Foundations of Data Science Project

Heart Disease Prediction

D P Ashutosh - AM.EN.U4CSE20222

N R Niranjan - AM.EN.U4CSE20249

R Adithya Kumar - AM.EN.U4CSE20258

V G Uday Krishnan - -AM.EN.U4CSE20273

Vishaal S Upot - AM.EN.U4CSE20276

Heart Disease Data Columns

1. age - age in years
2. sex - (1 = male; 0 = female)
3. cp - chest pain type
4. trestbps - resting blood pressure
5. chol - cholestrol
6. fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7. restecg - resting electrocardiographic results
8. thalach - maximum heart rate achieved
9. exang - exercise induced angina (1 = yes; 0 = no)
10. oldpeak - ST depression induced by exercise relative to rest
11. slope - the slope of the peak exercise ST segment
12. ca - number of major vessels (0-3) colored by flourosopy
13. thal - thalium stress result
14. target - have disease or not (1=yes, 0=no)

Dataset Link:

https://drive.google.com/drive/folders/1GCz9MLBu4Sx1_l_8kPkb8RHe02I4wlXM

Data Preprocessing

```
#Regular EDA and plotting libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import time
import seaborn as sns

#for our plots to appear in notebook
```

```
%matplotlib inline

# Models
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier

# Model evaluators
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import plot_roc_curve, accuracy_score
```

Data Collection

```
heart_disease = pd.read_csv("heart-disease.csv")
heart_disease.head()
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
slope  \
0   63    1   3       145   233    1        0      150      0      2.3
0
1   37    1   2       130   250    0        1      187      0      3.5
0
2   41    0   1       130   204    0        0      172      0      1.4
2
3   56    1   1       120   236    0        1      178      0      0.8
2
4   57    0   0       120   354    0        1      163      1      0.6
2

   ca  thal  target
0   0     1       1
1   0     2       1
2   0     2       1
3   0     2       1
4   0     2       1
```

Checking Missing Data

```
heart_disease.isnull()
```

```
       age    sex     cp  trestbps   chol    fbs  restecg  thalach
exang  \
0    False  False  False     False  False  False    False    False
False
1    False  False  False     False  False  False    False    False
False
2    False  False  False     False  False  False    False    False
False
3    False  False  False     False  False  False    False    False
```

```
False
4    False  False  False     False  False  False    False    False
False
..     ...    ...    ...       ...    ...    ...      ...      ...
...
298  False  False  False     False  False  False    False    False
False
299  False  False  False     False  False  False    False    False
False
300  False  False  False     False  False  False    False    False
False
301  False  False  False     False  False  False    False    False
False
302  False  False  False     False  False  False    False    False
False

     oldpeak  slope     ca   thal  target
0      False  False  False  False   False
1      False  False  False  False   False
2      False  False  False  False   False
3      False  False  False  False   False
4      False  False  False  False   False
..       ...    ...    ...    ...     ...
298    False  False  False  False   False
299    False  False  False  False   False
300    False  False  False  False   False
301    False  False  False  False   False
302    False  False  False  False   False

[303 rows x 14 columns]
```

The dataset does not contain any missing data.

Understanding the Data

```
heart_disease.head()
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
slope  \
0   63    1   3       145   233    1        0      150      0      2.3
0
1   37    1   2       130   250    0        1      187      0      3.5
0
2   41    0   1       130   204    0        0      172      0      1.4
2
3   56    1   1       120   236    0        1      178      0      0.8
2
4   57    0   0       120   354    0        1      163      1      0.6
2

   ca  thal  target
```

```
0   0     1       1
1   0     2       1
2   0     2       1
3   0     2       1
4   0     2       1
```

```
heart_disease.dtypes
```

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

There are no missing values and all of our columns are numerical in nature.

```
heart_disease.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Data Summarization

```
heart_disease.describe()
```

```
              age         sex          cp    trestbps        chol
fbs  \
count  303.000000  303.000000  303.000000  303.000000  303.000000
303.000000
mean    54.366337    0.683168    0.966997  131.623762  246.264026
0.148515
std      9.082101    0.466011    1.032052   17.538143   51.830751
0.356198
min     29.000000    0.000000    0.000000   94.000000  126.000000
0.000000
25%     47.500000    0.000000    0.000000  120.000000  211.000000
0.000000
50%     55.000000    1.000000    1.000000  130.000000  240.000000
0.000000
75%     61.000000    1.000000    2.000000  140.000000  274.500000
0.000000
max     77.000000    1.000000    3.000000  200.000000  564.000000
1.000000

          restecg     thalach       exang     oldpeak       slope
ca  \
count  303.000000  303.000000  303.000000  303.000000  303.000000
303.000000
mean     0.528053  149.646865    0.326733    1.039604    1.399340
0.729373
std      0.525860   22.905161    0.469794    1.161075    0.616226
1.022606
min      0.000000   71.000000    0.000000    0.000000    0.000000
0.000000
25%      0.000000  133.500000    0.000000    0.000000    1.000000
0.000000
50%      1.000000  153.000000    0.000000    0.800000    1.000000
0.000000
75%      1.000000  166.000000    1.000000    1.600000    2.000000
1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000
4.000000

             thal      target
count  303.000000  303.000000
mean     2.313531    0.544554
std      0.612277    0.498835
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

Correlation between independent features of the dataset

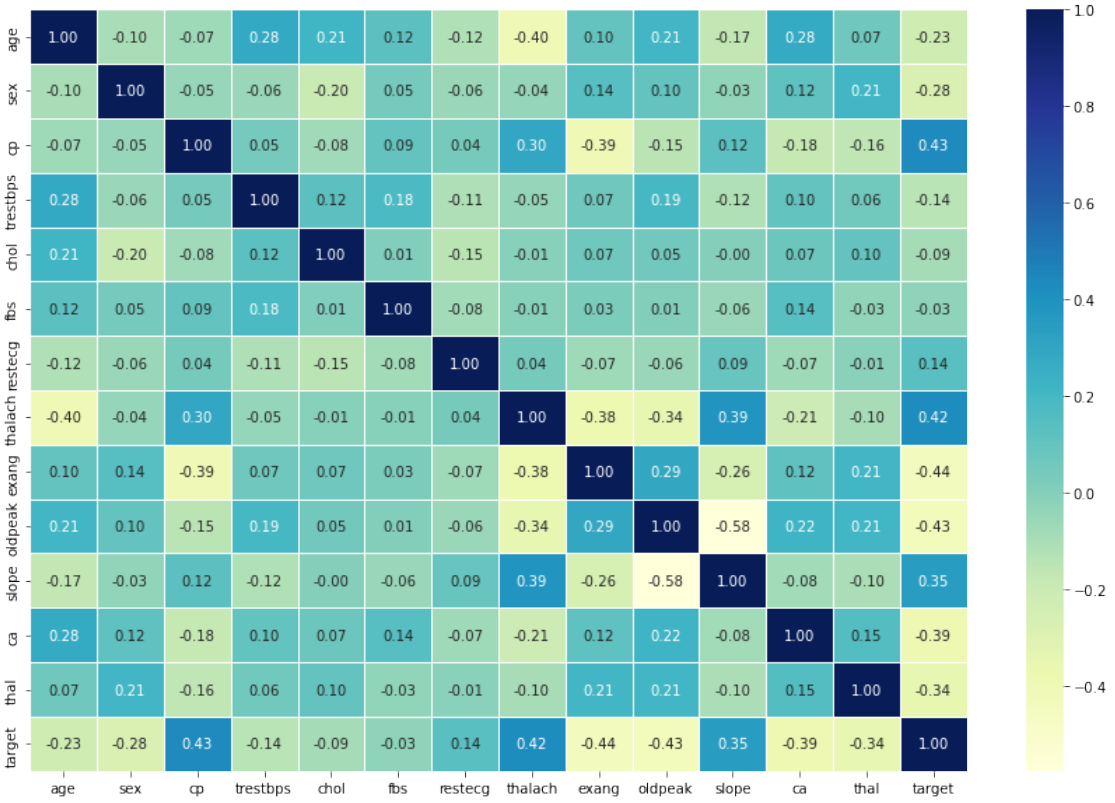
```
corr_matrix = heart_disease.corr()
corr_matrix
```

```
               age       sex        cp  trestbps      chol
fbs  \
age       1.000000 -0.098447 -0.068653  0.279351  0.213678  0.121308

sex      -0.098447  1.000000 -0.049353 -0.056769 -0.197912  0.045032

cp       -0.068653 -0.049353  1.000000  0.047608 -0.076904  0.094444

trestbps  0.279351 -0.056769  0.047608  1.000000  0.123174  0.177531

chol      0.213678 -0.197912 -0.076904  0.123174  1.000000  0.013294

fbs       0.121308  0.045032  0.094444  0.177531  0.013294  1.000000

restecg  -0.116211 -0.058196  0.044421 -0.114103 -0.151040 -0.084189

thalach  -0.398522 -0.044020  0.295762 -0.046698 -0.009940 -0.008567

exang     0.096801  0.141664 -0.394280  0.067616  0.067023  0.025665

oldpeak   0.210013  0.096093 -0.149230  0.193216  0.053952  0.005747

slope    -0.168814 -0.030711  0.119717 -0.121475 -0.004038 -0.059894

ca        0.276326  0.118261 -0.181053  0.101389  0.070511  0.137979

thal      0.068001  0.210041 -0.161736  0.062210  0.098803 -0.032019

target   -0.225439 -0.280937  0.433798 -0.144931 -0.085239 -0.028046


            restecg   thalach     exang   oldpeak     slope
ca  \
age      -0.116211 -0.398522  0.096801  0.210013 -0.168814  0.276326

sex      -0.058196 -0.044020  0.141664  0.096093 -0.030711  0.118261

cp        0.044421  0.295762 -0.394280 -0.149230  0.119717 -0.181053

trestbps -0.114103 -0.046698  0.067616  0.193216 -0.121475  0.101389

chol     -0.151040 -0.009940  0.067023  0.053952 -0.004038  0.070511

fbs      -0.084189 -0.008567  0.025665  0.005747 -0.059894  0.137979
```

```
restecg   1.000000  0.044123 -0.070733 -0.058770  0.093045 -0.072042

thalach   0.044123  1.000000 -0.378812 -0.344187  0.386784 -0.213177

exang    -0.070733 -0.378812  1.000000  0.288223 -0.257748  0.115739

oldpeak  -0.058770 -0.344187  0.288223  1.000000 -0.577537  0.222682

slope     0.093045  0.386784 -0.257748 -0.577537  1.000000 -0.080155

ca       -0.072042 -0.213177  0.115739  0.222682 -0.080155  1.000000

thal     -0.011981 -0.096439  0.206754  0.210244 -0.104764  0.151832

target    0.137230  0.421741 -0.436757 -0.430696  0.345877 -0.391724
```

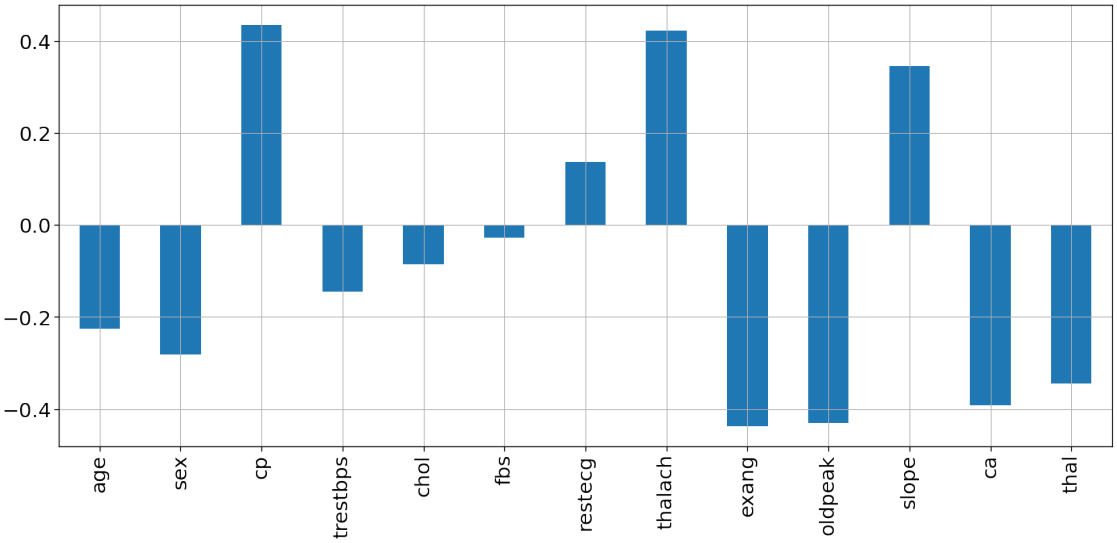
```
              thal    target
age       0.068001 -0.225439
sex       0.210041 -0.280937
cp       -0.161736  0.433798
trestbps  0.062210 -0.144931
chol      0.098803 -0.085239
fbs      -0.032019 -0.028046
restecg  -0.011981  0.137230
thalach  -0.096439  0.421741
exang     0.206754 -0.436757
oldpeak   0.210244 -0.430696
slope    -0.104764  0.345877
ca        0.151832 -0.391724
thal      1.000000 -0.344029
target   -0.344029  1.000000
```

```
plt.figure(figsize=(15, 10))
corr_matrix = heart_disease.corr()
sns.heatmap(corr_matrix,
            annot=True,
            linewidths=0.5,
            fmt= ".2f",
            cmap="YlGnBu");
```



Correlation with the target feature

```
heart = pd.read_csv("heart-disease.csv")
sns.set_context('notebook',font_scale = 2.3)
h=heart.drop('target', axis=1)
h.corrwith(heart['target']).plot(kind='bar', grid=True, figsize=(20,
10))
plt.tight_layout()
```

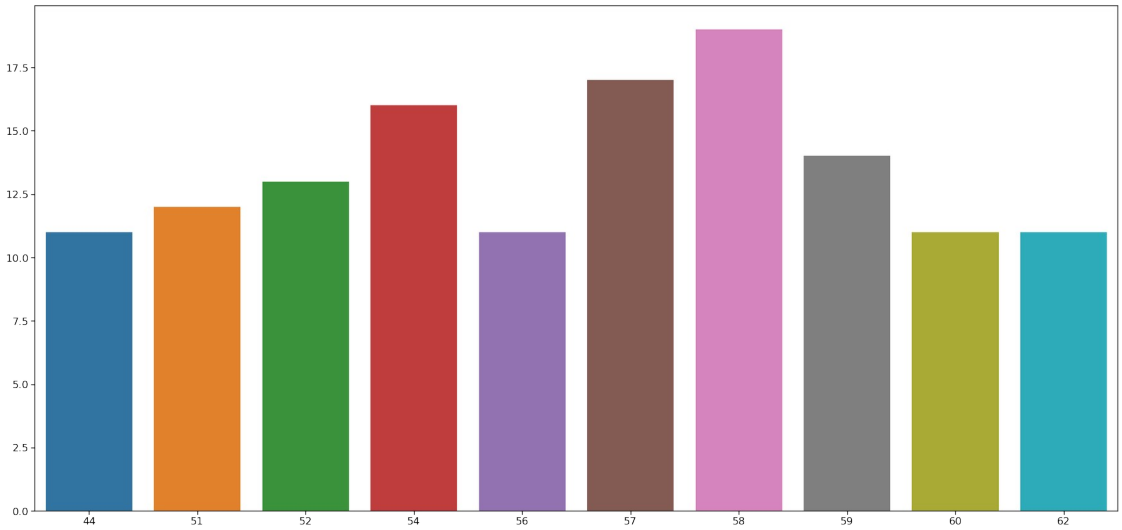


The target feature has negative correlation with attributes like “age”, “sex”, “trestbps”, “chol”, “fbs”, “exang”, “oldpeak”, “ca”, “thal”. All other features have a positive correlation with the target value.

Data Visualizaton

From the graph below we can understand that age 58 has the highest frequency. From this we can understandthat inthis given dataset the patients near to age 58 are more likely to appear in this study

```
plt.figure(figsize=(25,12))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=heart_disease['age'].value_counts()
[:10].index,y=heart_disease['age'].value_counts()[:10].values)
plt.tight_layout()
```



CHECKING THE RANGE OF AGES OF THE PATIENTS:

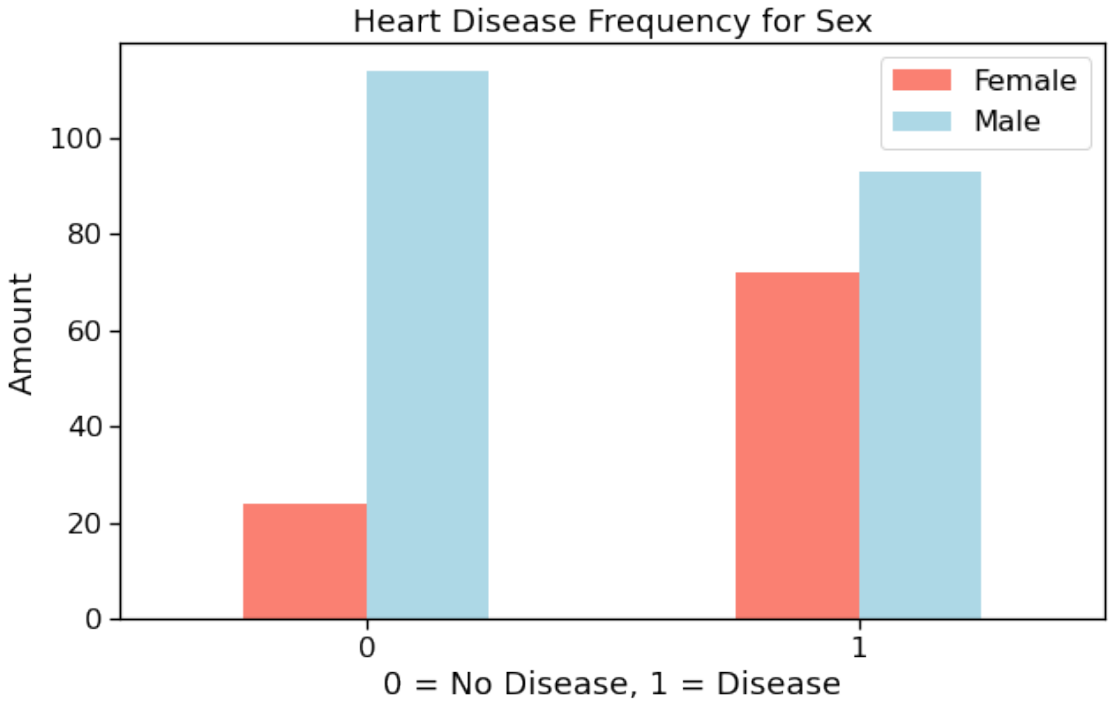
```
minage=min(heart_disease['age'])
maxage=max(heart_disease['age'])
meanage=heart_disease['age'].mean()
print('Minimum Age :',minage)
print('Maximum Age :',maxage)
print('Mean Age :',meanage)
```

```
Minimum Age : 29
Maximum Age : 77
Mean Age : 54.366336633663366
```

PLOTTING THE RELATION B/W SEX AND SICK PATIENTS:

```
pd.crosstab(heart_disease.target, heart_disease.sex).plot(kind="bar",
figsize=(10,6), color=["salmon", "lightblue"])
plt.title("Heart Disease Frequency for Sex")
plt.xlabel("0 = No Disease, 1 = Disease")
```

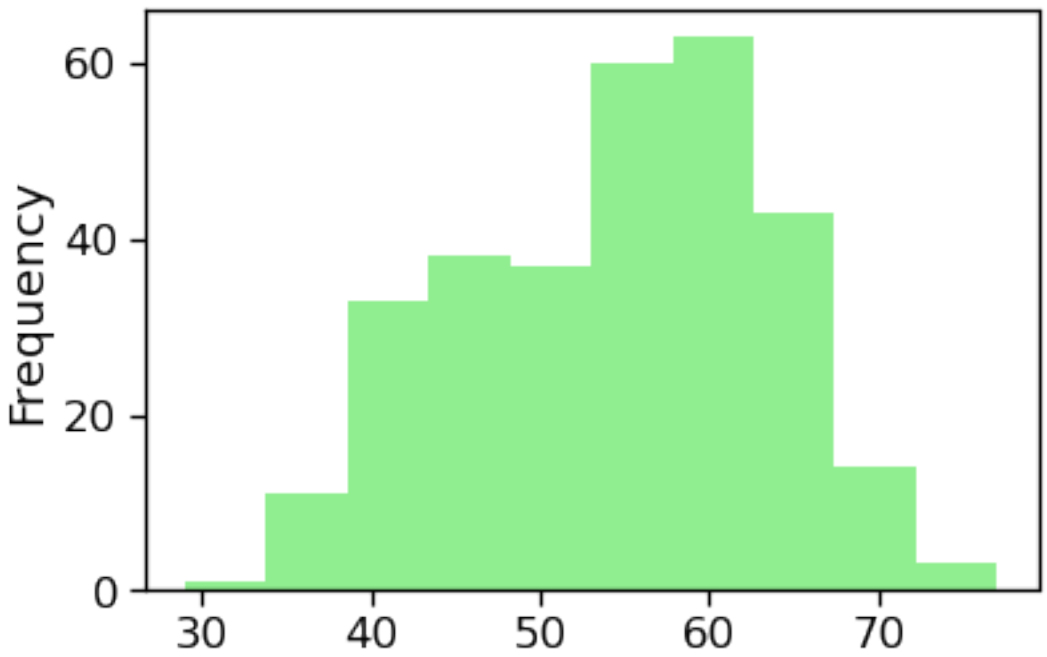
```
plt.ylabel("Amount")
plt.legend(["Female", "Male"])
plt.xticks(rotation=0);
```



Here, we can see the number of females with no disease is low. This shows that in this dataset the proportionofwomen who have heart disease is much higher than that of men. But the number of men who have heart diseaseismore because there are more men in this survey than women.

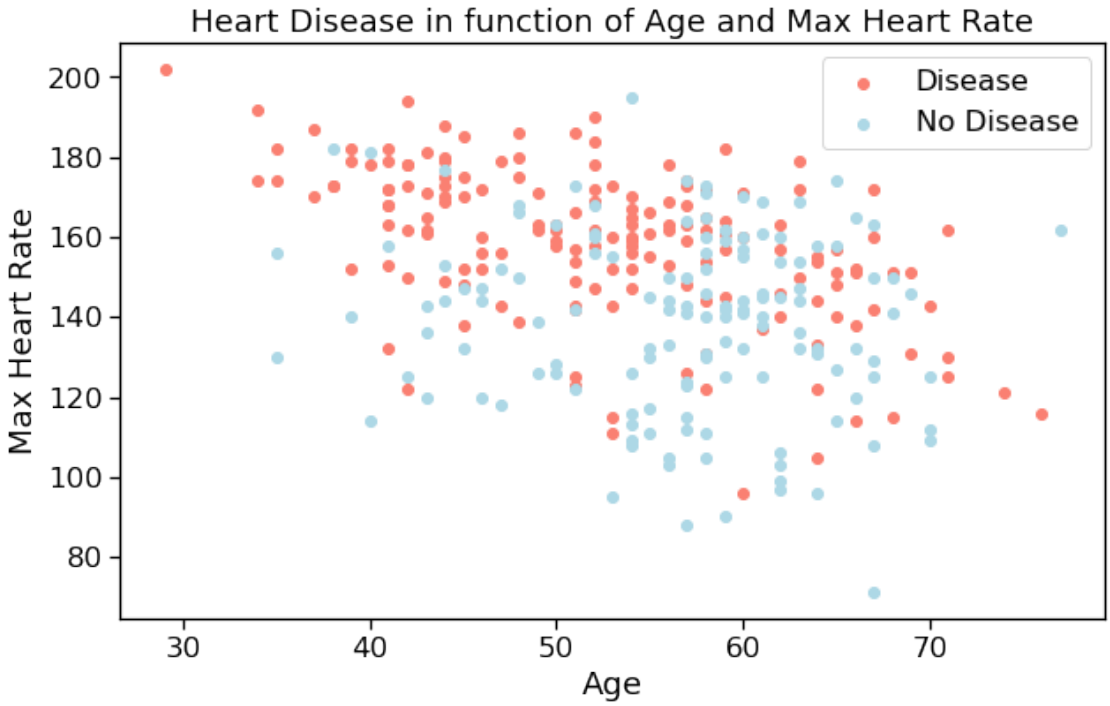
PLOTTING A HISTOGRAM FOR AGE:

```
heart_disease.age.plot.hist(color='lightgreen');
```



PLOTTING A SCATTERPLOT WITH MAX HEART RATE AND AGE:

```
plt.figure(figsize=(10,6))
plt.scatter(heart_disease.age[heart_disease.target==1],
            heart_disease.thalach[heart_disease.target==1],
            c="salmon")
plt.scatter(heart_disease.age[heart_disease.target==0],
            heart_disease.thalach[heart_disease.target==0],
            c="lightblue")
plt.title("Heart Disease in function of Age and Max Heart Rate")
plt.xlabel("Age")
plt.legend(["Disease", "No Disease"])
plt.ylabel("Max Heart Rate");
```

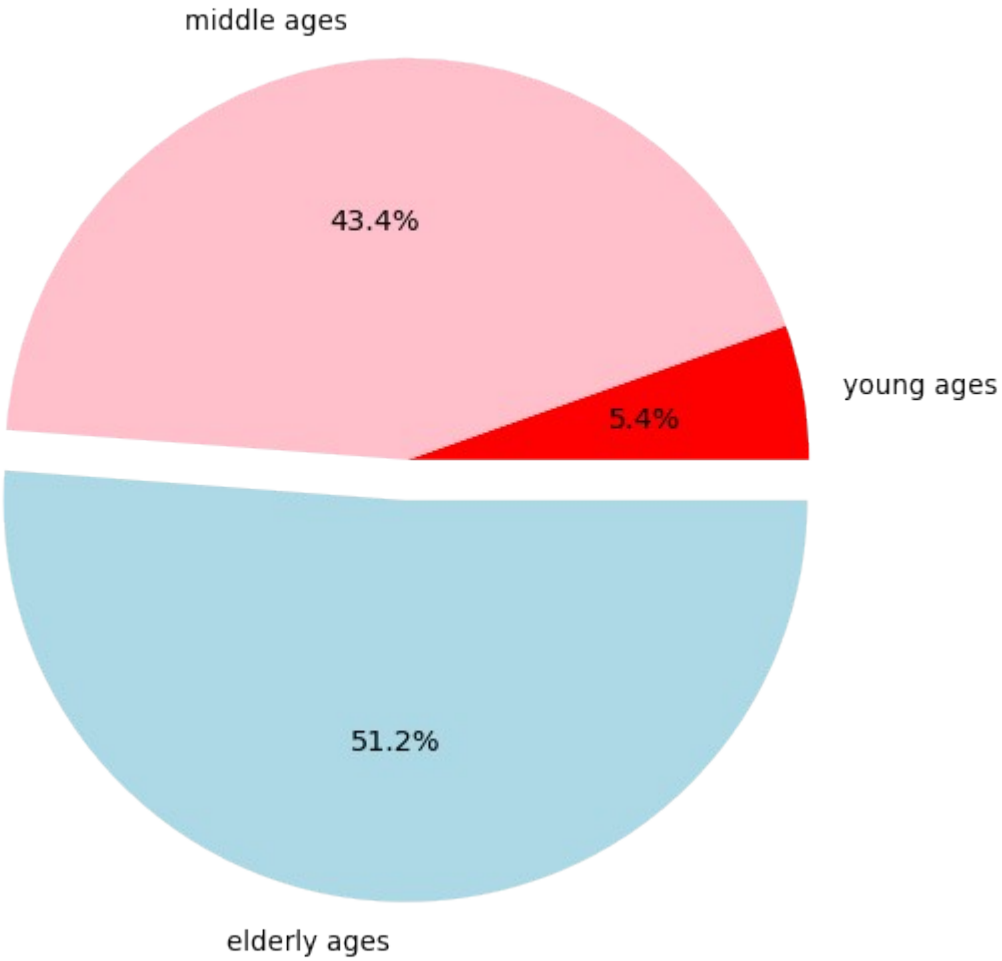


Here, it seems the younger someone is, the higher their max heart rate (dots are higher on the left of the graph) andtheolder someone is, the more green dots there are. But this may be because there are more dots all together ontherightside of the graph (older participants).

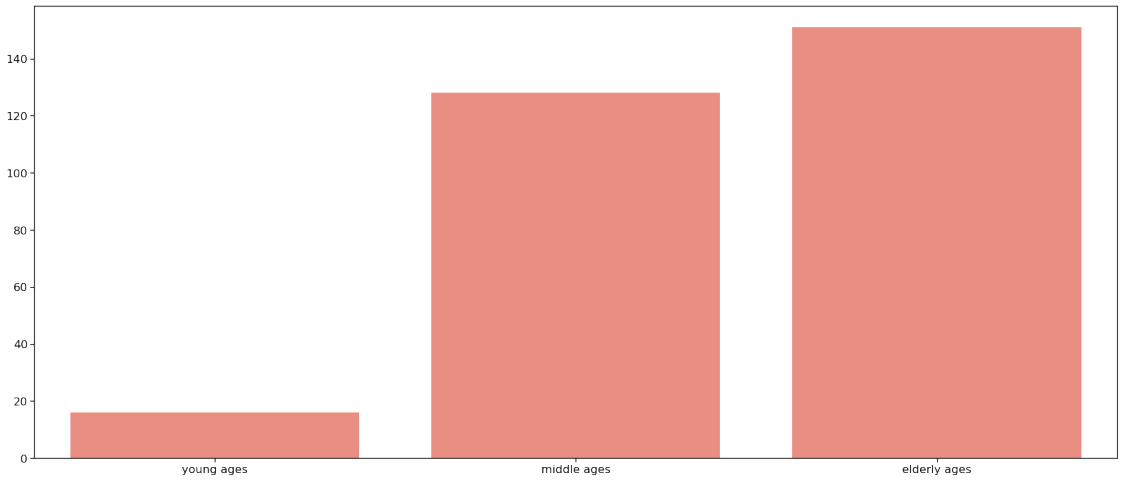
PLOTTING A PIE CHART ON AGE:

Here, 43.4% of the patients are middle aged, 5.4% of the patients are young aged and 51.2%of the patients areoldaged.

```
Young =
heart_disease[(heart_disease['age']>=29)&(heart_disease['age']<40)]
Middle =
heart_disease[(heart_disease['age']>=40)&(heart_disease['age']<55)]
Elder = heart_disease[(heart_disease['age']>55)]
colors = ['red','pink','lightblue']
explode = [0,0,0.1]
plt.figure(figsize=(8,8))
sns.set_context('notebook',font_scale = 1.2)
plt.pie([len(Young),len(Middle),len(Elder)],labels=['young
ages','middle ages','elderly ages'],explode=explode,colors=colors,
autopct='%1.1f%%')
plt.tight_layout()
```

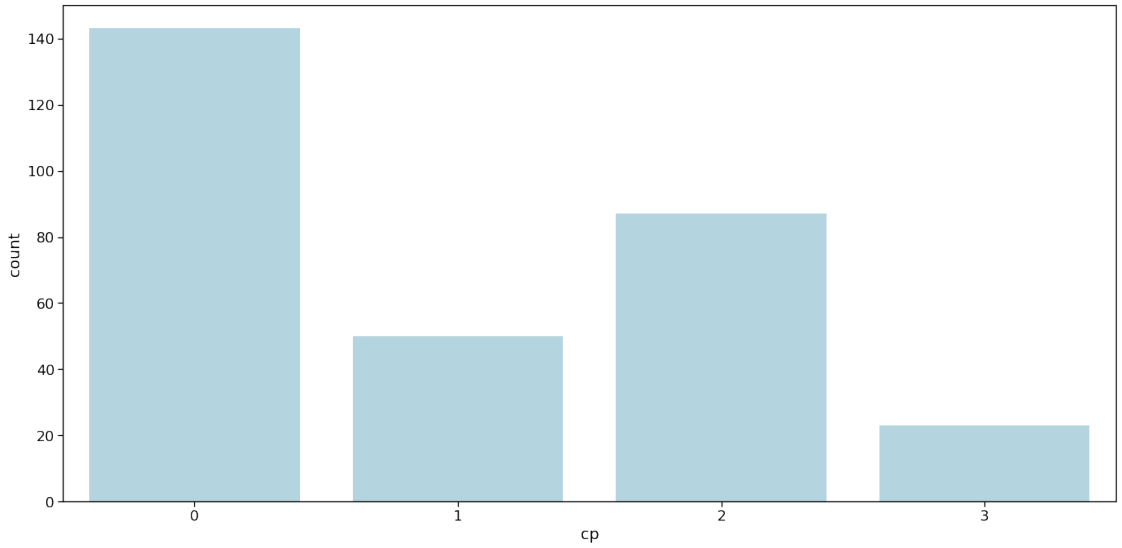



```
plt.figure(figsize=(23,10))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=['young ages','middle ages','elderly
ages'],y=[len(Young),len(Middle),len(Elder)],color='salmon')
plt.tight_layout()
```



PLOTTING A COUNTPLOT ON CHEST PAIN TYPE:

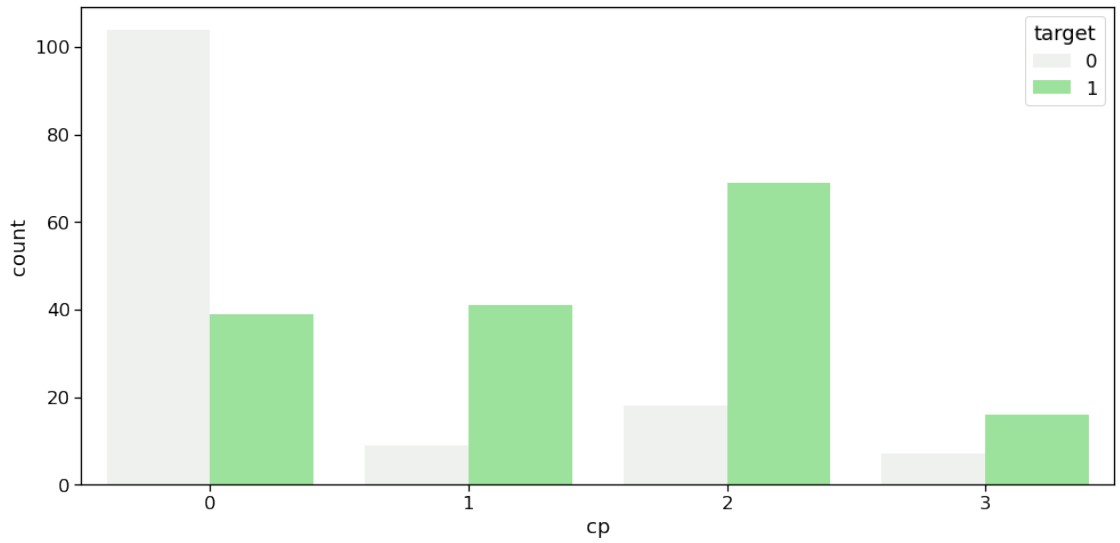
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(heart_disease['cp'], color='lightblue')
plt.tight_layout()
```



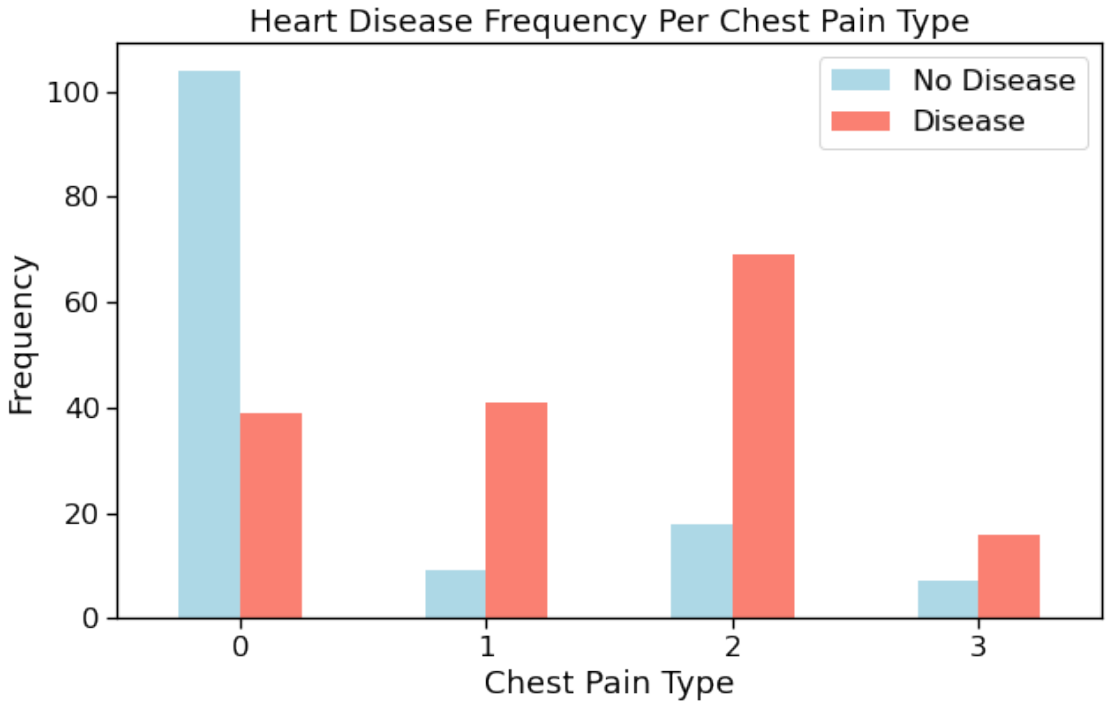
People with the chest pain type - 1 are more likely to come up with a heart disease compared to all other types. This is also brought out in the graph below.

COUNTPLOT SPLIT INTO TARGET VALUES 0 AND 1

```
plt.figure(figsize=(14,7))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(heart_disease['cp'],hue=heart_disease["target"],color='l
ightgreen')
plt.tight_layout()
```

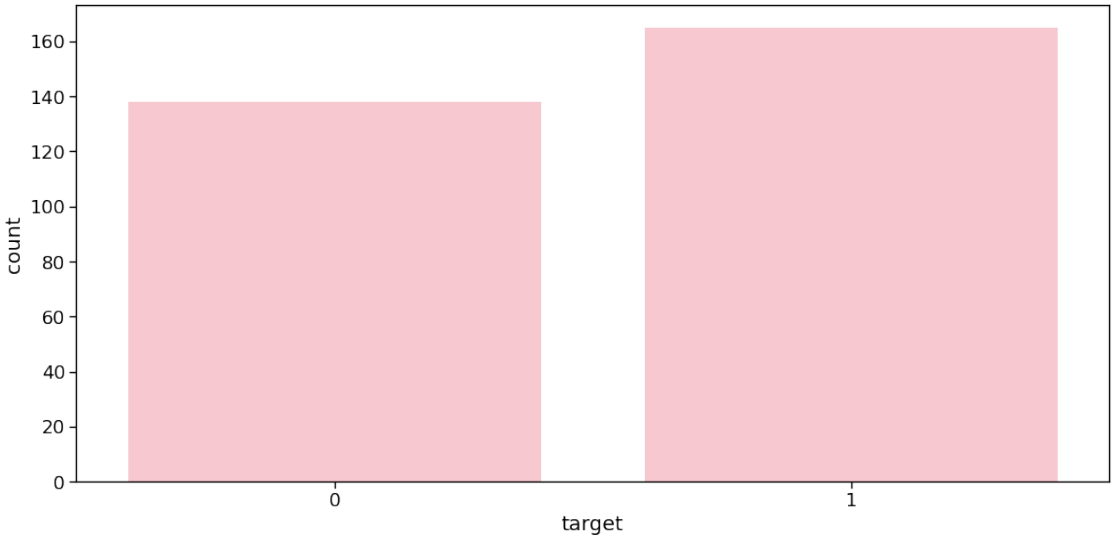



```
# Create a new crosstab and base plot
pd.crosstab(heart_disease.cp, heart_disease.target).plot(kind="bar",
                                                         figsize=(10,6),
                                                         color=["lightblue", "salmon"])

# Add attributes to the plot to make it more readable
plt.title("Heart Disease Frequency Per Chest Pain Type")
plt.xlabel("Chest Pain Type")
plt.ylabel("Frequency")
plt.legend(["No Disease", "Disease"])
plt.xticks(rotation = 0);
```



ANALYING THE TARGET VARIABLE:

```
plt.figure(figsize=(14,7))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(heart_disease['target'],color='pink')
plt.tight_layout()
```



This graph correctly shows that the given dataset is not imbalanced as the ratio b/w 1 and 0 is less than 1.5.

Hence, we can use this data for creating a machine learning model as it is well balancedandcontainsrelevant data for creating a relationship with the causes for a heart disease in patients.

Each relation we have found from the graphs are specified under the graph itself

5. Modeling

```
# Everything except target variable
data=pd.read_csv('heart-disease.csv')
X = data.drop("target", axis=1)

# Target variable
y = data.target.values

## Models
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier

## Model evaluators
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report
```

```
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import plot_roc_curve, accuracy_score

heart_disease.head()
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
slope  \
0   63    1   3       145   233    1        0      150      0      2.3
0
1   37    1   2       130   250    0        1      187      0      3.5
0
2   41    0   1       130   204    0        0      172      0      1.4
2
3   56    1   1       120   236    0        1      178      0      0.8
2
4   57    0   0       120   354    0        1      163      1      0.6
2

   ca  thal  target
0   0     1       1
1   0     2       1
2   0     2       1
3   0     2       1
4   0     2       1
```

```
X.head()
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
slope  \
0   63    1   3       145   233    1        0      150      0      2.3
0
1   37    1   2       130   250    0        1      187      0      3.5
0
2   41    0   1       130   204    0        0      172      0      1.4
2
3   56    1   1       120   236    0        1      178      0      0.8
2
4   57    0   0       120   354    0        1      163      1      0.6
2

   ca  thal
0   0     1
1   0     2
2   0     2
3   0     2
4   0     2
```

```
y
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
```

```
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
dtype=int64)
```

```
from sklearn.model_selection import train_test_split
np.random.seed(42)

# Split into train & test set
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size = 0.2)

X_train.head()
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang
oldpeak  \
132   42    1   1       120   295    0        1      162      0
0.0
202   58    1   0       150   270    0        0      111      1
0.8
196   46    1   2       150   231    0        1      147      0
3.6
75    55    0   1       135   250    0        0      161      0
1.4
176   60    1   0       117   230    1        1      160      1
1.4

     slope  ca  thal
```

```
132      2   0     2
202      2   0     3
196      1   0     2
75       1   0     2
176      2   2     3
```

```
y_train, len(y_train)
```

```
(array([1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
1,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
1,
       0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1,
0,
       0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1,
0,
       1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0,
1,
       1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1,
1,
       1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1,
0,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1,
1,
       1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0,
1,
       1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0,
1],
      dtype=int64),
 242)
```

```
X_test.head()
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang
oldpeak  \
179   57    1   0       150   276    0        0      112      1
0.6
228   59    1   3       170   288    0        0      159      0
0.2
111   57    1   2       150   126    1        1      173      0
0.2
246   56    0   0       134   409    0        0      150      1
1.9
60    71    0   2       110   265    1        0      130      0
0.0

     slope  ca  thal
179      1   1     1
228      1   0     3
```

```
111      2   1     3
246      1   2     3
60       2   1     2
```

```
y_test, len(y_test)
```

```
(array([0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1,
0,
        0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1,
        1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0],
dtype=int64),
 61)
```

Creating the models and estimating their accuracy on unseen data usingthespecified ML algorithms

```
models = {"KNN": KNeighborsClassifier(n_neighbors=5),
          "Logistic Regression": LogisticRegression(),
          "Random Forest": RandomForestClassifier()}

def fit_and_score(models, X_train, X_test, y_train, y_test):


    np.random.seed(42)
    # Make a list to keep model scores
    model_scores = {}
    # Loop through models
    for name, model in models.items():
        # Fit the model to the data
        model.fit(X_train, y_train)
        # Evaluate the model and append its score to model_scores
        model_scores[name] = model.score(X_test, y_test)

#         print("Predicted by ",model)
#         print(accuracy_score(y_test, model.predict(X_test)))
#         print(" ")

    return model_scores

model_scores = fit_and_score(models=models,
                             X_train=X_train,
                             X_test=X_test,
                             y_train=y_train,
                             y_test=y_test)


model_scores
```

```
{'KNN': 0.6885245901639344,
 'Logistic Regression': 0.8852459016393442,
 'Random Forest': 0.8360655737704918}
```

Comparing Accuracy of the three models used

```
model_compare = pd.DataFrame(model_scores, index=['accuracy'])
model_compare.T.plot.bar();
```