SWE485: Selected Topics in Software Engineering
Software Engineering Department
King Saud University

Due. **April. 26th**
3rd term 1444
project phase 1

Heart Attack Analysis & Prediction

Group #1 | section 60120

Group members:

Student name	ID
Fai Alotaibi	**441201811**
Lina nasser	**441204555**
Reem alshareef	**441200087**

Table of content:

1- Introduction:

The Heart Attack Analysis & Prediction dataset is a collection of data related to the risk factors for heart attacks. The dataset was compiled by Kaggle and consists of 303 observations with 14 variables, including demographic information such as age and gender, as well as medical information such as blood pressure and cholesterol level. The goal of the dataset is to predict the likelihood of a person having a heart attack based on various risk factors. The dataset is available for download on Kaggle and can be used for exploratory data analysis, predictive modeling, and other data science tasks.

2- Machine learning tasks

In this project we have a main goal is to characterize patients as a prime target for heart attacks or not. To predict and do this we will use 2 types of machine learning :

- Supervised learning
- Unsupervised learning

For supervised learning we will use 2 algorithms (linear regression and decision tree).
We will use Linear regression which is regression method in heart attacks target diagnosis to make predictions about the patients if they are a prime target for heart attacks or not. We can do this by applying some of statistical processes for estimates the relationship between target which is dependent variable and their independent variables which are(age, sex, Cp, Tresbps, Chol, Fbs, restecg, thalach, exang, oldpeak, slope, ca, thal).
And we will use decision trees which is classification method. C4.5 is an algorithm that we will use to build our decision tree in this project.
It can be used for classifying the patients if they are a prime target for heart attacks or not. This can be done based on how a previous questions were answered.

For unsupervised learning we will use K-Means which is clustering algorithm finds observations in a dataset that are like each other and places them in a cluster. [1] and each cluster will have a centroid , The inputs of the algorithm are the number of clusters K which will be 2 in our project and our dataset and the outputs our dataset divided into K clusters.

3- Data

Data kind and explanation :

The Heart Attack Analysis & Prediction dataset is a tabular dataset that consists of 303 observations and 14 variables. The dataset is in a comma-separated values (CSV) format and can be loaded into a pandas DataFrame using the read_csv() function.

The variables in the dataset are as follows:

age: age of the individual in years (numeric)

sex: gender of the individual (categorical - 1: male, 0: female)

cp: chest pain type (categorical - 0: typical angina, 1: atypical angina, 2: non-anginal pain, 3: asymptomatic)

trtbps: resting blood pressure (mm Hg) (numeric)

chol: serum cholesterol (mg/dL) (numeric)

fbs: fasting blood sugar > 120 mg/dL (categorical - 1: true, 0: false)

restecg: resting electrocardiographic results (categorical - 0: normal, 1: having ST-T wave abnormality, 2: showing probable or definite left ventricular hypertrophy)

thalachh: maximum heart rate achieved (numeric)

exng: exercise induced angina (categorical - 1: yes, 0: no)

oldpeak: ST depression induced by exercise relative to rest (numeric)

slp: slope of the peak exercise ST segment (categorical - 0: upsloping, 1: flat, 2: downsloping)

caa: number of major vessels (0-3) colored by fluoroscopy (categorical)

thall: thalassemia (categorical - 0: normal, 1: fixed defect, 2: reversible defect)

output: presence of heart attack (categorical - 1: yes, 0: no)

The goal of the dataset is to predict the likelihood of an individual having a heart attack based on their demographic and medical information. The 'output' variable is the target variable, which indicates the presence or absence of a heart attack. The other variables are the features that can be used to build predictive models.

data source :

The Heart Attack Analysis & Prediction dataset was created by a Kaggle user named "pritsheta". The dataset is available on Kaggle, which is an online community of data scientists and machine learning practitioners.

We get the dataset from : https://www.kaggle.com/datasets/pritsheta/heart-attack

To display the data type for each variable in the dataset, we used data.dtypes:

```
In [10]: data.dtypes
         # display the data type ( for each column  )

Out[10]: 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
```

Figure 1 : data type

to display sample rows using head method which retrieves the first 5 indexes:

```
In [22]: data.head()
         #retrive first 5 indexes
```

Out[22]:

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

Figure 2 : row sample

The following figure showsThe number and type of variables :

```
In [71]: data.dtypes
         # display the variable type ( for each column  )
```

```
Out[71]: 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
```

```
In [91]: # Count the number of variables
         num_variables = len(data.columns)

         # Print the number of variables
         print("Number of variables:", num_variables)
```

```
Number of variables: 13
```

Figure 3 : number and type of variable

To display Correlation between Variables using data.corr method :

The following chart shows the relationships between the variables, it shows us that if the colors are light, this means that the relationship between these two variables is very weak, such as the relationship between cp and target, and if the color is dark, this means that there is a strong relationship.

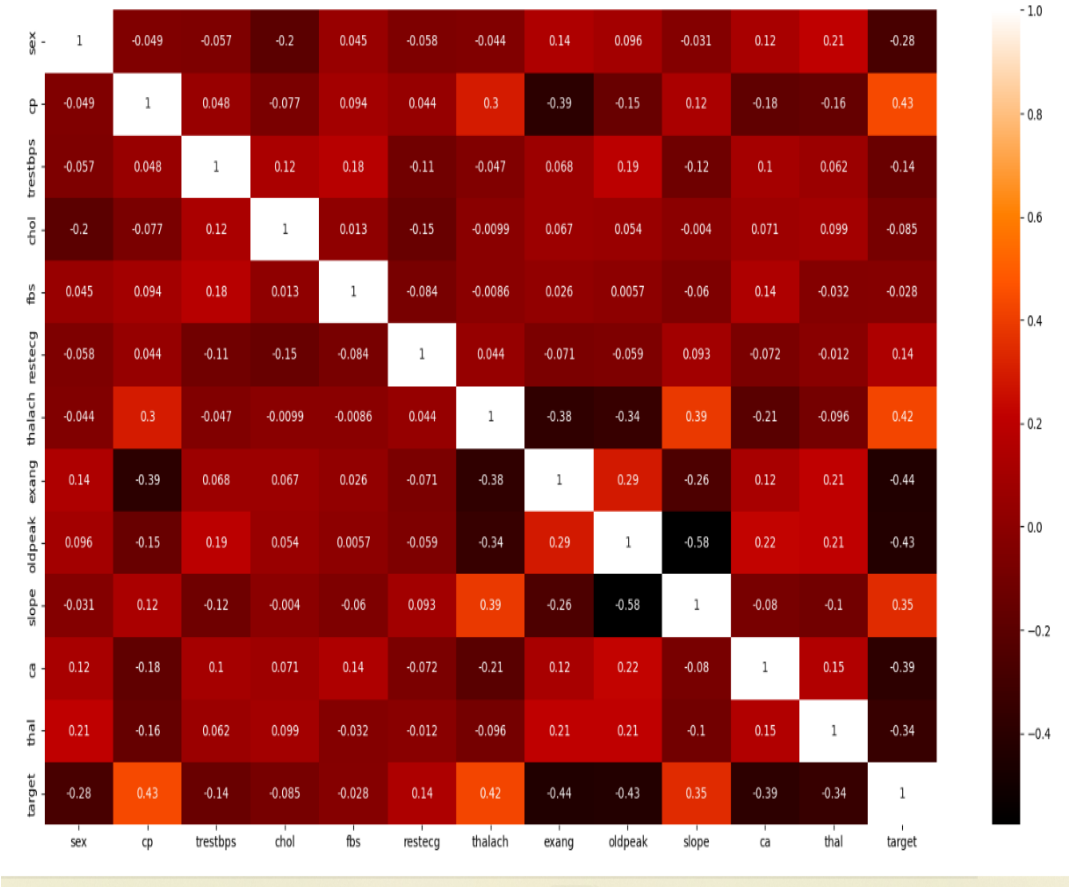

Figure 4: heatmap of data

4- Variables distribution:

4.1Histogram

The following chart in figure 4 displays the distribution of serum cholesterol levels in the Heart Attack Analysis & Prediction dataset using a histogram with a kernel density estimate (KDE) line overlaid on top of the histogram bars. The resulting chart provides insights into the distribution of serum cholesterol levels in the dataset and can be used to identify potential relationships with other variables.

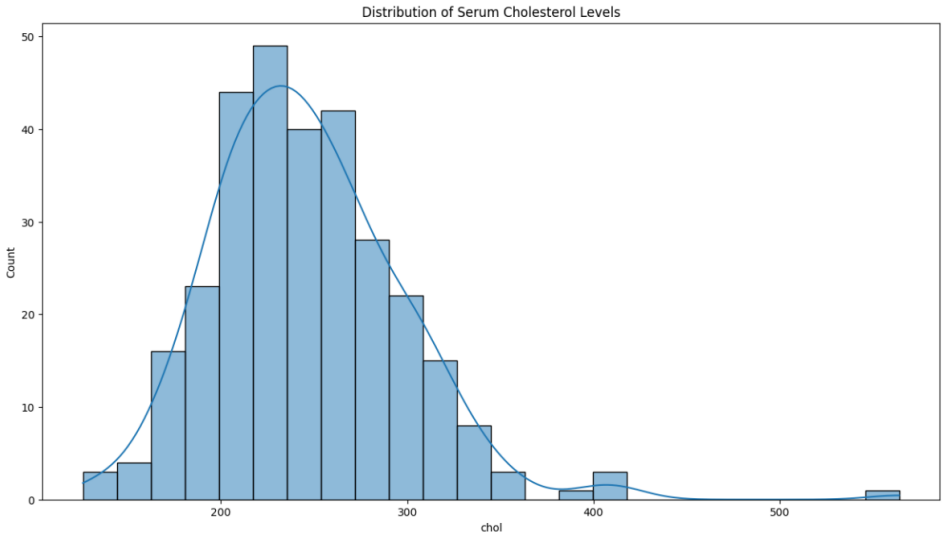

Figure 5 : chol histogram chart

4.2 Bar plot

The following bar plot displays the frequency of chest pain types in the dataset. Chest pain type 0 is the most common type followed by types 2, 1, and 3. This can help identify potential risk factors and develop prevention and treatment strategies for heart disease.

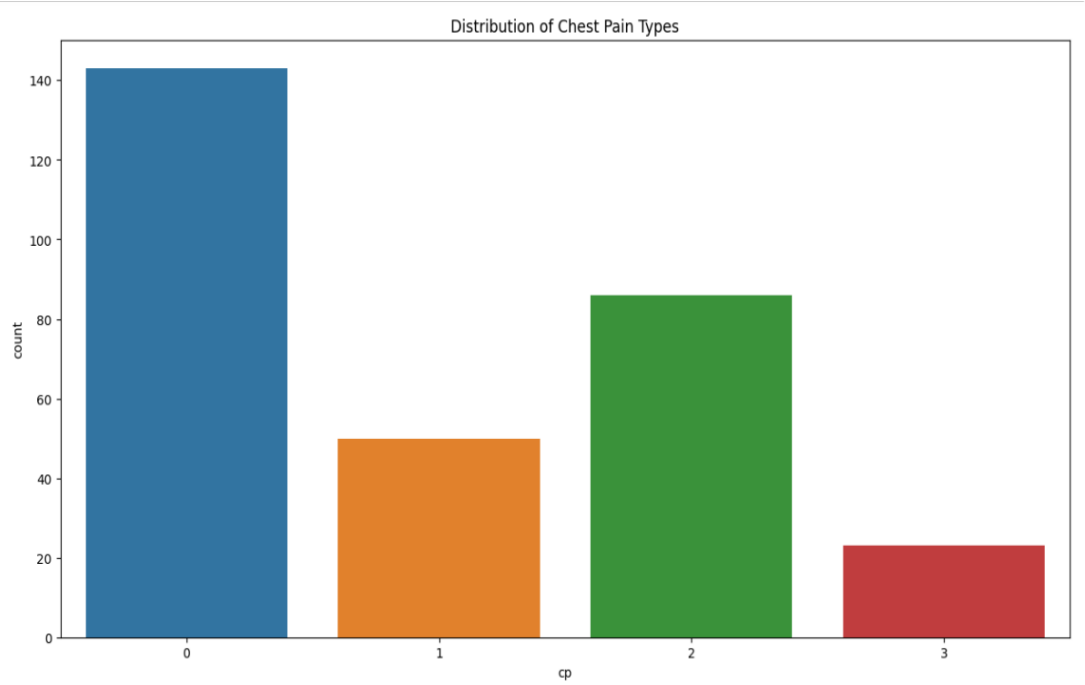

Figure 6: cp bar plot

4.3 Pie chart

The pie chart displays the percentage of males and females in the Heart Attack Analysis & Prediction dataset , (1 for male , 0 for female) Males make up a larger proportion of the dataset, which may indicate a higher likelihood of males experiencing heart attacks or participating in heart attack studies.

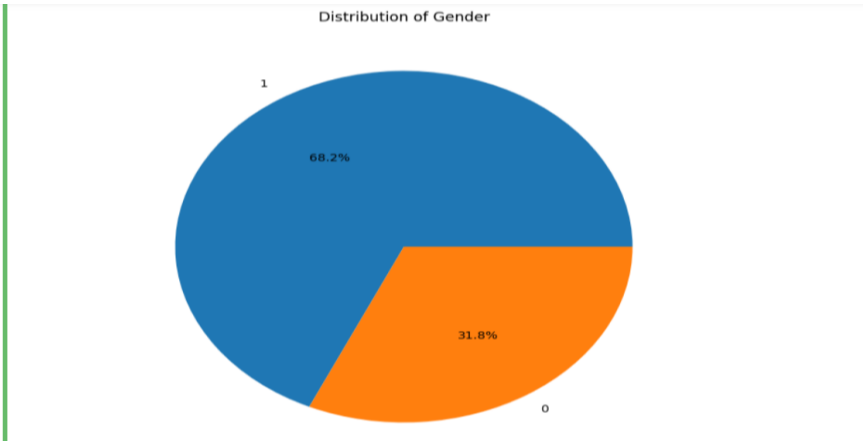

Figure 7 : sex pie chart

(null) Missing values :

To capture any missing values in an array-like object, we can use the isnull() method.

```
In [45]: #show if there are any null values
         data.isnull()
```

Out[45]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	False	False	False	False	False	False	False	False	False	False	False	False	False	False
299	False	False	False	False	False	False	False	False	False	False	False	False	False	False
300	False	False	False	False	False	False	False	False	False	False	False	False	False	False
301	False	False	False	False	False	False	False	False	False	False	False	False	False	False
302	False	False	False	False	False	False	False	False	False	False	False	False	False	False

303 rows × 14 columns

Figure 8: missing values

Statistical summaries:

To calculate the mean, IQR (Interquartile Range), and stf (standard deviation) for the numeric columns in our DataFrame, we can use the .describe() method from pandas library and .var() method to show the variances .

```
In [72]: data.describe()
```

Out[72]:

	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	t
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00
mean	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.54
std	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.49
min	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.00
50%	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.00
75%	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.00
max	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.00

```
In [92]: #to show the variances
variances = data.var()

# Print the variances
print(variances)
```

```
sex            0.217553
cp             1.065114
trestbps     308.472817
chol        2678.423588
fbs            0.127225
restecg        0.276705
thalach      524.571561
exang          0.221084
oldpeak        1.348971
slope          0.379794
ca             1.013542
thal           0.375800
target         0.248971
dtype: float64
```

Figure 9: Statistical summaries

5- Data preprocessing

Data preprocessing it is important step in data analysis to have a best and clean result.

data cleaning

In the data cleaning we can check whether there is a duplication in some rows or not and in the output we found one row duplicated, then we will remove the duplication.

```
In [84]: #The number of duplicated rows
         print(data.duplicated().sum())

         1

In [85]: #the duplicate value
         duplicates = data[data.duplicated()]
         print(duplicates)

              sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope
         age
         38     1   2       138   175    0        1      173      0      0.0      2  \

              ca  thal  target
         age
         38    4     2       1

In [86]: #remove the dublicate
         data=data.drop_duplicates(keep="last")

In [87]: #The number of duplicated now
         print(data.duplicated().sum())

         0

In [65]: #the duplicate value after rwmove the duplicate
         duplicates = data[data.duplicated()]
         print(duplicates)

         Empty DataFrame
         Columns: [sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal, target]
         Index: []
```

Figure 10: Remove duplication row code

Our dataset does not contain any missing values so we did not need to delete missing values or replace it with another value. Here is the justification:

```
In [77]: data.isnull().sum()
         #counting missing value

Out[77]: sex         0
         cp          0
         trestbps    0
         chol        0
         fbs         0
         restecg     0
         thalach     0
         exang       0
         oldpeak     0
         slope       0
         ca          0
         thal        0
         target      0
         dtype: int64

In [78]: #show if there are any null values
         data.isnull()
```

Out[78]:

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
63	False	False	False	False	False	False	False	False	False	False	False	False	False
37	False	False	False	False	False	False	False	False	False	False	False	False	False
41	False	False	False	False	False	False	False	False	False	False	False	False	False
56	False	False	False	False	False	False	False	False	False	False	False	False	False
57	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
57	False	False	False	False	False	False	False	False	False	False	False	False	False
45	False	False	False	False	False	False	False	False	False	False	False	False	False
68	False	False	False	False	False	False	False	False	False	False	False	False	False
57	False	False	False	False	False	False	False	False	False	False	False	False	False
57	False	False	False	False	False	False	False	False	False	False	False	False	False

303 rows × 13 columns

Figure 11 : the number and Existence of null values

Transformation:

In data transformation we will apply minimum maximum normalization to rescale the data

```
In [173]: # Separate the target variable from the features
target = data['oldpeak']
features = data.drop('oldpeak', axis=1)

# Normalize the features using the z-score method
features = (features - features.mean()) / features.std()

# Combine the normalized features with the target variable
normalized_data = pd.concat([features, target], axis=1)

# Print the normalized data
print(normalized_data.head())
```

```
           sex        cp  trestbps      chol  fbs   restecg     exang  \
age
63    0.681525  1.225290  0.762800 -0.175218  NaN -1.000880 -0.697187
37    0.681525  1.225290 -0.091249  0.171195  NaN  0.900163 -0.697187
41   -1.462439  0.123988 -0.091249 -0.828762  NaN -1.000880 -0.697187
56    0.681525  0.123988 -0.660615 -0.112294  NaN  0.900163 -0.697187
57   -1.462439 -0.977315 -0.660615  1.883529  NaN  0.900163  1.429586

        slope        ca      thal    target   oldpeak
age
63  -2.267418 -0.713727 -2.144396  0.915793  1.193922
37  -2.267418 -0.713727 -0.513143  0.915793  1.504077
41   0.977891 -0.713727 -0.513143  0.915793  0.875469
56   0.977891 -0.713727 -0.513143  0.915793  0.587787
57   0.977891 -0.713727 -0.513143  0.915793  0.470004
```

```
In [174]: # Print the preprocessed data
print(data.head())
```

```
     sex   cp  trestbps      chol  fbs  restecg  exang   oldpeak  slope  ca
age
63     1  2.0       145  5.455321  0.0        0      0  1.193922      0   0  \
37     1  2.0       130  5.525453 -inf        1      0  1.504077      0   0
41     0  1.0       130  5.323010 -inf        0      0  0.875469      2   0
56     1  1.0       120  5.468060 -inf        1      0  0.587787      2   0
57     0  0.0       120  5.872118 -inf        1      1  0.470004      2   0

     thal  target
age
63      1       1
37      2       1
41      2       1
56      2       1
57      2       1
```

Figure 13 : normalization code

References:

[1] J. Frost, “What is K means clustering? with an example,” Statistics By Jim, 31-Aug-2022. [Online]. Available: https://statisticsbyjim.com/basics/k-means-clustering/#:~:text=The%20K%20Means%20Clustering%20algorithm%20finds%20observations%20in%20a%20dataset,the%20center%20of%20the%20group. [Accessed: 26-Apr-2023].

[2] Prit Sheta, "Heart Attack Analysis & Prediction Dataset," Kaggle, Dec. 20, 2020. [Online]. Available: https://www.kaggle.com/datasets/pritsheta/heart-attack?resource=download. [Accessed: Month day, year].

[3] David Venturi, "Jupyter Notebook Tutorial: The Definitive Guide," Dataquest, Jan. 19, 2021. [Online]. Available: https://www.dataquest.io/blog/jupyter-notebook-tutorial/. [Accessed: Month day, year].