***Introduction to Data Analytics SGA -1 (Data Loading, Cleaning, and Exploration)***

1. ***Data Acquisition***
   1. ***Downloading the Dataset***

First the imports are declared and as I am using macOS, it is throwing SSL error when trying to download the dataset from UCI Repository so will be using this code:

from ucimlrepo import fetch\_ucirepo

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import ssl

import certifi

# For macOS SSL Error

ssl.\_create\_default\_https\_context = lambda: ssl.create\_default\_context(cafile=certifi.where())

I will be using the Heart Disease Dataset from UCI Repository as it is a very common data set that contains only Numerical and Categorical Data.

Now, the code to download the dataset:

# 1.1 Download the Dataset from UCI Repository

heart\_disease = fetch\_ucirepo(id=45) # Heart Disease dataset

* 1. ***Converting the data into a pandas DataFrame***

The code to convert the data into DataFrame is as follows and also additionally storing the data as a CSV file “heart\_disease.csv”:

# 1.2 Convert dataset to DataFrame

X = heart\_disease.data.features

y = heart\_disease.data.targets

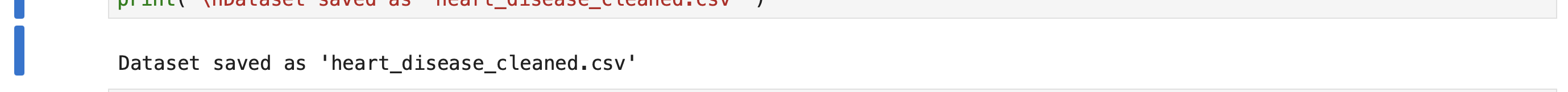
df = pd.concat([X, y], axis=1)

# Additionally saved the data in the form of a CSV File

df.to\_csv("heart\_disease.csv", index=False)

print("\nDataset saved as 'heart\_disease\_cleaned.csv'")

***Output of the Code:***



* 1. ***Displaying the First & Last Data of the Dataset***

Code to display the First & Last Data of the Dataset:

# 1.3 Display first and last five records

print("Displays first 5 Records:")

print(df.head())

print("\nDisplays last 5 Records:")

print(df.tail())

***Output of the Code:***

A screenshot of a computer

AI-generated content may be incorrect.

* 1. ***Using functions to display the given data***

Code to display column headings, statistical information, and description of the data:

# 1.4 Display column headings, statistical info, and description

# Shows the column headings

print("\nDisplay the Column Headings")

print(df.columns.tolist())

# Shows the Statistical Information (like count, mean, standard deviation, etc.)

print("\nStatistical Info:")

print(df.describe())

# Shows the Description of the Data (as in structure, meaning, distributing, types)

print("\nDescription of the data:")

print(heart\_disease.variables)

***Output of the Code:***

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* 1. ***Observations from the data***

Code to show the number of features and the examples in the dataset and the types of data attributes:

# 1.5 Observations

# Shows No. of Features and Examples in the Dataset

print("\nObservations:")

print(f"Number of Features: {df.shape[1]}")

print(f"Number of Examples: {df.shape[0]}")

# Shows the Types of Data Attributes

print("Types of Attributes:")

print("Data Attributes\tType")

print(df.dtypes)

print(df.dtypes.value\_counts())

* So, from the dataset it can be concluded that the dataset consist of 303 patients (example) and 14 features related to medicine and heart disease diagnosis.
* The attributes “ca” and “thal” have missing values.
* The dataset includes medically significant features such as blood pressure (trestbps), cholesterol level (chol), chest pain type (cp), and maximum heart rate (thalach), which are crucial for diagnosis.

***Output of the Code:***

A screenshot of a computer

AI-generated content may be incorrect.

1. ***Data Preparation***
   1. ***Checking for Duplicate, missing, inconsistent and Outliers***

The python code for checking are:

# 2.1 Check for duplicates, missing values, inconsistencies

# Check for Duplicate Data

print("\nDuplicate Data:")

print(df.duplicated().sum())

# Check for Missing Values

print("\nMissing Values:")

print(df.isnull().sum())

# Check for inconsistent data (i.e. unexpected values in categorical and numeric columns)

print("\nUnique values for categorical columns (to detect unexpected codes):")

for col in categorical\_cols:

print(f"{col}: {sorted(df[col].unique())}")

print("\nCheck for out-of-range numerical values (e.g., chol < 100 or > 600):")

print(df[(df['chol'] < 100) | (df['chol'] > 600)])

# Check for outliers using boxplot

print("\nBoxPlot")

plt.figure(figsize=(15, 8))

df.select\_dtypes(include='number').boxplot(rot=90)

plt.title("Boxplots to Detect Outliers")

plt.show()

From the inconsistent data checking and output it is clear that the data is not inconsistent and there is no duplicate data and there is missing data. The outliers can be seen using the Boxplots.

***Output of the Code:***

A screenshot of a computer program

AI-generated content may be incorrect.

A graph with numbers and lines

AI-generated content may be incorrect.

* 1. ***Applying techniques to remove duplicate, missing, and outliers data***

The code for removing duplicate, missing and outlier data is:

# 2.2 Applying Techniques to remove duplicate, missing, inconsistent and outlier data

# Remove duplicate data (although from the above data it is clear that there is no duplicate data)

df = df.drop\_duplicates()

# Drop missing data (will remove the thal one)

df = df.dropna()

# Outlier removal using IQR for all numeric columns (excluding categorical ones)

# To check for outliers — exclude encoded categorical columns if needed i.e. only numerical columns

numerical\_cols = df.select\_dtypes(include=[np.number]).columns.tolist()

# Remove outliers for each column using IQR (Inter-Quartile Range)

for col in numerical\_cols:

Q1 = df[col].quantile(0.25)

Q3 = df[col].quantile(0.75)

IQR = Q3 - Q1

before = df.shape[0]

df = df[~((df[col] < Q1 - 1.5 \* IQR) | (df[col] > Q3 + 1.5 \* IQR))]

after = df.shape[0]

print(f"Removed {before - after} outliers from '{col}' using IQR method.")

***Output of the Code (only showing the outlier removal using IQR):***

A screenshot of a computer program

AI-generated content may be incorrect.

* 1. ***Encode Categorical using either One-hot Encoding or Label Encoding***

***One Hot Encoding*** is a method for converting categorical variables into a binary format. It creates new columns for each category where 1 means the category is present and 0 means it is not. The primary purpose of One Hot Encoding is to ensure that categorical data can be effectively used in machine learning models.

Therefore, in this case of the chosen dataset, One Hot Encoding have to be used as the Encoding Technique.

The code for the same is :

# 2.3 Encode categorical variables

# Manually define categorical columns from the 1.4 chapter data (Description of the Data)

categorical\_cols = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'thal']

# Apply one-hot encoding

df = pd.get\_dummies(df, columns=categorical\_cols, drop\_first=True)

print("\nOne-Hot Encoding applied to:", categorical\_cols)

***Output of the Code:***



* 1. ***Report of the Observation***

The final observations that can be made about the dataset is:

* Even though there is no duplicate data, the code is given to ensure clarity of knowledge.
* From the check of data inconsistency, the data is not inconsistent.
* There were few missing values which are removed using the “dropna()” function of pandas.
* There were quite a few outliers, which were removed using the IQR (Inter-Quartile Range) Technique.
* Lastly, for encoding the categorical data, One Hot Encoding Technique is used.

1. ***Data Exploration using Visualizations***
   1. ***Creation of Scatter Plots for each feature against the target variable***

The code for Scatter Plot creation for each feature is as follow:

# 3.1 Scatter plots of features vs target

target = df.columns[-1]

for col in df.columns[:-1]:

if pd.api.types.is\_numeric\_dtype(df[col]):

plt.figure(figsize=(5, 3))

plt.scatter(df[col], df[target], alpha=0.5)

plt.title(f"{col} vs {target}")

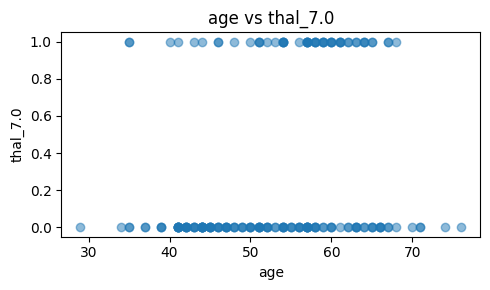
plt.xlabel(col)

plt.ylabel(target)

plt.tight\_layout()

plt.show()

The Scatter Plots are as follows:

 A graph of blue dots

AI-generated content may be incorrect.

A chart with blue dots

AI-generated content may be incorrect.A graph with blue dots

AI-generated content may be incorrect.

A graph of blue dots

AI-generated content may be incorrect. A graph with numbers and a line

AI-generated content may be incorrect.

A graph with numbers and lines

AI-generated content may be incorrect. A graph with numbers and lines

AI-generated content may be incorrect.

A graph with numbers and lines

AI-generated content may be incorrect. A graph with numbers and lines

AI-generated content may be incorrect.

A graph with numbers and lines

AI-generated content may be incorrect. A graph with numbers and lines

AI-generated content may be incorrect.

A graph with numbers and symbols

AI-generated content may be incorrect. A graph with lines and numbers

AI-generated content may be incorrect.

A graph with lines and numbers

AI-generated content may be incorrect. A graph with numbers and lines

AI-generated content may be incorrect.

* 1. ***Performing EDA using 2 additional visualization (like pair plot, heat map, correlation plot, regression plot)***

For performing EDA, these 2 visualization can be used namely, Heatmaps and Pair Plots for the following reasons:

* **Correlation Heatmap** helps in identifying which numerical features are strongly correlated with the target.
* **Pair plots** provide a visual understanding of relationships and class separability between top features and the target variable, especially in binary or multiclass problems.
* **Regression plots** are mainly useful for showing linear trends between two continuous variables. Since the target variable is categorical (heart disease presence), regression plots are not viable here.

Therefore, the code for the heat maps and pair plot are as follows:

# 3.2 Heatmap and pairplot

corr = df.corr()

plt.figure(figsize=(12, 8))

sns.heatmap(corr, annot=True, fmt=".2f", cmap="coolwarm")

plt.title("Correlation Heatmap")

plt.show()

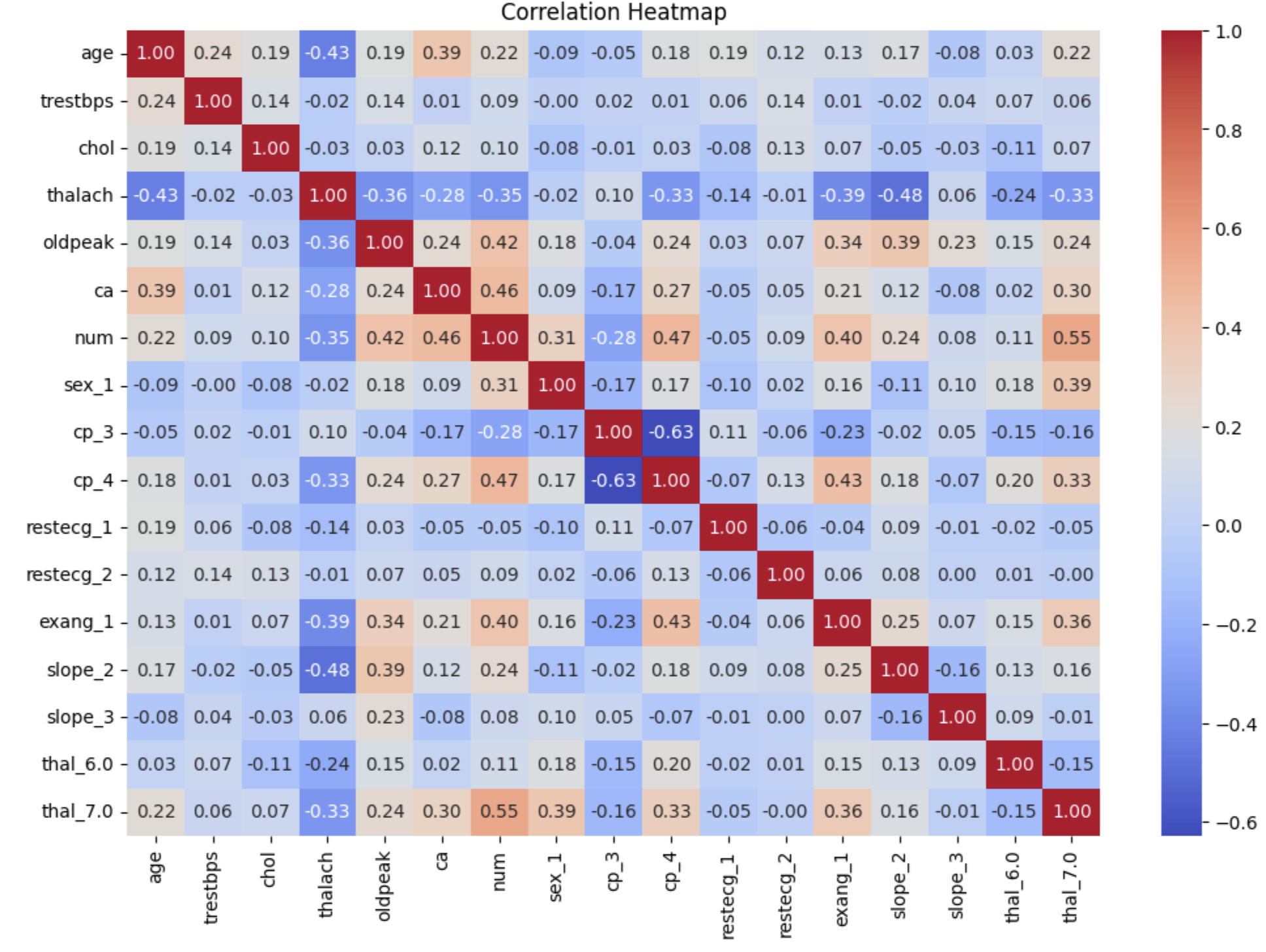
top\_corr = corr[target].abs().sort\_values(ascending=False).index[1:5].tolist()

sns.pairplot(df[top\_corr + [target]], hue=target)

plt.suptitle("Pairplot of Top Correlated Features with Target", y=1.02)

plt.show()

***Correlation Heatmap***



***Pair Plot***

