Introduction to Data Science

Project report for the partial fulfillment
of the requirements of the course

submitted by

Nitish Kumar Gupta - 19UCC131
Sanyam Lodha - 19UCS258
Tanmay Lodha - 19UCS221
Aryan Seth - 19UCC132

submitted to
Dr. Aloke Datta

Department of Computer Science and Engineering, The LNM Institute of Information Technology, Jaipur.

December 2021

Description of the dataset

Cardiovascular diseases (CVDs) or heart disease are the number one cause of death globally with 17.9 million death cases each year. CVDs are concertedly contributed by hypertension, diabetes, overweight and unhealthy lifestyles. This project aims to visualize the factors on which the possibility of a person having a heart disease depends.

This is a heart disease dataset. Initially, the dataset contains 76 features from 303 patients from Cleveland. However, published studies chose only 14 features that are relevant in predicting heart disease. Hence, here we will be using the dataset consisting of 14 features from 303 patients.

This is a multivariate type of dataset which means providing or involving a variety of separate mathematical or statistical variables, multivariate numerical data analysis. It is composed of 14 attributes which are age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise induced angina, oldpeak — ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment, number of major vessels and Thalassemia. This database includes 76 attributes, but all published studies relate to the use of a subset of 14 of them. The Cleveland database is the only one used by ML researchers to date. One of the major tasks on this dataset is to predict based on the given attributes of a patient whether that particular person has a heart disease or not and another is the experimental task to diagnose and find out various insights from this dataset which could help in understanding the problem more.

- Data Set Characteristics: Multivariate
- Number of Instances: 303
- Area: Life
- Attribute Characteristics: Categorical, Integer, Real
- Number of Attributes: 76
- Date Donated: 1988-07-01
- Associated Tasks: Classification
- Missing Values: Yes

Source:

Creators:

1. Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
2. University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
3. University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.
4. V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D.

Donor:

David W. Aha (aha '@' ics.uci.edu) (714) 856-8779

Attribute Information:

1. **age** : age in years
2. **sex** : (1 = male; 0 = female)
3. **cp** : chest pain type
    a. Value 1: typical angina
    b. Value 2: atypical angina
    c. Value 3: non-anginal pain
    d. Value 4: asymptomatic
4. **trestbps**: resting blood pressure (in mm Hg on admission to the hospital)
5. **chol**: serum cholesterol in mg/dl
6. **fbs**: (fasting blood sugar > 120 mg/dl), (1 = true; 0 = false)
7. **restecg**: resting electrocardiographic results
    a. Value 0: normal
    b. Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
    c. Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
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
    a. Value 1: upsloping
    b. Value 2: flat
    c. Value 3: downsloping

12. **ca**: number of major vessels (0-3) colored by fluoroscopy
13. **thal**: 3 = normal; 6 = fixed defect; 7 = reversible defect
14. **num**: diagnosis of heart disease (angiographic disease status)
    a. 0 : No disease
    b. 1 : Threat level 1
    c. 2 : Threat level 2
    d. 3 : Threat level 3
    e. 4 : Threat level 4

Source of our dataset: https://archive.ics.uci.edu/ml/datasets/Heart+Disease

The outline of this project:

1. Import and get to know the data
2. Data Cleaning
    a. Check the data type
    b. Check for the data characters mistakes
    c. Check for missing values and replace them
    d. Check for duplicate rows
    e. Statistics summary
3. Data Visualization

Data Cleaning

- First, we import all the necessary libraries which will be required in our code.

Importing required libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sb
```

- Next, we read our dataset in a variable and output it.

Collect Cleveland data

In [2]:
```
data = pd.read_csv('processed.cleveland.data',names=['age','sex','cp','trestbps','chol','fbs',
                                                     'restecg','thalach','exang','oldpeak','slope',
                                                     'ca','thal','num'])
data
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	45.0	1.0	1.0	110.0	264.0	0.0	0.0	132.0	0.0	1.2	2.0	0.0	7.0	1
299	68.0	1.0	4.0	144.0	193.0	1.0	0.0	141.0	0.0	3.4	2.0	2.0	7.0	2
300	57.0	1.0	4.0	130.0	131.0	0.0	0.0	115.0	1.0	1.2	2.0	1.0	7.0	3
301	57.0	0.0	2.0	130.0	236.0	0.0	2.0	174.0	0.0	0.0	2.0	1.0	3.0	1
302	38.0	1.0	3.0	138.0	175.0	0.0	0.0	173.0	0.0	0.0	1.0	?	3.0	0

303 rows × 14 columns

- Using the info() method, we find information about our dataset.

```
In [3]: data.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    float64
 1   sex       303 non-null    float64
 2   cp        303 non-null    float64
 3   trestbps  303 non-null    float64
 4   chol      303 non-null    float64
 5   fbs       303 non-null    float64
 6   restecg   303 non-null    float64
 7   thalach   303 non-null    float64
 8   exang     303 non-null    float64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    float64
 11  ca        303 non-null    object 
 12  thal      303 non-null    object 
 13  num       303 non-null    int64  
dtypes: float64(11), int64(1), object(2)
memory usage: 33.3+ KB
```

- Changing the datatype of some attributes for better representation. To do that we first need to handle missing values.

```
In [4]: columns = data.columns.tolist()
        for column in columns:
            if '?' in data[column].value_counts().keys():
                print(data[column].value_counts())
```

```
0.0    176
1.0     65
2.0     38
3.0     20
?        4
Name: ca, dtype: int64
3.0    166
7.0    117
6.0     18
?        2
Name: thal, dtype: int64
```

As we can see, ‘ca’ and ‘thal’ have 4 and 2 missing values respectively.

- We filled them with the mode of the respective features.

```
In [5]: data['ca'].replace('?',data['ca'].mode()[0],inplace=True)
        data['thal'].replace('?',data['thal'].mode()[0],inplace=True)
```

- Now changing the necessary data types for better representation.

```
In [6]: data['age'] = data['age'].astype('int64')
        data['sex'] = data['sex'].astype('int64')
        data['cp'] = data['cp'].astype('int64')
        data['trestbps'] = data['trestbps'].astype('int64')
        data['chol'] = data['chol'].astype('int64')
        data['fbs'] = data['fbs'].astype('int64')
        data['restecg'] = data['restecg'].astype('int64')
        data['thalach'] = data['thalach'].astype('int64')
        data['exang'] = data['exang'].astype('int64')
        data['slope'] = data['slope'].astype('int64')
        data['ca'] = data['ca'].astype(str).astype('float64').astype('int64')
        data['thal'] = data['thal'].astype(str).astype('float64').astype('int64')
```

- Using the info() method again to see the changes.

```
In [7]: data.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  num       303 non-null    int64
        dtypes: float64(1), int64(13)
        memory usage: 33.3 KB
```

- Checking for duplicates

```
In [8]: data.duplicated().sum()
Out[8]: 0
```

There are no duplicates in the dataset.

- We now used the describe() method to view basic statistical information about our dataset.

```
In [9]: data.describe()
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
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.
mean	54.438944	0.679868	3.158416	131.689769	246.693069	0.148515	0.990099	149.607261	0.326733	1.039604	1.600660	0.663366	4.
std	9.038662	0.467299	0.960126	17.599748	51.776918	0.356198	0.994971	22.875003	0.469794	1.161075	0.616226	0.934375	1.
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000	3.
25%	48.000000	0.000000	3.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	3.
50%	56.000000	1.000000	3.000000	130.000000	241.000000	0.000000	1.000000	153.000000	0.000000	0.800000	2.000000	0.000000	3.
75%	61.000000	1.000000	4.000000	140.000000	275.000000	0.000000	2.000000	166.000000	1.000000	1.600000	2.000000	1.000000	7.
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	3.000000	3.000000	7.

Inference:

In our data, we can observe the following things:

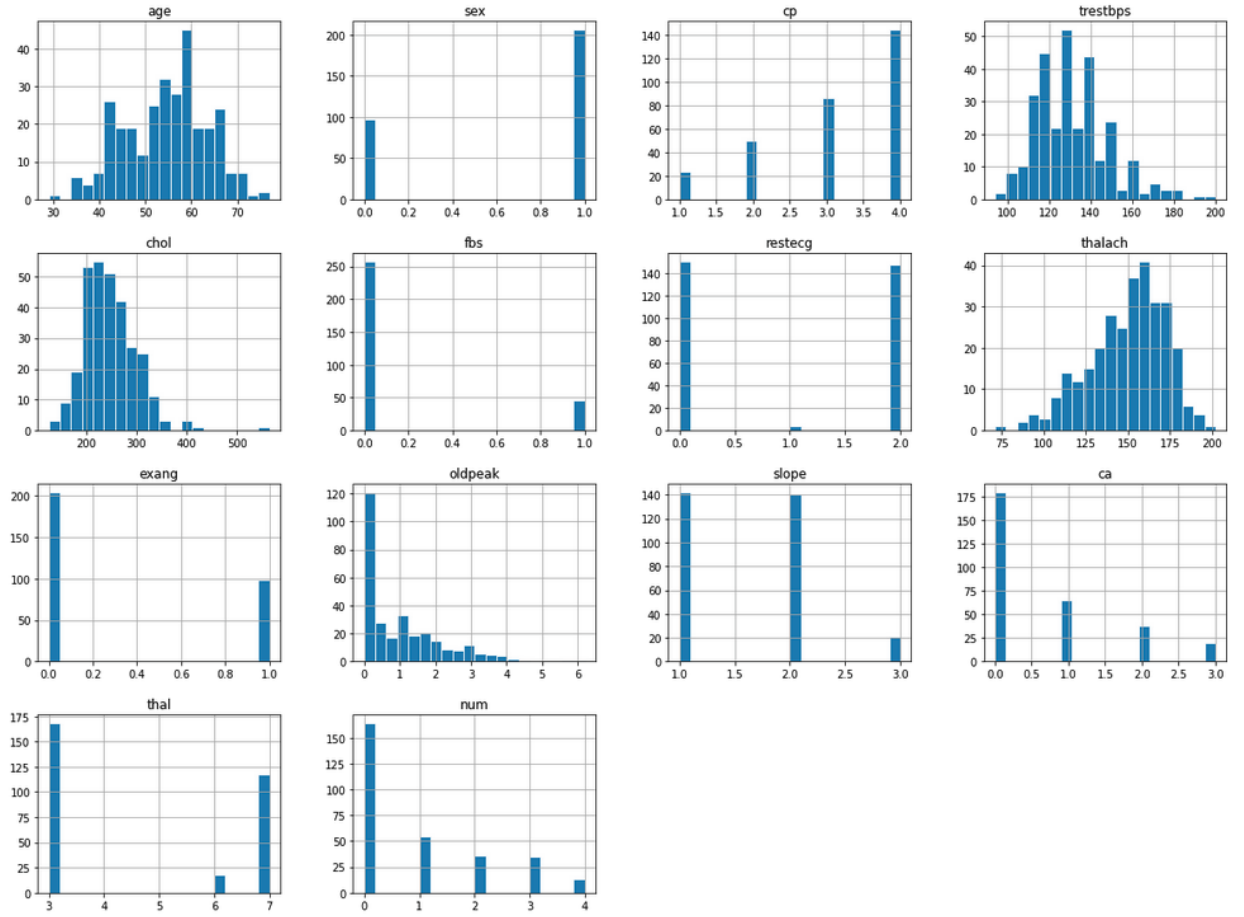
- All of our data is numeric.
- There are no unnecessary attributes in our dataset that interfere with our visualization.
- From the age attribute, we can observe that the data varies from 29 to 77 and is more dense after the second quartile as the difference between the median and the minimum is more than the difference between median and the maximum.

We can continue to visualize our data to get more inference out of it.

Data Visualization

1. We used hist() to plot histogram for all the attributes in the dataset

```
In [10]: data.hist(figsize = (20,15),ec='white',bins=20,alpha=0.9)
         plt.show()
```

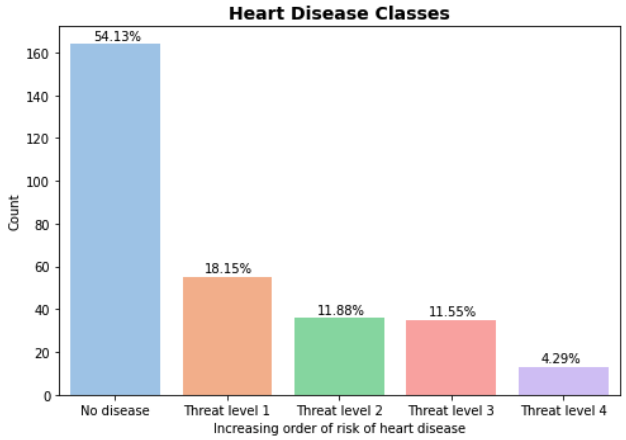


Inference:

- This graph gives the histogram plots for each attribute in our dataset.
- The x-value gives the range of values and the y-value gives the count(frequency).
- age, trestbps are normally distributed
- chol is almost normally distributed
- oldpeak is left-skewed
- thalac is right skewed
- Most of the patients are in their 50s and 60s.

2. Next, we visualize the “num” attribute. It tells about the diagnosis of heart disease for every instance.

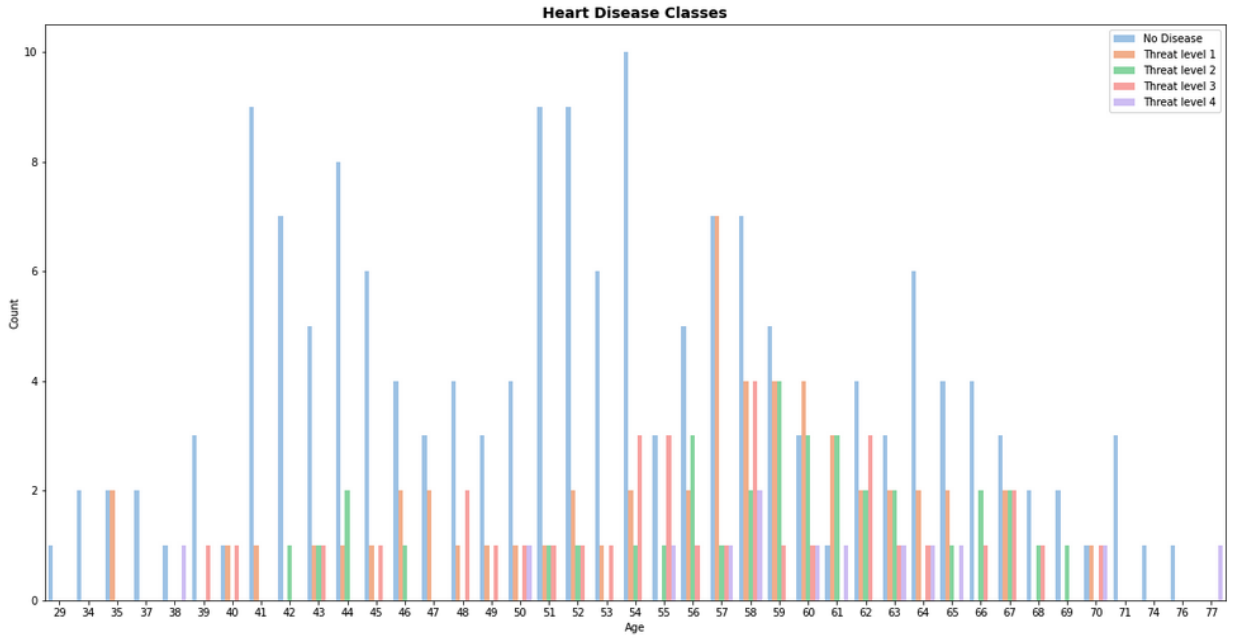
```
In [15]: fig, ax = plt.subplots(figsize=(7,5))
ax = sb.countplot(x="num", data=data, palette="pastel")
ax.set_title("Heart Disease Classes", fontsize = 14, weight = 'bold')
plt.xlabel("Increasing order of risk of heart disease")
plt.ylabel("Count")
name = ["No disease","Threat level 1","Threat level 2","Threat level 3","Threat level 4"]
ax.set_xticklabels (name, rotation = 0)
totals = []
for i in ax.patches:
    totals.append(i.get_height())
total = sum(totals)
for i in ax.patches:
    ax.text(i.get_x()+0.2, i.get_height()+2,
            str(round((i.get_height()/total)*100, 2))+'%')
plt.tight_layout()
plt.show()
```



54.13% of patients are healthy and 45.87% of patients have some sort of threat to heart disease.

3. Now we look at of age according to target class

```
In [36]: fig, ax = plt.subplots(figsize=(20,10))
         ax = sb.countplot(x="age", hue="num", data=data, palette="pastel")
         ax.set_title("Age Distribution according to classes", fontsize = 14, weight = 'bold')
         plt.xlabel("Age")
         plt.ylabel("Count")
         plt.legend(["No Disease","Threat level 1","Threat level 2","Threat level 3","Threat level 4"],loc='upper right')
         plt.show()
```

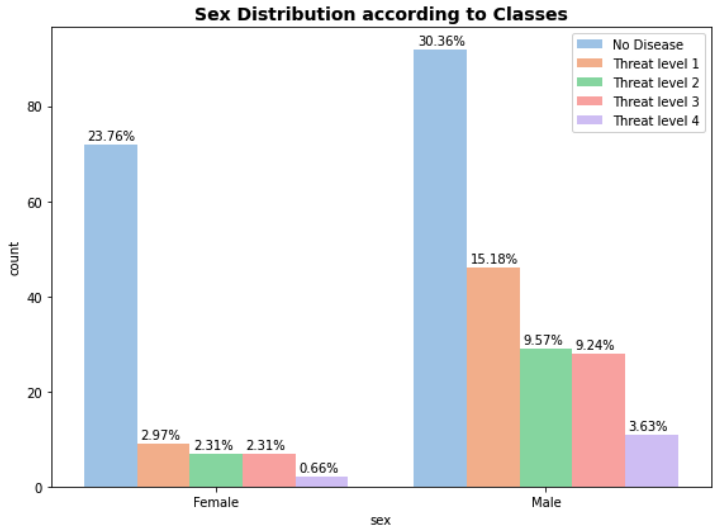


We see that most people who are suffering are of the age of 58, followed by 57. Majorly, people belonging to the age group 50+ are suffering from the disease.

Heart Disease is very common in the seniors who are of age group 60 and above and common among adults which belong to the age group of 41 to 60. But it's rare among the age group of 19 to 40 and very rare among the age group of 0 to 18.

4. Now we look at plot of sex accroding to target classes

```
In [13]: fig, ax = plt.subplots(figsize=(8,6))
name = ["Female","Male"]
ax = sb.countplot(x='sex', hue='num', data=data, palette='pastel')
ax.set_title("Sex Distribution according to Classes", fontsize = 14, weight = 'bold')
ax.set_xticklabels (name, rotation = 0)
totals = []
for i in ax.patches:
    totals.append(i.get_height())
total = sum(totals)
for i in ax.patches:
    ax.text(i.get_x()+0.01, i.get_height()+1,
            str(round((i.get_height()/total)*100, 2))+'%')
plt.tight_layout()
plt.legend(["No Disease","Threat level 1","Threat level 2","Threat level 3","Threat level 4"])
plt.show()

female = len(data[data.sex == 0])
male = len(data[data.sex == 1])
total = len(data.num)
print(f'Female patients: {(female/total)*100:0.2f}%')
print(f'Male patients: {(male/total)*100:0.2f}%')
```

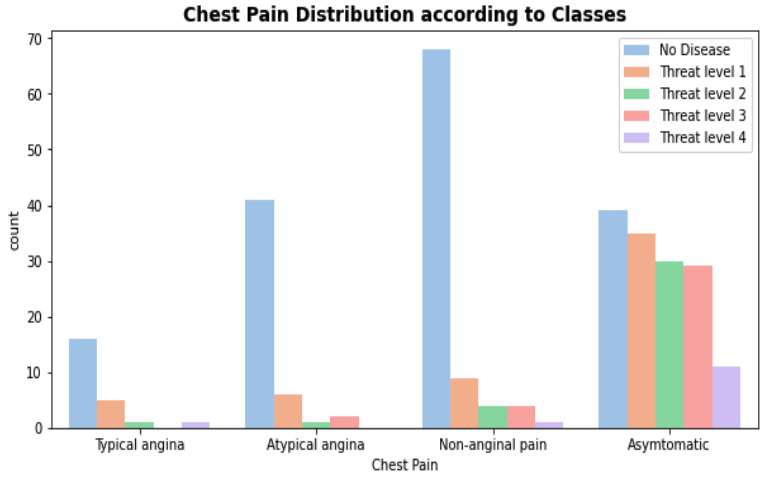



```
Female patients: 32.01%
Male patients: 67.99%
```

According to this Cleveland dataset males are more susceptible to get Heart Disease than females. Men experience heart attacks more than women. Sudden Heart Attacks are experienced by men between 70% — 89%. Women may experience a heart attack with no chest pressure at all, they usually experience nausea or vomiting which are often confused with acid reflux or the flu.

5. Now we look at plot of chest pain according to target class

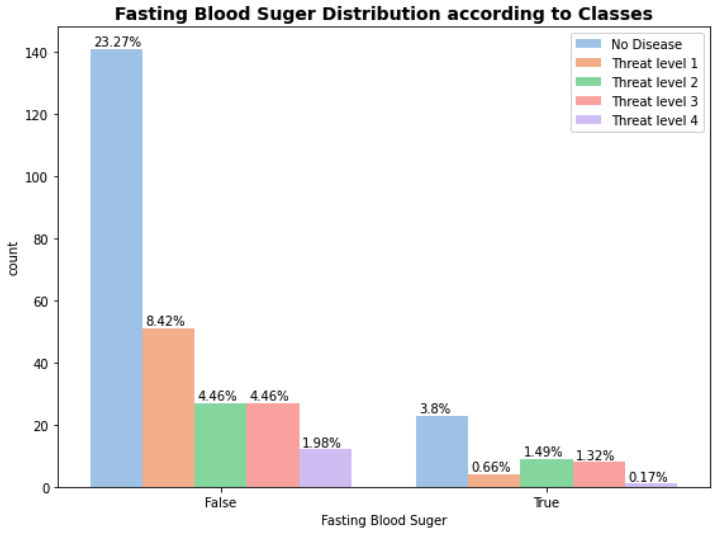
```
In [14]: fig, ax = plt.subplots(figsize=(10,5))
name = ["Typical angina", "Atypical angina", "Non-anginal pain" ,"Asymtomatic"]
ax = sb.countplot(x='cp', hue='num', data=data, palette='pastel')
ax.set_title("Chest Pain Distribution according to Classes", fontsize = 14, weight = 'bold')
ax.set_xticklabels (name, rotation = 0)
plt.xlabel("Chest Pain")
plt.legend(["No Disease","Threat level 1","Threat level 2","Threat level 3","Threat level 4"])
plt.show()
```



There are four types of chest pain, asymptomatic, atypical angina, non-anginal pain and typical angina. Most of the Heart Disease patients are found to have asymptomatic chest pain. These groups of people might show atypical symptoms like indigestion, flu or a strained chest muscle. A asymptomatic attack, like any heart attack, involves blockage of blood flow to your heart and possible damage to the heart muscle.

6. Now we look at plot of fasting blood sugar according to target class

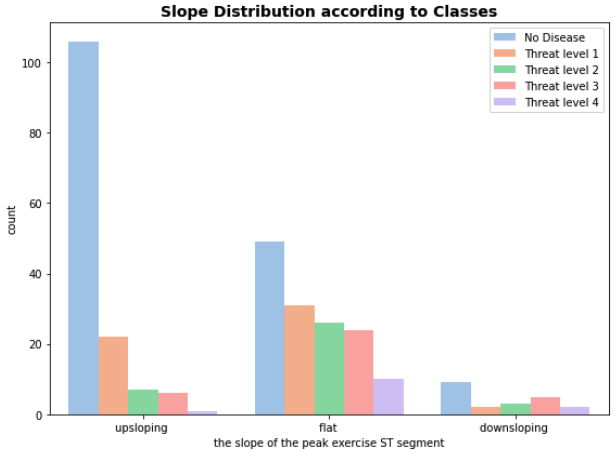
```
In [37]: fig, ax = plt.subplots(figsize=(8,6))
name = ["False","True"]
ax = sb.countplot(x='fbs', hue='num', data=data, palette='pastel')
ax.set_title("Fasting Blood Suger Distribution according to Classes", fontsize = 14, weight = 'bold')
ax.set_xticklabels (name, rotation = 0)
plt.xlabel("Fasting Blood Suger")
for i in ax.patches:
    totals.append(i.get_height())
total = sum(totals)
for i in ax.patches:
    ax.text(i.get_x()+0.01, i.get_height()+1,
            str(round((i.get_height()/total)*100, 2))+'%')

plt.legend(["No Disease","Threat level 1","Threat level 2","Threat level 3","Threat level 4"])
plt.tight_layout()
plt.show()
```



Fasting blood sugar or fbs is a diabetes indicator with fbs >120 mg/d is considered diabetic (True class). Here, we observe that the number for class true is lower compared to class false.

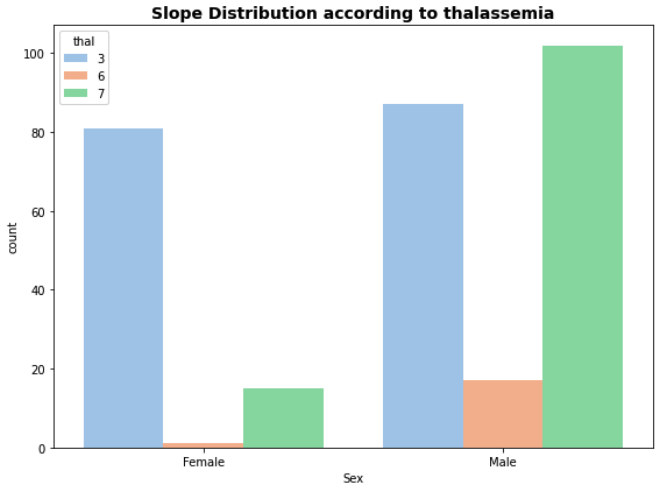
7. Now we look at plot of slope distribution according target class

```
In [38]: fig, ax = plt.subplots(figsize=(8,6))
name = ["upsloping ","flat ","downsloping "]
ax = sb.countplot(x='slope', hue='num', data=data, palette='pastel')
ax.set_title("Slope Distribution according to Classes", fontsize = 14, weight = 'bold')
ax.set_xticklabels (name, rotation = 0)
plt.xlabel("the slope of the peak exercise ST segment")
plt.legend(["No Disease","Threat level 1","Threat level 2","Threat level 3","Threat level 4"])
plt.tight_layout()
plt.show()
```



8. Now we look at plot of sex according to thalassemia

```
In [23]: fig, ax = plt.subplots(figsize=(8,6))
ax = sb.countplot(x='sex', hue='thal', data=data, palette='pastel')
ax.set_title("Slope Distribution according to thalassemia", fontsize = 14, weight = 'bold')
name = ["Female","Male"]
ax.set_xticklabels (name, rotation = 0)
plt.xlabel("Sex")
plt.tight_layout()
plt.show()
```



9. Now we look at jointplot between cholesterol and age

To do this, we first created a temp data that only consists of records of patients with some severity of disease.

In [17]:
```
temp_data = data.copy()
for i in range(0,len(data)):
    if temp_data.num[i] == 0:
        temp_data.drop(labels=i,axis=0,inplace=True)
temp_data
```

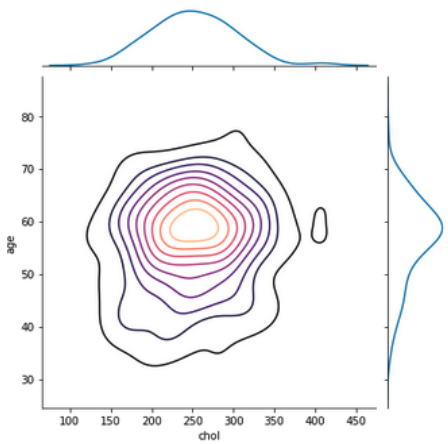
Out[17]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
1	67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
2	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
6	62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
8	63	1	4	130	254	0	2	147	0	1.4	2	1	7	2
9	53	1	4	140	203	1	2	155	1	3.1	3	0	7	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
297	57	0	4	140	241	0	0	123	1	0.2	2	0	7	1
298	45	1	1	110	264	0	0	132	0	1.2	2	0	7	1
299	68	1	4	144	193	1	0	141	0	3.4	2	2	7	2
300	57	1	4	130	131	0	0	115	1	1.2	2	1	7	3
301	57	0	2	130	236	0	2	174	0	0.0	2	1	3	1

139 rows × 14 columns

In [18]:
```
sb.jointplot(data=temp_data,
             x='chol',
             y='age',
             kind='kde',
            cmap='magma'
             )
```

Out[18]: <seaborn.axisgrid.JointGrid at 0x119361040>



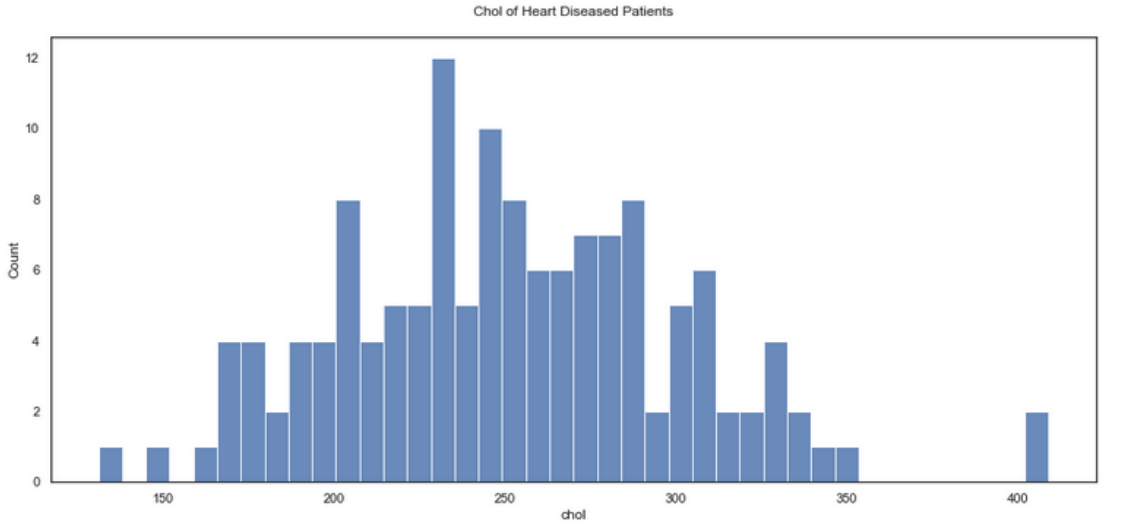
Inference:

Joint plots in seaborn helps us to understand the trend seen among two features. As observed from the above plot we can see that most of the Heart diseased patients in their age of upper 50s or lower 60s tend to have Cholesterol between 200 mg/dl to 300 mg/dl.

10. Now we look at jointplot between cholesterol and age

Taking the same test_data made for the previous plot, we explore the cholesterol level in diseased patients.

```
In [24]: plt.figure(figsize=(16,7))
         sb.histplot(temp_data['chol'],kde=False,bins=40)
         plt.title('Chol of Heart Diseased Patients\n')
         plt.show()
```



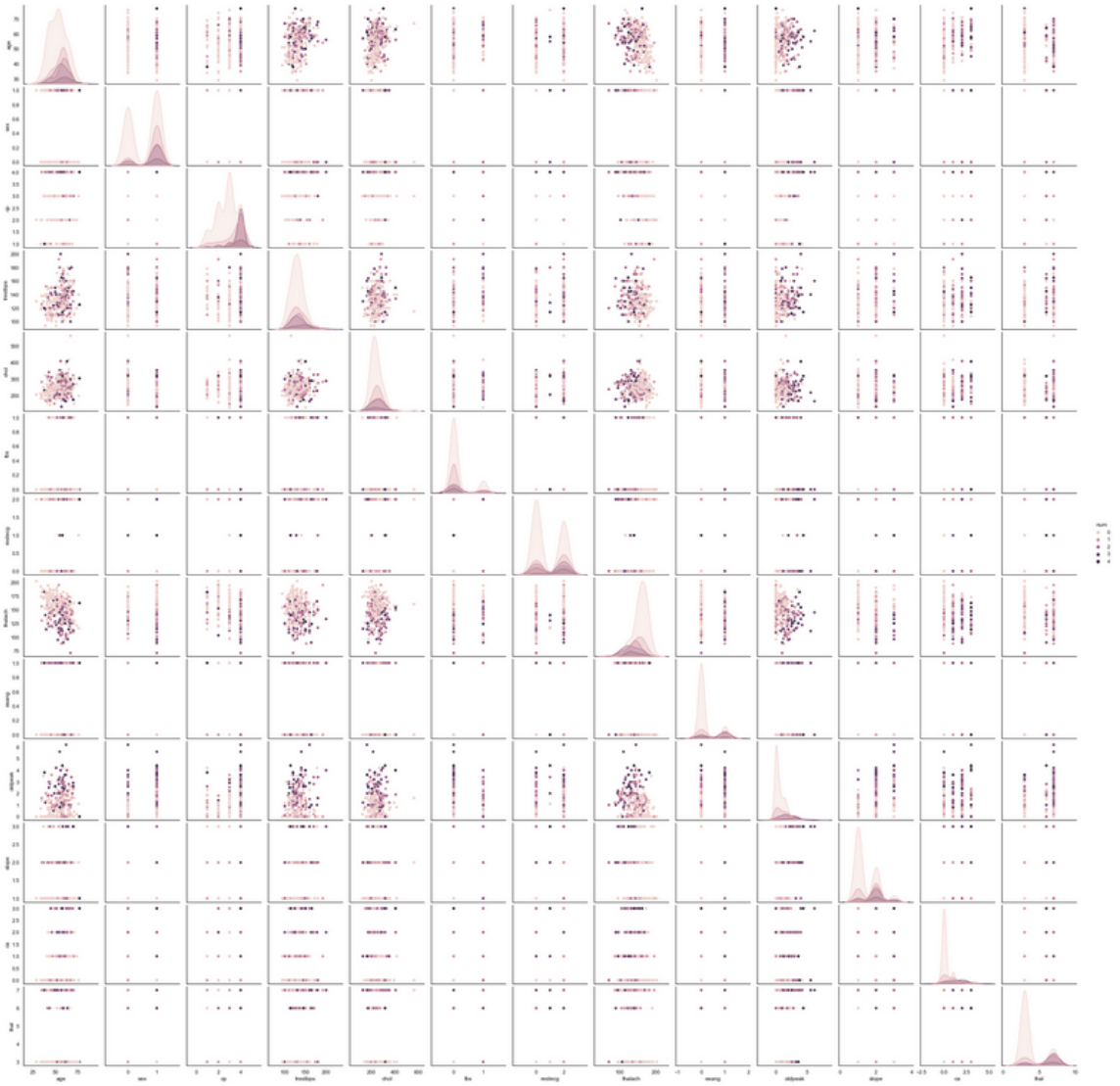
In adults, the total cholesterol levels are considered desirable less than 200 milligram per decilitre (mg / dL). Borderlines are considered to be high between 200 to 239 mg / dL and 240 mg / dL and above.

Bad cholesterol should contain less than 100 mg / dL of cholesterol. 100 mg / dl rates for individuals without any health issue are appropriate but may be more relevant for those with cardiac problems or risk factors for heart disease.

Levels of good cholesterol are to be maintained higher. The risk factor for cardiovascular diseases is called a reading less than 40 mg / dL. The good cholesterol level can be measured with a maximum of 60 mg / dL.

11. Now we look at pairplot to visualize the distribution.

```
In [60]: sb.pairplot(data,hue="num")
Out[60]: <seaborn.axisgrid.PairGrid at 0x12447bbb0>
```

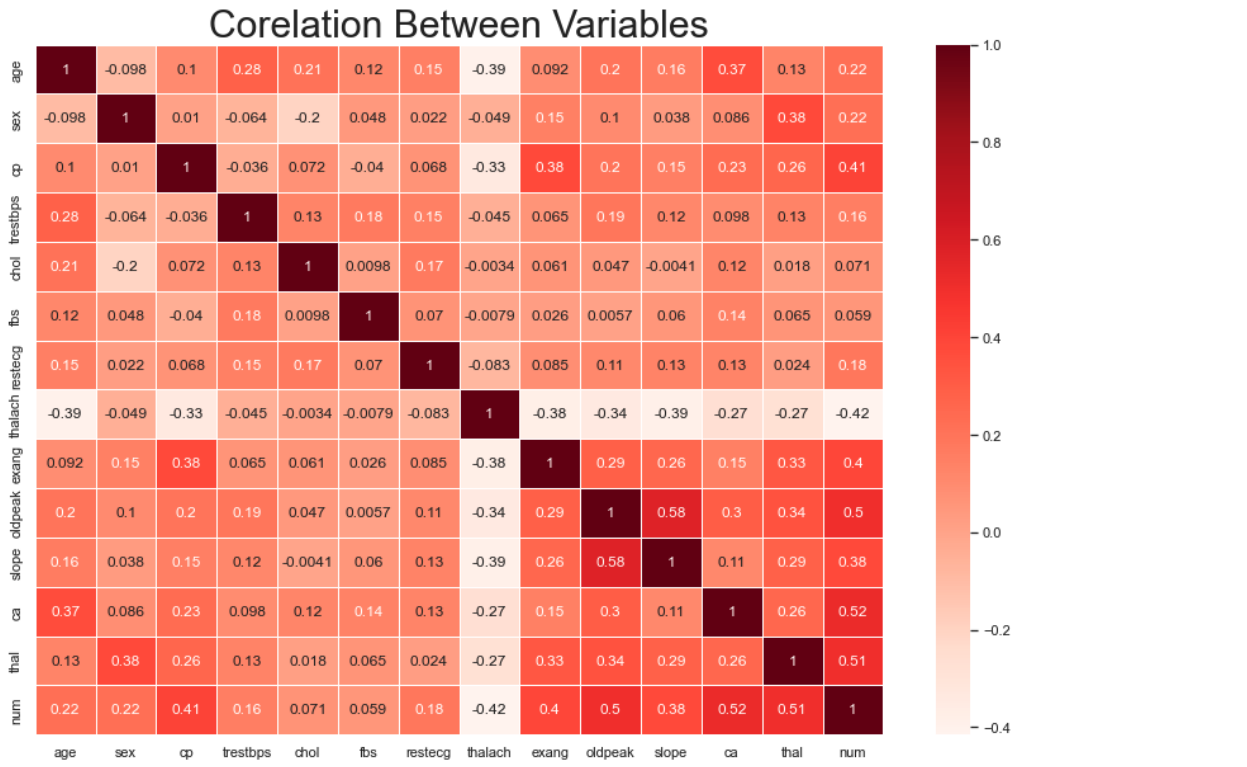


Inference:

- oldpeak having a linear separation relation between disease and non-disease.
- thalach having a mild separation relation between disease and non-disease.
- Other features don’t form any clear separation

12. Now we look at the correlation between the attributes.

```
In [58]: sb.set(style="white")
plt.rcParams['figure.figsize'] = (15, 10)
sb.heatmap(data.corr(), annot = True, linewidths=.5, cmap="Reds")
plt.title('Corelation Between Variables', fontsize = 30)
plt.show()
```



Inference:

- ‘cp’, ‘thalach’, ‘slope’ show a good positive correlation with the target.
- ‘oldpeak’, ‘exang’, ‘ca’, ‘thal’, ‘sex’, ‘age’ shows a good negative correlation with target.
- ‘fbs’ ‘chol’, ‘trestbps’, ‘restecg’ have low correlation with our target.

Conclusion

Heart Disease is one of the major concerns for society today. It is difficult to manually determine the odds of getting heart disease based on risk factors.

However, by using data preprocessing and analysis we get a detailed insight about the trends and causes for heart disease. By applying further ML algorithms we can predict the output from existing data.

Acknowledgement

We would like to thank Prof. Aloke Datta for giving us an opportunity to work upon this project on Data Science. This project has been completed only because of the team effort and cooperation from Tanmay Lodha, Sanyam Lodha, Aryan Seth and Nitish Kumar Gupta.

References

- Class lectures.
- seaborn: statistical data visualization — seaborn 0.11.2 documentation
- Matplotlib — Visualization with Python
- Medium – Where good ideas find you.
- Towards Data Science
- Heart Disease Data Set - UCI Machine Learning Repository

We have also uploaded the jupyter notebook of this project on GitHub