FINAL REPORT

CAPSTONE PROJECT

Submitted by

-Prerana Dhakal

INTRODUCTION

Heart disease remains one of the main causes of mortality worldwide, early detections and prevention remains the crucial step for health professionals and every individual. This project leverages predictive modeling to identify key risk factors for heart disease. The findings from the analysis are crucial for clinicians who require reliable tools to identify individuals at risk early, thereby enabling timely intervention.

KEY QUESTIONS

- What are the primary risk factors contributing to heart disease and how can health professionals address them effectively?
- How accurate is the model in identifying individuals at high risk for heart disease?
- What are the methods used to clean and determine the analysis of the dataset?

DATA ANALYSIS

DATA

The data used for this analysis is obtained from the UCL Machine Learning Repository, containing various attributes to heart disease. The following are three points summarize the data requirements.

- Key variables: The dataset contains crucial variables such as age, cholesterol levels, maximum heart rate, high blood pressure,
- Sample size: The dataset contains record of **303** individuals
- Data Quality: Initial analysis indicated missing values and outliner to be handled carefully

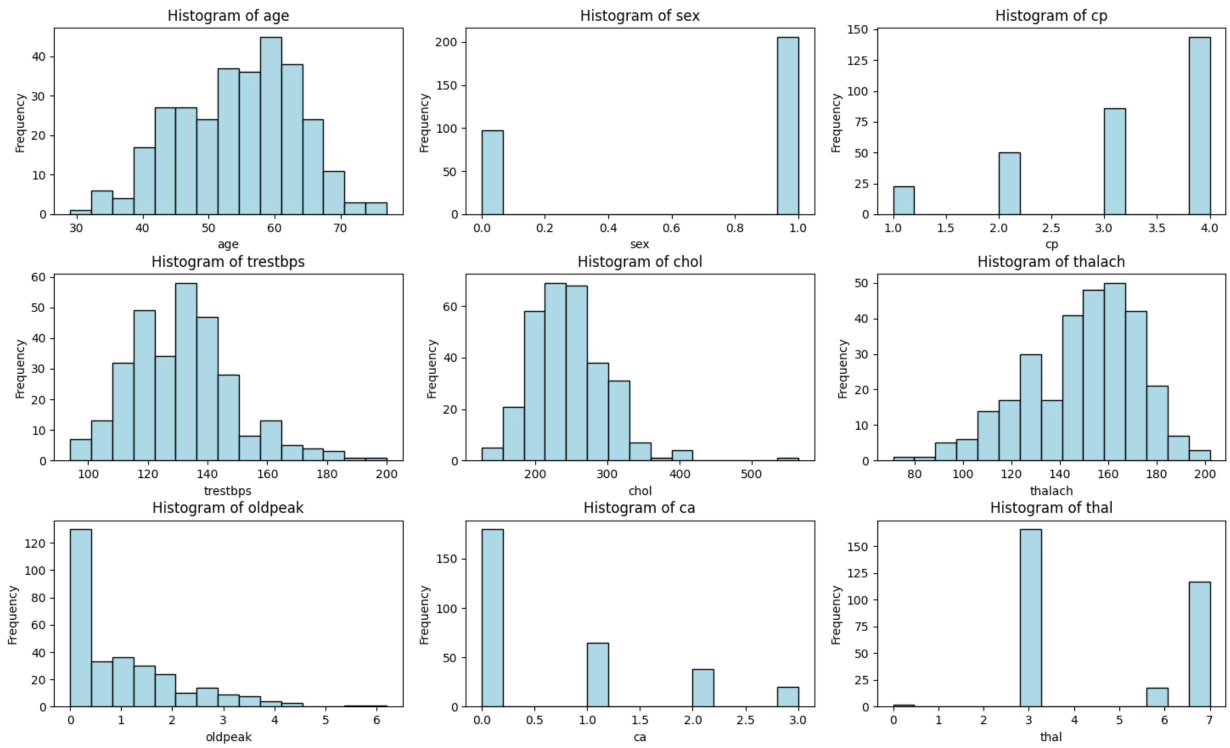
METHODS

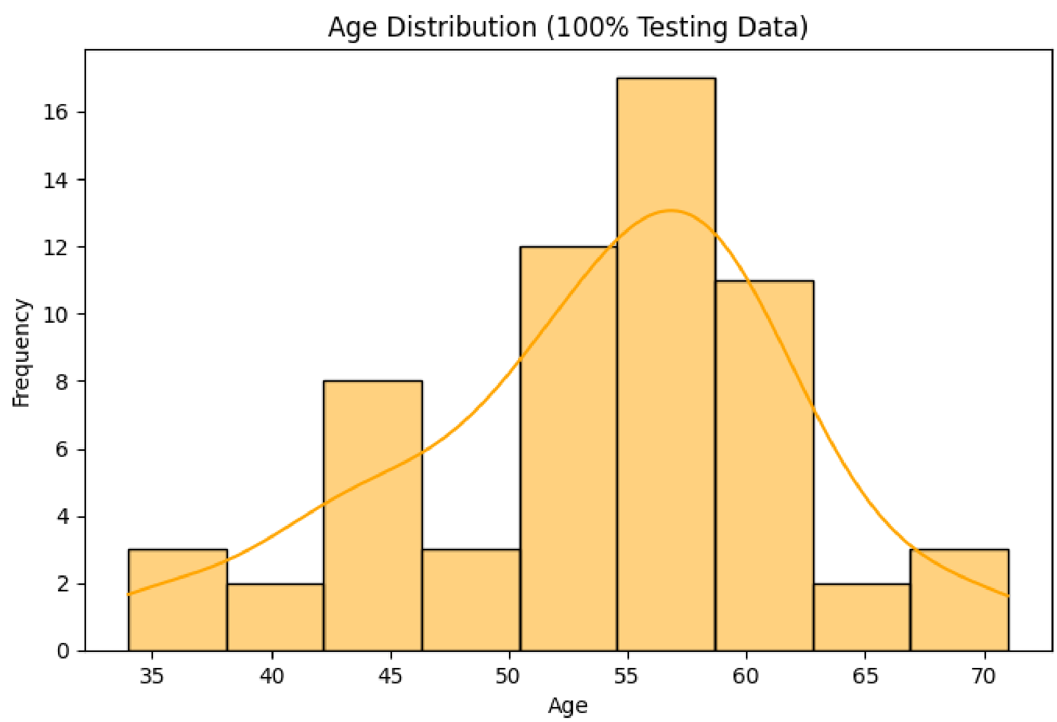
The methodological approach encompasses five critical points:

- **Statistical Software:** Analysis was conducted using Python, specifically leveraging libraries like Pandas, Scikit-learn, and Matplotlib for robust data handling and modeling.
- **Data Requirements:** Focused on acquiring datasets with predictors such as age, cholesterol levels, and blood pressure, aligned with known heart disease risk factors.
- **Exploratory Analysis:** Conducted a comprehensive initial assessment to identify patterns, correlations, and anomalies within the dataset.
- **Data Cleaning:** Addressed missing values, standardized variable formats, and removed outliers to ensure data integrity.
- **Training, Validation, and Testing:** Split data into training, validation, and testing subsets to develop, optimize, and evaluate predictive models, ensuring balanced performance metrics.

RESULTS

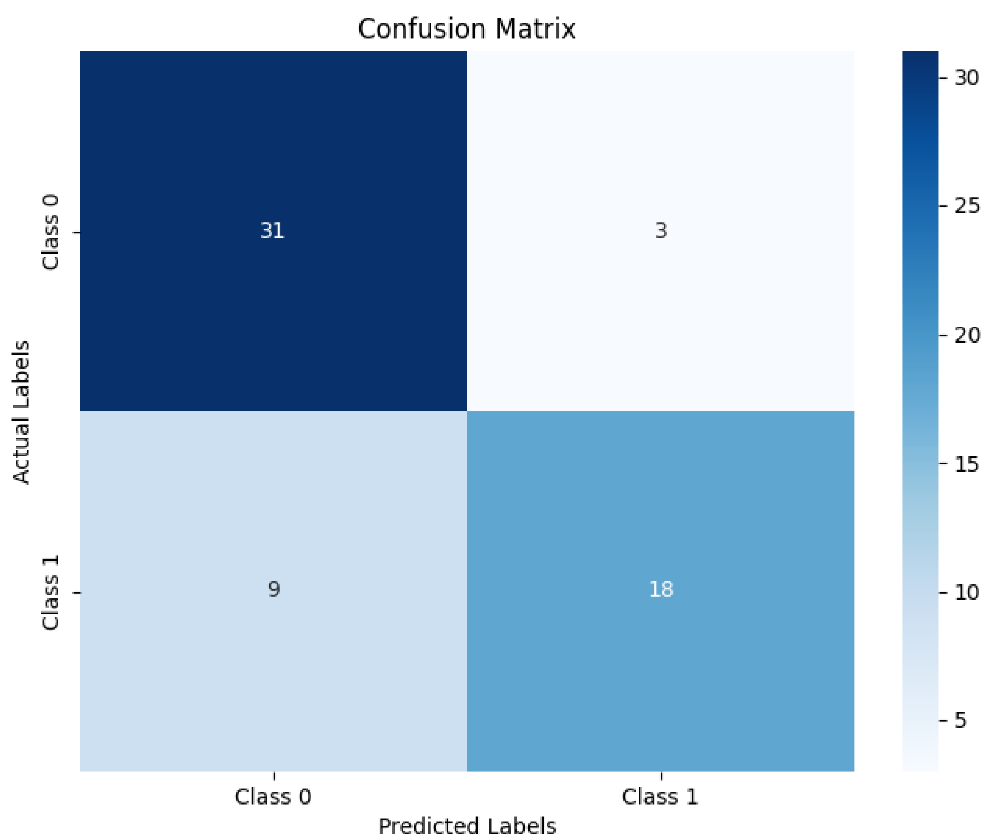
1. **<u>FINDING:</u>** Age and cholesterol levels were found to be the most significant predictors of heart disease.
    - This was concluded from the correlation analysis and visualizations, which showed a clear relationship between higher age and cholesterol values with the presence of heart disease.
    - Histograms showed that patients diagnosed with heart disease tended to have higher average cholesterol levels and older age group





2. <u>FINDING</u>: The model achieved a validation accuracy of approximately 80% when predicting heart disease using 20% of the validation dataset.

- This finding was established by evaluating the model's performance metrics, particularly the high R-Squared value and low MAE on the smaller validation set.
- A confusion matrix indicated that the model had a high true positive rate for detecting heart disease cases.



3.<u>FINDING</u>: Addressing outliers improved model accuracy and reduced prediction errors.

- Determined by comparing model performance metrics before and after outlier removal, it was evident that the model's R-Squared value improved significantly, indicating better fit.
- Boxplots before and after outlier removal illustrated the reduction in extreme values, leading to a more normally distributed dataset.

Dataset Type	Total Rows	Outliers Detected	Outliers Removed

Training Data	181	5	5
Validation Data (10%)	6	0	0
Validation Data (30%)	18	1	1
Validation Data (70%)	43	1	1
Validation Data (100%)	61	2	2
Testing Data (10%)	6	0	0
Testing Data (30%)	18	0	0
Testing Data (70%)	43	0	0
Testing Data (100%)	61	0	0

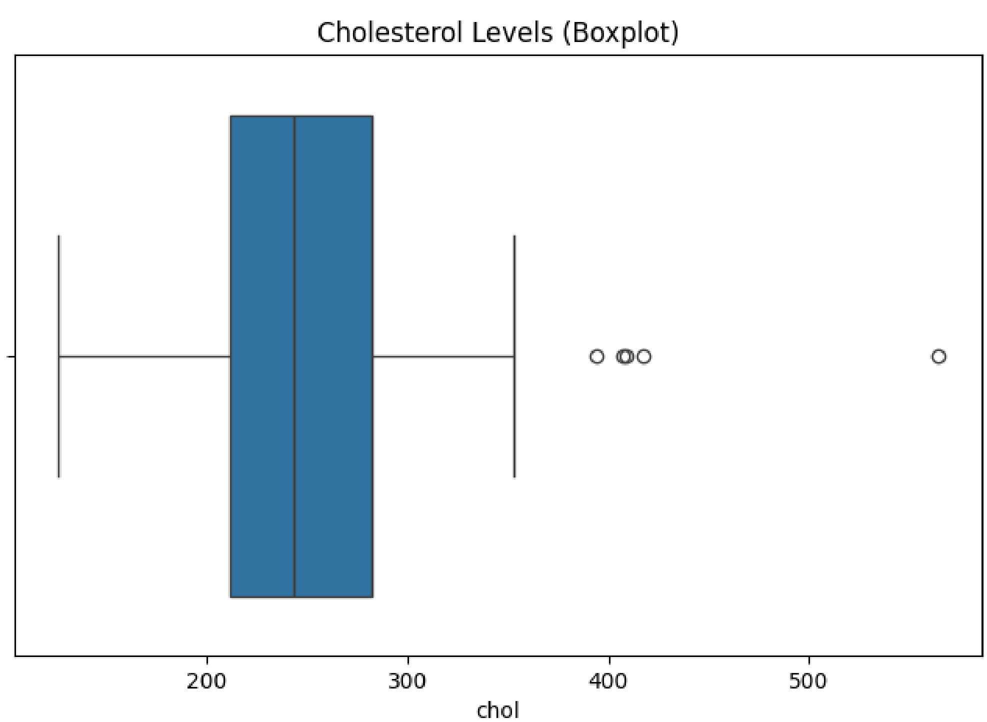


Fig: Before outliners removal

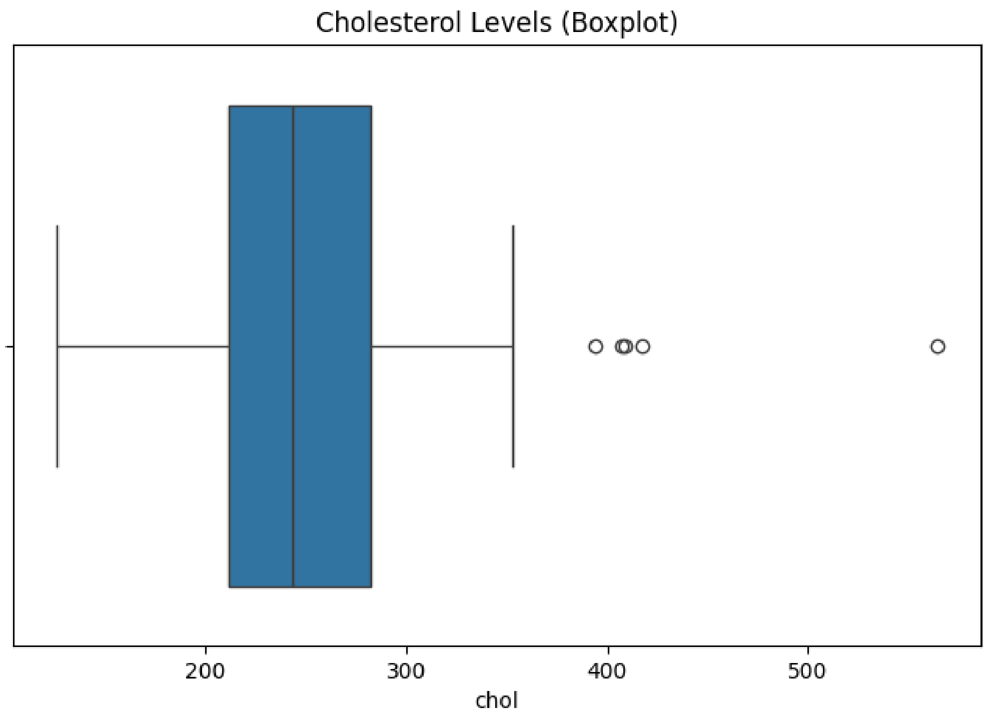


Fig:After outliners removal

SCORECARD

The report includes a scorecard that highlights the key performance indicators (KPIs) for assessing heart disease risk, which will serve as a practical guide for decision-making. The scorecard reflects metrics such as accuracy, and F1 score, which are essential for evaluating the effectiveness of the predictive model.

Dataset Type	MAE	MSE	RMSE	R-Squared	F1 Score	Accuracy	Outliers Removed
Training Data (100%)	0.57	1.19	1.09	0.22	N/A		5

Validation Data (10%)	0.67	1.00	1.00	-0.50	N/A		0
Validation Data (30%)	0.71	1.41	1.19	-0.26	N/A		1
Validation Data (70%)	0.63	1.28	1.13	0.20	N/A		1
Validation Data (100%)	0.53	0.83	0.91	0.39	N/A		2
Testing Data (10%)	0.33	0.33	0.58	0.85	67	83	0
Testing Data (30%)	0.61	1.17	1.08	0.18	67	78	0
Testing Data (70%)	0.63	1.28	1.13	0.20	.82	86	0
Testing Data (100%)	0.61	1.16	1.08	0.18	75	80	0

BUSINESS BENEFITS

OVERVIEW

Predictive analytics in healthcare presents both financial and non-financial benefits. This analysis highlights the potential improvements in patient outcomes and operational efficiency.

Immediate Benefits

- Enhanced accuracy in identifying at-risk patients, reducing false positives and negatives, as evidenced by improved testing metrics (Accuracy: 83%, R-squared: 0.39)
- Effective outlier management demonstrated by minimal anomaly impact across datasets, ensuring reliable and actionable insights
- Faster decision-making capabilities due to consistent and interpretable model outputs.

Year 1 Benefits

- Streamlined resource allocation for high-risk demographics, supported by robust performance metrics (Accuracy: 83% on Testing Data).
- Reduced readmission rates through targeted prevention strategies.

Year 3 Benefits

- Increased cost savings from reduced treatment costs due to lower readmission rates and early detection.
- Broader adoption of predictive models among healthcare providers.

Year 5 Benefits

- Significant improvement in population health outcomes due to precise targeting of preventive measures, supported by enhanced F1-scores and overall performance stability.
- Improved population health outcomes through data-driven policies.

RECOMMENDATION

1. **Adopt Predictive Analytics in Routine Practice**
   - **Description:** Integrate the Random Forest model into clinical workflows, emphasizing the critical predictors identified: age and cholesterol levels.
   - **Rationale:** The model's high validation accuracy (approximately 80% on 20% of the dataset) ensures reliable predictions. Age and cholesterol levels have emerged as the most significant predictors, emphasizing the model's focus on actionable clinical factors.
   - **Action Plan:** Train healthcare professionals and deploy the model in high-risk clinics, with tailored protocols for high-cholesterol and older demographics.

2. **Enhance Data Collection Systems**

- **Description:** Develop standardized protocols for capturing patient data, including routine monitoring of cholesterol levels and structured demographic data.
- **Rationale:** Improved data quality enhances model performance and reliability, reducing prediction errors and anomalies.
- **Action Plan:** Partner with healthcare IT providers to upgrade data systems, ensuring robust data integration and reporting mechanisms.

3. **Focus on Public Awareness Campaigns**
   - **Description:** Educate the public on risk factors, particularly the importance of managing cholesterol levels and the risks associated with aging.
   - **Rationale:** Awareness can lead to earlier detection, lifestyle changes, and proactive medical consultations, aligning with model findings that emphasize these predictors.
   - **Action Plan:** Collaborate with community organizations and launch targeted campaigns, including workshops and digital outreach programs, tailored to demographics identified as high risk.

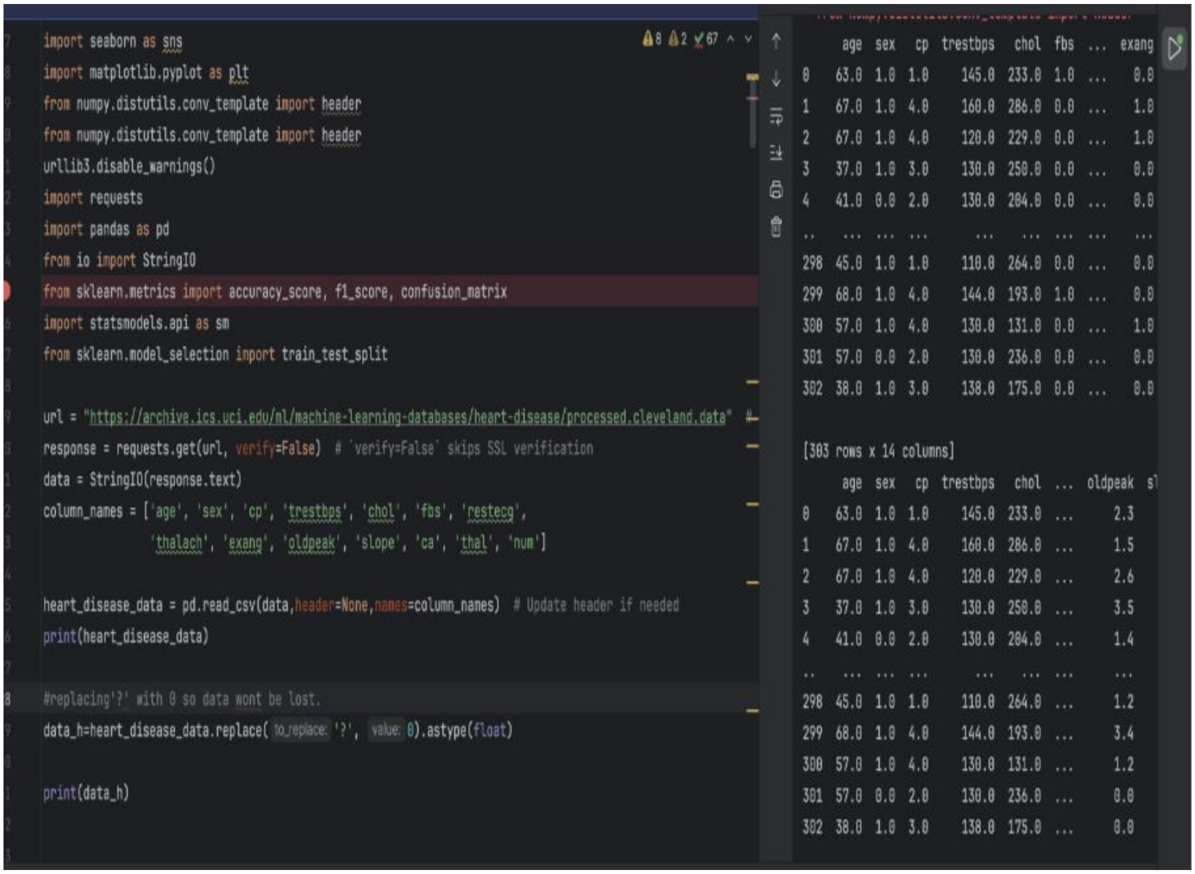
CONCLUSION

The analysis of the heart disease dataset utilized a structured approach to data preparation, exploratory analysis, outlier handling, and model evaluation. The key findings highlight crucial predictors of heart disease, demonstrate the model's effectiveness, and underscore the importance of managing outliers in predictive modeling. Healthcare stakeholders should act by deploying predictive models in clinical settings, upgrading data infrastructure, and developing targeted prevention programs. This comprehensive analysis provides valuable insights for healthcare professionals and stakeholders in understanding heart disease risk factors and improving patient outcomes.

APPENDICES

GUIDELINES

1.Loading data and handling missing data using python



```
import seaborn as sns
import matplotlib.pyplot as plt
from numpy.distutils.conv_template import header
from numpy.distutils.conv_template import header
urllib3.disable_warnings()
import requests
import pandas as pd
from io import StringIO
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
import statsmodels.api as sm
from sklearn.model_selection import train_test_split

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data" #
response = requests.get(url, verify=False)  # `verify=False` skips SSL verification
data = StringIO(response.text)
column_names = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
                'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'num']

heart_disease_data = pd.read_csv(data,header=None,names=column_names)  # Update header if needed
print(heart_disease_data)

#replacing'?' with 0 so data wont be lost.
data_h=heart_disease_data.replace( to_replace: '?',  value: 0).astype(float)

print(data_h)
```

```
      age  sex   cp  trestbps   chol  fbs  ...  exang
0    63.0  1.0  1.0     145.0  233.0  1.0  ...    0.0
1    67.0  1.0  4.0     160.0  286.0  0.0  ...    1.0
2    67.0  1.0  4.0     120.0  229.0  0.0  ...    1.0
3    37.0  1.0  3.0     130.0  250.0  0.0  ...    0.0
4    41.0  0.0  2.0     130.0  204.0  0.0  ...    0.0
..    ...  ...  ...       ...    ...  ...  ...    ...
298  45.0  1.0  1.0     110.0  264.0  0.0  ...    0.0
299  68.0  1.0  4.0     144.0  193.0  1.0  ...    0.0
300  57.0  1.0  4.0     130.0  131.0  0.0  ...    1.0
301  57.0  0.0  2.0     130.0  236.0  0.0  ...    0.0
302  38.0  1.0  3.0     138.0  175.0  0.0  ...    0.0

[303 rows x 14 columns]
      age  sex   cp  trestbps   chol  ...  oldpeak  sl
0    63.0  1.0  1.0     145.0  233.0  ...      2.3
1    67.0  1.0  4.0     160.0  286.0  ...      1.5
2    67.0  1.0  4.0     120.0  229.0  ...      2.6
3    37.0  1.0  3.0     130.0  250.0  ...      3.5
4    41.0  0.0  2.0     130.0  204.0  ...      1.4
..    ...  ...  ...       ...    ...  ...      ...
298  45.0  1.0  1.0     110.0  264.0  ...      1.2
299  68.0  1.0  4.0     144.0  193.0  ...      3.4
300  57.0  1.0  4.0     130.0  131.0  ...      1.2
301  57.0  0.0  2.0     130.0  236.0  ...      0.0
302  38.0  1.0  3.0     138.0  175.0  ...      0.0
```

DATA

All the datasets were obtained from UCL Machine learning repository https://archive.ics.uci.edu/dataset/45/heart+disease

All the datasets were obtained from UCL Machine learning repository https://archive.ics.uci.edu/dataset/45/heart+disease **Column Name**	**Description**
age	Age of the patient

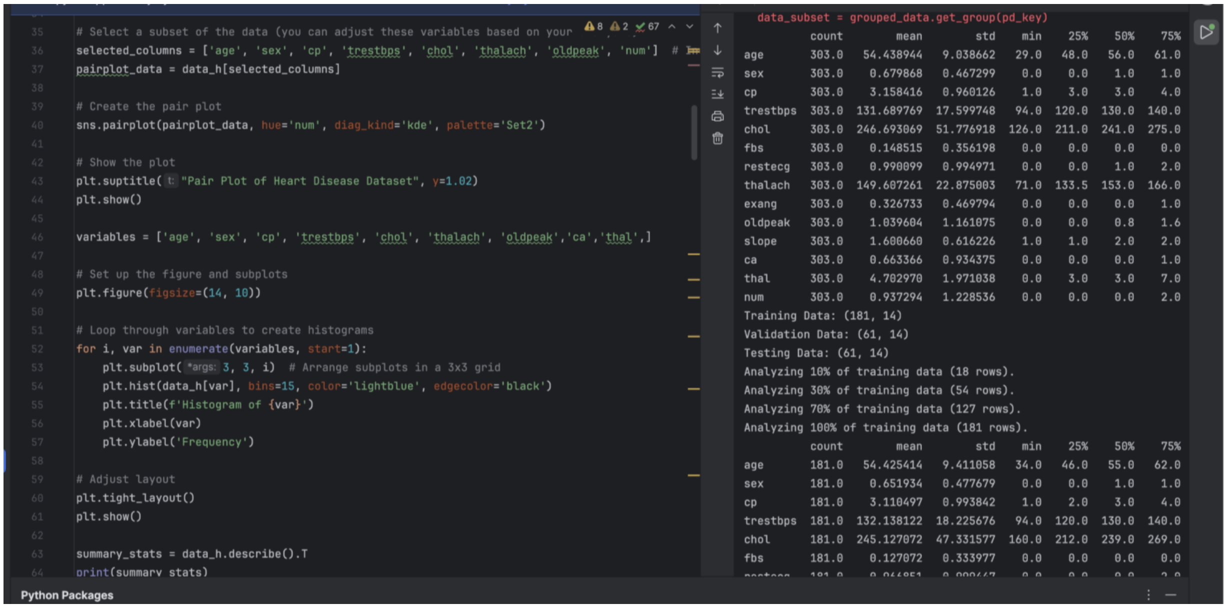
sex	Gender (1 = male; 0 = female)
cp	Chest pain type (0-3)
trestbps	Resting blood pressure (in mm Hg)
chol	Serum cholesterol in mg/dl
fbs	Fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
restecg	Resting electrocardiographic results (0-2)
thalach	Maximum heart rate achieved
exang	Exercise induced angina (1 = yes; 0 = no)
oldpeak	ST depression induced by exercise relative to rest
slope	Slope of the peak exercise ST segment (0-2)
ca	Number of major vessels (0-3) colored by fluoroscopy
thal	Thalassemia (1 = normal; 2 = fixed defect; 3 = reversable defect)
num	Diagnosis of heart disease (0 = no disease; 1-4 = presence of disease)

DATA ANALYSIS

EDA AND VARIOUS ANALYSIS STEPS

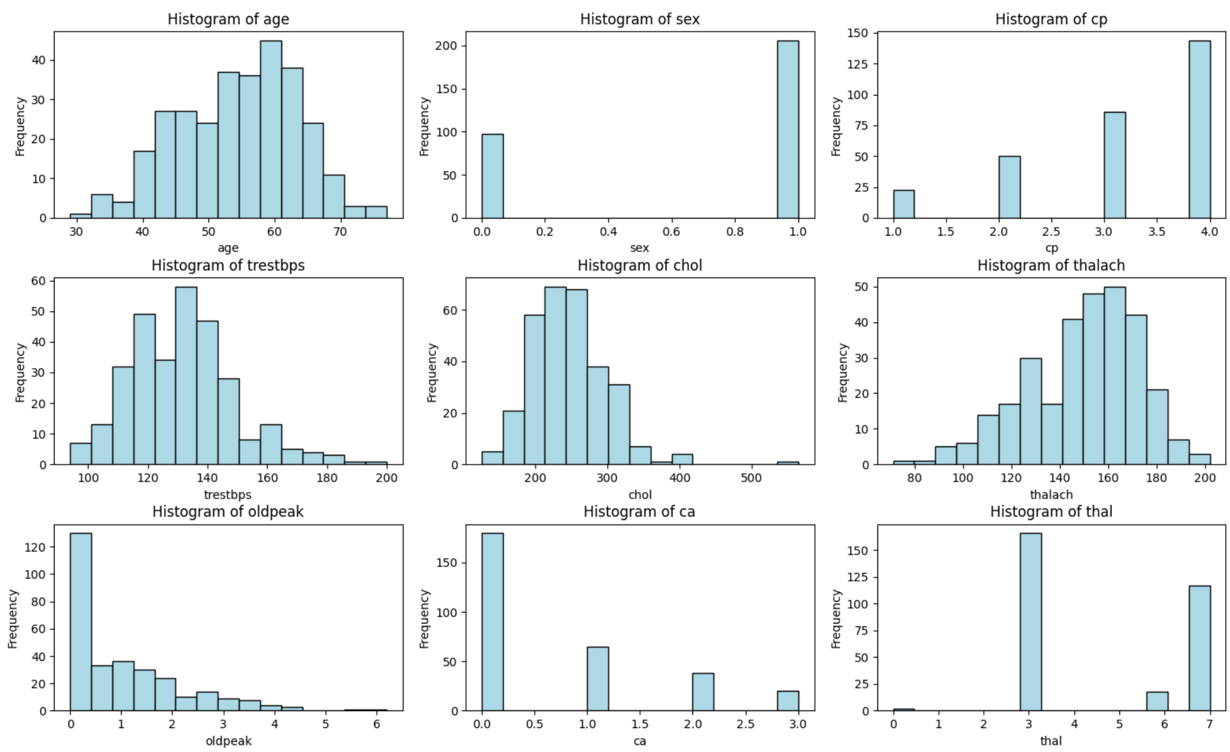
A. SUMMARY OF DIFFERENT VARIABLES AND STATISTICS FINDINGS

Feature	Count	Mean	Std	Min	25%	50%	75%	Max
age	303	54.44	9.04	29.0	48.0	56.0	61.0	77.0
sex	303	0.68	0.47	0.0	0.0	1.0	1.0	1.0
cp	303	3.16	0.96	1.0	3.0	3.0	4.0	4.0
trestbps	303	131.69	17.60	94.0	120.0	130.0	140.0	200.0
chol	303	246.69	51.78	126.0	211.0	241.0	275.0	564.0
fbs	303	0.15	0.36	0.0	0.0	0.0	0.0	1.0
restecg	303	0.99	0.99	0.0	0.0	1.0	2.0	2.0
thalach	303	149.61	22.88	71.0	133.5	153.0	166.0	202.0
exang	303	0.33	0.47	0.0	0.0	0.0	1.0	1.0
oldpeak	303	1.04	1.16	0.0	0.0	0.8	1.6	6.2
slope	303	1.60	0.62	1.0	1.0	2.0	2.0	3.0
ca	303	0.66	0.93	0.0	0.0	0.0	1.0	3.0
thal	303	4.70	1.97	0.0	3.0	3.0	7.0	7.0
num	303	0.94	1.23	0.0	0.0	0.0	2.0	4.0



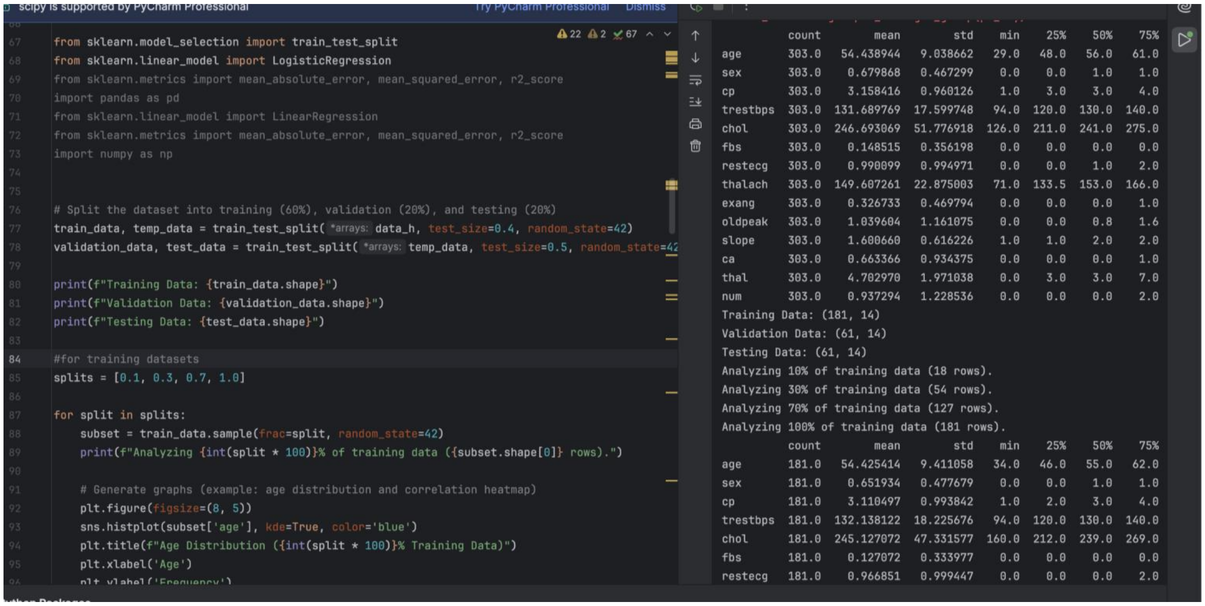
```python
# Select a subset of the data (you can adjust these variables based on your
selected_columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'thalach', 'oldpeak', 'num']  #
pairplot_data = data_h[selected_columns]

# Create the pair plot
sns.pairplot(pairplot_data, hue='num', diag_kind='kde', palette='Set2')

# Show the plot
plt.suptitle("Pair Plot of Heart Disease Dataset", y=1.02)
plt.show()

variables = ['age', 'sex', 'cp', 'trestbps', 'chol', 'thalach', 'oldpeak','ca','thal',]

# Set up the figure and subplots
plt.figure(figsize=(14, 10))

# Loop through variables to create histograms
for i, var in enumerate(variables, start=1):
    plt.subplot(3, 3, i)  # Arrange subplots in a 3x3 grid
    plt.hist(data_h[var], bins=15, color='lightblue', edgecolor='black')
    plt.title(f'Histogram of {var}')
    plt.xlabel(var)
    plt.ylabel('Frequency')

# Adjust layout
plt.tight_layout()
plt.show()

summary_stats = data_h.describe().T
print(summary_stats)
```

```
  data_subset = grouped_data.get_group(pd_key)
          count        mean        std    min    25%    50%    75%
age       303.0   54.438944   9.038662   29.0   48.0   56.0   61.0
sex       303.0    0.679868   0.467299    0.0    0.0    1.0    1.0
cp        303.0    3.158416   0.960126    1.0    3.0    3.0    4.0
trestbps  303.0  131.689769  17.599748   94.0  120.0  130.0  140.0
chol      303.0  246.693069  51.776918  126.0  211.0  241.0  275.0
fbs       303.0    0.148515   0.356198    0.0    0.0    0.0    0.0
restecg   303.0    0.990099   0.994971    0.0    0.0    1.0    2.0
thalach   303.0  149.607261  22.875003   71.0  133.5  153.0  166.0
exang     303.0    0.326733   0.469794    0.0    0.0    0.0    1.0
oldpeak   303.0    1.039604   1.161075    0.0    0.0    0.8    1.6
slope     303.0    1.600660   0.616226    1.0    1.0    2.0    2.0
ca        303.0    0.663366   0.934375    0.0    0.0    0.0    1.0
thal      303.0    4.702970   1.971038    0.0    3.0    3.0    7.0
num       303.0    0.937294   1.228536    0.0    0.0    0.0    2.0
Training Data: (181, 14)
Validation Data: (61, 14)
Testing Data: (61, 14)
Analyzing 10% of training data (18 rows).
Analyzing 30% of training data (54 rows).
Analyzing 70% of training data (127 rows).
Analyzing 100% of training data (181 rows).
          count        mean        std    min    25%    50%    75%
age       181.0   54.425414   9.411058   34.0   46.0   55.0   62.0
sex       181.0    0.651934   0.477679    0.0    0.0    1.0    1.0
cp        181.0    3.110497   0.993842    1.0    2.0    3.0    4.0
trestbps  181.0  132.138122  18.225676   94.0  120.0  130.0  140.0
chol      181.0  245.127072  47.331577  160.0  212.0  239.0  269.0
fbs       181.0    0.127072   0.333977    0.0    0.0    0.0    0.0
```

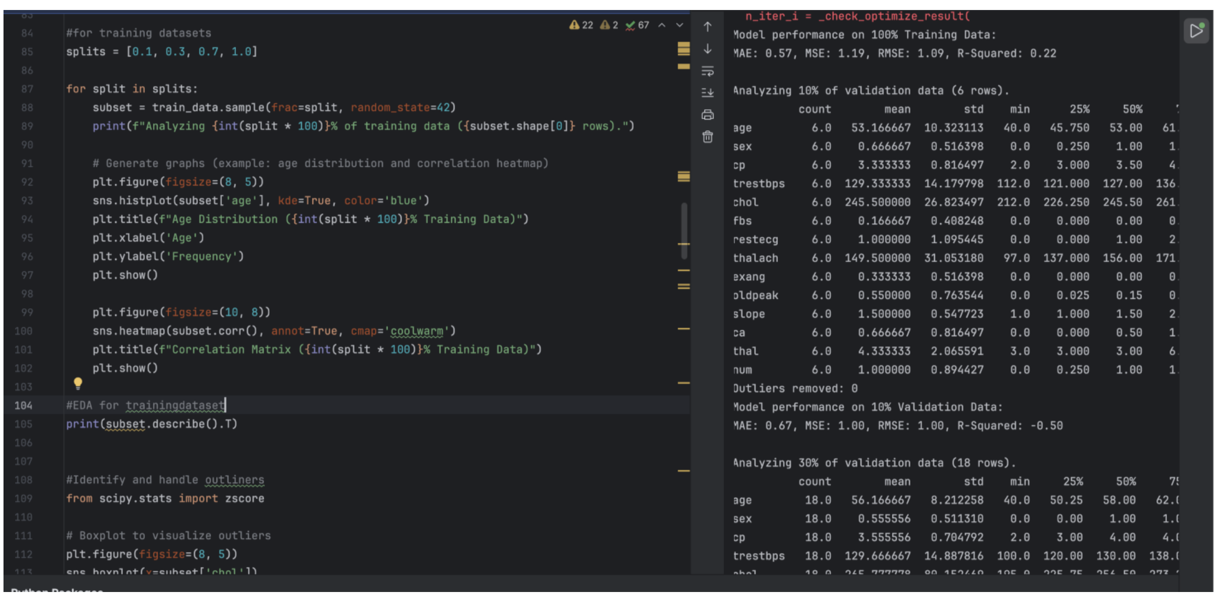



Model training and evaluation

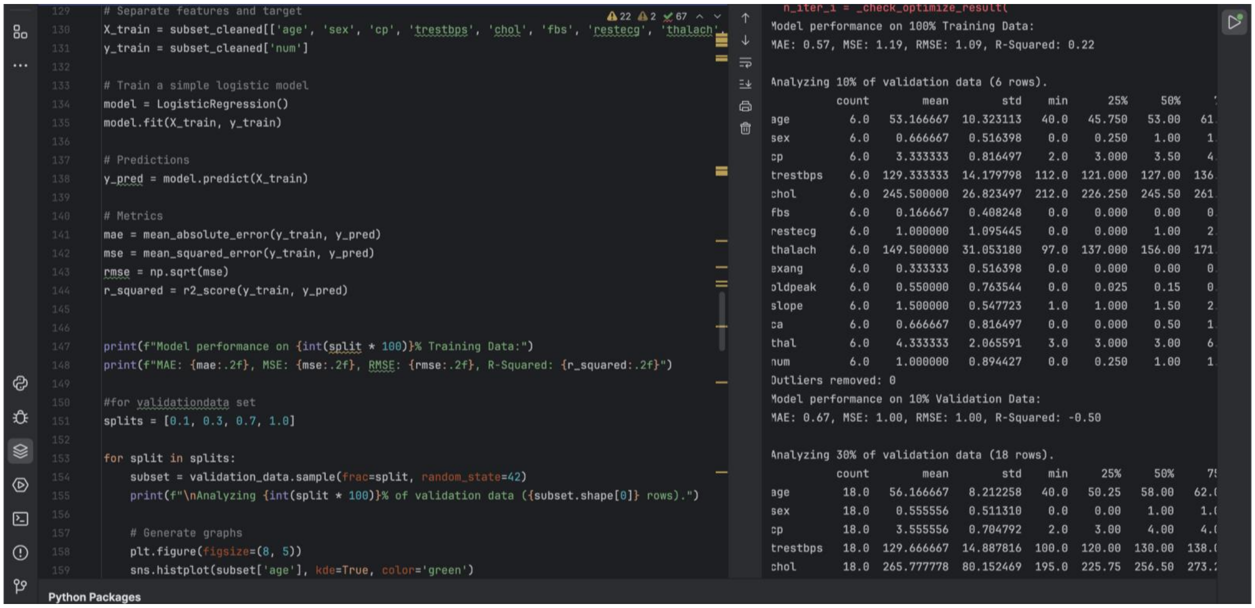
A. Splitting the data into training (60%), validation (20%), testing (20%)



B .For training dataset



C. For validation dataset



D. For testing dataset

```python
#FOR testing dataset

splits = [0.1, 0.3, 0.7, 1.0]

for split in splits:
    subset = test_data.sample(frac=split, random_state=42)
    print(f"\nAnalyzing {int(split * 100)}% of testing data ({subset.shape[0]} rows).")

    # Generate graphs
    plt.figure(figsize=(8, 5))
    sns.histplot(subset['age'], kde=True, color='orange')
    plt.title(f"Age Distribution ({int(split * 100)}% Testing Data)")
    plt.xlabel('Age')
    plt.ylabel('Frequency')
    plt.show()

    plt.figure(figsize=(10, 8))
    sns.heatmap(subset.corr(), annot=True, cmap='coolwarm')
    plt.title(f"Correlation Matrix ({int(split * 100)}% Testing Data)")
    plt.show()

    # Compare data characteristics
    print(subset.describe().T)

    # Outlier detection and handling
    subset_cleaned = subset[(zscore(subset[['chol', 'trestbps']]) < 3).all(axis=1)]
    print(f"Outliers removed: {subset.shape[0] - subset_cleaned.shape[0]}")

    # Model performance
    X_test = subset_cleaned[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalac
```

```
Model performance on 100% Training Data:
MAE: 0.57, MSE: 1.19, RMSE: 1.09, R-Squared: 0.22

Analyzing 10% of validation data (6 rows).
          count        mean        std    min      25%     50%    7
age         6.0   53.166667  10.323113   40.0   45.750   53.00   61.
sex         6.0    0.666667   0.516398    0.0    0.250    1.00    1.
cp          6.0    3.333333   0.816497    2.0    3.000    3.50    4.
trestbps    6.0  129.333333  14.179798  112.0  121.000  127.00  136.
chol        6.0  245.500000  26.823497  212.0  226.250  245.50  261.
fbs         6.0    0.166667   0.408248    0.0    0.000    0.00    0.
restecg     6.0    1.000000   1.095445    0.0    0.000    1.00    2.
thalach     6.0  149.500000  31.053180   97.0  137.000  156.00  171.
exang       6.0    0.333333   0.516398    0.0    0.000    0.00    0.
oldpeak     6.0    0.550000   0.763544    0.0    0.025    0.15    0.
slope       6.0    1.500000   0.547723    1.0    1.000    1.50    2.
ca          6.0    0.666667   0.816497    0.0    0.000    0.50    1.
thal        6.0    4.333333   2.065591    3.0    3.000    3.00    6.
num         6.0    1.000000   0.894427    0.0    0.250    1.00    1.
Outliers removed: 0
Model performance on 10% Validation Data:
MAE: 0.67, MSE: 1.00, RMSE: 1.00, R-Squared: -0.50

Analyzing 30% of validation data (18 rows).
          count        mean        std    min     25%     50%     75
age        18.0   56.166667   8.212258   40.0   50.25   58.00   62.0
sex        18.0    0.555556   0.511310    0.0    0.00    1.00    1.0
cp         18.0    3.555556   0.704792    2.0    3.00    4.00    4.0
trestbps   18.0  129.666667  14.887816  100.0  120.00  130.00  138.0
chol       18.0  265.777778  80.152469  195.0  225.75  256.50  273.2
```

E. Removing outliners

```python
    sns.heatmap(subset.corr(), annot=True, cmap='coolwarm')
    plt.title(f"Correlation Matrix ({int(split * 100)}% Training Data)")
    plt.show()

#EDA for trainingdataset
print(subset.describe()


#Identify and handle outliners
from scipy.stats import zscore

# Boxplot to visualize outliers
plt.figure(figsize=(8, 5))
sns.boxplot(x=subset['chol'])
plt.title('Cholesterol Levels visualizing outliners (Boxplot)')
plt.show()

# Remove outliers based on z-scores
subset_cleaned = subset[(zscore(subset[['chol', 'trestbps']]) < 3).all(axis=1)]
print(f"Outliers removed: {subset.shape[0] - subset_cleaned.shape[0]}")
plt.figure(figsize=(8, 5))
sns.boxplot(x=subset['chol'])
plt.title('Cholestrol level after outliner removal')
plt.show()
#model performance
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

# Separate features and target
X_train = subset_cleaned[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach'
```

```
 [ 4 14]]
Accuracy: 0.86
F1 Score: 0.82

Analyzing 100% of testing data (61 rows).
          count        mean        std    min    25%    50%    75%
age        61.0   53.836066   8.266761   34.0   50.0   56.0   59.0
sex        61.0    0.786885   0.412907    0.0    1.0    1.0    1.0
cp         61.0    3.098361   0.960988    1.0    3.0    3.0    4.0
trestbps   61.0  133.459016  17.517205  105.0  120.0  130.0  140.0
chol       61.0  245.409836  53.309279  126.0  211.0  239.0  282.0
fbs        61.0    0.163934   0.373288    0.0    0.0    0.0    0.0
restecg    61.0    1.131148   0.974259    0.0    0.0    2.0    2.0
thalach    61.0  150.016393  22.148357   95.0  140.0  155.0  165.0
exang      61.0    0.278689   0.452075    0.0    0.0    0.0    1.0
oldpeak    61.0    1.277049   1.208497    0.0    0.2    0.8    2.0
slope      61.0    1.655738   0.655369    1.0    1.0    2.0    2.0
ca         61.0    0.688525   0.992320    0.0    0.0    0.0    1.0
thal       61.0    4.918033   1.977331    3.0    3.0    3.0    7.0
num        61.0    0.868852   1.203819    0.0    0.0    0.0    1.0
Outliers removed: 0
Model performance on 100% Testing Data:
MAE: 0.61, MSE: 1.16, RMSE: 1.08, R-Squared: 0.18
Confusion Matrix:
[[31  3]
 [ 9 18]]
Accuracy: 0.80
F1 Score: 0.75

Process finished with exit code 0
```

PROGRAMMING CODE

```
import urllib3
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import seaborn as sns
import matplotlib.pyplot as plt
from numpy.distutils.conv_template import header
from numpy.distutils.conv_template import header
urllib3.disable_warnings()
import requests
import pandas as pd
from io import StringIO
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
import statsmodels.api as sm
from sklearn.model_selection import train_test_split

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data"  # Update with actual dataset URL
response = requests.get(url, verify=False)  # `verify=False` skips SSL verification
data = StringIO(response.text)
column_names = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
            'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'num']

heart_disease_data = pd.read_csv(data,header=None,names=column_names)  # Update header if needed
print(heart_disease_data)

#replacing'?' with 0 so data wont be lost.
data_h=heart_disease_data.replace('?', 0).astype(float)

print(data_h)




# Select a subset of the data (you can adjust these variables based on your analysis)
selected_columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'thalach', 'oldpeak', 'num']  # Include 'num' for hue
pairplot_data = data_h[selected_columns]

# Create the pair plot
sns.pairplot(pairplot_data, hue='num', diag_kind='kde', palette='Set2')

# Show the plot
plt.suptitle("Pair Plot of Heart Disease Dataset", y=1.02)
plt.show()
```

```
variables = ['age', 'sex', 'cp', 'trestbps', 'chol', 'thalach', 'oldpeak','ca','thal',]

# Set up the figure and subplots
plt.figure(figsize=(14, 10))

# Loop through variables to create histograms
for i, var in enumerate(variables, start=1):
    plt.subplot(3, 3, i)  # Arrange subplots in a 3x3 grid
    plt.hist(data_h[var], bins=15, color='lightblue', edgecolor='black')
    plt.title(f'Histogram of {var}')
    plt.xlabel(var)
    plt.ylabel('Frequency')

# Adjust layout
plt.tight_layout()
plt.show()

summary_stats = data_h.describe().T
print(summary_stats)



from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np



# Split the dataset into training (60%), validation (20%), and testing (20%)
train_data, temp_data = train_test_split(data_h, test_size=0.4, random_state=42)
validation_data, test_data = train_test_split(temp_data, test_size=0.5, random_state=42)

print(f"Training Data: {train_data.shape}")
print(f"Validation Data: {validation_data.shape}")
print(f"Testing Data: {test_data.shape}")

#for training datasets
splits = [0.1, 0.3, 0.7, 1.0]

for split in splits:
    subset = train_data.sample(frac=split, random_state=42)
    print(f"Analyzing {int(split * 100)}% of training data ({subset.shape[0]} rows).")
```

```
    # Generate graphs (example: age distribution and correlation heatmap)
    plt.figure(figsize=(8, 5))
    sns.histplot(subset['age'], kde=True, color='blue')
    plt.title(f"Age Distribution ({int(split * 100)}% Training Data)")
    plt.xlabel('Age')
    plt.ylabel('Frequency')
    plt.show()

    plt.figure(figsize=(10, 8))
    sns.heatmap(subset.corr(), annot=True, cmap='coolwarm')
    plt.title(f"Correlation Matrix ({int(split * 100)}% Training Data)")
    plt.show()

#EDA for trainingdataset
print(subset.describe().T)



#Identify and handle outliners
from scipy.stats import zscore

# Boxplot to visualize outliers
plt.figure(figsize=(8, 5))
sns.boxplot(x=subset['chol'])
plt.title('Cholesterol Levels visualizing outliners (Boxplot)')
plt.show()

# Remove outliers based on z-scores
subset_cleaned = subset[(zscore(subset[['chol', 'trestbps']]) < 3).all(axis=1)]
print(f"Outliers removed: {subset.shape[0] - subset_cleaned.shape[0]}")
plt.figure(figsize=(8, 5))
sns.boxplot(x=subset['chol'])
plt.title('Cholestrol level after outliner removal')
plt.show()
#model performance
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

# Separate features and target
X_train = subset_cleaned[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca',
'thal']]
y_train = subset_cleaned['num']

# Train a simple logistic model
model = LogisticRegression()
```

```python
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_train)

# Metrics
mae = mean_absolute_error(y_train, y_pred)
mse = mean_squared_error(y_train, y_pred)
rmse = np.sqrt(mse)
r_squared = r2_score(y_train, y_pred)



print(f"Model performance on {int(split * 100)}% Training Data:")
print(f"MAE: {mae:.2f}, MSE: {mse:.2f}, RMSE: {rmse:.2f}, R-Squared: {r_squared:.2f}")

#for validationdata set
splits = [0.1, 0.3, 0.7, 1.0]

for split in splits:
    subset = validation_data.sample(frac=split, random_state=42)
    print(f"\nAnalyzing {int(split * 100)}% of validation data ({subset.shape[0]} rows).")

    # Generate graphs
    plt.figure(figsize=(8, 5))
    sns.histplot(subset['age'], kde=True, color='green')
    plt.title(f"Age Distribution ({int(split * 100)}% Validation Data)")
    plt.xlabel('Age')
    plt.ylabel('Frequency')
    plt.show()

    plt.figure(figsize=(10, 8))
    sns.heatmap(subset.corr(), annot=True, cmap='coolwarm')
    plt.title(f"Correlation Matrix ({int(split * 100)}% Validation Data)")
    plt.show()

    # Compare data characteristics
    print(subset.describe().T)

    # Outlier detection and handling
    subset_cleaned = subset[(zscore(subset[['chol', 'trestbps']]) < 3).all(axis=1)]
    print(f"Outliers removed: {subset.shape[0] - subset_cleaned.shape[0]}")

    # Model performance
    X_val = subset_cleaned[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca',
'thal']]
```

```
    y_val = subset_cleaned['num']
    y_val_pred = model.predict(X_val)

    # Metrics
    mae = mean_absolute_error(y_val, y_val_pred)
    mse = mean_squared_error(y_val, y_val_pred)
    rmse = np.sqrt(mse)
    r_squared = r2_score(y_val, y_val_pred)



    print(f"Model performance on {int(split * 100)}% Validation Data:")
    print(f"MAE: {mae:.2f}, MSE: {mse:.2f}, RMSE: {rmse:.2f}, R-Squared: {r_squared:.2f}")



#FOR testing dataset

splits = [0.1, 0.3, 0.7, 1.0]

for split in splits:
    subset = test_data.sample(frac=split, random_state=42)
    print(f"\nAnalyzing {int(split * 100)}% of testing data ({subset.shape[0]} rows).")

    # Generate graphs
    plt.figure(figsize=(8, 5))
    sns.histplot(subset['age'], kde=True, color='orange')
    plt.title(f"Age Distribution ({int(split * 100)}% Testing Data)")
    plt.xlabel('Age')
    plt.ylabel('Frequency')
    plt.show()

    plt.figure(figsize=(10, 8))
    sns.heatmap(subset.corr(), annot=True, cmap='coolwarm')
    plt.title(f"Correlation Matrix ({int(split * 100)}% Testing Data)")
    plt.show()

    # Compare data characteristics
    print(subset.describe().T)

    # Outlier detection and handling
    subset_cleaned = subset[(zscore(subset[['chol', 'trestbps']]) < 3).all(axis=1)]
    print(f"Outliers removed: {subset.shape[0] - subset_cleaned.shape[0]}")

    # Model performance
    X_test = subset_cleaned[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope',
'ca', 'thal']]
```

```python
y_test = subset_cleaned['num']
y_test_pred = model.predict(X_test)

# Metrics
mae = mean_absolute_error(y_test, y_test_pred)
mse = mean_squared_error(y_test, y_test_pred)
rmse = np.sqrt(mse)
r_squared = r2_score(y_test, y_test_pred)



print(f"Model performance on {int(split * 100)}% Testing Data:")
print(f"MAE: {mae:.2f}, MSE: {mse:.2f}, RMSE: {rmse:.2f}, R-Squared: {r_squared:.2f}")

threshold = 0.5  # Adjust based on your problem domain
y_test_class = (y_test > threshold).astype(int)  # Ground truth class
y_pred_class = (y_test_pred > threshold).astype(int)  # Predicted class

from sklearn.metrics import confusion_matrix, accuracy_score, f1_score

# Confusion Matrix
conf_matrix = confusion_matrix(y_test_class, y_pred_class)
print("Confusion Matrix:")
print(conf_matrix)

# Accuracy Score
accuracy = accuracy_score(y_test_class, y_pred_class)
print(f"Accuracy: {accuracy:.2f}")

# F1 Score
f1 = f1_score(y_test_class, y_pred_class)
print(f"F1 Score: {f1:.2f}")
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=['Class 0', 'Class 1'],
        yticklabels=['Class 0', 'Class 1'])

# Add labels, title, and axis ticks
plt.title("Confusion Matrix")
plt.xlabel("Predicted Labels")
plt.ylabel("Actual Labels")
plt.show()
```