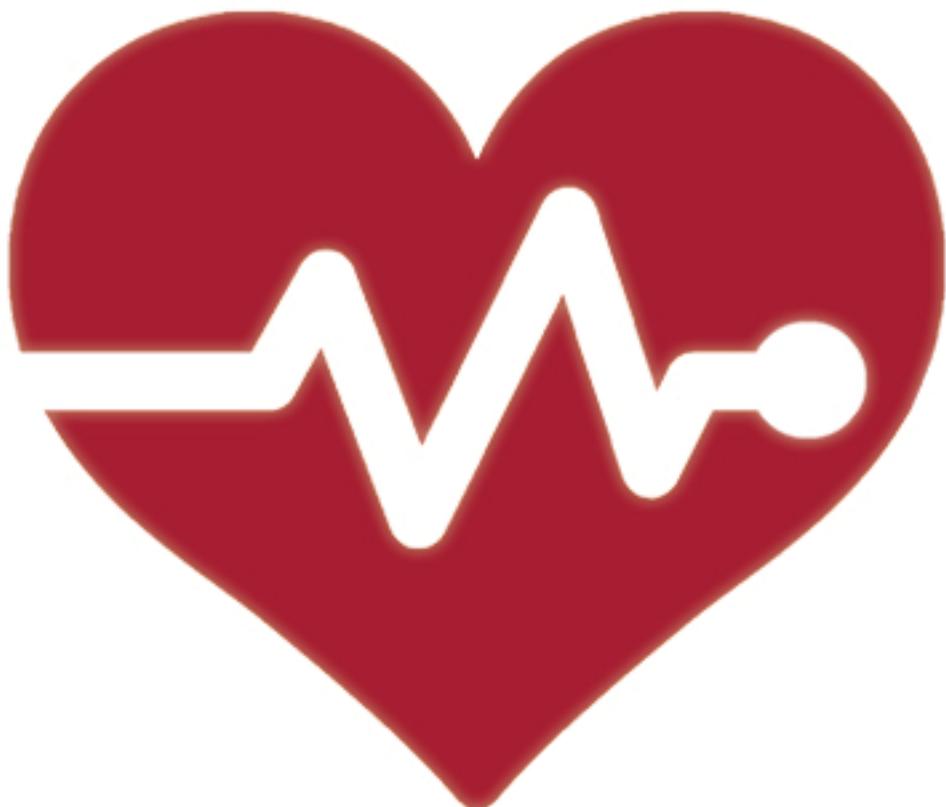


DATA ANALYTICS TRAINING

FIDELITY INTERNATIONAL

HEART - SEIZURE PREDICTION



SUBMITTED BY -

SHUBHAM

Project Location

GITHUB URL

This project is stored on GitHub - an online project repository platform. This platform provides with many features that helps its user to keep their projects safely and securely. It also provides feature like version control of the file and allows others collaboration on projects.

https://github.com/Shubham14793/UCDPA_shubham

ABSTRACT

In this project I have tried to use and implement the learning that was provided by the trainer during the online course and data camp worksheets. During development of main project developed I realised that few topics that were discussed might not be directly utilised in project. Hence, I have created two project files-

1. DATA_IMPORT_&_ANALYSIS.IPYNB

This first file contains concepts like Importing a dataset using an api and then processing the response data. Creating function to reutilise the code and using regular expression to extract useful data from response of an api.

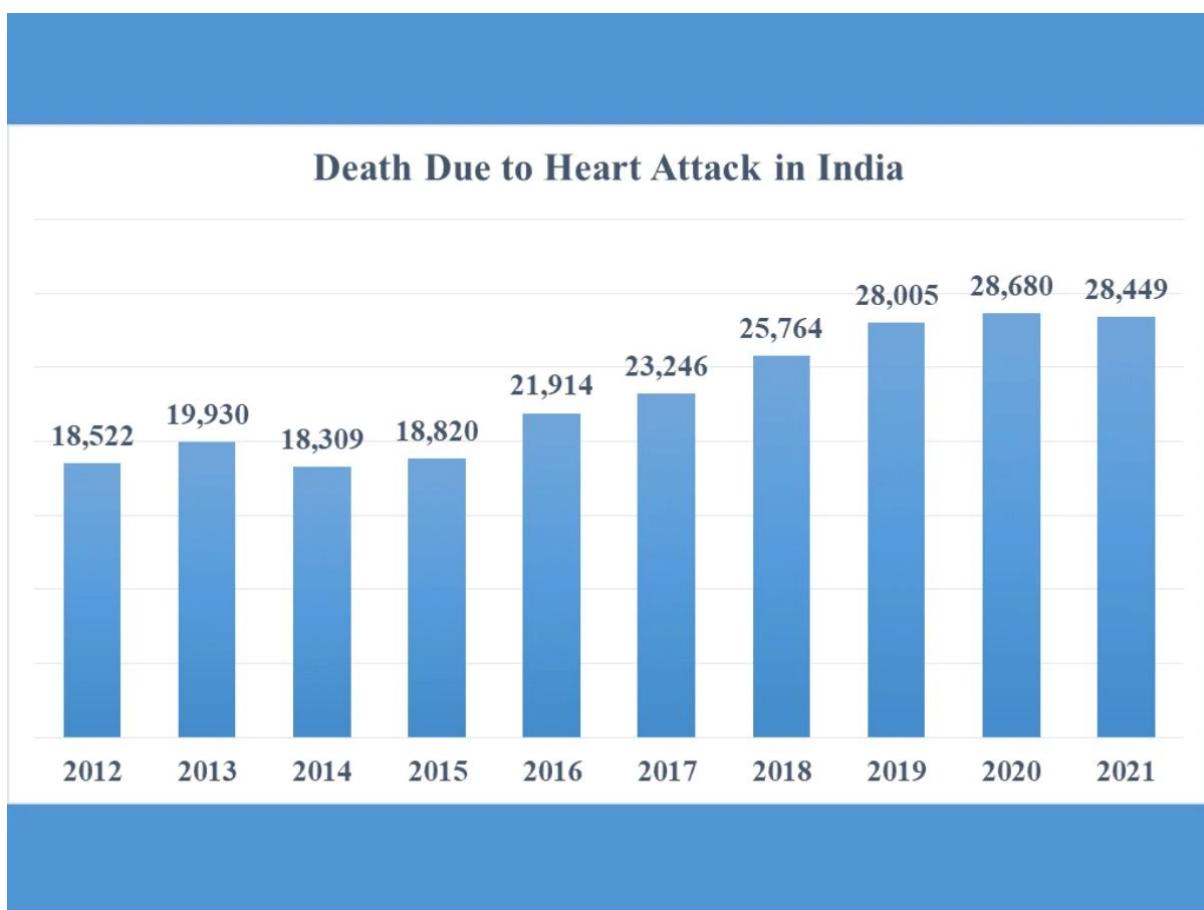
2. HEART_SEIZURE_DETECTION_MAIN.IPYNB

The second file contains the main project. Heart-Seizure analysis is a predictive data analysis project which helps us identifying whether or not a person or subject is susceptible to Heart Seizure or Heart Attack. This project utilises different aspect of a patient or subject to predict its conclusion. We will be using a dataset available from highly reliable datasource. There are 2 separate datasets available of the patients. We will combine these dataset together and doing the analysis on the combined dataframe.

First we will be importing the data from the data source and storing it locally. We will understand the data through various methods available from python language and then perform certain operations that will provide us deep insights of the data. We will use python packages to do a comparative analysis of dataset through charts. We then will move on to using AI models to create a predictive model which will help us make decision on possibility of random person being diagnosed with Heart-Seizure.

INTRODUCTION

As the world is emerging out of COVID-19 pandemic, deteriorating heart conditions of the citizen has been another big concern for the countries. Developing countries like India have seen sudden increase in Heart-Attack cases where in most of the cases the victim resulted in loss of life. In the last 10 years, from 2012 to 2021, there had been 54% increase in heart attack deaths, according to NCRB.



According to an article published by INDIA TODAY:

"Heart-related ailments among young Indians have become a concern for health experts. The issue is being debated why there has been a surge in heart attack cases in relatively younger age groups in India. Many people have associated it with Covid-19."

This issue requires some immediate actions in order to understand and reduce down cardiovascular related cases. A healthy lifestyle has always been advised but there can be other sources as well which can contribute in addressing this long prevailing situation. Early detection of cardiac diseases and continuous supervision of clinicians can reduce the mortality rate. However, accurate detection of heart diseases in all cases and consultation of a patient for 24 hours by a doctor is not available since it requires more sapience, time and expertise.

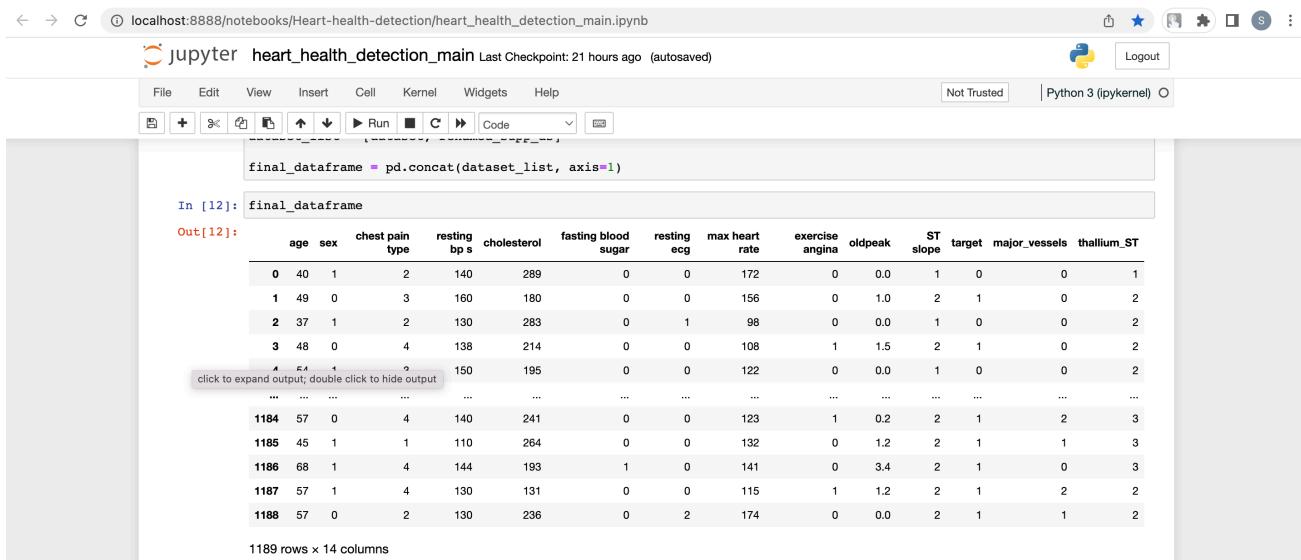
For the accurate detection of the heart disease we can utilise the advanced computer methodologies and techniques. Using Data Science and Artificial Intelligence for anomalies related to Heart condition can be a wonderful tool. In this project we are presenting a machine learning-based heart attack prediction method in which the analysis of different risk factors and prediction for heart attacks is done using ML approaches of Support Vector Machines, Logistic Regression. The data of heart disease symptoms has been collected dataset provided by IEEE.ORG and analysis has been performed on the data using ML methods. The focus has been on optimising the prediction on the basis of different parameters.

We observed that using ML approaches of Support Vector Machines, Logistic Regression helped AI model to predict possibility of Heart-Attack in a patient with more than 85% accuracy.

DATASET

The dataset that was utilised for this project is from ieee-dataport.org. The reason why I chose this dataset is that this heart disease dataset is created by combining 5 popular heart disease datasets already available independently but not combined before. In this dataset, 5 heart datasets are combined over multiple common features which makes it the largest heart disease dataset available so far for research purposes.

The dataset is consisting of 14 total columns and 1190 total rows. Each column is either of type int64 or float64.

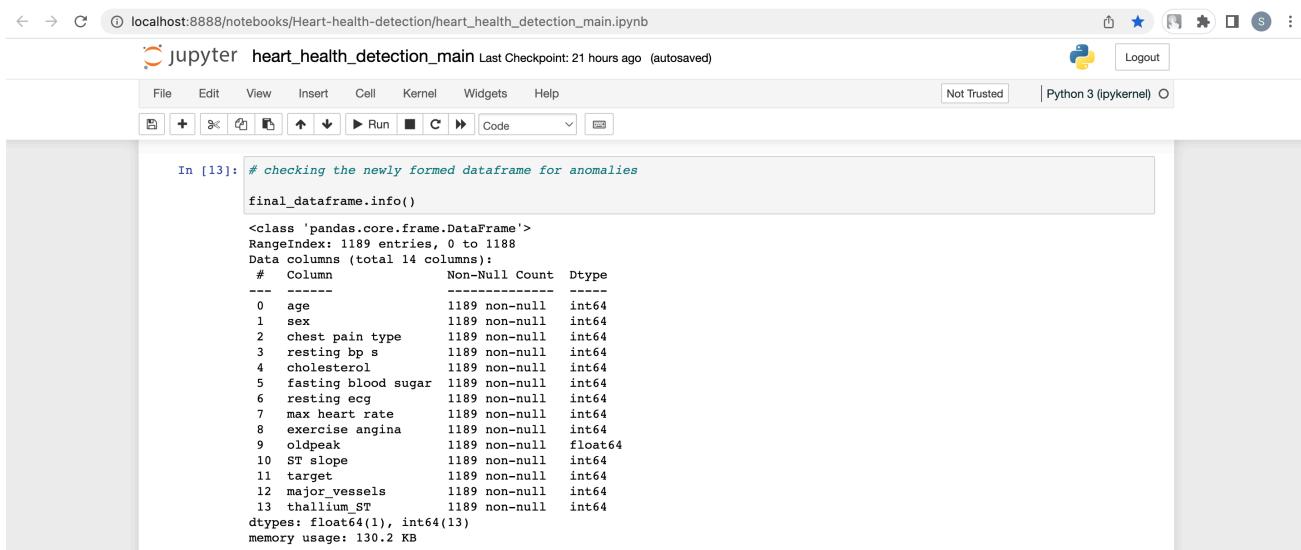


```
final_dataframe = pd.concat(dataset_list, axis=1)
```

In [12]: final_dataframe

	age	sex	chest pain type	resting bp s	cholesterol	fasting blood sugar	resting ecg	max heart rate	exercise angina	oldpeak	ST slope	target	major_vessels	thallium_ST
0	40	1	2	140	289	0	0	172	0	0.0	1	0	0	1
1	49	0	3	160	180	0	0	156	0	1.0	2	1	0	2
2	37	1	2	130	283	0	1	98	0	0.0	1	0	0	2
3	48	0	4	138	214	0	0	108	1	1.5	2	1	0	2
4	54	1	2	150	195	0	0	122	0	0.0	1	0	0	2
...
1184	57	0	4	140	241	0	0	123	1	0.2	2	1	2	3
1185	45	1	1	110	264	0	0	132	0	1.2	2	1	1	3
1186	68	1	4	144	193	1	0	141	0	3.4	2	1	0	3
1187	57	1	4	130	131	0	0	115	1	1.2	2	1	2	2
1188	57	0	2	130	236	0	2	174	0	0.0	2	1	1	2

1189 rows × 14 columns



```
# checking the newly formed dataframe for anomalies
```

```
final_dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1189 entries, 0 to 1188
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   age              1189 non-null   int64  
 1   sex              1189 non-null   int64  
 2   chest pain type 1189 non-null   int64  
 3   resting bp s    1189 non-null   int64  
 4   cholesterol      1189 non-null   int64  
 5   fasting blood sugar 1189 non-null   int64  
 6   resting ecg       1189 non-null   int64  
 7   max heart rate   1189 non-null   int64  
 8   exercise angina  1189 non-null   int64  
 9   oldpeak          1189 non-null   float64 
 10  ST slope          1189 non-null   int64  
 11  target            1189 non-null   int64  
 12  major_vessels     1189 non-null   int64  
 13  thallium ST       1189 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 130.2 KB
```

DATASET SOURCE

The dataset can be downloaded from below URL-

<https://ieee-dataport.org/open-access/heart-disease-dataset-comprehensive>

DATASET COLUMNS DESCRIPTION

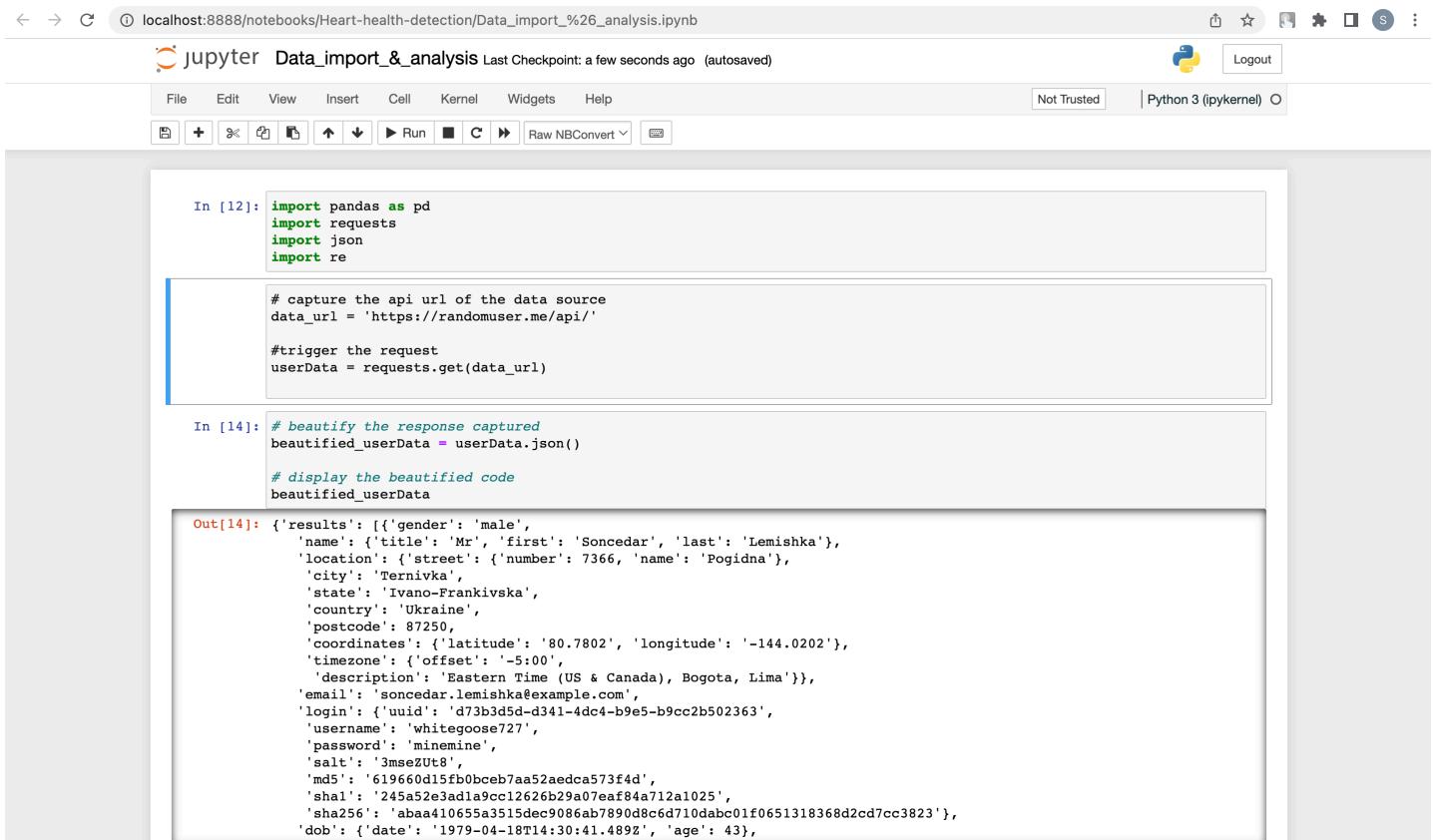
S.No.	Column Name	Column Description
1	Age	Age of the Subject
2	Sex	Gender of the Subject
3	Chest pain type	Categories - [0, 1, 2, 3] 0 -> Typical Pain 1 -> Atypical Pain 2 -> Non-specific chest pain 3 -> Asymptomatic
4	Resting BP S	Blood Pressure at resting state of Subject
5	Cholesterol	Cholesterol value measure by a device of Subject
6	Fasting blood sugar	High Blood Sugar during fasting (if > 120 then 1 otherwise 0)
7	Resting ECG	Electrocardiography during resting. Categories - [0, 1, 2] 0 -> Normal 1 -> ST-T wave normality () 2 -> Left ventricular hypertrophy (wall thickening of heart chamber)
8	Maximum Heart Rate	Maximum Heart Rate of the subject
9	Exercise Angina	Pain due to exercise [0 -> no, 1 -> yes]
10	Oldpeak	Previous Peak
11	ST Slope	Predictor of coronary artery disease
12	Target	Output variable
13	Major Vessel	Heart's Major Vessels
14	Thallium ST	Thallium stress test is an indicator of how well your blood flows to your heart.

IMPLEMENTATION PROCESS

ADDITIONAL GRADING STEPS

In order to meet the grading requirement of the project, I have performed several steps additional to the main project. These additional steps include -

1. Importing Data using an API and processing the response.



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** localhost:8888/notebooks/Heart-health-detection/Data_import_%26_analysis.ipynb
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Run, Stop, Kernel, Raw NBConvert, etc.
- Status Bar:** Not Trusted | Python 3 (ipykernel) ○
- In [12]:**

```
import pandas as pd
import requests
import json
import re
```
- In [13]:**

```
# capture the api url of the data source
data_url = 'https://randomuser.me/api/'

#trigger the request
userData = requests.get(data_url)
```
- In [14]:**

```
# beautify the response captured
beautified_userData = userData.json()

# display the beautified code
beautified_userData
```
- Out[14]:**

```
{'results': [{}{'gender': 'male',
  'name': {'title': 'Mr', 'first': 'Soncedar', 'last': 'Lemishka'},
  'location': {'street': {'number': 7366, 'name': 'Pogidna'},
    'city': 'Ternivka',
    'state': 'Ivano-Frankivska',
    'country': 'Ukraine',
    'postcode': 87250,
    'coordinates': {'latitude': '80.7802', 'longitude': '-144.0202'},
    'timezone': {'offset': '-5:00',
      'description': 'Eastern Time (US & Canada), Bogota, Lima'},
    'email': 'soncedar.leminshka@example.com',
    'login': {'uuid': 'd73b3d5d-d341-4dc4-b9e5-b9cc2b502363',
      'username': 'whitegoose727',
      'password': 'minemine',
      'salt': '3mseZUt8',
      'md5': '619660d15fb0bceb7aa52aedca573f4d',
      'sha1': '245a52e3ad1a9cc12626b29a07eaaf84a712a1025',
      'sha256': 'abaa410655a3515dec9086ab7890d8c6d710dabc01f0651318368d2cd7cc3823'},
    'dob': {'date': '1979-04-18T14:30:41.489Z', 'age': 43},
```

2. Analysing Data using Regular expressions and iterating through the Data.

The screenshot shows a Jupyter Notebook interface with the title "Data_import_&_analysis". The notebook has two cells:

```
In [15]: # process the data received from api
responseText = json.loads(userData.text)
regexString = json.dumps(responseText)

In [16]: # creating a function that extracts the profile pic of the user from the api response
# this function uses a regular expression to match a pattern to extract data

def getUserProfileImages (targetData=""):
    if len(targetData) == 0:
        return []
    regexPattern = '[a-z]{5}://\S*\.jpg'
    return re.findall(regexPattern, targetData)

Out[16]: ['https://randomuser.me/api/portraits/men/80.jpg',
          'https://randomuser.me/api/portraits/med/men/80.jpg',
          'https://randomuser.me/api/portraits/thumb/men/80.jpg']
```

3. Implementing python language specific function to make reusable code and creating Lists or Dictionary data structure to store the values.

The screenshot shows a Jupyter Notebook interface with two cells:

```
In [68]: userDetails = dict(personObject["results"][0])
userDetails["gender"]

Out[68]: 'male'

In [72]: personalDetailList = list(("name", "gender", "email", "phone"))

def extractPersonalDetailsData (personalDetailList, userData):
    personalDetailsMappings = {}
    for pdl in personalDetailList:
        personalDetailsMappings[pdl] = userData[pdl]
    return personalDetailsMappings

extractPersonalDetailsData(personalDetailList, userDetails)

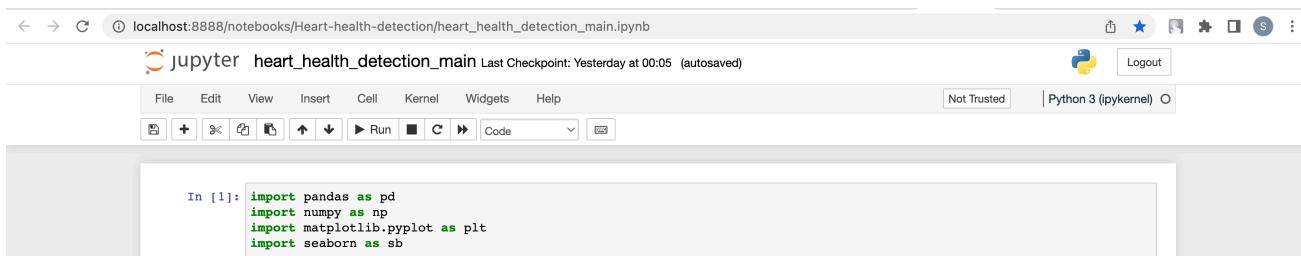
Out[72]: {'name': {'title': 'Mr', 'first': 'zakaria', 'last': 'Smakman'},
          'gender': 'male',
          'email': 'zakaria.smakman@example.com',
          'phone': '(0754) 800710'}
```

MAIN PROJECT

With goal of creating a model that is capable of predicting if a person is prone to a Heart Attack, I have performed series of operations on the dataset. The whole process of creating the model can be divided into several sub-sections.

1. PACKAGE IMPORT

In this section I imported all the project dependency. These dependencies are required to perform operations on the datasets that will be imported.



The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** localhost:8888/notebooks/Heart-health-detection/heart_health_detection_main.ipynb
- Title Bar:** jupyter heart_health_detection_main Last Checkpoint: Yesterday at 00:05 (autosaved)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help
- Status Bar:** Not Trusted | Python 3 (ipykernel) ○
- Code Cell:** In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
```

Pandas is a Python package **providing fast, flexible, and expressive data structures** designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.

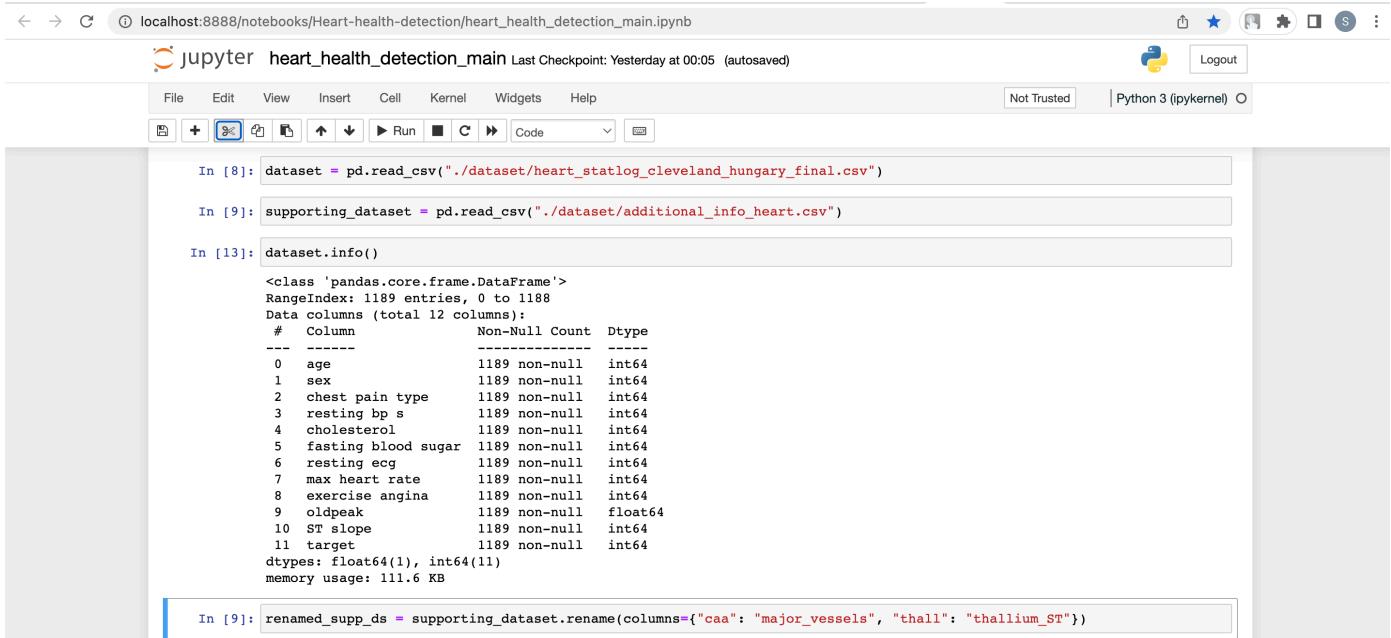
NumPy is **a Python library used for working with arrays**. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

Matplotlib is **a plotting library for Python**. It is used along with NumPy to provide an environment that represents the data Graphically. This library has lot of functionalities that can be helpful in visual representation of the processed data

Seaborn is an amazing visualisation library for statistical graphics plotting in Python. It provides beautiful default styles and colour palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.

2. DATA IMPORT

In this section the dataset was imported after downloading it locally. Its benefit is that there is no dependency on internet connection. Using pandas based method can be utilised for importing the dataset.



The screenshot shows a Jupyter Notebook interface with the following code cells:

```

In [8]: dataset = pd.read_csv("./dataset/heart_statlog_cleveland_hungary_final.csv")
In [9]: supporting_dataset = pd.read_csv("./dataset/additional_info_heart.csv")
In [13]: dataset.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1189 entries, 0 to 1188
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   age              1189 non-null    int64  
 1   sex              1189 non-null    int64  
 2   chest pain type 1189 non-null    int64  
 3   resting bp s    1189 non-null    int64  
 4   cholesterol      1189 non-null    int64  
 5   fasting blood sugar 1189 non-null  int64  
 6   resting ecg       1189 non-null    int64  
 7   max heart rate   1189 non-null    int64  
 8   exercise angina   1189 non-null    int64  
 9   oldpeak          1189 non-null    float64 
 10  ST slope         1189 non-null    int64  
 11  target            1189 non-null    int64  
dtypes: float64(1), int64(11)
memory usage: 111.6 KB
In [9]: renamed_supp_ds = supporting_dataset.rename(columns={"caa": "major_vessels", "thall": "thallium_ST"})

```

Once the data is imported successfully. Information about the data can be checked using **info() method** provided by pandas package.

3. DATA CLEANING AND PROCESSING.

This part is one of the most important step of whole process. Effective data cleaning is a vital part of the data analytics process. If the data has inconsistencies or error, there will be very high chances of flawed results. Data cleaning activity provides several benefits like - **Organised data, avoids mistake, productivity improvement, cost reduction and improved mapping of the data.**

Pandas describe() method is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values. When this method is applied to a series of string, it returns a different output which is shown below.

```
In [15]: final_dataframe.describe().transpose()
Out[15]:
      count      mean       std    min   25%   50%   75%   max
age     1189.0  53.733389  9.351015  28.0  47.0  54.0  60.0  77.0
sex     1189.0  0.763667  0.425008  0.0   1.0   1.0   1.0   1.0
chest pain type  1189.0  3.232969  0.935850  1.0   3.0   4.0   4.0   4.0
resting bp s    1189.0  132.148665 18.375769  0.0  120.0  130.0  140.0  200.0
cholesterol    1189.0  210.393608 101.457973  0.0  188.0  229.0  270.0  603.0
fasting blood sugar  1189.0  0.213625  0.410037  0.0   0.0   0.0   0.0   1.0
resting ecg     1189.0  0.698907  0.870489  0.0   0.0   0.0   2.0   2.0
max heart rate  1189.0  139.704794 25.510105  60.0  121.0  140.0  160.0  202.0
exercise angina  1189.0  0.387721  0.487435  0.0   0.0   0.0   1.0   1.0
oldpeak        1189.0  0.923541  1.086464  -2.6   0.0   0.6   1.6   6.2
ST slope        1189.0  1.624895  0.610447  0.0   1.0   2.0   2.0   3.0
target          1189.0  0.529016  0.499367  0.0   0.0   1.0   1.0   1.0
major_vessels   1189.0  0.724138  1.023099  0.0   0.0   0.0   1.0   4.0
thallium_ST     1189.0  2.316232  0.602392  0.0   2.0   2.0   3.0   3.0
```

While making a Data Frame from a csv file, many blank columns are imported as null value into the Data Frame which later creates problems while operating that data frame. Pandas isnull() and notnull() methods are used to check and manage NULL values in a data frame

```
In [19]: # checking for the presence of null values
final_dataframe.isnull().sum()
Out[19]:
age          0
sex          0
chest pain type  0
resting bp s    0
cholesterol    0
fasting blood sugar  0
resting ecg     0
max heart rate  0
exercise angina  0
oldpeak        0
ST slope        0
target          0
major_vessels   0
thallium_ST     0
dtype: int64
```

4. DATASET CATEGORISATION

The data columns provided by the Dataset are of different type i.e for few fields the value can be anything between a large range and it can be one of the limited options for other fields. Hence the dataset can be divided into categories-

1) Categorical columns

2) Continuous columns

But, why do we need to categorise the dataset ?

Well since we are doing analysis on the dataset, there are many techniques that Pandas library provide to do that. Using categories can bring some significant benefits:

- **Memory usage** for columns where there are many repeated values, categories can drastically reduce the amount of memory required to store the data in memory.
- **Runtime performance** are optimisations in place which can improve execution speed for certain operations.
- **library integrations** have special functionality for categorical columns.

Here is an screenshot of categorisation that I did for the Dataset columns:

The screenshot shows a Jupyter Notebook interface with the following content:

```
In [43]: #now we will divide the data into three categories 1) categorical data 2) Continuous data 3) target data
category_columns = [
    'sex',
    'chest pain type',
    'fasting blood sugar',
    'major_vessels',
    'resting ecg',
    'ST slope',
    'thalium ST'
]

continuous_columns = [
    'age',
    'resting bp s',
    'cholesterol',
    'max heart rate',
    'oldpeak'
]

target_column = ['target']
```

```
In [44]: final_dataframe[category_columns].describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
sex	1189.0	0.763667	0.425008	0.0	1.0	1.0	1.0	1.0
chest pain type	1189.0	3.232969	0.935850	1.0	3.0	4.0	4.0	4.0
fasting blood sugar	1189.0	0.213625	0.410037	0.0	0.0	0.0	0.0	1.0
major_vessels	1189.0	0.724138	1.023099	0.0	0.0	0.0	1.0	4.0
resting ecg	1189.0	0.698907	0.870489	0.0	0.0	0.0	2.0	2.0
ST slope	1189.0	1.624895	0.610447	0.0	1.0	2.0	2.0	3.0
thalium_ST	1189.0	2.316232	0.602392	0.0	2.0	2.0	3.0	3.0

The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** The URL is `localhost:8888/notebooks/Heart-health-detection/heart_health_detection_main.ipynb`. The title bar says "jupyter heart_health_detection_main Last Checkpoint: an hour ago (unsaved changes)". On the right, there are icons for file operations, a Python logo, and a "Logout" button.
- Toolbar:** A standard toolbar with buttons for File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and various execution and cell manipulation icons.
- Status Bar:** Shows "Not Trusted" and "Python 3 (ipykernel) O".
- Code Cell:** An input cell labeled "In [29]:" containing the Python command: `fdf_corr = final_dataframe[continuous_columns].describe().transpose()
fdf_corr`.
- Output Cell:** An output cell labeled "Out[29]:" showing a summary statistics table for continuous columns. The table has columns: count, mean, std, min, 25%, 50%, 75%, max. The data rows are:

	count	mean	std	min	25%	50%	75%	max
age	1189.0	53.733389	9.351015	28.0	47.0	54.0	60.0	77.0
resting bp s	1189.0	132.148865	18.375769	0.0	120.0	130.0	140.0	200.0
cholesterol	1189.0	210.393608	101.457973	0.0	188.0	229.0	270.0	603.0
max heart rate	1189.0	139.704794	25.510105	60.0	121.0	140.0	160.0	202.0
oldpeak	1189.0	0.923549	1.086464	-2.6	0.0	0.6	1.6	6.2

5. EXPLORATORY DATA ANALYSIS & VISUAL REPRESENTATION

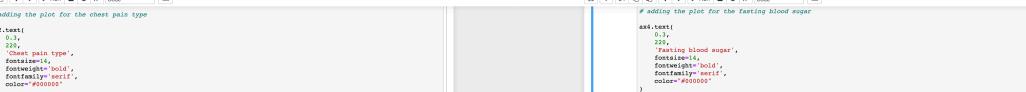
This section is responsible for the deep analysis of the dataset and is applied to investigate the dataset and summarise the key insights. This section provides the basic understanding of the data and its distribution. The exploration can be either done using graphs or python functions.

For this project I have used univariate analysis in which analysis on single attribute is done. This analysis is done for the continuous and categorical attributes of the dataset. For better and intuitive understanding of the data and its distribution I have utilised graphical feature provided by the ‘matplotlib’ and ‘seaborn’ libraries.

Count plotting on categorical data

`countplot()` method is used to Show the counts of observations in each categorical bin using graph bars. A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable, so you can compare counts across nested variables.

HEART-SEIZURE ANALYSIS



The image shows two side-by-side Jupyter Notebook interfaces. Both notebooks have the title "jupyter heart_health_detection_main.ipynb". The left notebook has a status bar at the bottom indicating "Not Trusted | Python 3 (ipykernel)". The right notebook also has a status bar indicating "Not Trusted | Python 3 (ipykernel)". The code in both notebooks is identical, consisting of several sections of text and plot-related code. The code includes imports like `ax2.text()`, `ax3.text()`, and `ax4.text()`. It also contains calls to `ab.countplot()` and `ax3.set_xlabel("")` and `ax3.set_ylabel("")` methods. The overall structure suggests a step-by-step analysis or visualization of heart health data.

```
# adding the plot for the chest pain type
ax2.text(0.1,
         225,
         'Chest pain type',
         fontsize=14,
         fontweight='bold',
         fontfamily='serif',
         color="#000000")
ax2.grid(color="#000000", linestyle='-', axis='y', zorder=0, dashes=[1,1])
ab.countplot(x=ax2, data_final_dataframe, x='chest pain type', palette=countplot_colors)
ax2.set_xlabel('')
ax2.set_ylabel('')

# adding the plot for the major vessels
ax3.text(0.1,
         225,
         'Major vessels',
         fontsize=14,
         fontweight='bold',
         fontfamily='serif',
         color="#000000")
ax3.grid(color="#000000", linestyle='-', axis='y', zorder=0, dashes=[1,1])
ab.countplot(x=ax3, data_final_dataframe, x='major_vessels', palette=countplot_colors)
ax3.set_xlabel('')
ax3.set_ylabel('')

# adding the plot for the fasting blood sugar
ax4.text(0.1,
         225,
         'Fasting blood sugar',
         fontsize=14,
         fontweight='bold',
         fontfamily='serif',
         color="#000000")
ax4.grid(color="#000000", linestyle='-', axis='y', zorder=0, dashes=[1,1])
ab.countplot(x=ax4, data_final_dataframe, x='fasting_blood_sugar', palette=countplot_colors)
ax4.set_xlabel('')
ax4.set_ylabel('')

# adding the plot for the resting ecg
ax5.text(0.1,
         225,
         'Resting ecg',
         fontsize=14,
         fontweight='bold',
         fontfamily='serif',
         color="#000000")
ax5.grid(color="#000000", linestyle='-', axis='y', zorder=0, dashes=[1,1])
ab.countplot(x=ax5, data_final_dataframe, x='resting_ecg', palette=countplot_colors)
ax5.set_xlabel('')
ax5.set_ylabel('')

# adding the plot for the ST slope
ax6.text(0.1,
```

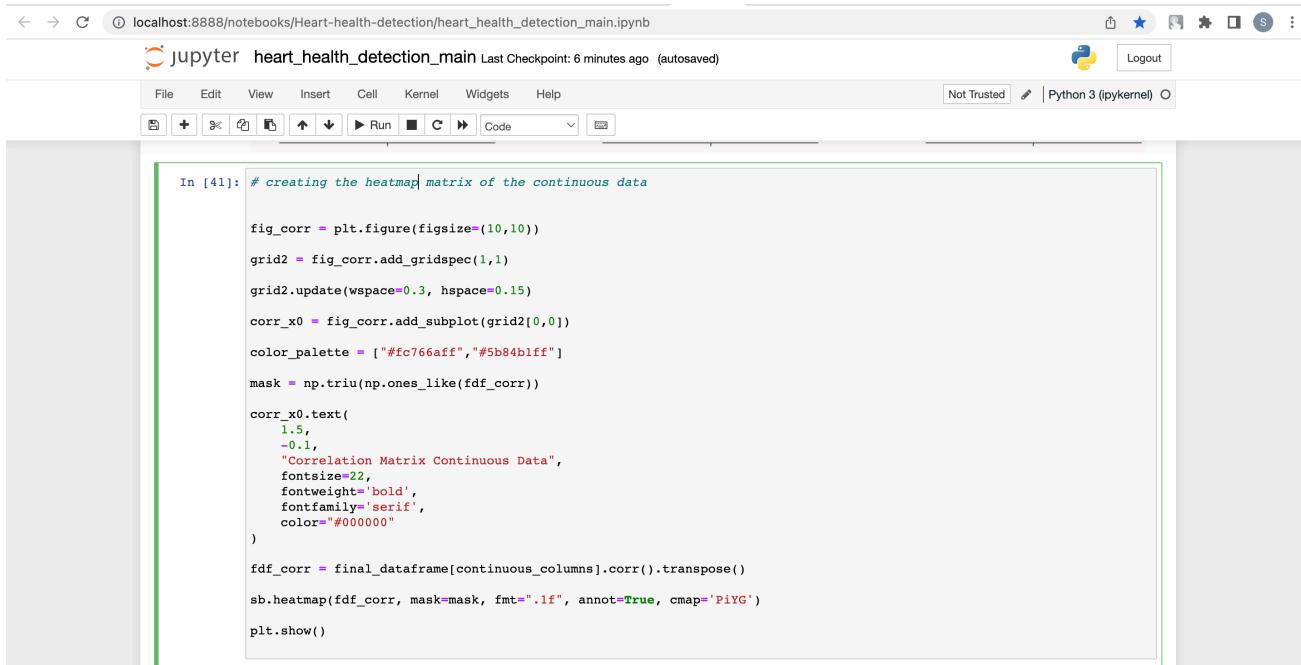
```
# locating each plot of the figure created
for s in ["top","right","left"]:
    ax[s].set_axis_off()
    ax[s].spines[s].set_visible(False)
    ax[s].spines[s].set_color("black")
    ax[s].spines[s].set_alpha(0.5)
    ax[s].spines[s].set_zorder(1)
    ax[s].spines[s].set_alpha(0.5)
```

Box plotting on continuous data

A box plot, also known as a whisker plot, is used to display a summary of a set of data values with attributes such as minimum value, first quartile, median, third quartile, and maximum value. A box is created from the first quartile to the third quartile, where there is also a box with a vertical line through the median. Here the x-axis represents the data to be plotted and the y-axis represents the frequency distribution.

Heatmap matrix of continuous data

A heatmap is defined as a graphical representation of data using colours to visualise matrix values. In this case, to represent more common values or higher activity, a lighter colour is usually used, while to represent less common values or activity values, darker colours are preferred. Heatmaps are also defined by the name of the shading matrix. Heatmaps in Seaborn can be drawn using the `seaborn.heatmap()` function.



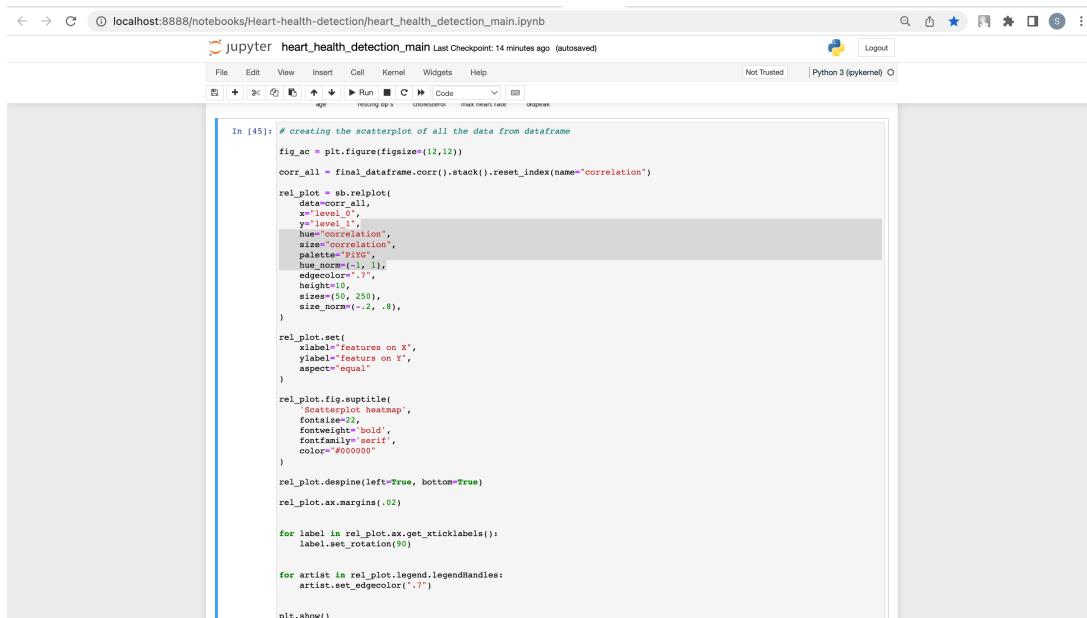
```
In [41]: # creating the heatmap matrix of the continuous data

fig_corr = plt.figure(figsize=(10,10))
grid2 = fig_corr.add_gridspec(1,1)
grid2.update(wspace=0.3, hspace=0.15)
corr_x0 = fig_corr.add_subplot(grid2[0,0])
color_palette = ["#fc766aff", "#5b84bfff"]
mask = np.triu(np.ones_like(fdf_corr))

corr_x0.text(
    1.5,
    -0.1,
    "Correlation Matrix Continuous Data",
    fontsize=22,
    fontweight='bold',
    fontfamily='serif',
    color="#000000"
)
fdf_corr = final_dataframe[continuous_columns].corr().transpose()
sb.heatmap(fdf_corr, mask=mask, fmt=".1f", annot=True, cmap='PiYG')
plt.show()
```

Scatterplot matrix of all data

Scatterplot is used with several semantic groupings which can help to understand well in a graph. They can plot two-dimensional graphics that can be



```
In [45]: # creating the scatterplot of all the data from dataframe

fig_ac = plt.figure(figsize=(12,12))
corr_all = final_dataframe.corr().stack().reset_index(name="correlation")
rel_plot = sb.relplot(
    data=corr_all,
    x="level_0",
    y="level_1",
    hue="correlation",
    size="correlation",
    palette="PiYG",
    hue_norm=(-1, 1),
    edgecolor=".7",
    height=10,
    sizes=(50, 250),
    size_norm=(-.2, .8),
)
rel_plot.set(
    xlabel="features on X",
    ylabel="features on Y",
    aspect="equal"
)
rel_plot.fig.subtitle(
    "Scatterplot heatmap",
    fontsize=22,
    fontweight='bold',
    fontfamily='serif',
    color="#000000"
)
rel_plot.despine(left=True, bottom=True)
rel_plot.ax.margins(.02)

for label in rel_plot.ax.get_xticklabels():
    label.set_rotation(90)

for artist in rel_plot.legend.legendHandles:
    artist.set_edgecolor('.7')

plt.show()
```

enhanced by mapping up to three additional variables while using the semantics of hue, size, and style parameters. All the parameter control visual semantic which are used to identify the different subsets. Using redundant semantics can be helpful for making graphics more accessible.

6. PACKAGES FOR AI MODEL

There are set of packages that we need in order to ready and train a predictive model. Here is the list of packages that will be needed-

RobustScaler- This technique of ML is used when there is a presence of outliers and the dataset.

train_test_split- This package is used to divide the dataset for training and testing suite of data.

Torch- this package provide high level features like - **Tensor computation (like NumPy) with strong GPU acceleration & Deep neural networks built on a tape-based autograd system**

Scikit Learn- It is a simple and efficient tools for predictive data analysis. It is built on NumPy, SciPy, and matplotlib. For regression related problems such as the one used in this project Scikit provides feature like Linear Regression learning models. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

7. TRAINING MODEL WITH DATASET

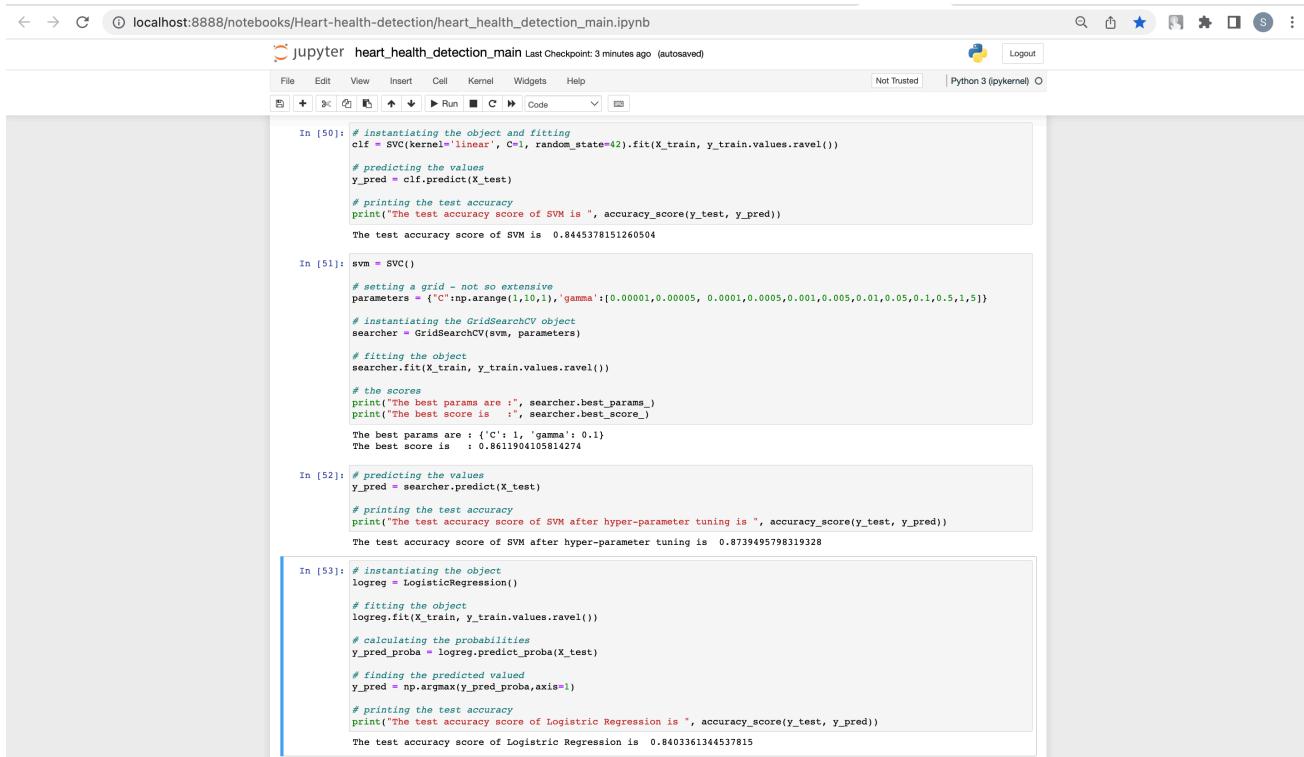
The process of training the model is quite simple. We first break the dataset into continuous and categorical sub-datasets. We then divide our whole dataset into two parts. One is used for training the model and other is used for testing the prediction accuracy. For splitting we use `train_test_split()` method and we keep 20% of our total data for testing purposes.

We then proceed to use method like `fit_transform()` for training the model. The method `SVC()` is responsible to creating instance of model that predicts the target variable for the testing suite. We can also check the efficiency of the model using `accuracy_score()`.

8. HYPER PARAMETER TUNING FOR IMPROVING PREDICTION.

By training a model with existing data, we are able to fit the model parameters. However, there is another kind of parameter, known as Hyperparameters, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.

For this project the methodology I have used is GridSearchCV(). In this methodology it searches for the best set of hyperparameters from a grid of hyperparameters values. GridSearchCV will go through all the intermediate combinations of hyperparameters which makes grid search computationally very expensive and will return the best possible combination.



The screenshot shows a Jupyter Notebook interface with three code cells (In [50], In [51], In [52]) and one additional cell (In [53]) below them. The notebook title is "heart_health_detection_main".

```

In [50]: # instantiating the object and fitting
clf = SVC(kernel='linear', C=1, random_state=42).fit(X_train, y_train.values.ravel())
# predicting the values
y_pred = clf.predict(X_test)
# printing the test accuracy
print("The test accuracy score of SVM is ", accuracy_score(y_test, y_pred))
The test accuracy score of SVM is  0.8445378151260504

In [51]: svm = SVC()
# setting a grid - not so extensive
parameters = {'C':np.arange(1,10,1),'gamma':[0.0001,0.0005, 0.0001,0.0005,0.001,0.005,0.01,0.05,0.1,0.5,1,5]}
# instantiating the GridSearchCV object
searcher = GridSearchCV(svm, parameters)
# fitting the object
searcher.fit(X_train, y_train.values.ravel())
# the scores
print("The best params are : ", searcher.best_params_)
print("The best score is : ", searcher.best_score_)
The best params are : {'C': 1, 'gamma': 0.1}
The best score is : 0.8611904105814274

In [52]: # predicting the values
y_pred = searcher.predict(X_test)
# printing the test accuracy
print("The test accuracy score of SVM after hyper-parameter tuning is ", accuracy_score(y_test, y_pred))
The test accuracy score of SVM after hyper-parameter tuning is  0.8739495798319328

In [53]: # instantiating the object
logreg = LogisticRegression()
# fitting the object
logreg.fit(X_train, y_train.values.ravel())
# calculating the probabilities
y_pred_proba = logreg.predict_proba(X_test)
# finding the predicted values
y_pred = np.argmax(y_pred_proba, axis=1)
# printing the test accuracy
print("The test accuracy score of Logistic Regression is ", accuracy_score(y_test, y_pred))
The test accuracy score of Logistic Regression is  0.8402361344537815

```

We can also measure the accuracy of the model using method like accuracy_score() on the testing suite of the sub-dataset. We have package methods like logisticRegression() which builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”.

RESULTS

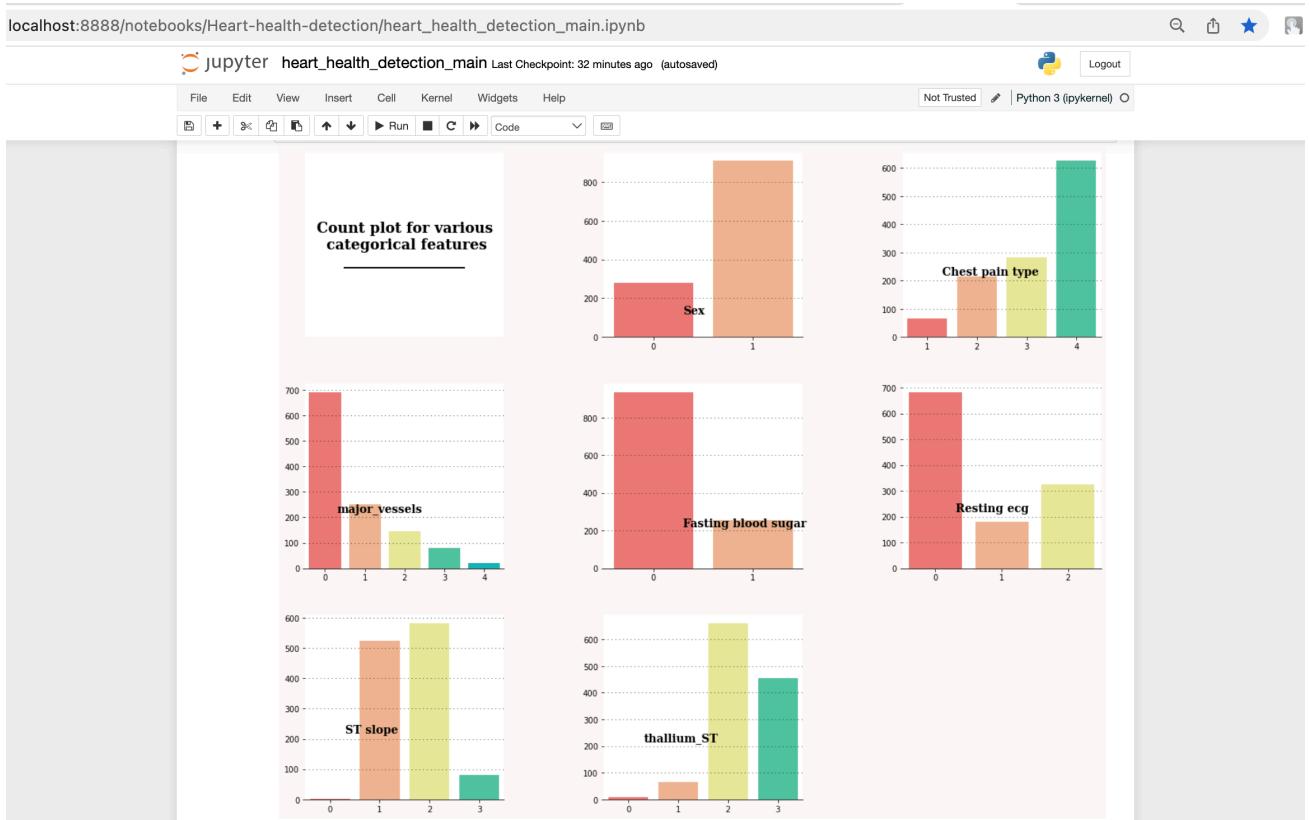
In this section of the project report we will discuss about the results of the operations that we performed on dataset. We used techniques that describes the data graphically and mathematically.

GRAPHICAL RESULTS ON DATASETS

COUNTPLOT OF THE CATEGORICAL COLUMNS

This graph represents the count plot of categorical columns from Dataset. It provides insight on the distribution of the dataset among the possible categories. e.g for Sex column there is only two possible categories, this graph informs us that around 300 subjects are of 'Sex-0' type and rest are of 'Sex-1' type.

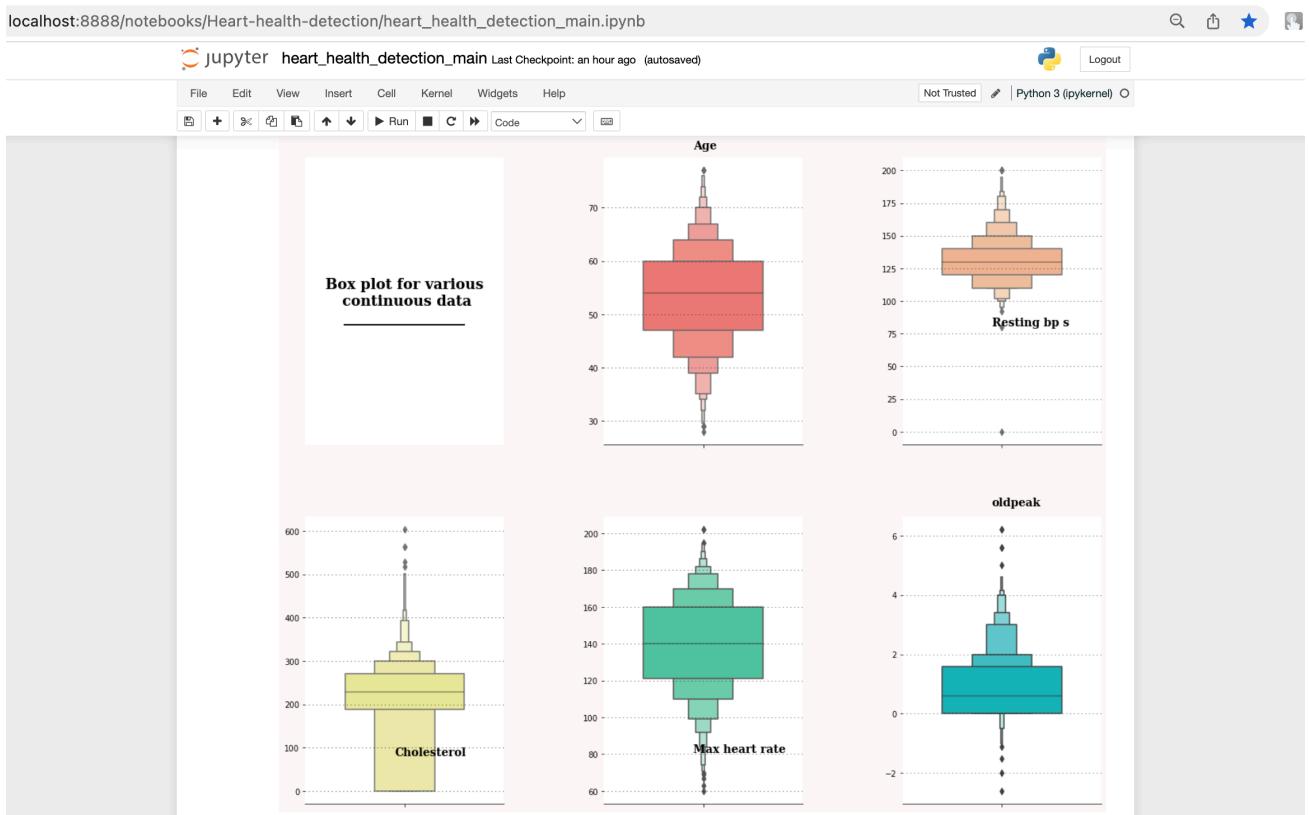
Similarly other columns categories can also be easily understood with the help of this graph.



BOXPLOT OF THE CONTINUOUS COLUMNS

This graph represents the boxplot of the continuous columns from Dataset. It provides an insight on minimum value, first quartile, median, third quartile, and maximum value of the column data. A box is created from first quartile to the third quartile and there is a median line from the box. Box plot also informs about the location of the outliers present in the column data.

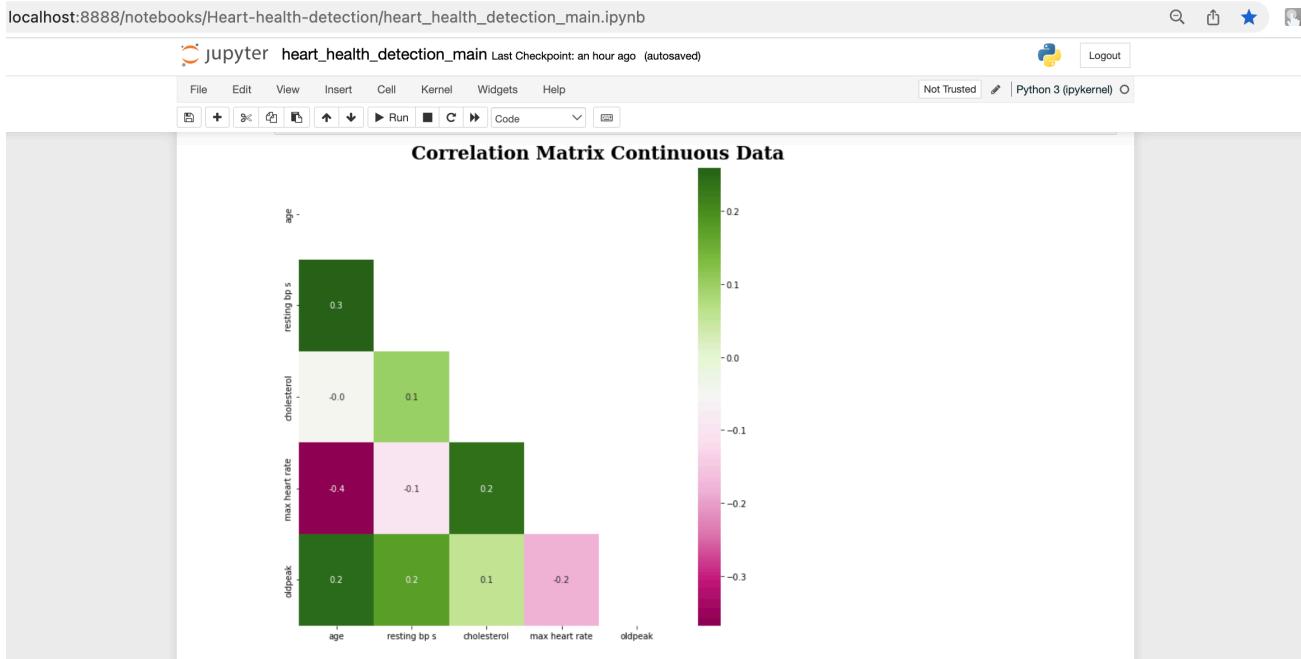
e.g. for the column ‘Resting Blood pressure’ there is one outlier which has the value 0. This value might not be useful for the prediction process and might lead to affect the prediction accuracy. Hence with help from this graphical data we can conclude that during Training of the AI model we should adopt a technique that removes the outliers from the columns data.



HEATMAP MATRIX OF THE CONTINUOUS COLUMNS

This graph represents the Heatmap of the continuous columns from Dataset. It provides an insight on the correlation between the two columns. The lighter colour represent the strong correlation between two columns and darker colour

Represents weak correlation between two columns.

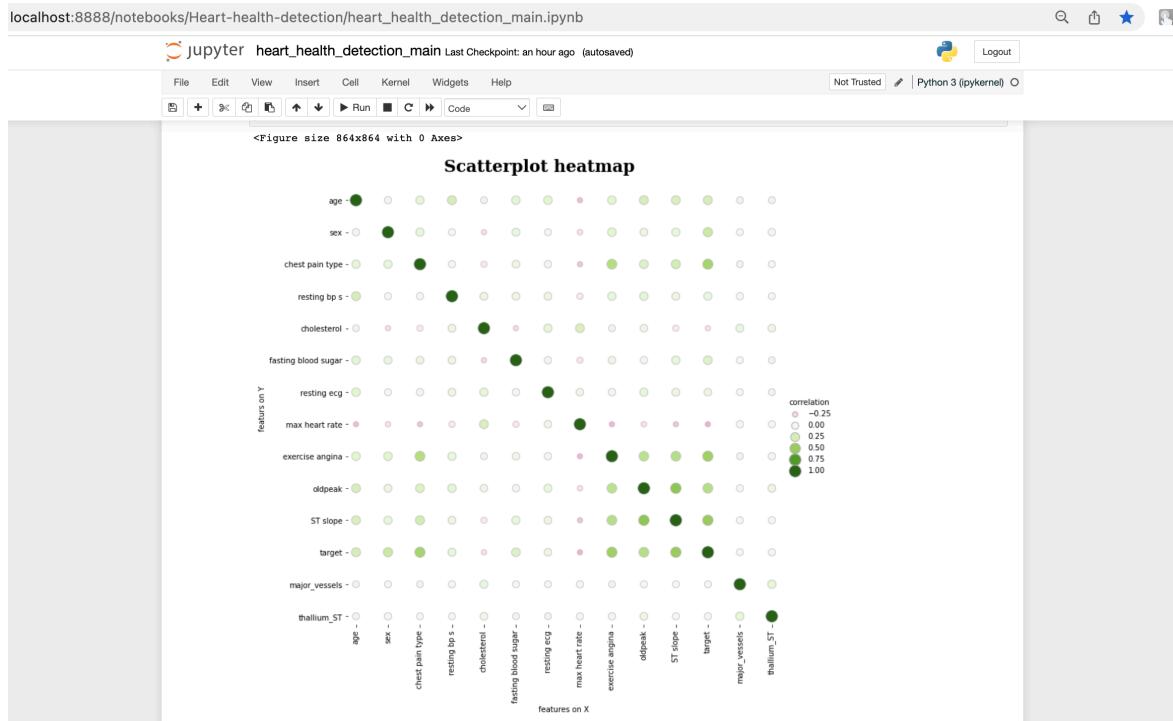


SCATTERPLOT MATRIX OF ALL COLUMNS

This graph represents the scatterplot of all the columns in dataset. It provides an insight on the relationship between the two columns. If there is a strong relation between two columns then they are presented with darker colour.

e.g It can be seen that there is a strong relationship between 'chest type pain' and 'exercise angina'. Which can help conclude that a person who is prone to heart attack will experience pain in the chest during physical activities or workout sessions.

HEART-SEIZURE ANALYSIS



INSIGHTS

From the operation that I have performed and visual presentation of the dataset, here are the insights on the data

1. The dataset does not contain the null values and the data type of each column is of either INTEGER type or FLOAT type.
2. It can be seen that there are outliers present in continuous columns of Dataset.
3. There are less than twice subjects who have high Fasting blood sugar.
4. The scatterplot suggests that there is a strong relation between columns like 'Chest Pain', 'Exercise Angina' and 'ST Slope'.
5. There is no strong relation between the two categories of columns evident from scatterplot of all the columns from dataset.

REFERENCES

1. <https://news.abplive.com/science/world-heart-day-2022-70-of-heart-attack-deaths-last-year-occurred-in-30-60-age-group-1555818>
2. <https://www.indiatoday.in/health/story/why-there-is-a-sudden-surge-in-cardiac-arrest-cases-explain-experts-1998096-2022-09-08>
3. <https://www.datacamp.com/courses/introduction-to-data-visualization-with-matplotlib>
4. <https://www.youtube.com/watch?v=j9KpNo34Rp0>
5. <https://www.youtube.com/watch?v=CpWzLuQOg8g>
6. www.youtube.com/watch?v=wetW1TFk1Sk
7. www.youtube.com/watch?v=ssNyZlc7_i0
8. https://practice.geeksforgeeks.org/courses/data-science-live?utm_source=GfG&utm_medium=gfg_submenu&utm_campaign=DS_Submenu/