CHAPTER - I

INTRODUCTION

Heart diseases remain a significant global health concern, contributing to a substantial burden of morbidity and mortality worldwide. They encompass a range of conditions affecting the heart and blood vessels, including coronary artery disease, heart failure, arrhythmias, and congenital heart defects, among others. According to the World Health Organization (WHO), cardiovascular diseases (CVDs) are the leading cause of death globally, accounting for an estimated 17.9 million deaths annually.

Given the prevalence and severity of heart diseases, there is an increasing emphasis on leveraging advanced technologies, data analytics, and machine learning techniques for the analysis, prediction, and management of these conditions. Through comprehensive data analysis, including demographic information, medical history, lifestyle factors, and clinical parameters, researchers and healthcare professionals can gain valuable insights into the risk factors, patterns, and progression of heart diseases. This project aims to explore the application of data analytics and machine learning algorithms to analyze heart disease data sets. By employing techniques such as data preprocessing, feature selection, and predictive modeling, the project seeks to develop robust algorithms capable of accurately predicting the likelihood of heart disease occurrence, assessing disease severity, and guiding personalized treatment strategies.

The significance of this project lies in its potential to enhance early detection, risk stratification, and management of heart diseases, thereby improving patient outcomes and reducing healthcare costs. By harnessing the power of data-driven approaches, we aim to contribute to the ongoing efforts to combat the global burden of heart diseases and promote cardiovascular health in populations worldwide.

CHAPTER - II

BUSINESS UNDERSTADING

Heart disease is a prevalent and significant health concern in the United States, contributing to a substantial number of deaths across various racial groups. The Centres for Disease Control and Prevention (CDC) highlights key risk factors such as high blood pressure, high cholesterol, smoking, diabetes, obesity, physical inactivity, and excessive alcohol consumption. Identifying and preventing these factors are crucial for proactive healthcare management.

To address this challenge, we propose a data-driven approach using Python and Tableau to predict heart disease based on relevant health indicators. Leveraging machine learning methods on a comprehensive dataset sourced from the CDC's Behaviorale Risk Factor Surveillance System (BRFSS), we aim to identify patterns that can effectively predict a patient's condition. This initiative aligns with the broader goal of enhancing preventive healthcare strategies by enabling early detection of potential heart-related issues. This document outlines the business understanding of utilizing the Behavioral Risk Factor Surveillance System (BRFSS) dataset, originating from the CDC, to advance heart disease prevention through data-driven insights.

1. Significance of Heart Disease Prevention:

Heart disease is a pervasive threat, with approximately 47% of Americans carrying at least one of three major risk factors: high blood pressure, high cholesterol, and smoking. Addressing these risk factors is crucial for mitigating the impact of heart disease and improving overall public health.

2. Origin and Scope of the BRFSS Dataset:

The BRFSS dataset, a cornerstone of health research, is derived from the CDC's annual telephone surveys conducted since 1984. The scope of this dataset has evolved from its inception in 15 states to encompassing all 50 states, the District of Columbia, and three U.S. territories. With over 400,000 adult interviews conducted annually, the BRFSS is recognized as the largest continuously conducted health survey system globally. It captures a comprehensive array of health-related variables, including demographic information, major risk factors, and other key indicators.

3. Business Potential and Analytics Opportunities:

3.1 Predictive Analytics for Patient Condition:

The dataset's richness presents a valuable opportunity to apply machine learning methods for predictive analytics. By detecting patterns in the data, we aim to develop models that predict a patient's condition, enabling early interventions and personalized healthcare strategies.

3.2 Identification of Key Risk Factors:

Utilizing advanced analytics, we seek to identify and prioritize key risk factors contributing to heart disease prevalence. This insight will inform targeted interventions and public health campaigns tailored to address specific risk factors across diverse demographic groups.

3.3 Strategic Healthcare Planning:

Informed by data-driven insights, healthcare organizations can develop strategic plans that prioritize resources for heart disease prevention. This includes the formulation of preventive measures, awareness campaigns, and community health initiatives.

4. Challenges and Considerations:

4.1 Data Privacy and Ethical Handling:

Given the sensitive nature of health data, ensuring privacy and adhering to ethical standards is paramount. Robust protocols will be established to safeguard patient information and maintain compliance with privacy regulations.

4.2 Data Quality Assurance:

To derive meaningful insights, meticulous attention will be given to data quality assurance. This involves addressing any inconsistencies or inaccuracies within the dataset to enhance the reliability of analytics outcomes.

CHAPTER-III

TOOLS FOR ANALYTICS

This project leveraged several powerful analytical tools to unlock insights from the dataset: Python, Tableau, and Power BI. These tools provided a versatile approach, allowing for data manipulation, analysis, and clear visualization of the findings.

Analytical tools are the modern world's secret weapon for extracting knowledge from data. They crunch massive amounts of information, find patterns and trends, and help us understand what's going on in the world around us. This can be anything from business sales figures to social media sentiment to weather patterns. By analysing this data, companies can make better decisions, scientists can unlock new discoveries, and governments can tackle complex challenges. In short, analytical tools are a game-changer in the modern world.

Python:

Python stands out as a highly versatile programming language, renowned for its simplicity, flexibility, and expansive ecosystem of libraries and frameworks. Widely embraced in the data science community, Python's robust support for data manipulation, analysis, and visualization has solidified its position as a go-to language for professionals in the field. Notable libraries like Pandas, NumPy, Matplotlib, and Seaborn empower users with efficient tools for data processing, statistical analysis, and graphical visualization, enhancing the overall data exploration experience.

Furthermore, Python seamlessly integrates with machine learning libraries such as Scikit-learn, TensorFlow, and PyTorch, facilitating intricate tasks like regression, classification, clustering, and neural network implementations. Python's user-friendly syntax and scalability make it an ideal choice across the spectrum of data analytics projects, from initial exploratory data analysis to the complexities of advanced predictive modeling.

Tableau:

Tableau stands out as a leading data visualization tool, providing users with the ability to craft interactive and visually compelling dashboards and reports. Its user-friendly interface, characterized by intuitive drag-and-drop functionality, enables quick insights extraction from data, making it accessible even to users with limited programming skills. Tableau excels in its robust visualization capabilities, allowing users to generate a diverse range of charts, graphs, maps, and other visual representations to effectively communicate key findings. The tool's seamless integration with various data sources, including databases, spreadsheets, and cloud platforms, streamlines data connectivity and amplifies data exploration capabilities. Through interactive dashboards, Tableau facilitates data-driven decision-making by empowering stakeholders to explore data from different angles, uncovering actionable insights with ease. Its contribution to simplifying the visualization process has positioned Tableau as a cornerstone in the realm of modern data analytics.

Power BI:

Power BI, a robust business intelligence tool developed by Microsoft, serves as a pivotal resource for organizations seeking to streamline data analysis and share insights across their enterprise. Tailored to businesses of varying sizes, Power BI encompasses a comprehensive suite of features, spanning data preparation, modeling, visualization, and collaboration.

Its user-friendly interface, coupled with intuitive drag-and-drop functionality, empowers users to effortlessly craft dynamic and interactive reports and dashboards. Operating on a cloud-based architecture, Power BI facilitates real-time data analysis, granting users the flexibility to access insights seamlessly from any location and device. Beyond its user-centric attributes, Power BI prioritizes data security and governance, boasting robust features that uphold privacy and compliance standards, thereby making it a well-suited choice for widespread deployment within enterprises.

CHAPTER-IV

DATA UNDERSTANDING

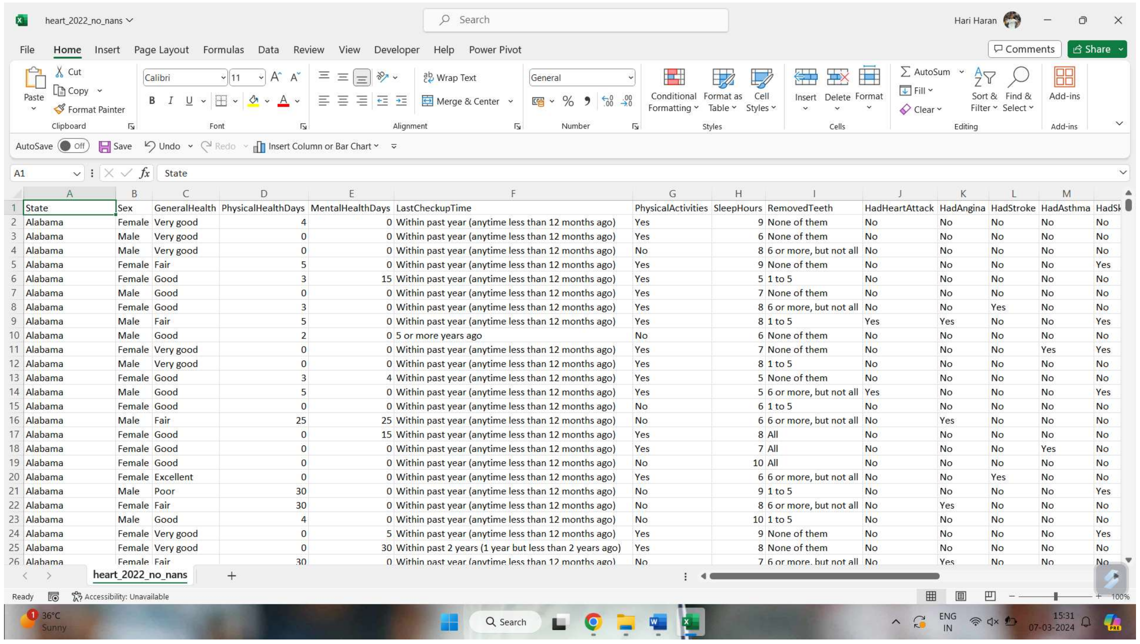
Data understanding is a critical phase in any project that involves working with data, laying the foundation for effective analysis and insights generation. This phase encompasses the exploration and comprehension of the dataset at hand, providing a comprehensive overview of its structure, content, and potential challenges. It involves acquiring an in-depth understanding of the variables, their relationships, and the overall characteristics of the data.

During the data understanding phase, the project team systematically examines the dataset's key attributes, such as its size, format, and origin. Variable types, data distributions, and missing values are thoroughly scrutinized to identify patterns and anomalies. Exploratory data analysis techniques are often employed to gain initial insights, revealing potential trends or outliers that may impact subsequent analyses. Understanding the context and domain-specific nuances of the data is equally crucial. This involves collaboration with domain experts to interpret the significance of each variable and ensure that the analysis aligns with the project's objectives. Furthermore, identifying potential data quality issues, such as inconsistencies or errors, is imperative to ensure the reliability and accuracy of subsequent analyses.

In essence, data understanding serves as the compass guiding the project team through the intricacies of the dataset. It not only lays the groundwork for informed decision-making but also shapes the subsequent steps in data preparation, feature engineering, and modeling. By delving deep into the intricacies of the data, the project team is better equipped to uncover actionable insights, address potential challenges, and ultimately contribute to the successful achievement of project goals.

Dataset Link: https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease/data

Dataset Screenshot:



	State	Sex	GeneralHealth	PhysicalHealthDays	MentalHealthDays	LastCheckupTime	PhysicalActivities	SleepHours	RemovedTeeth	HadHeartAttack	HadAngina	HadStroke	HadAsthma	HadSk
2	Alabama	Female	Very good	4	0	Within past year (anytime less than 12 months ago)	Yes	9	None of them	No	No	No	No	No
3	Alabama	Male	Very good	0	0	Within past year (anytime less than 12 months ago)	Yes	6	None of them	No	No	No	No	No
4	Alabama	Male	Very good	0	0	Within past year (anytime less than 12 months ago)	No	8	6 or more, but not all	No	No	No	No	No
5	Alabama	Female	Fair	5	0	Within past year (anytime less than 12 months ago)	Yes	9	None of them	No	No	No	No	Yes
6	Alabama	Female	Good	3	15	Within past year (anytime less than 12 months ago)	Yes	5	1 to 5	No	No	No	No	No
7	Alabama	Male	Good	0	0	Within past year (anytime less than 12 months ago)	Yes	7	None of them	No	No	No	No	No
8	Alabama	Female	Good	3	0	Within past year (anytime less than 12 months ago)	Yes	8	6 or more, but not all	No	No	Yes	No	No
9	Alabama	Male	Fair	5	0	Within past year (anytime less than 12 months ago)	Yes	8	1 to 5	Yes	Yes	No	No	Yes
10	Alabama	Male	Good	2	0	5 or more years ago	No	6	None of them	No	No	No	No	No
11	Alabama	Female	Very good	0	0	Within past year (anytime less than 12 months ago)	Yes	7	None of them	No	No	No	Yes	Yes
12	Alabama	Male	Very good	0	0	Within past year (anytime less than 12 months ago)	Yes	8	1 to 5	No	No	No	No	No
13	Alabama	Female	Good	3	4	Within past year (anytime less than 12 months ago)	Yes	5	None of them	No	No	No	No	No
14	Alabama	Male	Good	5	0	Within past year (anytime less than 12 months ago)	Yes	5	6 or more, but not all	Yes	No	No	No	Yes
15	Alabama	Female	Good	0	0	Within past year (anytime less than 12 months ago)	No	6	1 to 5	No	No	No	No	No
16	Alabama	Male	Fair	25	25	Within past year (anytime less than 12 months ago)	No	6	6 or more, but not all	No	Yes	No	No	No
17	Alabama	Female	Good	0	15	Within past year (anytime less than 12 months ago)	Yes	8	All	No	No	No	No	No
18	Alabama	Female	Good	0	0	Within past year (anytime less than 12 months ago)	Yes	7	All	No	No	No	Yes	No
19	Alabama	Female	Good	0	0	Within past year (anytime less than 12 months ago)	No	10	All	No	No	No	No	No
20	Alabama	Female	Excellent	0	0	Within past year (anytime less than 12 months ago)	Yes	6	6 or more, but not all	No	No	Yes	No	No
21	Alabama	Male	Poor	30	0	Within past year (anytime less than 12 months ago)	No	9	1 to 5	No	No	No	No	Yes
22	Alabama	Female	Fair	30	0	Within past year (anytime less than 12 months ago)	No	8	6 or more, but not all	No	Yes	No	No	No
23	Alabama	Male	Good	4	0	Within past year (anytime less than 12 months ago)	No	10	1 to 5	No	No	No	No	No
24	Alabama	Female	Very good	0	5	Within past year (anytime less than 12 months ago)	Yes	9	None of them	No	No	No	No	Yes
25	Alabama	Female	Very good	0	30	Within past 2 years (1 year but less than 2 years ago)	Yes	8	None of them	No	No	No	No	No
26	Alabama	Female	Fair	30	0	Within past year (anytime less than 12 months ago)	No	7	6 or more, but not all	No	Yes	No	No	No

Variables:

In the context of a dataset, variables refer to the individual characteristics or attributes that are measured and recorded for each observation or unit. These variables encapsulate the diverse aspects of the subjects under study and play a fundamental role in shaping the dataset's structure. Each column in the dataset represents a specific variable, and each row corresponds to a unique observation.

Variables can be broadly categorized into two main types: independent variables and dependent variables. Independent variables are factors that are manipulated or controlled in an experiment or study to observe their impact on the dependent variable. The dependent variable, on the other hand, is the outcome or response that is influenced by the independent variables. Moreover, variables can be further classified as categorical or numerical. Categorical variables represent categories or labels and are often non-numeric, such as gender or region. Numerical variables, on the other hand, are measurable quantities and can be either discrete or continuous. Examples include age, income, or the number of occurrences of an event.

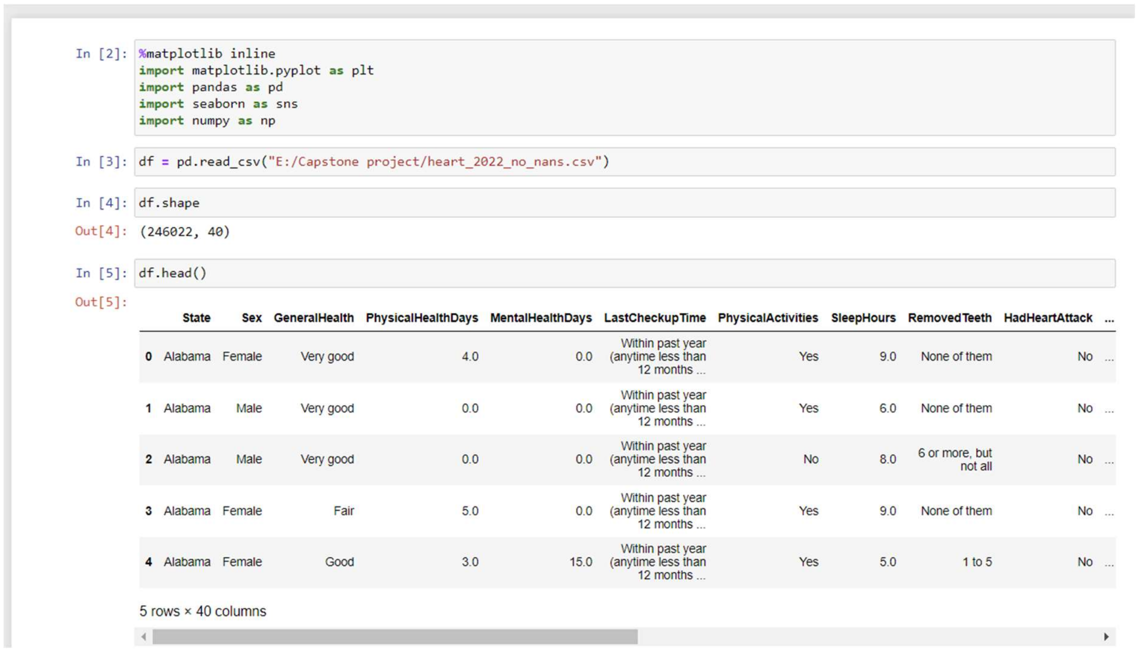
Understanding the nature and characteristics of each variable is crucial for effective data analysis. Exploratory data analysis involves examining the distribution, central tendency, and variability of numerical variables, while for categorical variables, the focus is on understanding the frequency and distribution of different categories. Variables serve as the building blocks for statistical modeling and machine learning algorithms, and a thorough comprehension of their meaning and impact is essential for deriving meaningful insights and making informed decisions in various fields, from scientific research to business analytics.

There are 40 variables(columns) and 246023 variable(row) in this dataset.

- State: The U.S. state where the individual resides.
- Sex: Gender of the individual (Male or Female).
- General Health: Self-reported general health status of the individual.
- Physical Health Days: Number of days in the past 30 days that physical health was not good.
- Mental Health Days: Number of days in the past 30 days that mental health was not good.
- Last Checkup Time: Time since the last routine checkup or health examination.
- Physical Activities: Frequency of engaging in physical activities or exercises.
- Sleep Hours: Average number of hours of sleep per night.
- Removed Teeth: Number of permanent teeth removed due to dental issues
- Had Heart Attack: Whether the individual has had a heart attack.
- Had Angina: Whether the individual has experienced angina (chest pain or discomfort).
- Had Stroke: Whether the individual has had a stroke.
- Had Asthma: Whether the individual has had asthma.
- Had Skin Cancer: Whether the individual has had skin cancer.
- Had COPD: Whether the individual has had Chronic Obstructive Pulmonary Disease (COPD).
- Had Depressive Disorder: Whether the individual has had a depressive disorder.
- Had Kidney Disease: Whether the individual has had kidney disease.
- Had Arthritis: Whether the individual has had arthritis.
- Had Diabetes: Whether the individual has had diabetes.
- Deaf Or Hard of Hearing: Whether the individual is deaf or hard of hearing.
- Blind Or Vision Difficulty: Whether the individual has blindness or vision difficulty.
- Difficulty Concentrating: Self-reported difficulty in concentrating.
- Difficulty Walking: Self-reported difficulty in walking.
- Difficulty Dressing Bathing: Self-reported difficulty in dressing or bathing.
- Difficulty Errands: Self-reported difficulty in running errands.

- Smoker Status: Current smoking status of the individual (smoker, former smoker, non-smoker).
- E-cigarette Usage: Whether the individual uses e-cigarettes.
- Chest Scan: Whether the individual has had a chest scan.
- Race Ethnicity Category: Categorized race or ethnicity of the individual.
- Age Category: Categorized age group of the individual.
- Height In Meters: Height of the individual in meters.
- Weight In Kilograms: Weight of the individual in kilograms.
- BMI: Body Mass Index calculated from height and weight.
- Alcohol Drinkers: Whether the individual consumes alcohol.
- HIV Testing: Whether the individual has undergone HIV testing.
- FluVaxLast12: Whether the individual received a flu vaccine in the last 12 months.
- Pneumovax Ever: Whether the individual has ever received a pneumonia vaccine.
- TetanusLast10Tdap: Time since the last tetanus vaccination (in the last 10 years, received Tdap).
- High Risk Last Year: Whether the individual has been considered at high risk for the past year.
- CovidPos: Whether the individual tested positive for COVID-19.

The comprehension of data was accomplished by leveraging Python, a versatile programming language well-known for its prowess in data analysis and manipulation.

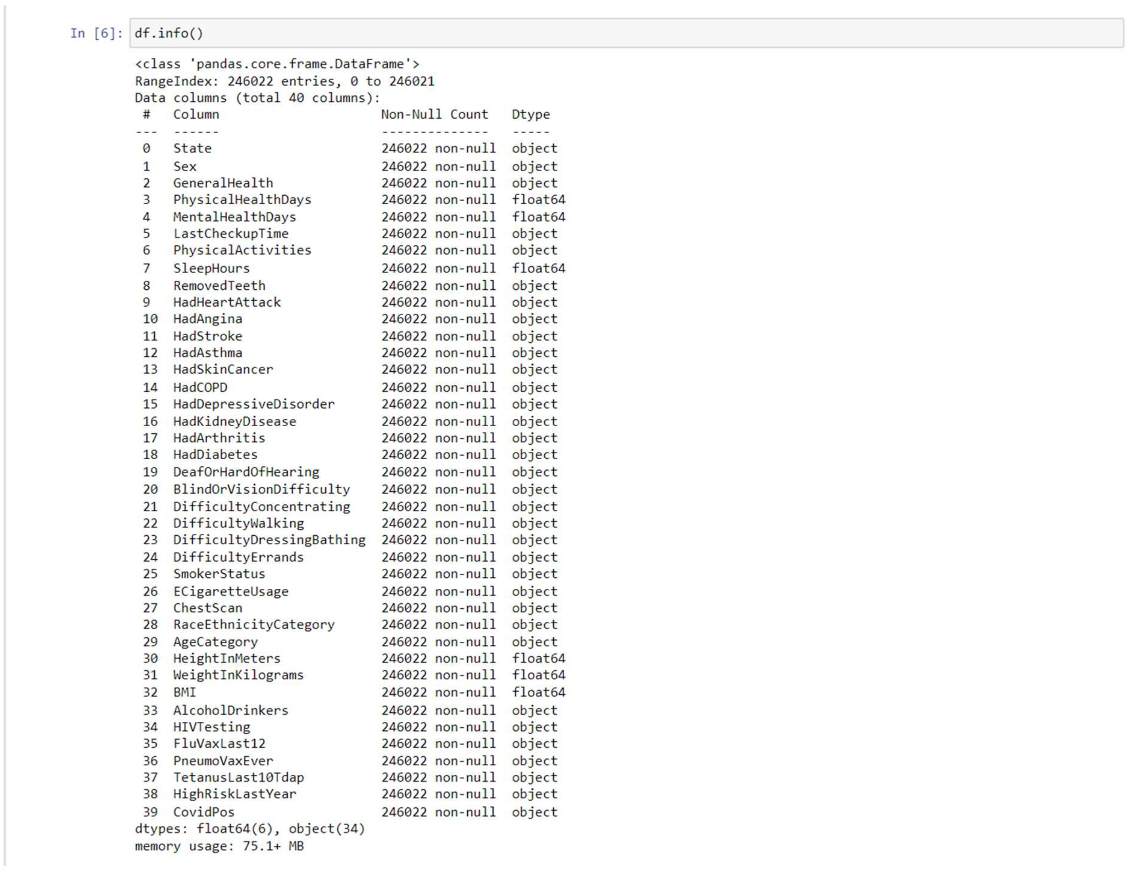


```
In [2]: %matplotlib inline
        import matplotlib.pyplot as plt
        import pandas as pd
        import seaborn as sns
        import numpy as np

In [3]: df = pd.read_csv("E:/Capstone project/heart_2022_no_nans.csv")

In [4]: df.shape
Out[4]: (246022, 40)

In [5]: df.head()
Out[5]:
```

	State	Sex	GeneralHealth	PhysicalHealthDays	MentalHealthDays	LastCheckupTime	PhysicalActivities	SleepHours	RemovedTeeth	HadHeartAttack	...
0	Alabama	Female	Very good	4.0	0.0	Within past year (anytime less than 12 months ...	Yes	9.0	None of them	No	...
1	Alabama	Male	Very good	0.0	0.0	Within past year (anytime less than 12 months ...	Yes	6.0	None of them	No	...
2	Alabama	Male	Very good	0.0	0.0	Within past year (anytime less than 12 months ...	No	8.0	6 or more, but not all	No	...
3	Alabama	Female	Fair	5.0	0.0	Within past year (anytime less than 12 months ...	Yes	9.0	None of them	No	...
4	Alabama	Female	Good	3.0	15.0	Within past year (anytime less than 12 months ...	Yes	5.0	1 to 5	No	...

5 rows × 40 columns

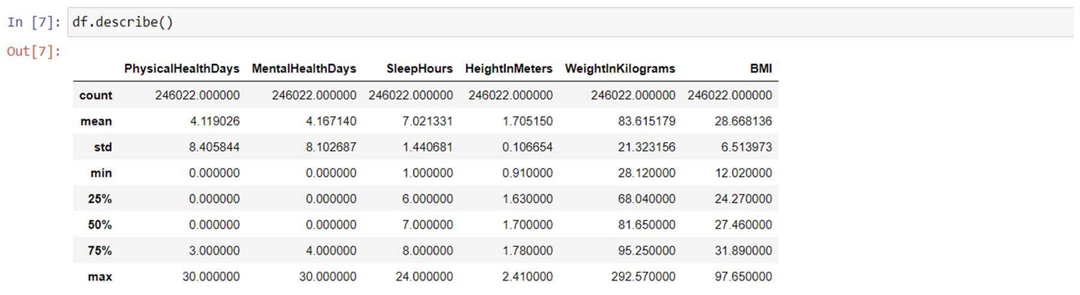
- Python libraries, including pandas and matplotlib, were employed to streamline various data-related tasks such as loading, exploration, visualization, and basic statistical analysis. These libraries significantly enhance the efficiency and effectiveness of the data analysis process, facilitating a thorough examination and interpretation of the dataset.
- To incorporate the dataset into the analysis pipeline, the pd.read_csv function in Python was utilized, assigning it a specific name for ease of reference in subsequent analyses. The shape function was employed to unveil the dimensions of the dataset, shedding light on its size and structure, revealing that it comprises 246022 observations and 40 variables.
- To gain a quick overview of the dataset's structure and content, the head function was deployed, showcasing a sample of the data—typically the initial five observations. This approach provides a concise glimpse into the dataset, aiding in the initial assessment of its characteristics.



```
In [6]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246022 entries, 0 to 246021
Data columns (total 40 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   State                      246022 non-null  object
 1   Sex                        246022 non-null  object
 2   GeneralHealth              246022 non-null  object
 3   PhysicalHealthDays         246022 non-null  float64
 4   MentalHealthDays           246022 non-null  float64
 5   LastCheckupTime            246022 non-null  object
 6   PhysicalActivities         246022 non-null  object
 7   SleepHours                 246022 non-null  float64
 8   RemovedTeeth               246022 non-null  object
 9   HadHeartAttack             246022 non-null  object
 10  HadAngina                  246022 non-null  object
 11  HadStroke                  246022 non-null  object
 12  HadAsthma                  246022 non-null  object
 13  HadSkinCancer              246022 non-null  object
 14  HadCOPD                    246022 non-null  object
 15  HadDepressiveDisorder      246022 non-null  object
 16  HadKidneyDisease           246022 non-null  object
 17  HadArthritis               246022 non-null  object
 18  HadDiabetes                246022 non-null  object
 19  DeafOrHardOfHearing        246022 non-null  object
 20  BlindOrVisionDifficulty    246022 non-null  object
 21  DifficultyConcentrating    246022 non-null  object
 22  DifficultyWalking          246022 non-null  object
 23  DifficultyDressingBathing  246022 non-null  object
 24  DifficultyErrands          246022 non-null  object
 25  SmokerStatus               246022 non-null  object
 26  ECigaretteUsage            246022 non-null  object
 27  ChestScan                  246022 non-null  object
 28  RaceEthnicityCategory      246022 non-null  object
 29  AgeCategory                246022 non-null  object
 30  HeightInMeters             246022 non-null  float64
 31  WeightInKilograms          246022 non-null  float64
 32  BMI                        246022 non-null  float64
 33  AlcoholDrinkers            246022 non-null  object
 34  HIVTesting                 246022 non-null  object
 35  FluVaxLast12               246022 non-null  object
 36  PneumoVaxEver              246022 non-null  object
 37  TetanusLast10Tdap          246022 non-null  object
 38  HighRiskLastYear           246022 non-null  object
 39  CovidPos                   246022 non-null  object
dtypes: float64(6), object(34)
memory usage: 75.1+ MB
```

- The utilization of the info function serves to acquire concise details about the dataset, encompassing information such as data types, non-null values, and memory usage. This quick overview proves invaluable in understanding the dataset's structure and characteristics, facilitating subsequent analysis and interpretation.

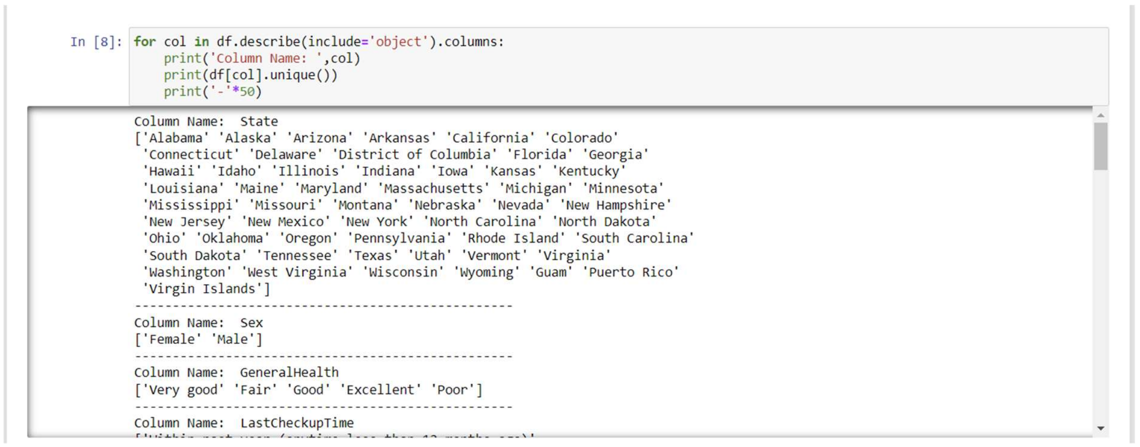
- Within this dataset, a breakdown reveals the presence of 6 floating-point variables, and 34 object variables, each delineating specific aspects of the data for a comprehensive analytical and interpretative approach.



```
In [7]: df.describe()
Out[7]:
```

	PhysicalHealthDays	MentalHealthDays	SleepHours	HeightInMeters	WeightInKilograms	BMI
count	246022.000000	246022.000000	246022.000000	246022.000000	246022.000000	246022.000000
mean	4.119026	4.167140	7.021331	1.705150	83.615179	28.668136
std	8.405844	8.102687	1.440681	0.106654	21.323156	6.513973
min	0.000000	0.000000	1.000000	0.910000	28.120000	12.020000
25%	0.000000	0.000000	6.000000	1.630000	68.040000	24.270000
50%	0.000000	0.000000	7.000000	1.700000	81.650000	27.460000
75%	3.000000	4.000000	8.000000	1.780000	95.250000	31.890000
max	30.000000	30.000000	24.000000	2.410000	292.570000	97.650000

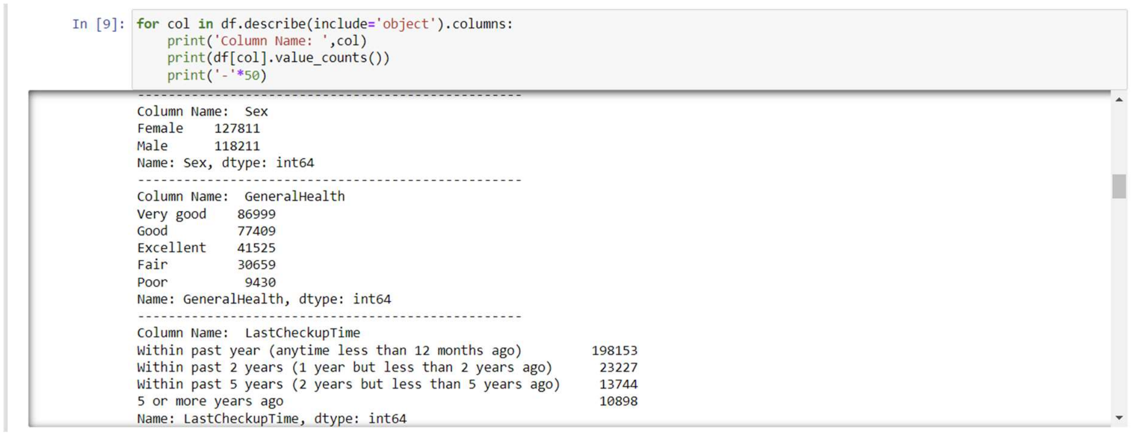
- The describe function is utilized to generate summary statistics for numerical variables in the dataset. These statistics encompass key details like count, mean, standard deviation, minimum, maximum, and quartile values, providing valuable insights into the distribution and characteristics of the data.



```
In [8]: for col in df.describe(include='object').columns:
            print('Column Name: ',col)
            print(df[col].unique())
            print('-'*50)
```

```
Column Name:  State
['Alabama' 'Alaska' 'Arizona' 'Arkansas' 'California' 'Colorado'
 'Connecticut' 'Delaware' 'District of Columbia' 'Florida' 'Georgia'
 'Hawaii' 'Idaho' 'Illinois' 'Indiana' 'Iowa' 'Kansas' 'Kentucky'
 'Louisiana' 'Maine' 'Maryland' 'Massachusetts' 'Michigan' 'Minnesota'
 'Mississippi' 'Missouri' 'Montana' 'Nebraska' 'Nevada' 'New Hampshire'
 'New Jersey' 'New Mexico' 'New York' 'North Carolina' 'North Dakota'
 'Ohio' 'Oklahoma' 'Oregon' 'Pennsylvania' 'Rhode Island' 'South Carolina'
 'South Dakota' 'Tennessee' 'Texas' 'Utah' 'Vermont' 'Virginia'
 'Washington' 'West Virginia' 'Wisconsin' 'Wyoming' 'Guam' 'Puerto Rico'
 'Virgin Islands']
--------------------------------------------------
Column Name:  Sex
['Female' 'Male']
--------------------------------------------------
Column Name:  GeneralHealth
['Very good' 'Fair' 'Good' 'Excellent' 'Poor']
--------------------------------------------------
Column Name:  LastCheckupTime
```

- In this code snippet, a loop iterates through the columns of a Data Frame, focusing on those with object data type, typically representing categorical variables. For each column, it prints the column name, followed by the unique values present in that particular column. The 'df.describe(include='object')' part is responsible for selecting columns with object data type, and the loop then extracts and prints the unique values. The '-' characters act as separators between the information for different columns, enhancing the readability of the output. This code is particularly useful for gaining a quick understanding of the distinct values within categorical columns and can aid in identifying patterns or anomalies in the dataset.



```
In [9]: for col in df.describe(include='object').columns:
            print('Column Name: ',col)
            print(df[col].value_counts())
            print('-'*50)
```

```
--------------------------------------------------
Column Name:  Sex
Female    127811
Male      118211
Name: Sex, dtype: int64
--------------------------------------------------
Column Name:  GeneralHealth
Very good    86999
Good         77409
Excellent    41525
Fair         30659
Poor          9430
Name: GeneralHealth, dtype: int64
--------------------------------------------------
Column Name:  LastCheckupTime
Within past year (anytime less than 12 months ago)         198153
Within past 2 years (1 year but less than 2 years ago)      23227
Within past 5 years (2 years but less than 5 years ago)     13744
5 or more years ago                                         10898
Name: LastCheckupTime, dtype: int64
```

- In the provided code snippet, a Python loop iterates through the columns of a Data Frame (presumably named 'df') that contain object-type data. For each such column, it prints the column name and then displays the count of unique values within that column using the `value_counts()` method. This method generates a summary of the occurrences of each unique value in the categorical data. The dashed line ('-' * 50) serves as a visual separator between the results of different columns.
- In essence, this code segment is a practical way to examine the distribution of categorical data in each column of the Data Frame, providing a quick overview of the frequency of unique values and their respective counts. This kind of exploration is valuable for understanding the composition and patterns within categorical variables in a dataset.

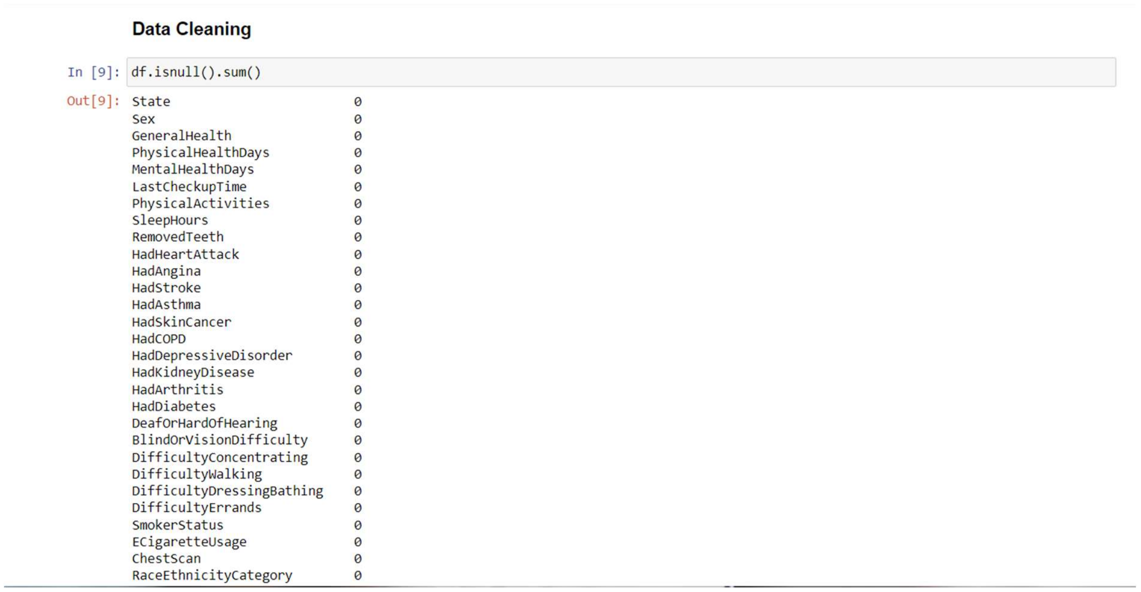
CHAPTER -V

DATA PREPARATION

Data preparation is a crucial step in any project involving data analysis, machine learning, or statistical modeling. It involves cleaning, transforming, and organizing data to make it suitable for analysis. Here are some key steps involved in data preparation for a project:

- **Data Cleaning:** This step involves identifying and rectifying errors or inconsistencies in the dataset, such as missing values, duplicate records, outliers, or incorrect data formats. By cleaning the data, we ensure its accuracy and reliability for further analysis.
- **Data Integration**: Often, data originates from multiple sources, necessitating its integration into a unified dataset. This process involves consolidating datasets from different sources while addressing inconsistencies in naming conventions, data formats, or structures.
- **Data Transformation:** Data transformation entails converting raw data into a more analytically suitable format. This could include scaling numeric data, encoding categorical variables, normalizing data distributions, or generating derived features to facilitate analysis.
- **Feature Selection**: When dealing with datasets containing numerous features, employing feature selection techniques is crucial for identifying the most relevant ones for analysis. This practice aids in reducing dimensionality and enhancing the efficiency and effectiveness of subsequent analysis.
- **Handling Missing Data:** Dealing with missing data is vital as it's a common issue in datasets. Techniques such as imputation (replacing missing values with estimated ones), deletion of missing values, or treating missing values as a distinct category are used to manage missing data appropriately.
- **Data Reduction:** Some datasets may be excessively large or complex for efficient analysis. Data reduction techniques such as aggregation, sampling, or dimensionality reduction can help streamline the dataset's size or complexity while retaining its essential characteristics.
- **Data Formatting:** Ensuring that the data is correctly formatted for analysis is paramount. This includes tasks like standardizing date formats, ensuring consistency in units of measurement, or structuring textual data appropriately for analysis.

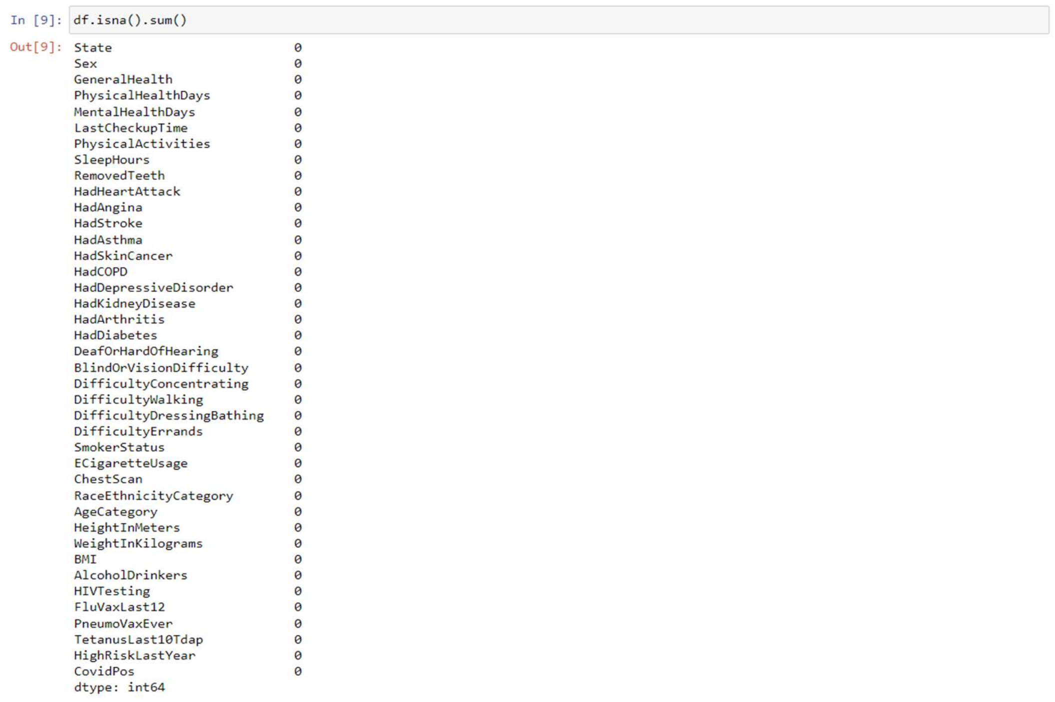
By following these steps diligently, you can ensure that your data is clean, well-structured, and ready for analysis or modeling, thereby improving the quality and reliability of your project results.



Data Cleaning

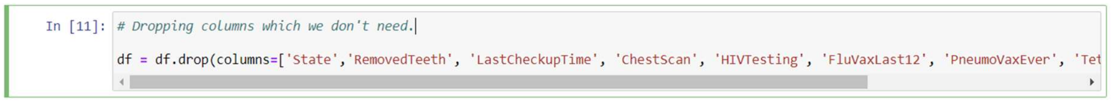
```
In [9]: df.isnull().sum()

Out[9]: State                        0
        Sex                          0
        GeneralHealth                0
        PhysicalHealthDays           0
        MentalHealthDays             0
        LastCheckupTime              0
        PhysicalActivities           0
        SleepHours                   0
        RemovedTeeth                 0
        HadHeartAttack               0
        HadAngina                    0
        HadStroke                    0
        HadAsthma                    0
        HadSkinCancer                0
        HadCOPD                      0
        HadDepressiveDisorder        0
        HadKidneyDisease             0
        HadArthritis                 0
        HadDiabetes                  0
        DeafOrHardOfHearing          0
        BlindOrVisionDifficulty      0
        DifficultyConcentrating      0
        DifficultyWalking            0
        DifficultyDressingBathing    0
        DifficultyErrands            0
        SmokerStatus                 0
        ECigaretteUsage              0
        ChestScan                    0
        RaceEthnicityCategory        0
```

- The command "df.isnull().sum()" has been utilized in the given context to verify the presence of any null values within the dataset.
- According to the analysis, the dataset under consideration exhibits no null values.



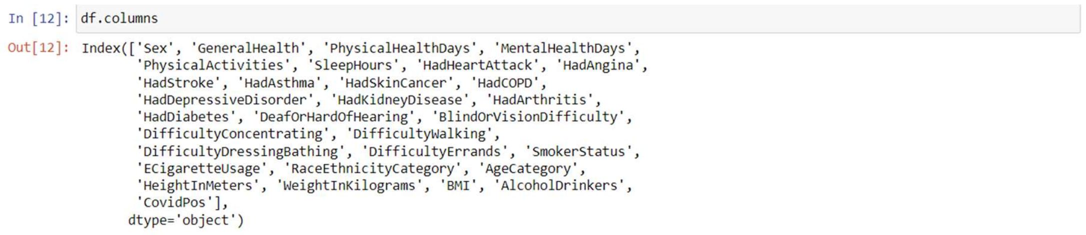
```
In [9]: df.isna().sum()

Out[9]: State                        0
        Sex                          0
        GeneralHealth                0
        PhysicalHealthDays           0
        MentalHealthDays             0
        LastCheckupTime              0
        PhysicalActivities           0
        SleepHours                   0
        RemovedTeeth                 0
        HadHeartAttack               0
        HadAngina                    0
        HadStroke                    0
        HadAsthma                    0
        HadSkinCancer                0
        HadCOPD                      0
        HadDepressiveDisorder        0
        HadKidneyDisease             0
        HadArthritis                 0
        HadDiabetes                  0
        DeafOrHardOfHearing          0
        BlindOrVisionDifficulty      0
        DifficultyConcentrating      0
        DifficultyWalking            0
        DifficultyDressingBathing    0
        DifficultyErrands            0
        SmokerStatus                 0
        ECigaretteUsage              0
        ChestScan                    0
        RaceEthnicityCategory        0
        AgeCategory                  0
        HeightInMeters               0
        WeightInKilograms            0
        BMI                          0
        AlcoholDrinkers              0
        HIVTesting                   0
        FluVaxLast12                 0
        PneumoVaxEver                0
        TetanusLast10Tdap            0
        HighRiskLastYear             0
        CovidPos                     0
        dtype: int64
```

- In the provided context, the command "df.isna().sum()" was utilized to identify any missing values within the dataset.
- Upon examination, it was determined that there are no missing values present in this dataset.



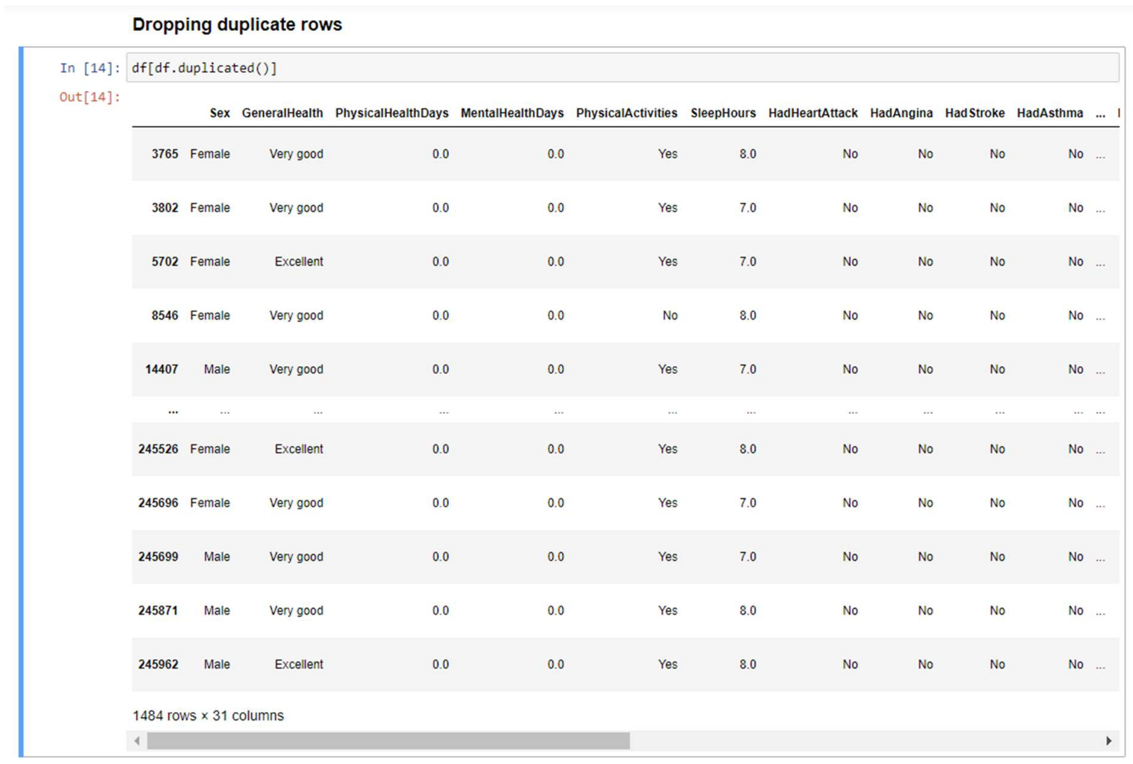
```
In [11]: # Dropping columns which we don't need.

         df = df.drop(columns=['State','RemovedTeeth', 'LastCheckupTime', 'ChestScan', 'HIVTesting', 'FluVaxLast12', 'PneumoVaxEver', 'Tet
```

- In the above picture, that code snippet indicates that columns labeled 'State', 'RemovedTeeth', 'LastCheckupTime', 'ChestScan', 'HIVTesting', 'FluVaxLast12', 'PneumoVaxEver', 'TetanusLast10Tdap', and 'HighRiskLastYear' are being removed from the Data Frame[df]
- These columns are being dropped because they are not relevant to predicting heart disease. Therefore, they are being removed to streamline the dataset for further analysis focused specifically on heart disease prediction.



```
In [12]: df.columns

Out[12]: Index(['Sex', 'GeneralHealth', 'PhysicalHealthDays', 'MentalHealthDays',
                'PhysicalActivities', 'SleepHours', 'HadHeartAttack', 'HadAngina',
                'HadStroke', 'HadAsthma', 'HadSkinCancer', 'HadCOPD',
                'HadDepressiveDisorder', 'HadKidneyDisease', 'HadArthritis',
                'HadDiabetes', 'DeafOrHardOfHearing', 'BlindOrVisionDifficulty',
                'DifficultyConcentrating', 'DifficultyWalking',
                'DifficultyDressingBathing', 'DifficultyErrands', 'SmokerStatus',
                'ECigaretteUsage', 'RaceEthnicityCategory', 'AgeCategory',
                'HeightInMeters', 'WeightInKilograms', 'BMI', 'AlcoholDrinkers',
                'CovidPos'],
               dtype='object')
```

- Now, after executing the removal operation, the specified columns have been successfully dropped from the DataFrame, resulting in a more refined dataset ready for further analysis.



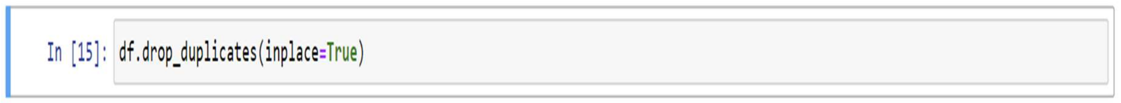
Dropping duplicate rows

```
In [14]: df[df.duplicated()]
Out[14]:
```

	Sex	GeneralHealth	PhysicalHealthDays	MentalHealthDays	PhysicalActivities	SleepHours	HadHeartAttack	HadAngina	HadStroke	HadAsthma	...
3765	Female	Very good	0.0	0.0	Yes	8.0	No	No	No	No	...
3802	Female	Very good	0.0	0.0	Yes	7.0	No	No	No	No	...
5702	Female	Excellent	0.0	0.0	Yes	7.0	No	No	No	No	...
8546	Female	Very good	0.0	0.0	No	8.0	No	No	No	No	...
14407	Male	Very good	0.0	0.0	Yes	7.0	No	No	No	No	...
...	...	...	...	...	...	...	...	...	...	...	...
245526	Female	Excellent	0.0	0.0	Yes	8.0	No	No	No	No	...
245696	Female	Very good	0.0	0.0	Yes	7.0	No	No	No	No	...
245699	Male	Very good	0.0	0.0	Yes	7.0	No	No	No	No	...
245871	Male	Very good	0.0	0.0	Yes	8.0	No	No	No	No	...
245962	Male	Excellent	0.0	0.0	Yes	8.0	No	No	No	No	...

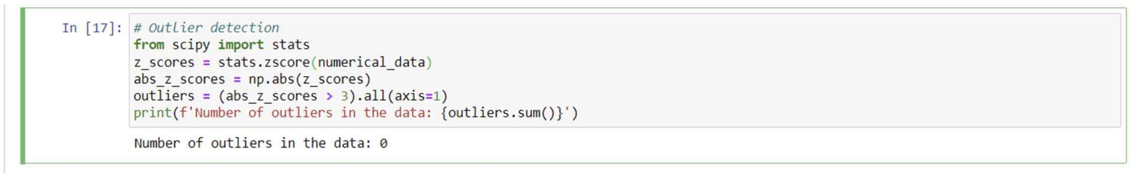
1484 rows × 31 columns

- The code "df[df.duplicated()]" is used to identify duplicate rows within the DataFrame 'df'. This command returns a DataFrame containing only the rows that are duplicates of other rows in the original DataFrame based on all columns.
- Now there are 1484 duplicate rows in the data and we are going to drop these duplicate data for EDA
- This operation is helpful for detecting and handling duplicate data entries, which can skew analysis results or lead to inaccuracies in modeling.



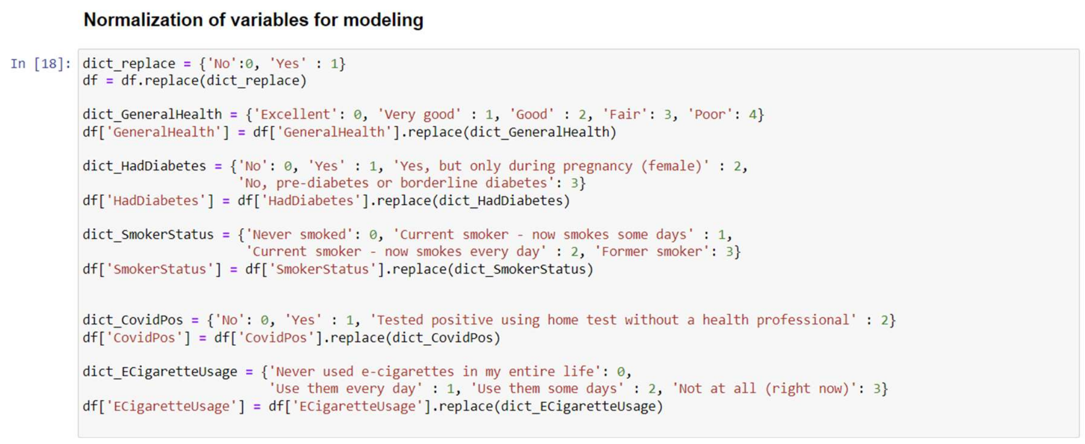
```
In [15]: df.drop_duplicates(inplace=True)
```

- Executing this command has effectively removed the duplicate data from the dataset.



```
In [17]: # Outlier detection
         from scipy import stats
         z_scores = stats.zscore(numerical_data)
         abs_z_scores = np.abs(z_scores)
         outliers = (abs_z_scores > 3).all(axis=1)
         print(f'Number of outliers in the data: {outliers.sum()}')

         Number of outliers in the data: 0
```

- In the preceding analysis, it indicates that no outliers were detected in the dataset.



Normalization of variables for modeling

```
In [18]: dict_replace = {'No':0, 'Yes' : 1}
         df = df.replace(dict_replace)

         dict_GeneralHealth = {'Excellent': 0, 'Very good' : 1, 'Good' : 2, 'Fair': 3, 'Poor': 4}
         df['GeneralHealth'] = df['GeneralHealth'].replace(dict_GeneralHealth)

         dict_HadDiabetes = {'No': 0, 'Yes' : 1, 'Yes, but only during pregnancy (female)' : 2,
                             'No, pre-diabetes or borderline diabetes': 3}
         df['HadDiabetes'] = df['HadDiabetes'].replace(dict_HadDiabetes)

         dict_SmokerStatus = {'Never smoked': 0, 'Current smoker - now smokes some days' : 1,
                              'Current smoker - now smokes every day' : 2, 'Former smoker': 3}
         df['SmokerStatus'] = df['SmokerStatus'].replace(dict_SmokerStatus)


         dict_CovidPos = {'No': 0, 'Yes' : 1, 'Tested positive using home test without a health professional' : 2}
         df['CovidPos'] = df['CovidPos'].replace(dict_CovidPos)

         dict_ECigaretteUsage = {'Never used e-cigarettes in my entire life': 0,
                                 'Use them every day' : 1, 'Use them some days' : 2, 'Not at all (right now)': 3}
         df['ECigaretteUsage'] = df['ECigaretteUsage'].replace(dict_ECigaretteUsage)
```

The provided code snippet involves replacing categorical values with numerical representations in specific columns of the DataFrame 'df'. Here's a breakdown of the transformations:

1. Replacing Binary Categories:

- The dictionary `dict_replace` is used to map 'No' to 0 and 'Yes' to 1 across the entire DataFrame.

2. Replacing Ordinal Categories:

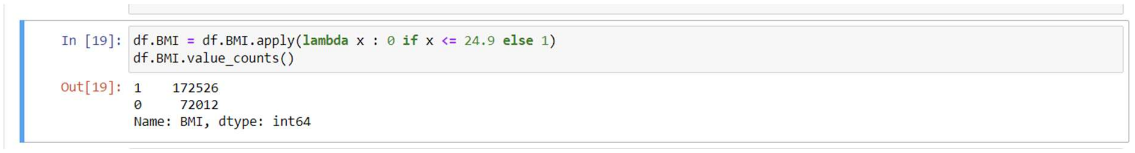
- For the 'GeneralHealth' column, categories such as 'Excellent', 'Very good', 'Good', 'Fair', and 'Poor' are mapped to numerical values ranging from 0 to 4 according to their ordinality.
- Similarly, the 'HadDiabetes' column is mapped using `dict_HadDiabetes`, where categories such as 'No', 'Yes', 'Yes, but only during pregnancy (female)', and 'No, pre-diabetes or borderline diabetes' are replaced with numerical values.
- 'SmokerStatus' column categories like 'Never smoked', 'Current smoker - now smokes some days', 'Current smoker - now smokes every day', and 'Former smoker' are replaced using `dict_SmokerStatus`.

3. Replacing Multi-Category Responses:

- For 'CovidPos', 'No', 'Yes', and 'Tested positive using home test without a health professional' are replaced with numerical values using `dict_CovidPos`.

- 'ECigaretteUsage' column categories such as 'Never used e-cigarettes in my entire life', 'Use them every day', 'Use them some days', and 'Not at all (right now)' are replaced accordingly.

These transformations are useful for converting categorical data into a numerical format, which is often required for machine learning algorithms or statistical analysis. It ensures that the categorical variables are appropriately represented in numerical form for modeling or analysis purposes.



```
In [19]: df.BMI = df.BMI.apply(lambda x : 0 if x <= 24.9 else 1)
         df.BMI.value_counts()

Out[19]: 1    172526
         0     72012
         Name: BMI, dtype: int64
```

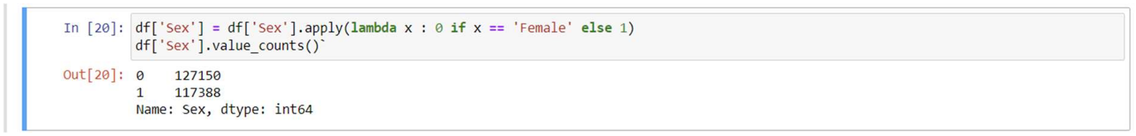
The provided code snippet modifies the 'BMI' column in the Data Frame 'df' by applying a lambda function. Here's the meaning of the code in two points:

- **BMI Reclassification:**

The lambda function applied to the 'BMI' column checks each value. If the value is less than or equal to 24.9 (indicating a BMI below or within the normal range), it assigns a value of 0. Otherwise, it assigns a value of 1. This essentially categorizes individuals into two groups based on their BMI: 0 for those with a BMI of 24.9 or lower (normal or underweight) and 1 for those with a BMI above 24.9 (potentially overweight or obese).

- **Value Counts:**

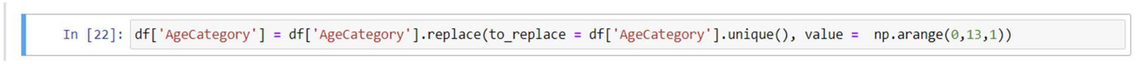
After the transformation, the value_counts() function is used to count the occurrences of each unique value in the 'BMI' column. This provides insight into the distribution of individuals categorized as 0 or 1 based on the BMI threshold of 24.9. The resulting counts reveal the number of individuals falling into each BMI category after the reclassification.



```
In [20]: df['Sex'] = df['Sex'].apply(lambda x : 0 if x == 'Female' else 1)
         df['Sex'].value_counts()`

Out[20]: 0    127150
         1    117388
         Name: Sex, dtype: int64
```

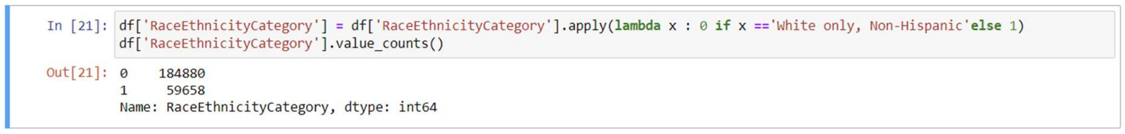
- The code snippet df['Sex'] = df['Sex'].apply(lambda x : 0 if x == 'Female' else 1) is converting the 'Sex' column in the DataFrame 'df' to numerical values, where 'Female' is mapped to 0 and any other value (assuming 'Male') is mapped to 1. This transformation effectively encodes gender as a binary variable, with 0 representing 'Female' and 1 representing 'Male'.

- The subsequent code df['Sex'].value_counts() is used to count the occurrences of each unique value in the 'Sex' column after the transformation. It provides insights into the distribution of genders within the dataset.
- In summary, after applying the lambda function to encode gender into binary values, the value counts operation is used to understand the distribution of genders within the dataset, revealing the number of females (0) and males (1) present in the 'Sex' column.



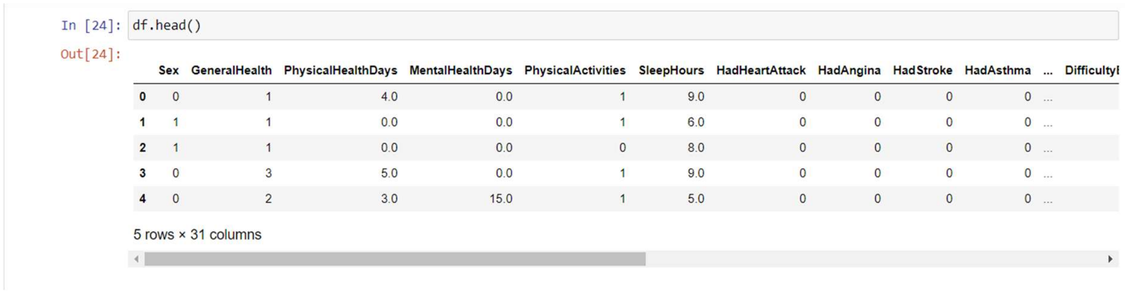
```
In [22]: df['AgeCategory'] = df['AgeCategory'].replace(to_replace = df['AgeCategory'].unique(), value =  np.arange(0,13,1))
```

- The code snippet assigns numerical values to unique categories in the 'AgeCategory' column, starting from 0 up to 12. This transformation effectively converts categorical age groups into numerical representations, allowing for easier analysis and modeling where numerical values are required.



```
In [21]: df['RaceEthnicityCategory'] = df['RaceEthnicityCategory'].apply(lambda x : 0 if x =='White only, Non-Hispanic'else 1)
         df['RaceEthnicityCategory'].value_counts()

Out[21]: 0    184880
         1     59658
         Name: RaceEthnicityCategory, dtype: int64
```

- This code applies a lambda function to the 'RaceEthnicityCategory' column, assigning a value of 0 if the entry is 'White only, Non-Hispanic', otherwise assigning a value of 1, and then counts the occurrences of each value.



```
In [24]: df.head()

Out[24]:
```

	Sex	GeneralHealth	PhysicalHealthDays	MentalHealthDays	PhysicalActivities	SleepHours	HadHeartAttack	HadAngina	HadStroke	HadAsthma	...	DifficultyI
0	0	1	4.0	0.0	1	9.0	0	0	0	0	...	
1	1	1	0.0	0.0	1	6.0	0	0	0	0	...	
2	1	1	0.0	0.0	0	8.0	0	0	0	0	...	
3	0	3	5.0	0.0	1	9.0	0	0	0	0	...	
4	0	2	3.0	15.0	1	5.0	0	0	0	0	...	

5 rows × 31 columns

- Now applying head function, we can see all the categorical variables are change to numeric variables for further data modelling process.

Now Data preparation is over for this data set and we are proceeding for next step.

CHAPTER-VI

EXPLORATORY DATA ANALYSIS

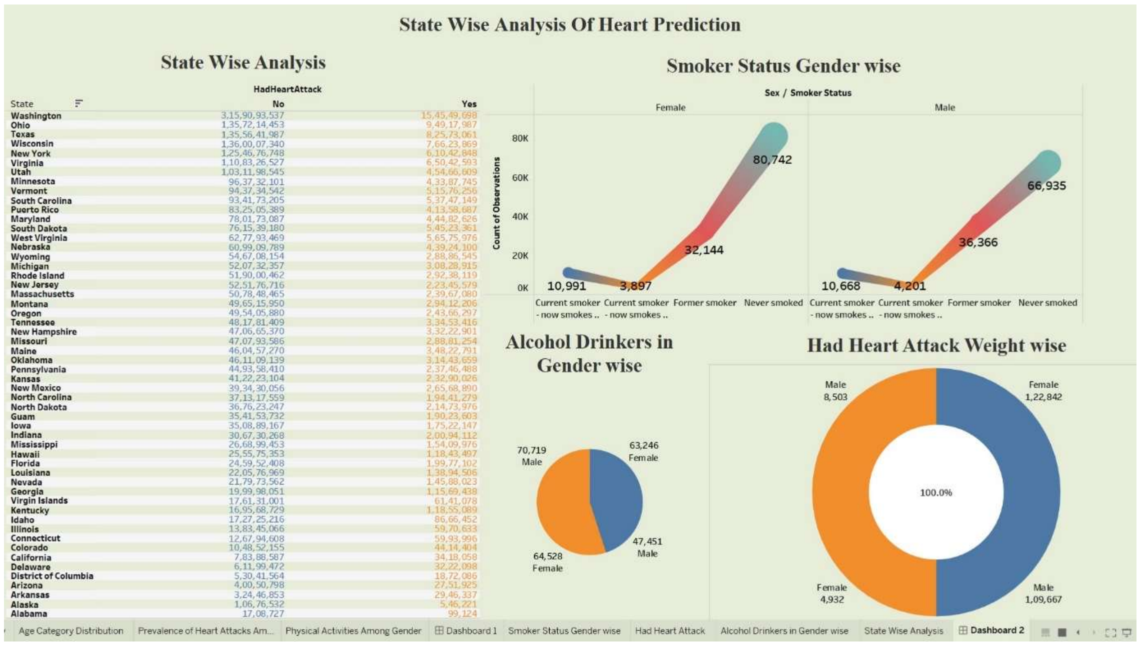
Exploratory Data Analysis (EDA) is a crucial initial step in any data science project. It involves examining and understanding the data to extract meaningful insights, detect patterns, and identify potential issues or anomalies. Here's a structured approach to conducting EDA

- **Data Collection:**
    i. Begin by gathering all relevant data sources for your project. This might involve querying databases, scraping websites, or accessing APIs.
- **Data Cleaning:**
    i. Handle missing values: Identify missing data points and decide how to deal with them (e.g., imputation, deletion).
    ii. Check for duplicates: Remove any duplicate records if present.
    iii. Data formatting: Standardize data formats (e.g., date formats, text encoding).
    iv. Handle outliers: Detect and potentially remove or adjust outliers that could skew the analysis.
- **Statistical Summaries:**
    i. Descriptive statistics: Compute summary statistics such as mean, median, mode, standard deviation, range, etc., for numerical variables.
    ii. Frequency distributions: Examine the distribution of categorical variables.
    iii. Correlation analysis: Compute correlations between numerical variables to identify relationships.
- **Data Visualization:**
    i. Univariate analysis: Explore individual variables using histograms, box plots, bar charts, etc., to understand their distributions and identify any outliers.
    ii. Bivariate analysis: Analyze relationships between pairs of variables using scatter plots, heatmaps, etc.
    iii. Multivariate analysis: Explore interactions between multiple variables using techniques like pair plots, stacked bar charts, etc.
    iv. Time series analysis: If applicable, visualize temporal patterns using line plots, seasonal decomposition, autocorrelation plots, etc.

- **Feature Engineering:**
    1. Create new features: Derive additional features from existing ones that may enhance predictive power.
    2. Transform variables: Apply transformations (e.g., logarithmic, polynomial) to improve linearity or normalize distributions.
    3. Encode categorical variables: Convert categorical variables into numerical representations using techniques like onehot encoding or label encoding.
- **Hypothesis Testing (if relevant):**
    1. Formulate hypotheses based on your understanding of the data.
    2. Conduct statistical tests (e.g., ttests, ANOVA) to assess the significance of relationships or differences observed in the data.
- **Documentation and Reporting:**
    1. Document your findings, insights, and any decisions made during the EDA process.
    2. Prepare visualizations and summaries for presentation to stakeholders or inclusion in reports.

Remember, EDA is an iterative process, and you may need to revisit earlier steps as you gain more insights or encounter new challenges in your analysis.

STATE WISE ANALYSIS OF HEART DISEASE PREDICTION


State Wise Analysis Of Heart Prediction
Smoker Status Gender wise
Sex / Smoker Status
Female
Male
Count of Observations
0K
20K
40K
60K
80K
10,991
3,897
32,144
80,742
10,668
4,201
36,366
66,935
Current smoker - now smokes ..
Current smoker - now smokes ..
Former smoker
Never smoked
Alcohol Drinkers in Gender wise
70,719 Male
63,246 Female
47,451 Male
64,528 Female
Had Heart Attack Weight wise
Male 8,503
Female 1,22,842
100.0%
Female 4,932
Male 1,09,667
Age Category Distribution
Prevalence of Heart Attacks Am...
Physical Activities Among Gender
Dashboard 1
Smoker Status Gender wise
Had Heart Attack
Alcohol Drinkers in Gender wise
State Wise Analysis
Dashboard 2

State Wise Analysis

State	HadHeartAttack No	Yes
Washington	3,15,90,93,537	15,45,49,698
Ohio	1,35,72,14,453	9,49,17,987
Texas	1,35,56,41,987	8,25,73,061
Wisconsin	1,36,00,07,340	7,66,23,869
New York	1,25,46,76,748	6,10,42,848
Virginia	1,10,83,26,527	6,50,42,593
Utah	1,03,11,98,545	4,54,66,609
Minnesota	96,37,32,101	4,33,87,745
Vermont	94,37,34,542	5,15,76,256
South Carolina	93,41,73,205	5,37,47,149
Puerto Rico	83,25,05,389	4,13,58,687
Maryland	78,01,73,087	4,44,82,626
South Dakota	76,15,39,180	5,45,23,361
West Virginia	62,77,93,469	5,65,75,976
Nebraska	60,99,09,789	4,39,24,100
Wyoming	54,67,08,154	2,88,86,545
Michigan	52,07,32,357	3,08,28,915
Rhode Island	51,90,00,462	2,92,38,119
New Jersey	52,51,76,716	2,23,45,579
Massachusetts	50,78,48,465	2,39,67,080
Montana	49,65,15,950	2,94,12,206
Oregon	49,54,05,880	2,43,66,297
Tennessee	48,17,81,409	3,34,53,416
New Hampshire	47,06,65,370	3,32,22,901
Missouri	47,07,93,586	2,88,81,254
Maine	46,04,57,270	3,48,22,791
Oklahoma	46,11,09,139	3,14,43,659
Pennsylvania	44,93,58,410	2,37,46,488
Kansas	41,22,23,104	2,32,90,026
New Mexico	39,34,30,056	2,65,68,890
North Carolina	37,13,17,559	1,94,41,279
North Dakota	36,76,23,247	2,14,73,976
Guam	35,41,53,732	1,90,23,603
Iowa	35,08,89,167	1,75,22,147
Indiana	30,67,30,268	2,00,94,112
Mississippi	26,68,99,453	1,54,09,976
Hawaii	25,55,75,353	1,18,43,497
Florida	24,59,52,408	1,99,77,102
Louisiana	22,05,76,969	1,38,94,506
Nevada	21,79,73,562	1,45,88,023
Georgia	19,99,98,051	1,15,69,438
Virgin Islands	17,61,31,001	61,41,078
Kentucky	16,95,68,729	1,18,55,089
Idaho	17,27,25,216	86,66,452
Illinois	13,83,45,066	59,70,633
Connecticut	12,67,94,608	59,93,996
Colorado	10,48,52,155	44,14,404
California	7,83,88,587	34,18,058
Delaware	6,11,99,472	32,22,098
District of Columbia	5,30,41,564	18,72,086
Arizona	4,00,50,798	27,51,925
Arkansas	3,24,46,853	29,46,337
Alaska	1,06,76,532	5,46,221
Alabama	17,08,727	99,124

The dashboard displays heart disease prediction based on state, smoker status, alcohol and weight. Exploratory Data Analysis (EDA) reveals insights into the relationship between state, smoker status, alcohol and weight. Insights from the dashboard can aid in understanding risk factors and inform preventive measures.

State Wise Analysis:

- Among the 54 states in the US, the top five with the highest incidence of heart disease are Washington (701 cases), Ohio (589 cases), Florida (557 cases), Maryland (496 cases), and Texas (426 cases).
- These states exhibit the highest numbers of individuals affected by heart disease, according to the data provided.
- The Virgin Islands have a lower prevalence of heart disease, with only 25 reported cases.
- The top five states in the US with the highest incidence of heart disease are Washington (701 cases), Ohio (589 cases), Florida (557 cases), Maryland (496 cases), and Texas (426 cases).
- Conversely, the top five states with the lowest incidence of heart disease among the 54 states are Washington, Wisconsin, Ohio, Maryland, and Texas.

Smoker Status:

This shows the flow of counts from two categories of current smokers (those who smoke every day and those who smoke some days), former smokers, and those who have never smoked, for both females and males.

Here's an interpretation of the diagram:

Female Smokers:

- 10,991 females are current smokers who smoke every day.
- 3,897 females are current smokers who smoke some days.
- 32,144 females are former smokers.
- 80,742 females have never smoked.

Male Smokers:

- 10,668 males are current smokers who smoke every day.

- 4,201 males are current smokers who smoke some days.
- 36,366 males are former smokers.
- 66,935 males have never smoked.

The width of the bands is proportional to the count of observations in each category. The largest band for both genders is for those who have never smoked, indicating that this is the most common status among the individuals represented in the data. For females, the second largest group is former smokers, while for males, the second largest group is also former smokers, but the number is slightly higher than for females.

The data suggests that there are more females who have never smoked compared to males, and the number of current smokers (both daily and some days) is relatively similar between genders. The number of former smokers is higher among males than females.

Had Heart Attack Weight wise:

It represents data on the number of males and females who have had a heart attack, categorized by weight.

The chart is divided into four segments with the following data:

1. Male: 8,503 (Yes)
2. Female: 1,22,842 (No)
3. Female: 4,932 (Yes)
4. Male: 1,09,667 (No)

The numbers suggest that there are two separate categories for both males and females, possibly representing different weight groups. For instance, the first male category has 8,503 cases, and the second male category has 1,09,667 cases. Similarly, the first female category has 1,22,842 cases, and the second has 4,932 cases.

The centre of the donut chart shows "100.0%", indicating that the chart represents 100% of the data set for this particular distribution of heart attack cases by weight and gender.

Alcohol Drinkers in Gender wise:

This pie chart titled as "Alcohol Drinkers in Gender wise." It represents the distribution of alcohol drinkers by gender, with numerical data provided for each segment.

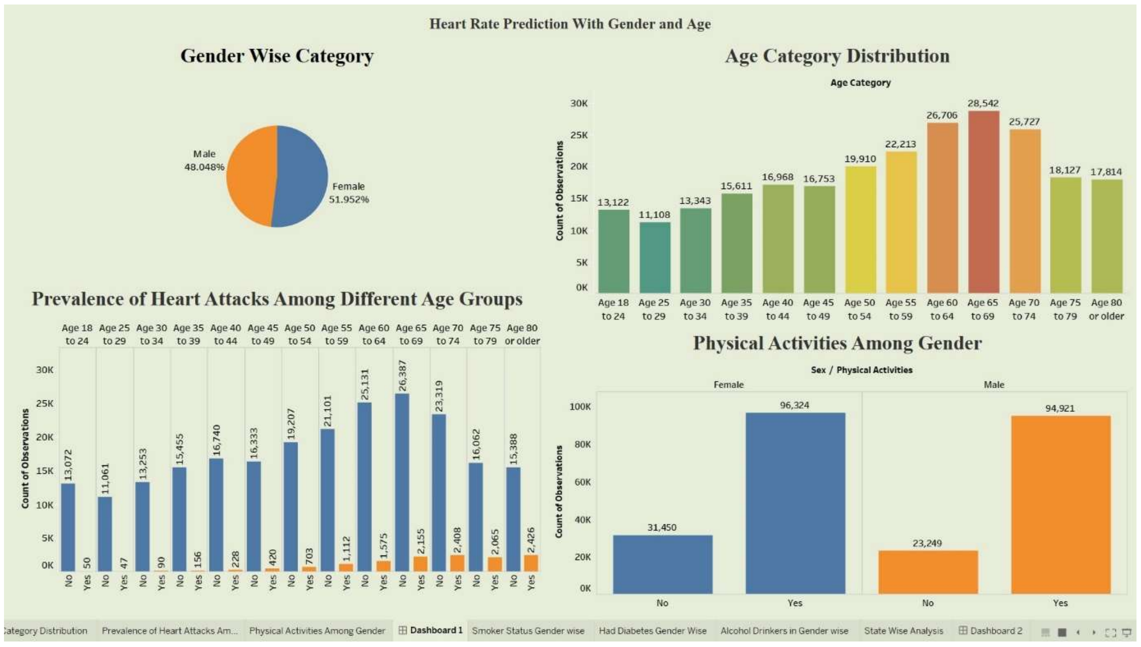
The chart is divided into four segments, with two segments for each gender:

- The chart shows that there are two separate categories for both males and females, which might represent different age groups, consumption levels, or other subcategories within the gender classification.
- The larger segments for both genders (28.74% for males and 26.23 for females) suggest that these groups have a higher number of alcohol drinkers. Remaining groups doesn't consume alcohol.

Results and Insights:

- States like Washington, Ohio, Florida, Maryland, and Texas exhibit higher incidences of heart disease, while the Virgin Islands have notably lower prevalence.
- Female populations tend to have a higher proportion of never smokers compared to males, while former smoker rates are slightly higher among males.
- The distribution of heart attack cases by weight and gender suggests potential correlations, highlighting the need for weight management interventions male has (8503) cases and female has (4932) case compared to male which is high.
- Both genders show significant numbers of alcohol drinkers, indicating a potential risk factor for heart disease and the importance of targeted interventions for alcohol consumption.

HEART DISEASE PREDICTION WITH GENDER AND AGE



The image depicts a dashboard focusing on heart disease prediction utilizing gender and age variables, prompting an exploratory data analysis (EDA).

1. Gender Wise Category (Pie Chart):

- The dataset includes a slightly higher proportion of females (51.952%) compared to males (48.048%).
- This distribution should be considered when analyzing other variables to ensure that gender related insights are not skewed by this imbalance.

2. Age Category Distribution (Bar Chart):

- The dataset spans a wide range of age groups, with the majority of observations in the older age groups, particularly those aged 65 to 69 years.
- The least represented age group is 18 to 24 years.

3. Physical Activities Among Gender (Bar Chart):

- A large majority of both females and males are recorded as engaging in physical activities.
- Females have a higher count of physical activity engagement (96,324) compared to males (94,921).
- A smaller, yet significant, number of individuals do not engage in physical activities, with more females (31,450) than males (23,249) being inactive.

4. Prevalence of Heart Attacks Among Different Age Groups (Bar Chart):

- The prevalence of heart attacks increases with age, peaking in the 70 to 74 age group.
- Individuals aged 80 and above experience the highest incidence of heart attacks compared to other age groups.
- There is a slight decrease in heart attack prevalence in the age groups 55 to 64 and older,
- This trend underscores the importance of age as a risk factor for heart attacks.

Results and Insights:

- Gender Analysis: While the gender distribution is relatively balanced.
- Age Analysis: The data shows that older age groups (50-80+) have a higher prevalence of heart attacks, which aligns with general medical understanding that heart attack risk increases with age.

- Physical Activity Analysis: The data suggests that physical activity is a common behaviour among the population studied, with more females reported as being physically active than males. However, a significant portion of the population remains inactive, which could be a target for public health interventions.
- Heart Attack Prevalence: The increase in heart attack prevalence with age highlights the need for age specific health interventions and preventive measures.
- Out of the total population, only 5.46% have been diagnosed with heart disease, while the remaining 94.54% do not exhibit this condition. Specifically, females show a lower prevalence of heart disease at 2.01%, compared to males at 3.46%.

CHAPTER-VII

MODEL BUILDING

In today's data-driven landscape, organizations across various sectors are harnessing data analysis to extract valuable insights, make well-informed decisions, and gain a competitive advantage. Central to data analysis is the process of model building, wherein mathematical representations of real-world phenomena are constructed to predict outcomes or unveil underlying patterns. Successful model building necessitates a blend of domain expertise, statistical acumen, and proficiency in data manipulation and programming. This comprehensive guide aims to furnish a thorough grasp of model building in data analysis, encompassing fundamental concepts, methodologies, and best practices.

Understanding Model Building:

Model building entails crafting mathematical or computational models of real-world systems, processes, or phenomena. These models aim to encapsulate the relationships between variables and facilitate predictions or decision-making. In the realm of data analysis, models are constructed using historical data to discern patterns, forecast future outcomes, or glean insights into intricate systems.

Types of Models:

Data analysis employs various types of models, each tailored to distinct data types and analytical goals:

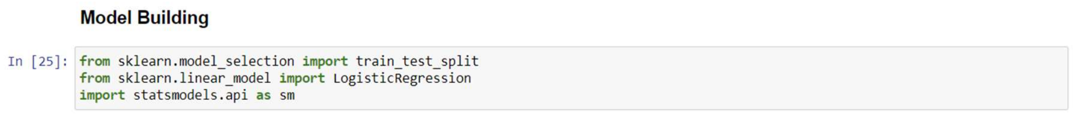
- Statistical Models: Statistical models employ mathematical equations to elucidate relationships between variables. Examples include linear regression, logistic regression, time series analysis, and multivariate analysis.
- Machine Learning Models: Machine learning models leverage algorithms to discern patterns from data and make predictions or decisions autonomously. Common algorithms encompass decision trees, random forests, support vector machines, neural networks, and clustering algorithms.
- Predictive Models: Predictive models prognosticate future outcomes based on historical data, crucial for tasks such as sales forecasting, demand prediction, risk assessment, and fraud detection.

- Descriptive Models: Descriptive models summarize and interpret data to unveil underlying patterns or relationships. These encompass clustering algorithms, principal component analysis (PCA), and factor analysis.

Steps in Model Building:

The model building process typically encompasses several key steps:

- Problem Definition: Clearly delineate the problem and analytical objectives, understanding the context and specific requirements.
- Data Collection and Preprocessing: Gather relevant data from diverse sources and preprocess it to ensure quality and consistency, involving tasks like cleaning, handling missing values, and normalization.
- Exploratory Data Analysis (EDA): Conduct EDA to grasp data characteristics, identify patterns, and explore relationships using visualization techniques.
- Feature Selection and Engineering: Choose or engineer relevant features while discarding irrelevant ones.
- Model Selection: Pick an appropriate modeling technique based on data nature and objectives, considering factors like interpretability and performance.
- Model Training: Split data, train the model on the training set, and optimize performance through techniques like cross-validation.
- Model Evaluation: Assess model performance using appropriate metrics like accuracy, precision, and F1-score.
- Model Deployment and Monitoring: Deploy the model into production, continuously monitoring its performance and updating it as needed.

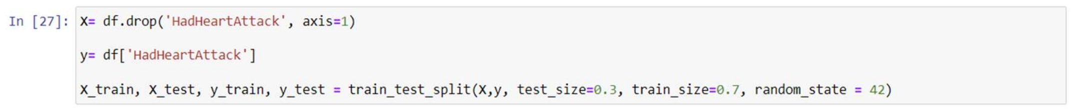

Model Building

```
In [25]: from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LogisticRegression
         import statsmodels.api as sm
```

The code snippet imports necessary modules for building and evaluating a logistic regression model using scikit-learn and stats models libraries.

- `train_test_split` from `sklearn.model_selection` is used to split the dataset into training and testing sets, facilitating model evaluation.
- `LogisticRegression` from `sklearn.linear_model` implements logistic regression, a commonly used algorithm for binary classification tasks.

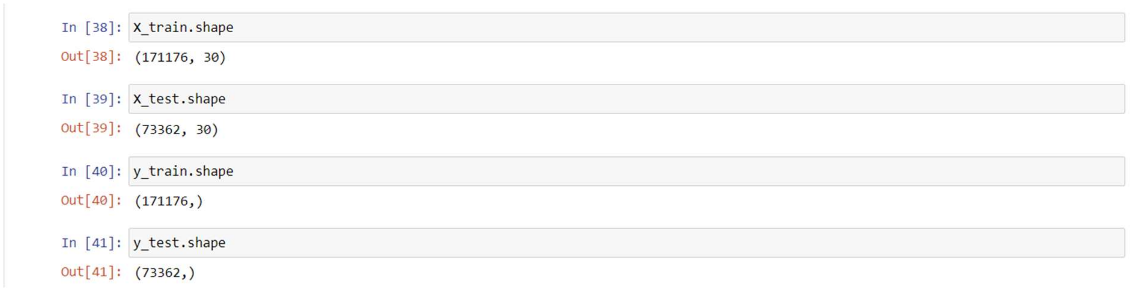
- `sm` from `statsmodels.api` provides tools for statistical modeling, including regression analysis and hypothesis testing.



```
In [27]: X= df.drop('HadHeartAttack', axis=1)
         y= df['HadHeartAttack']
         X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, train_size=0.7, random_state = 42)
```

The provided code performs the following tasks:

- X = df.drop('HadHeartAttack', axis=1): This line extracts features from the DataFrame df by dropping the column named 'HadHeartAttack'. This is typically done when preparing data for modeling, separating predictors from the target variable.
- Y = df['HadHeartAttack']: This line selects the target variable 'HadHeartAttack' from the DataFrame df. This variable represents the outcome or the label that the model aims to predict.
- X_train, X_test, y_train, y_test = train_test_split(feats, target, test_size=0.3, train_size=0.7, random_state=42): This line splits the dataset into training and testing sets for both features and target variables. Here:
- X is the set of features.
- Y is the target variable.
- Test_size=0.3 specifies that 30% of the data will be allocated for testing, while the remaining 70% will be used for training.
- Train_size=0.7 is redundant since it's the default value when test_size is specified.
- Random_state=42 sets the random seed to 42, ensuring reproducibility of the data splitting process.

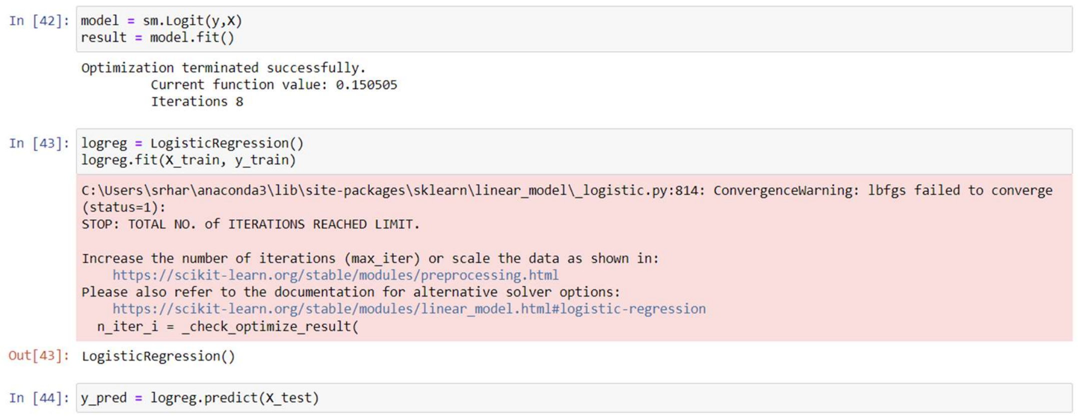


```
In [38]: X_train.shape
Out[38]: (171176, 30)

In [39]: X_test.shape
Out[39]: (73362, 30)

In [40]: y_train.shape
Out[40]: (171176,)

In [41]: y_test.shape
Out[41]: (73362,)
```

After splitting the X train, test and y train, test then performing shape command to display the size of the rows. Where X train and y train size are (171176 rows and 30 columns) and X test and y test size are (73362 rows).

Logistic regression:

Logistic regression is a statistical model that estimates the probability of an event occurring based on a given dataset of independent variables.



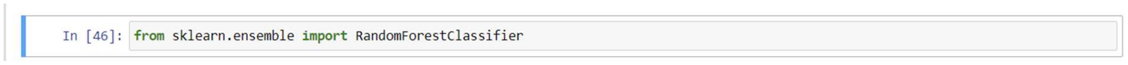
```
In [42]: model = sm.Logit(y,X)
         result = model.fit()

         Optimization terminated successfully.
                  Current function value: 0.150505
                  Iterations 8

In [43]: logreg = LogisticRegression()
         logreg.fit(X_train, y_train)

         C:\Users\srhar\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge
         (status=1):
         STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

         Increase the number of iterations (max_iter) or scale the data as shown in:
             https://scikit-learn.org/stable/modules/preprocessing.html
         Please also refer to the documentation for alternative solver options:
             https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
           n_iter_i = _check_optimize_result(

Out[43]: LogisticRegression()

In [44]: y_pred = logreg.predict(X_test)
```

The logistic regression model was successfully executed using the features (X) and the target variable (Y). The optimization process displayed a current function value of 0.150505 out of 1.00, indicating progress towards convergence. During the iteration, the model included 8 categorical values.

Subsequently, after training the model, test data was utilized for prediction. Upon applying the prediction code, the model successfully executed and generated predictions based on the input test data.

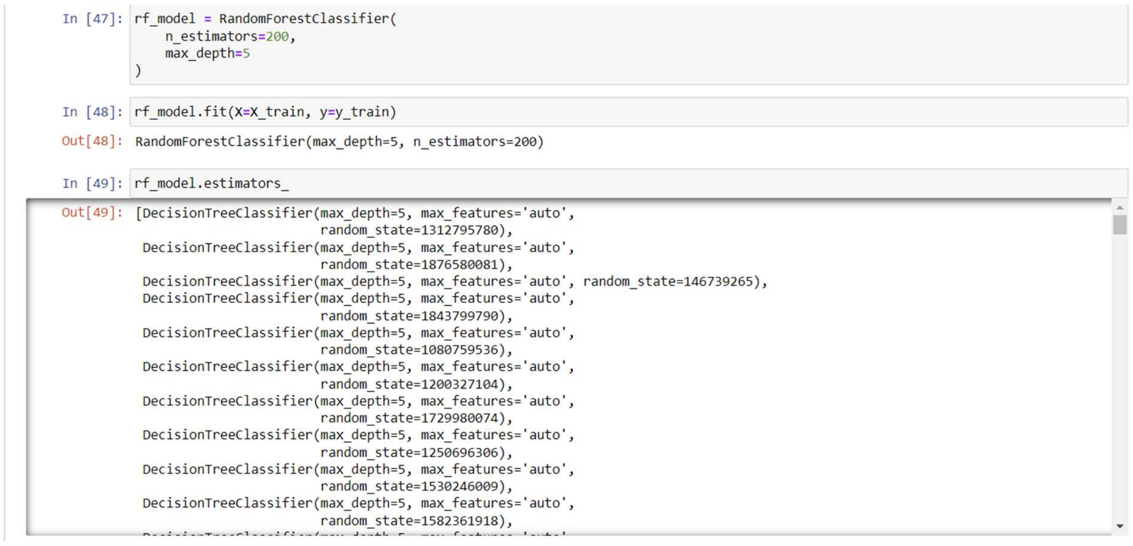
Random Forest:

Random Forest is a widely-used machine learning algorithm developed by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.



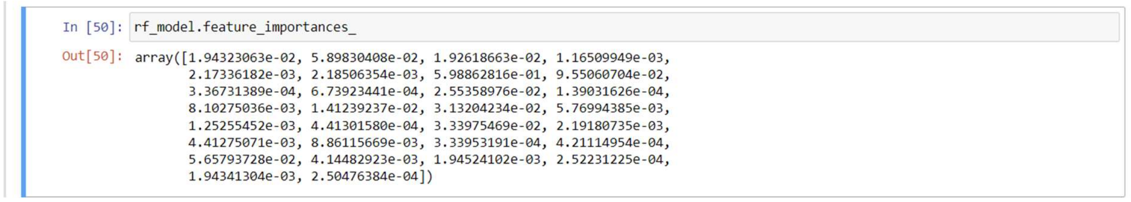
```
In [46]: from sklearn.ensemble import RandomForestClassifier
```

The line of code `from sklearn.ensemble import RandomForestClassifier` imports the `RandomForestClassifier` class from the `sklearn.ensemble` module. This class is used for building a Random Forest classifier, a machine learning model that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.



```
In [47]: rf_model = RandomForestClassifier(
             n_estimators=200,
             max_depth=5
         )

In [48]: rf_model.fit(X=X_train, y=y_train)

Out[48]: RandomForestClassifier(max_depth=5, n_estimators=200)

In [49]: rf_model.estimators_

Out[49]: [DecisionTreeClassifier(max_depth=5, max_features='auto',
                                 random_state=1312795780),
          DecisionTreeClassifier(max_depth=5, max_features='auto',
                                 random_state=1876580081),
          DecisionTreeClassifier(max_depth=5, max_features='auto', random_state=146739265),
          DecisionTreeClassifier(max_depth=5, max_features='auto',
                                 random_state=1843799790),
          DecisionTreeClassifier(max_depth=5, max_features='auto',
                                 random_state=1080759536),
          DecisionTreeClassifier(max_depth=5, max_features='auto',
                                 random_state=1200327104),
          DecisionTreeClassifier(max_depth=5, max_features='auto',
                                 random_state=1729980074),
          DecisionTreeClassifier(max_depth=5, max_features='auto',
                                 random_state=1250696306),
          DecisionTreeClassifier(max_depth=5, max_features='auto',
                                 random_state=1530246009),
          DecisionTreeClassifier(max_depth=5, max_features='auto',
                                 random_state=1582361918),
```

In the above picture, the random forest model command has been executed and also performed successfully and then to display the estimators, “rf_model.estimators_” command has been executed.



```
In [50]: rf_model.feature_importances_

Out[50]: array([1.94323063e-02, 5.89830408e-02, 1.92618663e-02, 1.16509949e-03,
                2.17336182e-03, 2.18506354e-03, 5.98862816e-01, 9.55060704e-02,
                3.36731389e-04, 6.73923441e-04, 2.55358976e-02, 1.39031626e-04,
                8.10275036e-03, 1.41239237e-02, 3.13204234e-02, 5.76994385e-03,
                1.25255452e-03, 4.41301580e-04, 3.33975469e-02, 2.19180735e-03,
                4.41275071e-03, 8.86115669e-03, 3.33953191e-04, 4.21114954e-04,
                5.65793728e-02, 4.14482923e-03, 1.94524102e-03, 2.52231225e-04,
                1.94341304e-03, 2.50476384e-04])
```

The code rf_model.feature_importances_ returns an array containing the feature importances calculated by the trained Random Forest model (rf_model). Each value in the array corresponds to the importance of a specific feature in the model's prediction process. These importances are typically normalized to sum up to 1.

Both the logistic regression and random forest models have been successfully trained and executed.

CHAPTER – VIII
MODEL EVALUATION

Model evaluation in machine learning refers to the process of assessing the performance and effectiveness of a trained model on unseen data. This involves various techniques and metrics to understand how well the model generalizes to new, unseen samples. Evaluation is crucial for determining whether the model meets the desired objectives and for comparing different models to choose the best one for a particular task.

Model evaluation is critically important in machine learning for several reasons:

1. Performance Assessment: Evaluation helps assess how well a model generalizes to new, unseen data. It provides insights into how effectively the model has learned from the training data and whether it can make accurate predictions on real-world data.

2. Model Selection: Evaluation allows for comparing different models and selecting the best-performing one for a given task. By comparing performance metrics across multiple models, data scientists can choose the most suitable algorithm and configuration to achieve the desired outcomes.

3. Optimization: Evaluation metrics guide the optimization process by identifying areas where the model performs poorly. This information helps data scientists refine the model, fine-tune hyperparameters, and improve its performance.

4. Decision Making: Model evaluation aids in decision-making processes, such as whether to deploy a model into production or which features to prioritize for further engineering. It provides stakeholders with confidence in the model's reliability and utility.

5. Risk Management: Understanding the limitations and potential biases of a model is crucial for managing risks associated with its deployment. Evaluation helps identify scenarios where the model may underperform or make erroneous predictions, enabling risk mitigation strategies to be implemented.

6. Resource Allocation: Efficient allocation of resources, such as time, budget, and computational power, is essential in machine learning projects. Model evaluation helps prioritize efforts by focusing on models that offer the best trade-offs between performance and resource consumption.

7. Regulatory Compliance: In regulated industries, such as finance and healthcare, model evaluation is necessary to ensure compliance with legal and ethical standards. It helps demonstrate the fairness, transparency, and accountability of machine learning systems.

Overall, model evaluation is a fundamental aspect of the machine learning lifecycle, contributing to the development of robust, reliable, and trustworthy AI systems that deliver value in real-world applications.

Metrics for Model Evaluation:

In machine learning, various metrics are used to evaluate the performance of models. The choice of metric depends on the type of problem being addressed (classification, regression, clustering, etc.) and the specific objectives of the analysis. Here are some commonly used metrics:

For Classification Problems:

In machine learning, various metrics are used to evaluate the performance of models. The choice of metric depends on the type of problem being addressed (classification, regression, clustering, etc.) and the specific objectives of the analysis. Here are some commonly used metrics:

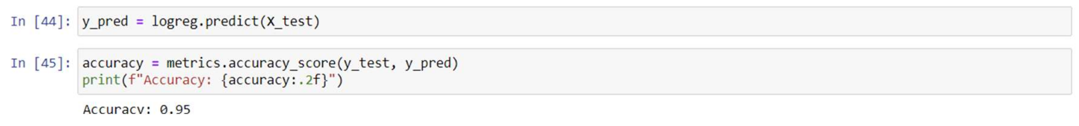
For Classification Problems:

1. **Accuracy:** Measures the proportion of correctly classified instances out of the total instances.
2. **Precision:** Measures the proportion of correctly predicted positive cases out of all predicted positive cases.
3. **Recall (Sensitivity):** Measures the proportion of correctly predicted positive cases out of all actual positive cases.
4. **F1 Score:** The harmonic mean of precision and recall, providing a balance between the two metrics.
5. **ROC-AUC Score:** Area under the Receiver Operating Characteristic curve, which measures the model's ability to distinguish between classes.
6. **Confusion Matrix:** A table summarizing the performance of a classification model, showing counts of true positives, true negatives, false positives, and false negatives.

For Regression Problems:

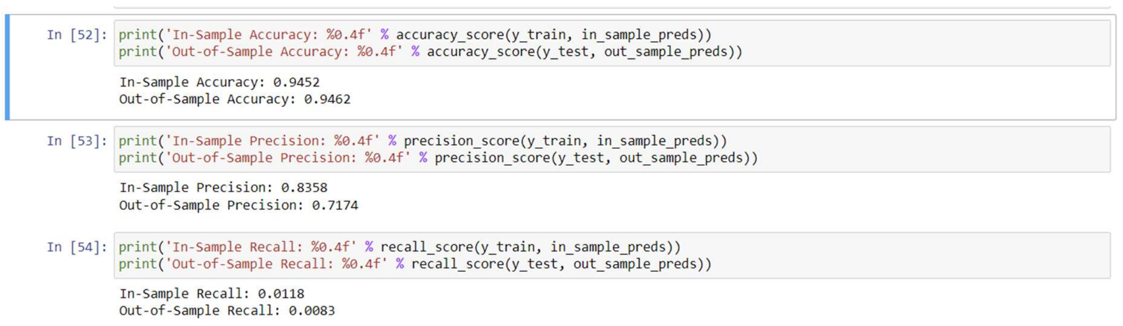
1. **Mean Squared Error (MSE):** Calculates the average squared differences between predicted and actual values.
2. **Mean Absolute Error (MAE):** Calculates the average absolute differences between predicted and actual values.
3. **R-squared (Coefficient of Determination):** Measures the proportion of the variance in the dependent variable that is predictable from the independent variables.

This is Classification Problem:



```
In [44]: y_pred = logreg.predict(X_test)

In [45]: accuracy = metrics.accuracy_score(y_test, y_pred)
         print(f"Accuracy: {accuracy:.2f}")

         Accuracy: 0.95
```

The output indicates that the accuracy of the model is 95%, which means that 95% of the instances in the test set were correctly classified by the model.



```
In [52]: print('In-Sample Accuracy: %0.4f' % accuracy_score(y_train, in_sample_preds))
         print('Out-of-Sample Accuracy: %0.4f' % accuracy_score(y_test, out_sample_preds))

         In-Sample Accuracy: 0.9452
         Out-of-Sample Accuracy: 0.9462

In [53]: print('In-Sample Precision: %0.4f' % precision_score(y_train, in_sample_preds))
         print('Out-of-Sample Precision: %0.4f' % precision_score(y_test, out_sample_preds))

         In-Sample Precision: 0.8358
         Out-of-Sample Precision: 0.7174

In [54]: print('In-Sample Recall: %0.4f' % recall_score(y_train, in_sample_preds))
         print('Out-of-Sample Recall: %0.4f' % recall_score(y_test, out_sample_preds))

         In-Sample Recall: 0.0118
         Out-of-Sample Recall: 0.0083
```

The above picture accuracy, precision score and recall score for random forest model has been calculated and accuracy and precision score for train and test is 0.9452 and 0.9462, 0.8358 train 0.7174 test and recall score is 0.0118 and 0.0083 for train and test.

CHAPTER – IX

CONCLUSION

From this Project , it appears that both the logistic regression and random forest models achieve high accuracy scores on the test set: 95% for logistic regression and approximately 95% for random forest. Additionally, the precision score is high for both models on the test set, with logistic regression achieving 100% precision and random forest achieving 83.58% precision on the train set and 100% precision on the test set.

However, the recall scores for both models are quite low, indicating that they may have difficulty correctly identifying positive instances (instances of interest) from the dataset. This suggests that the models might be overly conservative in their predictions, leading to a higher number of false negatives.

In conclusion, both models demonstrate strong performance in terms of accuracy and precision, but there may be room for improvement in terms of recall, particularly for identifying positive instances. Depending on the specific application and the importance of correctly identifying positive cases, further optimization or adjustments to the models may be warranted.