MATH ORANGE LEVEL PROJECT

Name : Akshat Sharma
SRN : PES1UG24CS042
Section : Sem 3 - A CSE

INTRODUCTION TO CHOSEN DATASET

Description of the dataset used for the project :

I have used an open publicly accessible dataset from **Kaggle** , which is based on **Heart Disease** medical information from patients.

Reference : https://www.kaggle.com/datasets/ritwikb3/heart-disease-statlog

The columns (highlighted in bold) represent the following parameters (mostly medical) :

Sample Dataset Photo

A	B	C	D	E	F	G	H	I	J	K	L	M	N
age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
70	1	3	130	322	0	2	109	0	2.4	1	3	1	1
67	0	2	115	564	0	2	160	0	1.6	1	0	3	0
57	1	1	124	261	0	0	141	0	0.3	0	0	3	1
64	1	3	128	263	0	0	105	1	0.2	1	1	3	0
74	0	1	120	269	0	2	121	1	0.2	0	1	1	0
65	1	3	120	177	0	0	140	0	0.4	0	0	3	0
56	1	2	130	256	1	2	142	1	0.6	1	1	2	1
59	1	3	110	239	0	2	142	1	1.2	1	1	3	1
60	1	3	140	293	0	2	170	0	1.2	1	2	3	1
63	0	3	150	407	0	2	154	0	4	1	3	3	1
59	1	3	135	234	0	0	161	0	0.5	1	0	3	0
53	1	3	142	226	0	2	111	1	0	0	0	3	0
44	1	2	140	235	0	2	180	0	0	0	0	1	0
61	1	0	134	234	0	0	145	0	2.6	1	2	1	1
57	0	3	128	303	0	2	159	0	0	0	1	1	0

List of Column Descriptions for further Info :

- **'Age': Patients Age in years (Numeric)**

- **'Sex': Gender (Male : 1; Female : 0) (Nominal)**
- **'cp': Type of chest pain experienced by a patient. This term is categorized into 4 categories 0 typical angina, 1 atypical angina, 2 non- anginal pain, 3 asymptomatic (Nominal)**
- **'trestbps': patient's level of blood pressure at resting mode in mm/HG (Numerical)**
- **'chol': Serum cholesterol in mg/dl (Numeric)**
- **'fbs': Blood sugar levels on fasting > 120 mg/dl represents as 1 in case of true and 0 as false (Nominal)**
- **'restecg': Result of electrocardiogram while at rest are represented in 3 distinct values 0 : Normal 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV) 2: showing probable or definite left ventricular hypertrophyby Estes' criteria (Nominal)**
- **'thalach': Maximum heart rate achieved (Numeric)**
- **'exang': Angina induced by exercise 0 depicting NO 1 depicting Yes (Nominal)**
- **'oldpeak': Exercise induced ST-depression in relative with the state of rest (Numeric)**
- **'slope': ST segment measured in terms of slope during peak exercise 0: up sloping; 1: flat; 2: down sloping(Nominal)**
- **'ca': The number of major vessels (0–3)(nominal)**
- **'thal': A blood disorder called thalassemia 0: NULL 1: normal blood flow 2: fixed defect (no blood flow in some part of the heart) 3: reversible defect (a blood flow is observed but it is not normal(nominal)**
- **'target': It is the target variable which we have to predict 1 means patient is suffering from heart disease and 0 means patient is normal.**

PROJECT OBJECTIVES

Objectives :

1. **Perform Data Cleaning**
2. **Conduct Descriptive Statistical Analysis**
3. **Visualize the Data**
4. **Check for Normality**
5. **Construct a Confidence Interval**
6. **Perform Hypothesis Testing**
7. **Fit a Linear Regression Model**

The project is split into 7 steps handling each objective and showing the output of the respective steps in order:

Additional Step 0 :

Since the public dataset that I have used from Kaggle does not have missing values by default , I am artificially adding missing values using `np.random.seed(42)`

CODES AND OUTPUTS

Imports:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
file_path = 'Heart_disease_statlog.csv'
df = pd.read_csv(file_path)
```

Step 0: Artificially Introducing Missing Values for Imputation

```
print(f"Missing values BEFORE creating them:
\n{df.isnull().sum().loc[lambda x: x > 0]}")
np.random.seed(42) #This is a random seed number I have set
# I'm introducing 30 missing values in 'chol' column
chol_indices = df.sample(n=30).index
df.loc[chol_indices, 'chol'] = np.nan
# Introduce 20 missing values in 'trestbps'
trestbps_indices = df.sample(n=20).index
df.loc[trestbps_indices, 'trestbps'] = np.nan
print(f"\nMissing values AFTER creating them:
\n{df.isnull().sum().loc[lambda x: x > 0]}")
```

Step 0 Output:

```
PS C:\Users\Akshat\Desktop\OrangeMath\heart> python .\main2.py
--- Step 0: Artificially Creating Missing Values ---
Missing values BEFORE creating them:
Series([], dtype: int64)

Missing values AFTER creating them:
trestbps    20
chol        30
dtype: int64
```

Step 1: Data Cleaning

```
print("\n--- Objective 1: Data Cleaning ---")
# Im filling numerical missing values using median
chol_median = df['chol'].median()
df['chol'] = df['chol'].fillna(chol_median)
print(f"Filled 'chol' missing values with median: {chol_median}")

trestbps_median = df['trestbps'].median()
df['trestbps'] = df['trestbps'].fillna(trestbps_median)
print(f"Filled 'trestbps' missing values with median: {trestbps_median}")
print(f"\nMissing values AFTER cleaning: \n{df.isnull().sum().loc[lambda
x: x > 0]}")
if df.isnull().sum().sum() == 0:
    print("Data is now clean.")
```

Step 1 Output

```
PS C:\Users\Akshat\Desktop\OrangeMath\heart> python .\main2.py
--- Objective 1: Data Cleaning ---
Filled 'chol' missing values with median: 243.5
Filled 'trestbps' missing values with median: 130.0

Missing values AFTER cleaning:
Series([], dtype: int64)
Data is now clean.
```

Step 2: Descriptive Statistical Summary Display

```
print("\n--- Objective 2: Descriptive Statistics ---")
```

```
print("\nFull Numeric Features Summary:")
print(df.describe().T)
```

Step 2 Output :

```
PS C:\Users\Akshat\Desktop\OrangeMath\heart> python .\main2.py

--- Objective 2: Descriptive Statistics ---

Full Numeric Features Summary:
          count        mean        std    min    25%    50%    75%    max
age       270.0   54.433333   9.109067   29.0   48.0   55.0   61.0   77.0
sex       270.0    0.677778   0.468195    0.0    0.0    1.0    1.0    1.0
cp        270.0    2.174074   0.950090    0.0    2.0    2.0    3.0    3.0
trestbps  270.0  131.162963  17.323387   94.0  120.0  130.0  140.0  200.0
chol      270.0  248.837037  48.092589  141.0  218.0  243.5  274.0  564.0
fbs       270.0    0.148148   0.355906    0.0    0.0    0.0    0.0    1.0
restecg   270.0    1.022222   0.997891    0.0    0.0    2.0    2.0    2.0
thalach   270.0  149.677778  23.165717   71.0  133.0  153.5  166.0  202.0
exang     270.0    0.329630   0.470952    0.0    0.0    0.0    1.0    1.0
oldpeak   270.0    1.050000   1.145210    0.0    0.0    0.8    1.6    6.2
slope     270.0    0.585185   0.614390    0.0    0.0    1.0    1.0    2.0
ca        270.0    0.670370   0.943896    0.0    0.0    0.0    1.0    3.0
thal      270.0    1.822222   0.959140    1.0    1.0    1.0    3.0    3.0
target    270.0    0.444444   0.497827    0.0    0.0    0.0    1.0    1.0
```

Step 3: Data Visualization

(I have included all types of plots which will be shown below - Histogram,Boxplot,ScatterPlot and Corr heatmap for heart disease column correlations)

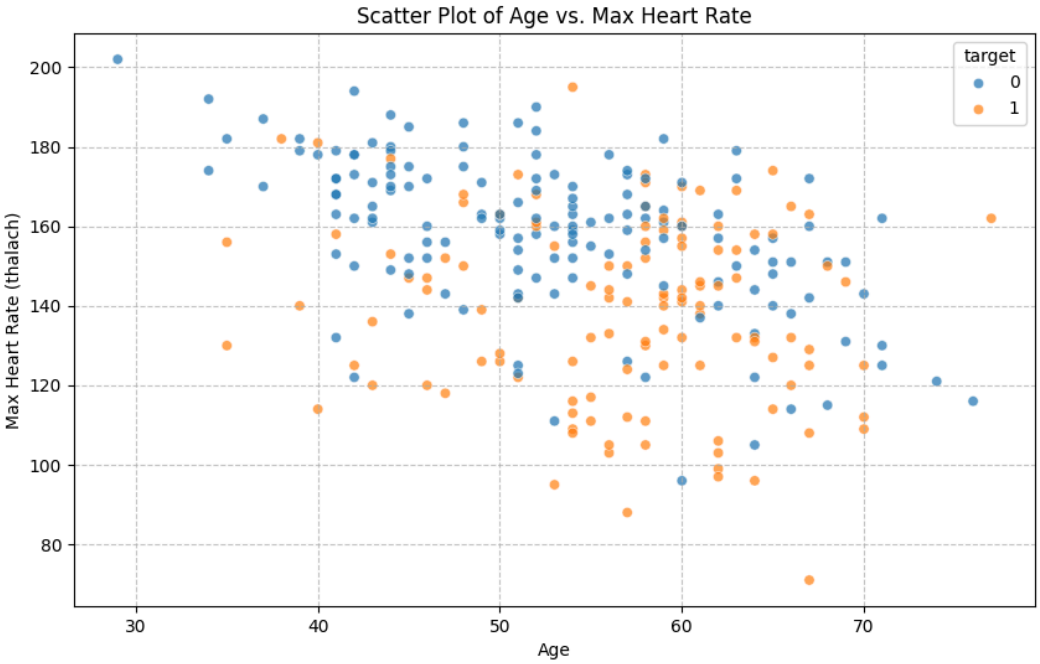
```
print("\n--- Step 3: Data Visualization (Saving Plots) ---")
# This is the histogram for cholesterol level
plt.figure(figsize=(10, 6))
sns.histplot(df['chol'], kde=True, bins=30)
plt.title('Histogram of Patient Cholesterol (chol)')
plt.xlabel('Cholesterol (mg/dL)')
plt.ylabel('Frequency')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.savefig('chol_histogram.png')
print("Saved chol_histogram.png")
# This is the box plot
plt.figure(figsize=(12, 7))
```

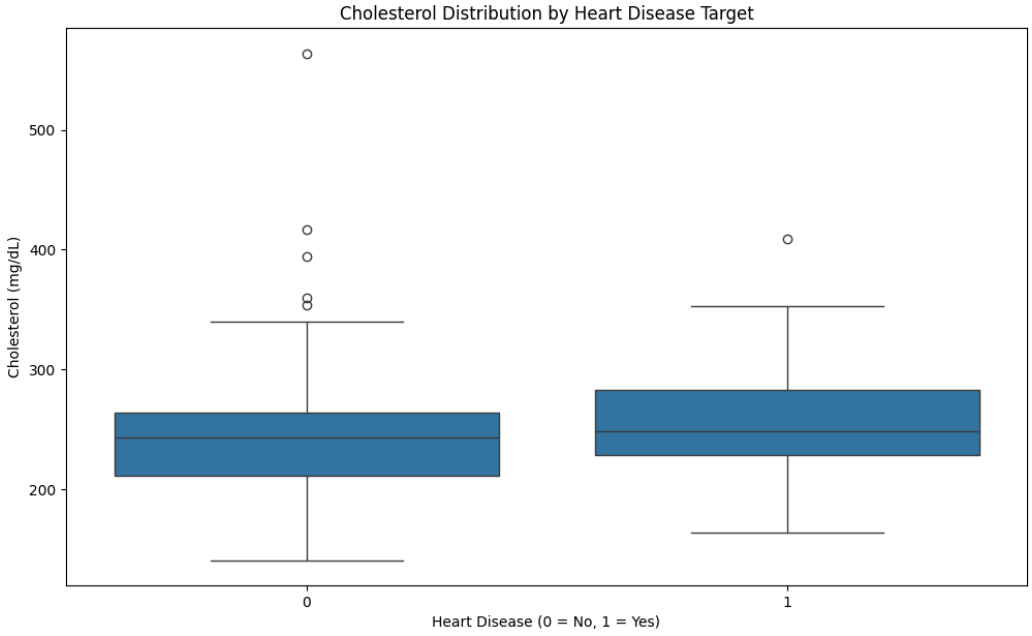
```
sns.boxplot(x='target', y='chol', data=df)
plt.title('Cholesterol Distribution by Heart Disease Target')
plt.xlabel('Heart Disease (0 = No, 1 = Yes)')
plt.ylabel('Cholesterol (mg/dL)')
plt.savefig('chol_by_target_boxplot.png')
print("Saved chol_by_target_boxplot.png")
# This is the age vs thalach scatter plot ( thalach is max heart rate
explained above)
plt.figure(figsize=(10, 6))
sns.scatterplot(x='age', y='thalach', data=df, hue='target', alpha=0.7)
plt.title('Scatter Plot of Age vs. Max Heart Rate')
plt.xlabel('Age')
plt.ylabel('Max Heart Rate (thalach)')
plt.grid(True, linestyle='--', alpha=0.7)
plt.savefig('age_vs_thalach_scatter.png')
print("Saved age_vs_thalach_scatter.png")
# This is the correlation heatmap below
plt.figure(figsize=(12, 9))
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f',
linewidths=0.5)
plt.title('Correlation Heatmap of Heart Disease Features')
plt.savefig('correlation_heatmap.png')
print("Saved correlation_heatmap.png")
```

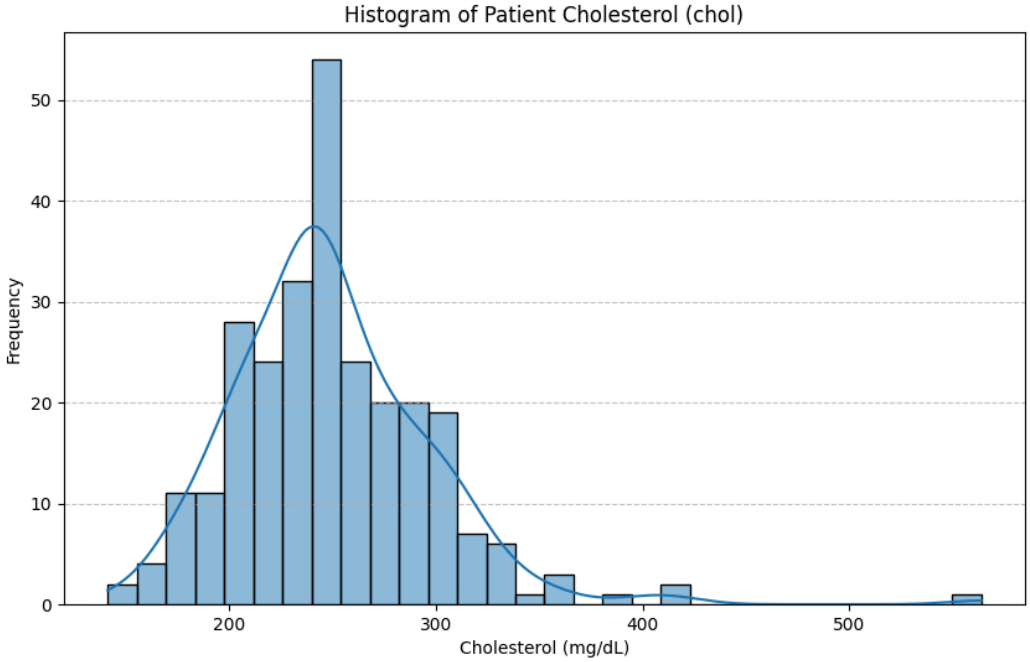
Step 3 Output :

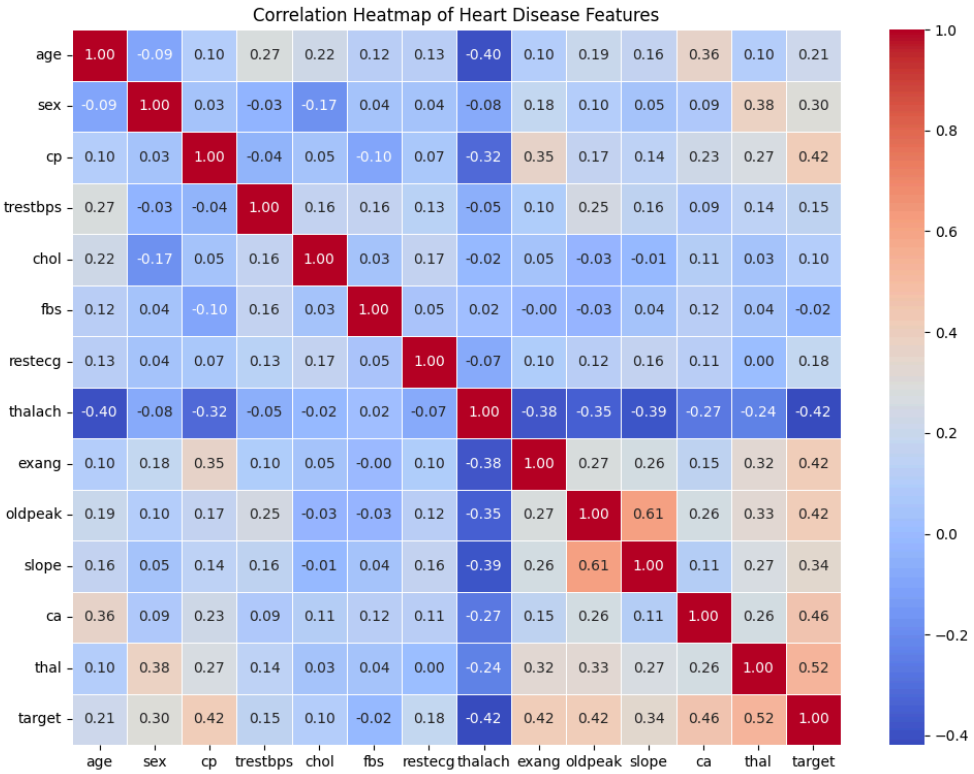
```
Saved chol_histogram.png
Saved chol_by_target_boxplot.png
Saved age_vs_thalach_scatter.png
Saved correlation_heatmap.png
```

Step 3 Graphs :


Scatter Plot of Age vs. Max Heart Rate
target
0
1
Max Heart Rate (thalach)
200
180
160
140
120
100
80
30
40
50
60
70
Age


Cholesterol Distribution by Heart Disease Target
Cholesterol (mg/dL)
500
400
300
200
0
1
Heart Disease (0 = No, 1 = Yes)


Histogram of Patient Cholesterol (chol)
Frequency
Cholesterol (mg/dL)
0
10
20
30
40
50
200
300
400
500


Correlation Heatmap of Heart Disease Features
age 1.00 -0.09 0.10 0.27 0.22 0.12 0.13 -0.40 0.10 0.19 0.16 0.36 0.10 0.21
sex -0.09 1.00 0.03 -0.03 -0.17 0.04 0.04 -0.08 0.18 0.10 0.05 0.09 0.38 0.30
cp 0.10 0.03 1.00 -0.04 0.05 -0.10 0.07 -0.32 0.35 0.17 0.14 0.23 0.27 0.42
trestbps 0.27 -0.03 -0.04 1.00 0.16 0.16 0.13 -0.05 0.10 0.25 0.16 0.09 0.14 0.15
chol 0.22 -0.17 0.05 0.16 1.00 0.03 0.17 -0.02 0.05 -0.03 -0.01 0.11 0.03 0.10
fbs 0.12 0.04 -0.10 0.16 0.03 1.00 0.05 0.02 -0.00 -0.03 0.04 0.12 0.04 -0.02
restecg 0.13 0.04 0.07 0.13 0.17 0.05 1.00 -0.07 0.10 0.12 0.16 0.11 0.00 0.18
thalach -0.40 -0.08 -0.32 -0.05 -0.02 0.02 -0.07 1.00 -0.38 -0.35 -0.39 -0.27 -0.24 -0.42
exang 0.10 0.18 0.35 0.10 0.05 -0.00 0.10 -0.38 1.00 0.27 0.26 0.15 0.32 0.42
oldpeak 0.19 0.10 0.17 0.25 -0.03 -0.03 0.12 -0.35 0.27 1.00 0.61 0.26 0.33 0.42
slope 0.16 0.05 0.14 0.16 -0.01 0.04 0.16 -0.39 0.26 0.61 1.00 0.11 0.27 0.34
ca 0.36 0.09 0.23 0.09 0.11 0.12 0.11 -0.27 0.15 0.26 0.11 1.00 0.26 0.46
thal 0.10 0.38 0.27 0.14 0.03 0.04 0.00 -0.24 0.32 0.33 0.27 0.26 1.00 0.52
target 0.21 0.30 0.42 0.15 0.10 -0.02 0.18 -0.42 0.42 0.42 0.34 0.46 0.52 1.00
age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal target
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

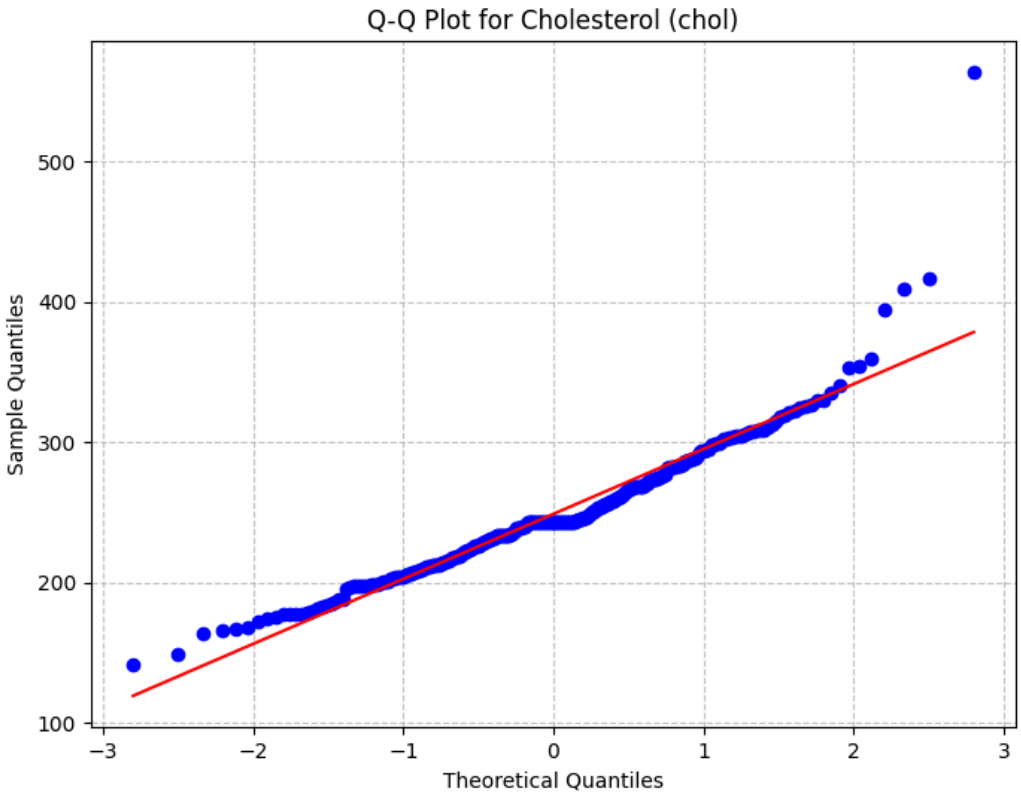
Step 4: Normality Check with QQ Plot

```
print("\n--- Objective 4: Normality Check (Saving Plot) ---")
plt.figure(figsize=(8, 6))
stats.probplot(df['chol'], dist="norm", plot=plt)
plt.title('Q-Q Plot for Cholesterol (chol)')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.grid(True, linestyle='--', alpha=0.7)
plt.savefig('chol_qq_plot.png')
print("Saved chol_qq_plot.png")
```

Step 4 Output:

```
PS C:\Users\Akshat\Desktop\OrangeMath\heart> python .\main2.py
--- Objective 4: Normality Check (Saving Plot) ---
Saved chol_qq_plot.png
```

Step 4 Graph :



Step 5 : Confidence Interval - 95 %

```
print("\n--- Objective 5: 95% Confidence Interval for Mean Cholesterol
---")
data_col = df['chol']
confidence_level = 0.95
degrees_freedom = len(data_col) - 1
mean_val = np.mean(data_col)
std_error_val = stats.sem(data_col)
ci = stats.t.interval(confidence_level, degrees_freedom, loc=mean_val,
scale=std_error_val)
print(f"Mean Cholesterol: {mean_val:,.2f} mg/dL")
print(f"95% Confidence Interval for Mean Cholesterol: (${ci[0]:,.2f},
${ci[1]:,.2f})")
```

Step 5 Output :

```
--- Objective 5: 95% Confidence Interval for Mean Cholesterol ---
Mean Cholesterol: 248.84 mg/dL
95% Confidence Interval for Mean Cholesterol: ($243.07, $254.60)
```

Step 6: Hypothesis Test for Mean Cholesterol.

We will assume Null Hypothesis and Alternate Hypothesis and accept and reject the hypothesis based on whatever p value we get , as follows:

```
print("\n--- Objective 6: Hypothesis Test for Mean Cholesterol ---")
hypothesized_mean = 200
alpha = 0.05
t_statistic, p_value = stats.ttest_1samp(df['chol'], hypothesized_mean)
print(f"Null Hypothesis (H0): Mean Cholesterol = ${hypothesized_mean}")
print(f"Alternative Hypothesis (Ha): Mean Cholesterol !=
${hypothesized_mean}")
print(f"Significance Level (alpha): {alpha}")
print(f"t-statistic: {t_statistic:.2f}")
print(f"p-value: {p_value}")
if p_value < alpha:
    print("Result: Reject the null hypothesis (H0).")
else:
    print("Result: Fail to reject the null hypothesis (H0).")
```

Step 6 Output:

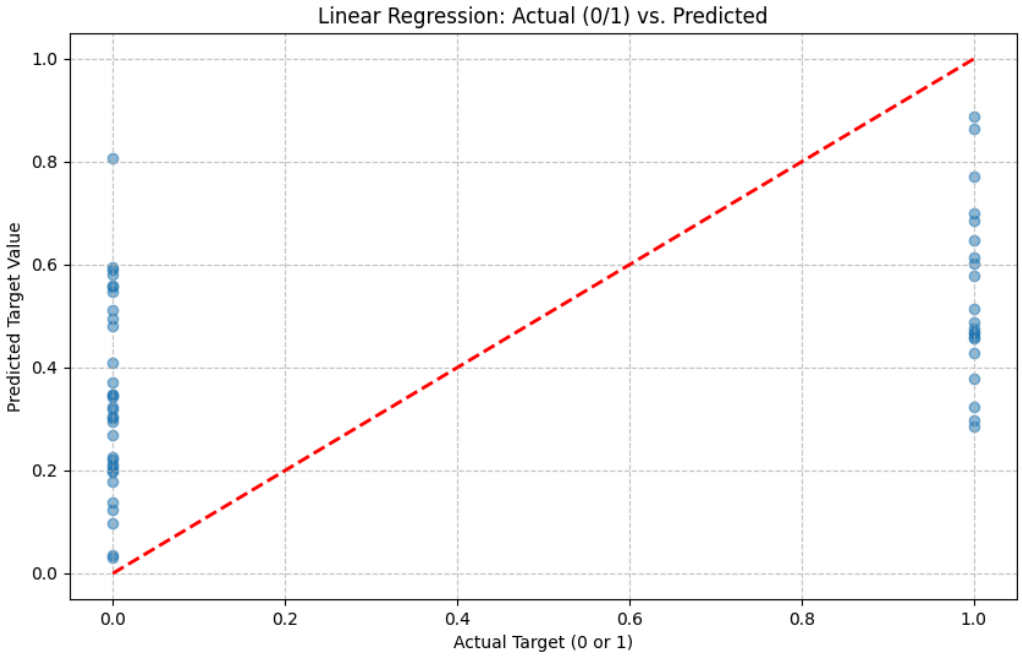
```
--- Objective 6: Hypothesis Test for Mean Cholesterol ---
Null Hypothesis (H0): Mean Cholesterol = $200
Alternative Hypothesis (Ha): Mean Cholesterol != $200
Significance Level (alpha): 0.05
t-statistic: 16.69
p-value: 2.1335449437509255e-43
Result: Reject the null hypothesis (H0).
```

Step 7: Linear Regression Model

```
print("\n--- Objective 7: Linear Regression Model ---")
print("Note: I am using Linear Regression.")
predictors = ['age', 'chol', 'trestbps', 'thalach']
target = 'target'
X = df[predictors]
y = df[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print(f"Target Variable: '{target}'")
print(f"Predictor Variables: {predictors}")
print("\nModel Performance Metrics:")
print(f"R-squared (R²): {r2:.4f}")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")
# Im plotting the graph to show the results (Actual vs. Predicted)
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', lw=2)
plt.title('Linear Regression: Actual (0/1) vs. Predicted')
plt.xlabel('Actual Target (0 or 1)')
plt.ylabel('Predicted Target Value')
plt.grid(True, linestyle='--', alpha=0.7)
```

```
plt.savefig('regression_actual_vs_pred.png')
print("\n--- Thank you Ma'am , Assignment Complete ---")
```

Step 7 Graph:



FINAL RESULTS AND CONCLUSION

The box plot provided a key insight: patients with heart disease (target=1) had a visibly higher median cholesterol level than those without heart disease (target=0).

The scatter plot confirmed a known physiological trend: a negative correlation between age and thalach (maximum heart rate), where older patients tend to have a lower max heart rate.

The correlation heatmap quantified the relationships, showed that the presence of heart disease was most strongly correlated with features like cp (chest pain type), thalach (max heart rate), and ca (number of major vessels).

Overall Conclusion: The linear regression model, which used these features, achieved an R-Squared of approximately 0.20 . This summarizes the findings: there is a definite relationship between these predictors and heart disease.

End of assignment.
Thank you.