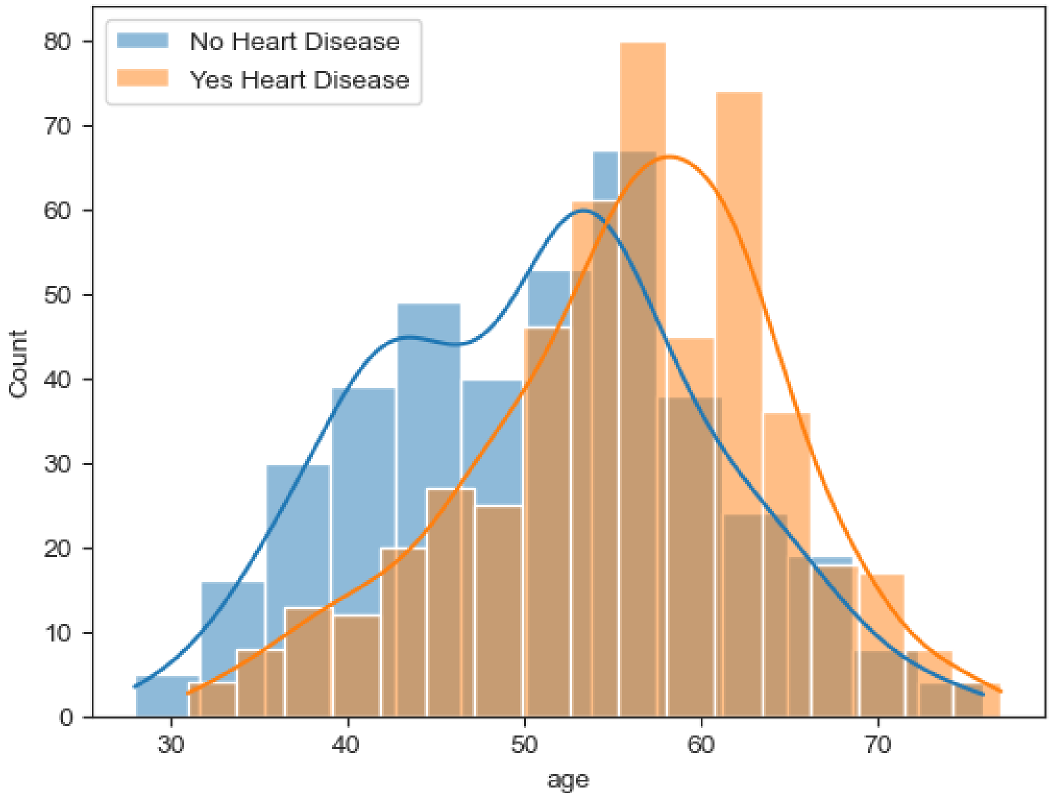
```
# Hypothesis Testing
ages_mean = df.groupby("num")['age'].mean()
ages_std = df.groupby("num")['age'].std()

stages_df = pd.DataFrame({'Stage':[0,1,2,3,4], 'Age Mean': ages_mean.values,␣
 ↪'Age Std': ages_std.values,
                          'Sample Size':[ len(df.age[df.num==0]), len(df.age[df.
 ↪num==1]), len(df.age[df.num==2]),
                                         len(df.age[df.num==3]), len(df.age[df.
 ↪num==4]) ] })
stages_df
```

[80]:

```
   Stage   Age Mean   Age Std  Sample Size
0      0  50.704082  9.507525          392
1      1  53.511628  8.755757          258
2      2  57.336449  7.650310          107
3      3  59.320755  7.952894          106
4      4  58.592593  7.747254           27
```

[103]:

```
# Population Density Function of Age

sns.histplot(age_heart_disease.get_group(0), label='No Heart Disease', kde=True)
sns.histplot(pd.concat([age_heart_disease.get_group(1),
                             age_heart_disease.get_group(2),
                            age_heart_disease.get_group(3),
                            age_heart_disease.get_group(4)]), label='Yes Heart␣
 ↪Disease', kde=True)
plt.legend()
plt.show()

# we can observe that the distribution of the age of the person who doesn't␣
 ↪have heart disease
# shifted downward and to the left of those who have heart disease.
```



Since we want to compare the cholesterol levels (numerical variable) between two groups (patients with and without heart disease), we will use an independent t-test.

[104]:
```
from scipy.stats import ttest_ind

noheart_disease = df[df.num == 0]['chol']
heart_disease = df[df.num > 0]['chol']

# perform independent t-test
t_stat, p_value = ttest_ind(heart_disease, noheart_disease)
```

```
# Print the results
print("t-statistic:", t_stat)
print("p-value:", p_value)

alpha = 0.05

if p_value < alpha:
    print("Reject Null Hypothesis. There is a significant difference in␣
  ↪cholesterol levels.")
else:
    print("Fail to Reject Null Hypothesis. There is no significant difference␣
  ↪in cholesterol levels.")
```

```
t-statistic: -7.061510014417096
p-value: 3.323201042049721e-12
Reject Null Hypothesis. There is a significant difference in cholesterol levels.
```

1.10 Confidence Intervals

[119]:

```
# Confidence Interval
# confidence intervals for the 'chol' (cholesterol levels) and 'thalach'␣
  ↪(maximum heart rate achieved) variables
import math

columns = df[['thalch', 'chol']]
columns_mean = columns.mean()
columns_std = columns.std()

columns_standard_error = columns_std/len(columns)
columns_margin_error = columns_standard_error/2

confidence_level = 0.95

upper_bound = columns_mean + confidence_level * (columns_std / math.sqrt(10))
lower_bound = columns_mean - confidence_level * (columns_std / math.sqrt(10))

confidence_intervals = pd.DataFrame({
    'Sample Size': columns.count(),
    'Sample Mean': columns_mean,
    'Standard Error': columns_standard_error,
    'Margin of Error': columns_margin_error,
    'Lower Bound (95% CI)': lower_bound,
    'Upper Bound (95% CI)': upper_bound
})

confidence_intervals
```

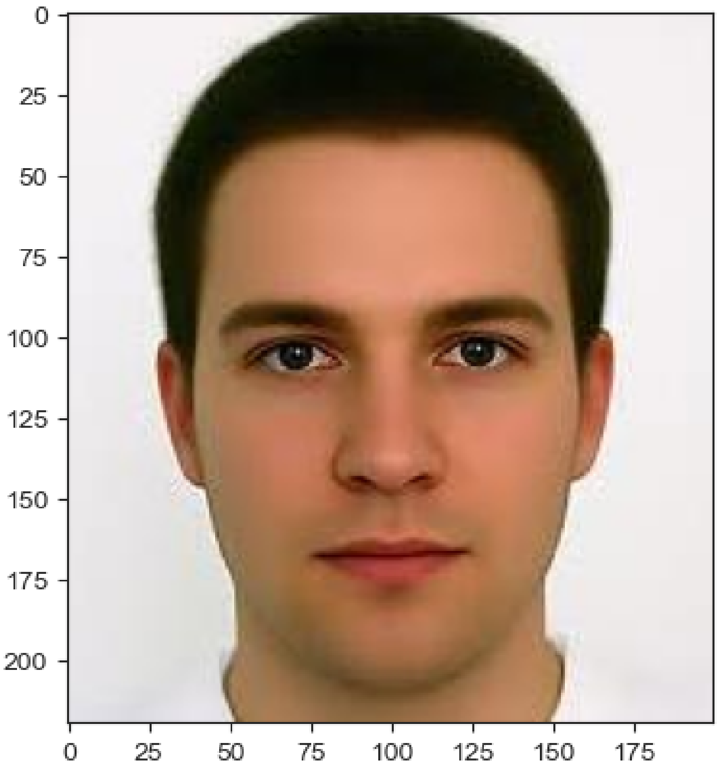
```
[119]:          Sample Size  Sample Mean  Standard Error  Margin of Error  \
        thalch          838   137.539379        0.029202         0.014601
        chol            890   199.130337        0.124473         0.062236

                Lower Bound (95% CI)  Upper Bound (95% CI)
        thalch            129.731645            145.347114
        chol              165.849967            232.410707
```

```
[ ]:
```

2 Task 2: K-Means: Document the results for K=2, 3, 5, 10, 15 and 20. Comment on the results.

2.1 Load Image

```
[167]: import cv2
       import numpy as np

       faceImage = cv2.imread("face.jpg")
       faceImage = cv2.cvtColor(faceImage, cv2.COLOR_BGR2RGB)
       plt.imshow(faceImage)
       plt.show()
```



2.2 Convert Image to Numerical Pixels Values

[168]:

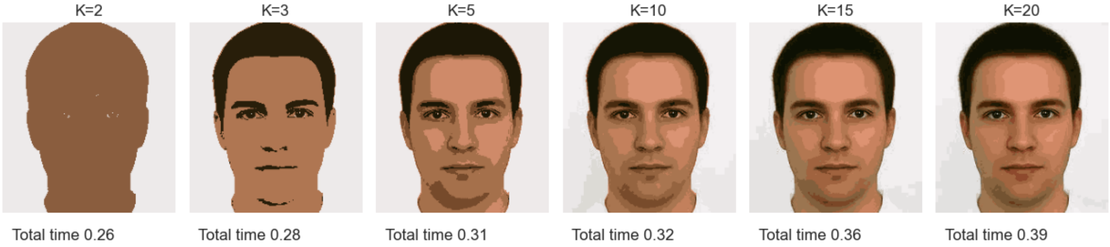
```
# Reshape the image to a 2D array of pixels
pixel_values = faceImage.reshape((-1, 3))
# pixel_values = np.float32(pixel_values)

pixel_values
print(pixel_values.shape)
```

```
(44000, 3)
```

2.3 Clustering (Getting the Cluster Centers) and Image Segmentation

[212]:

```
from sklearn.cluster import KMeans
import time

plt.figure(figsize=(12, 8))

#clusters values
K_values = [2, 3, 5, 10, 15, 20]

for i, K in enumerate(K_values):
    time_start = time.time()

    # CLUSTERING
    kmeans = KMeans(n_clusters = K, n_init='auto')
    kmeans.fit(pixel_values)

    # Get the cluster centers and labels
    cluster_centers = kmeans.cluster_centers_
    labels = kmeans.labels_

    # SEGMENTATION

    # Create a new image with the same shape as the original image
    segmented_image = np.zeros_like(pixel_values)

    # Replace each pixel value with the corresponding cluster center
    for j in range(len(cluster_centers)):
        segmented_image[labels == j] = cluster_centers[j]

    # Reshape the segmented_image back to the original image shape
    segmented_image = segmented_image.reshape(faceImage.shape)

    time_end = time.time()
    computational_time = time_end - time_start
    # segmented_image
    plt.subplot(1, len(K_values), i + 1)  # Use i + 1 for the subplot index
    plt.imshow(segmented_image)

    plt.axis('off')
    plt.text(10, 250, f'Total time {round(computational_time, 2)}', fontsize=12)
    plt.title(f'K={K}')

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



2.4 Observations

Number Of Segments As you increase the value of K, the number of segments or clusters in the segmented image increases. Smaller K values might group similar regions together, resulting in larger segments, whereas larger K values tend to create more detailed and smaller segments. #### Detail Level Higher K values generally produce more detailed segmented images with finer distinctions between different regions or objects in the image. Lower K values, on the other hand, might merge similar regions, leading to a loss of fine details. #### Computational Complexity The computational cost of k-means clustering increases with higher K values. Larger K values require more iterations and may be computationally expensive, especially for large images.

3 Task 3 Read and make a report

Link https://aws.amazon.com/getting-started/hands-on/build-train-deploy-machine-learning-model-sagemaker/

3.1 Here are the main steps covered in the article:

3.1.1 Create a SageMaker notebook instance

You start by creating a SageMaker notebook instance where you can work on your data and ML model. This instance is used to download and process the data.

3.1.2 Prepare the data

The tutorial uses the Bank Marketing Data Set, which contains customer demographics and responses to marketing events. The data is pre-labeled to identify whether a customer enrolls for a product offered by the bank.

3.1.3 Train the model

After preparing the data, you use gradient-based optimization to train the XGBoost model on the data. The trained model will predict whether a customer will enroll for a certificate of deposit (CD).

3.1.4 Deploy the model

Once the model is trained, you deploy it to a SageMaker endpoint, making it accessible for predictions.

3.1.5 Evaluate model performance

The tutorial evaluates the model's performance using a confusion matrix, comparing actual and predicted values.

3.1.6 Clean up

Finally, you clean up by deleting the endpoint and S3 bucket used in the tutorial.

The tutorial also provides code snippets and explanations for each step. The cost is less than $1, making it AWS Free Tier eligible.

[ ]:

```

```