Assn2PartA-GrpNo43

October 6, 2023

0.1 # Pattern Recognition And Machine Learning

0.2 Assignment 2 - Part A

Group No: 43 Group Members: - Ajul Thomas (u3253992) - Hamad Rasheed (u3224704)

0.3 # Heart Disease Prediction using Pattern Recognition and Machine Learning

0.4 Data set selection and description of dataset and features.

0.5 Description

Our project focuses on developing a Predictive Risk Model for Heart Disease (PRMHD) using Pattern Recognition and Machine Learning (PRML). The model aims to analyze a range of medical and lifestyle factors to predict the likelihood of an individual developing heart disease.

0.6 Motivation

Heart disease is a leading cause of death worldwide. Early diagnosis can lead to effective treatment, but traditional diagnostic methods are often slow and expensive. A computational model can provide quick, accurate, and cost-effective risk assessment.

0.7 Dataset

The Cleveland Heart Disease dataset from the UCI Machine Learning Repository was selected for this project. Although the database contains a total of 76 attributes, our focus is on a subset of 14 key attributes, as these are the ones most commonly cited in published research. Specifically, we are utilizing only the Cleveland database for this endeavor. The initial steps involve data cleaning to address any missing values, followed by Exploratory Data Analysis (EDA) to gain insights into dataset.

0.8 Dataset Description:

1) **age** - age of the individual in years
2) **sex** - 1 = Male , 2 = Female
3) **cp** - chest pain type
    - 1 - typical angina
    - 2 - atypical angina
    - 3 - non-anginal pain
    - 4 - asymptomatic

4) **trtbps** - resting blood pressure (in mm Hg on admission to the hospital)
5) **chol** - serum cholesterols in mg/dl
6) **fbs** - fasting blood sugar > 120 mg/dl
    - 1 - true
    - 0 - false
7) **restecg** - resting electrocardiographic results
    - 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 hypertrophy by Estes' criteria
8) **thalach** - maximum heart rate achieved
9) **exng**: exercise induced angina
    - 1 - Yes
    - 0 - No
10) **oldpeak** - ST depression induced by exercise relative to rest
11) **slp** - the slope of the peak exercise ST segmen
    - 1 - Upsloping
    - 2 - Flat
    - 3 - Downsloping
12) **ca** - number of major vessels (0–3) colored by fluoroscopy
13) **thall** - 3 = normal; 6 = fixed defect; 7 = reversible defect

0.8.1 Import required libraries

[1]:
```
import pandas as pd
import numpy as np
```

0.8.2 Import the Heart Disease Dataset

[2]:
```
data = pd.read_csv('./data/heart.csv')
# data = pd.read_csv('./data/processed.cleveland.data')
data.head()
```

[2]:
```
   age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak  slp  ca  \
0   63    1   3     145   233    1        0       150     0      2.3    0   0
1   37    1   2     130   250    0        1       187     0      3.5    0   0
2   41    0   1     130   204    0        0       172     0      1.4    2   0
3   56    1   1     120   236    0        1       178     0      0.8    2   0
4   57    0   0     120   354    0        1       163     1      0.6    2   0

   thall  output
0      1       1
1      2       1
2      2       1
3      2       1
4      2       1
```

0.9 ## Exploratory Data Analysis

EDA involves understanding the dataset's structure and basic statistics.

[3]:
```
data.shape
```

[3]:
```
(303, 14)
```

[4]:
```
data.columns
```

[4]:
```
Index(['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg', 'thalachh',
       'exng', 'oldpeak', 'slp', 'ca', 'thall', 'output'],
      dtype='object')
```

[5]:
```
data.dtypes
```

[5]:
```
age           int64
sex           int64
cp            int64
trtbps        int64
chol          int64
fbs           int64
restecg       int64
thalachh      int64
exng          int64
oldpeak     float64
slp           int64
ca            int64
thall         int64
output        int64
dtype: object
```

[6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trtbps    303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalachh  303 non-null    int64
 8   exng      303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64
 11  ca        303 non-null    int64
 12  thall     303 non-null    int64
 13  output    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

[7]:
```
data.describe().T
```

[7]:
```
          count        mean        std    min    25%    50%    75%    max
age       303.0   54.366337   9.082101   29.0   47.5   55.0   61.0   77.0
sex       303.0    0.683168   0.466011    0.0    0.0    1.0    1.0    1.0
cp        303.0    0.966997   1.032052    0.0    0.0    1.0    2.0    3.0
trtbps    303.0  131.623762  17.538143   94.0  120.0  130.0  140.0  200.0
chol      303.0  246.264026  51.830751  126.0  211.0  240.0  274.5  564.0
fbs       303.0    0.148515   0.356198    0.0    0.0    0.0    0.0    1.0
restecg   303.0    0.528053   0.525860    0.0    0.0    1.0    1.0    2.0
thalachh  303.0  149.646865  22.905161   71.0  133.5  153.0  166.0  202.0
exng      303.0    0.326733   0.469794    0.0    0.0    0.0    1.0    1.0
oldpeak   303.0    1.039604   1.161075    0.0    0.0    0.8    1.6    6.2
slp       303.0    1.399340   0.616226    0.0    1.0    1.0    2.0    2.0
ca        303.0    0.729373   1.022606    0.0    0.0    0.0    1.0    4.0
thall     303.0    2.313531   0.612277    0.0    2.0    2.0    3.0    3.0
output    303.0    0.544554   0.498835    0.0    0.0    1.0    1.0    1.0
```

0.10 ## Data Cleaning

Checks and addresses if any misssing values are there. If any will address these issues.

[8]:
```
data.nunique()
```

[8]:
```
age          41
sex           2
cp            4
trtbps       49
chol        152
fbs           2
restecg       3
thalachh     91
exng          2
oldpeak      40
slp           3
ca            5
thall         4
output        2
dtype: int64
```

[9]:
```
data['ca'].unique()
```

[9]:
```
array([0, 2, 1, 3, 4], dtype=int64)
```

[10]:
```
data.ca.value_counts()
```

[10]:
```
ca
0    175
1     65
2     38
3     20
4      5
Name: count, dtype: int64
```

[11]:
```
data.duplicated().sum()
```

[11]:
```
1
```

[12]:
```
data.isnull().sum()
```

[12]:
```
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
ca          0
thall       0
output      0
dtype: int64
```

0.11 ## Data Visualization

Create visualizations to gain insights into the data distribution, relationships, and patterns. Uses libraries like Matplotlib and Seaborn.

[13]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

changing the data for better visualization and plotting

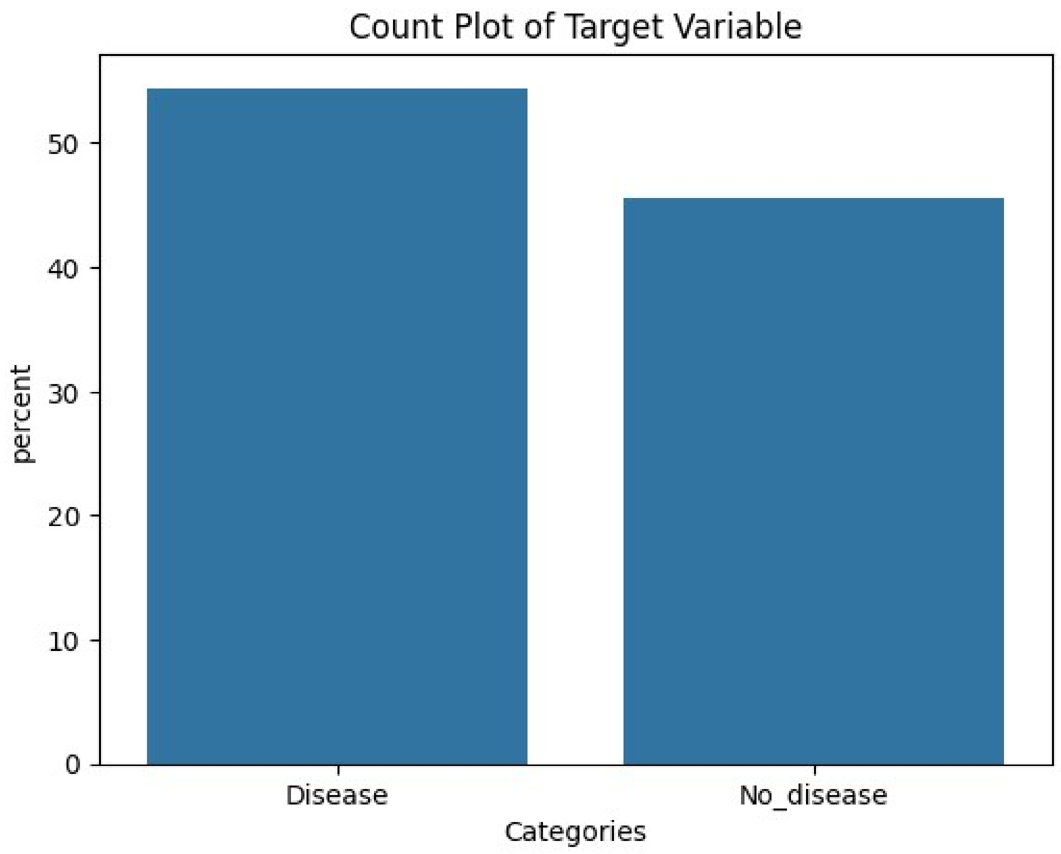
[14]:
```
df = data.copy(deep=True)
df['output'] = df.output.replace({1: "Disease", 0: "No_disease"})
df['sex'] = df.sex.replace({1: "Male", 0: "Female"})
df['cp'] = df.cp.replace({0: "typical_angina",
                          1: "atypical_angina",
                          2:"non-anginal pain",
                          3: "asymtomatic"})
df['exng'] = df.exng.replace({1: "Yes", 0: "No"})
df['fbs'] = df.fbs.replace({1: "True", 0: "False"})
df['slp'] = df.slp.replace({0: "upsloping", 1: "flat",2:"downsloping"})
df['thall'] = df.thall.replace({1: "fixed_defect", 2: "reversable_defect", 3:
 ↪"normal"})
```

[15]:
```
df
```

[15]:
```
     age     sex                cp  trtbps  chol    fbs  restecg  thalachh  \
0     63    Male       asymtomatic     145   233   True        0       150
1     37    Male  non-anginal pain     130   250  False        1       187
2     41  Female   atypical_angina     130   204  False        0       172
3     56    Male   atypical_angina     120   236  False        1       178
4     57  Female    typical_angina     120   354  False        1       163
..   ...     ...               ...     ...   ...    ...      ...       ...
298   57  Female    typical_angina     140   241  False        1       123
299   45    Male       asymtomatic     110   264  False        1       132
300   68    Male    typical_angina     144   193   True        1       141
301   57    Male    typical_angina     130   131  False        1       115
302   57  Female   atypical_angina     130   236  False        0       174

    exng  oldpeak          slp  ca              thall      output
0     No      2.3    upsloping   0       fixed_defect     Disease
1     No      3.5    upsloping   0  reversable_defect     Disease
2     No      1.4  downsloping   0  reversable_defect     Disease
3     No      0.8  downsloping   0  reversable_defect     Disease
4    Yes      0.6  downsloping   0  reversable_defect     Disease
..   ...      ...          ...  ..                ...         ...
298  Yes      0.2         flat   0             normal  No_disease
299   No      1.2         flat   0             normal  No_disease
300   No      3.4         flat   2             normal  No_disease
301  Yes      1.2         flat   1             normal  No_disease
302   No      0.0         flat   1  reversable_defect  No_disease

[303 rows x 14 columns]
```

0.11.1 Target Variable Distribution

[16]:
```
sns.countplot(df, x='output', stat="percent")
plt.title('Count Plot of Target Variable')
plt.xlabel('Categories')
plt.show()
```



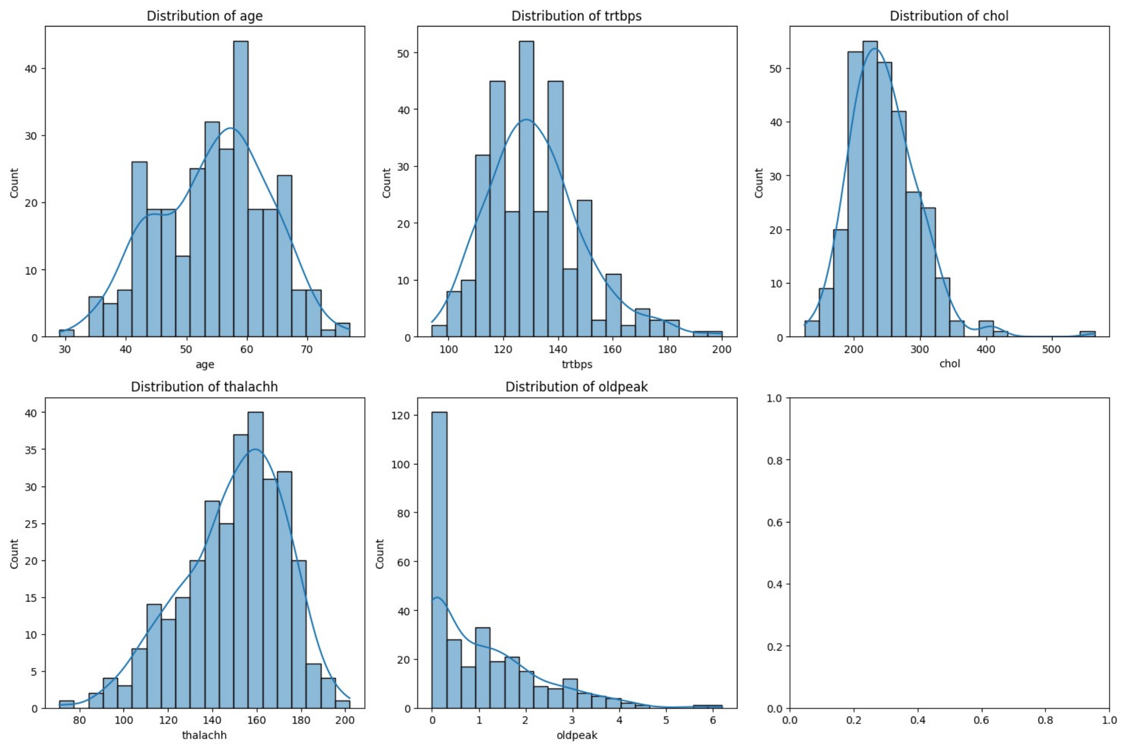
Observation:

- Our initial graph categorizes individuals based on the presence or absence of heart disease. In the dataset, just over 50% of participants have heart disease, while approximately 45% do not. This balanced distribution provides a solid foundation for more in-depth analysis.

0.11.2 Distribution of Numerical Variables

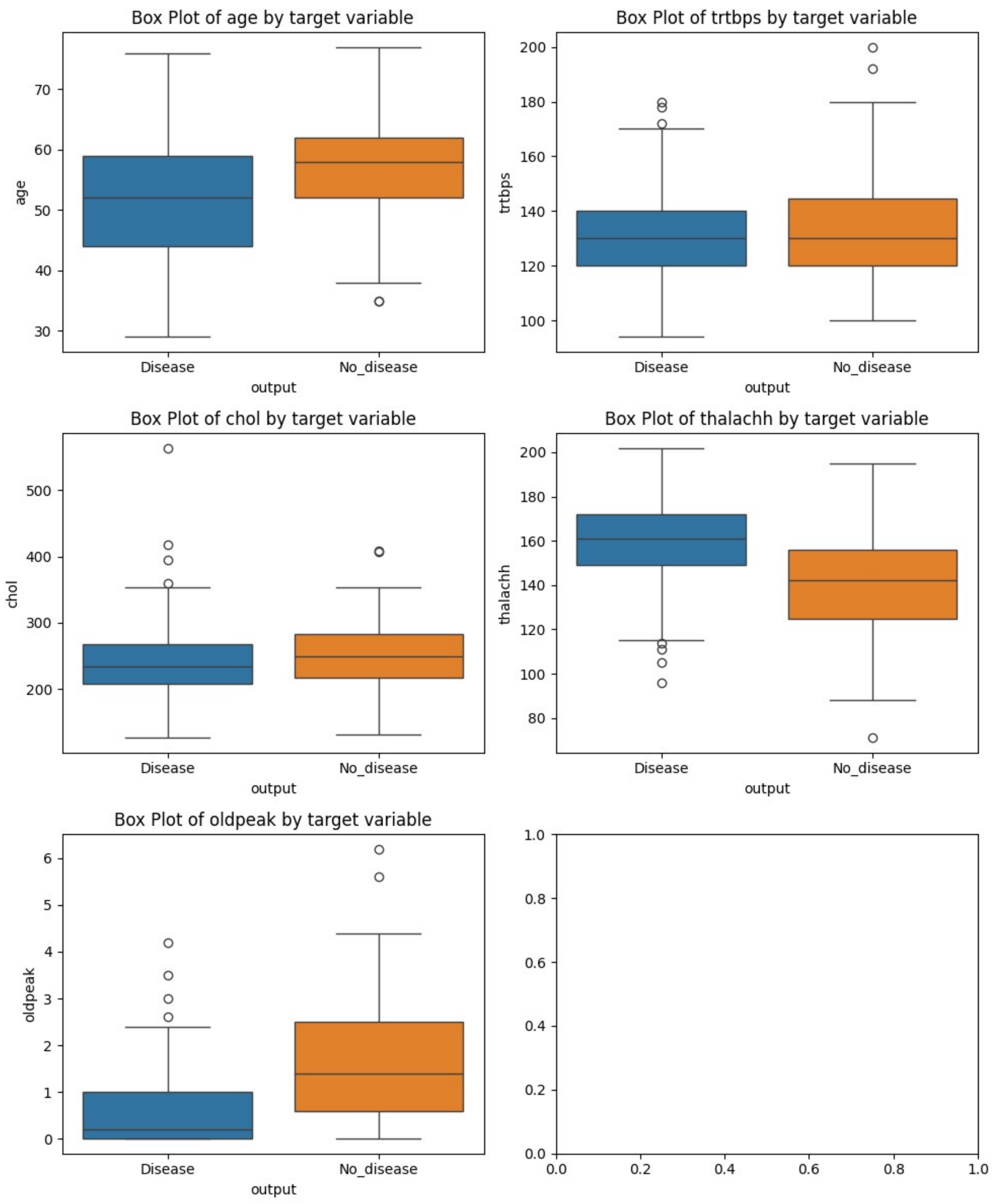
[17]:
```python
numerical_features = ['age', 'trtbps', 'chol', 'thalachh', 'oldpeak']
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
fig.subplots_adjust(hspace=0.5)
for i, feature in enumerate(numerical_features):
    row, col = i // 3, i % 3
    ax = axes[row, col]
    sns.histplot(data[feature], bins=20, kde=True, ax=ax)
    ax.set_title(f'Distribution of {feature}')
    ax.set_xlabel(feature)
    ax.set_ylabel('Count')

# Adjust layout
```

```
plt.tight_layout()
plt.show()
```



[18]:
```
# Assuming 'target' is the name of the column representing your target variable
target_variable = 'output'

fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(10, 12))
fig.subplots_adjust(hspace=0.5)

colors = ["blue", "green"]

# Loop through each attribute and plot a box plot for each target variable
for i, attribute in enumerate(numerical_features):
    row, col = i//2, i % 2
    ax = axes[row, col]
    # plt.figure(figsize=(10, 6))  # Set the figure size
    sns.boxplot(x=target_variable, y=attribute, hue=target_variable, data=df,␣
  ↪ax=ax)
    ax.set_title(f'Box Plot of {attribute} by target variable')
    ax.set_xlabel(target_variable)
    ax.set_ylabel(attribute)

# Adjust layout
```

```
plt.tight_layout()
plt.show()
```



Observations: As we examine the distributions of numerical variables such as age, resting heart rate (trtbps), cholesterol (chol), maximum heart rate achieved (thalachh), and exercise-induced ST depression (oldpeak).

The dataset reveals that the average age for those with heart disease is lower compared to those without it. Most individuals in the dataset are aged between 50 and 70, following a normal distribution.

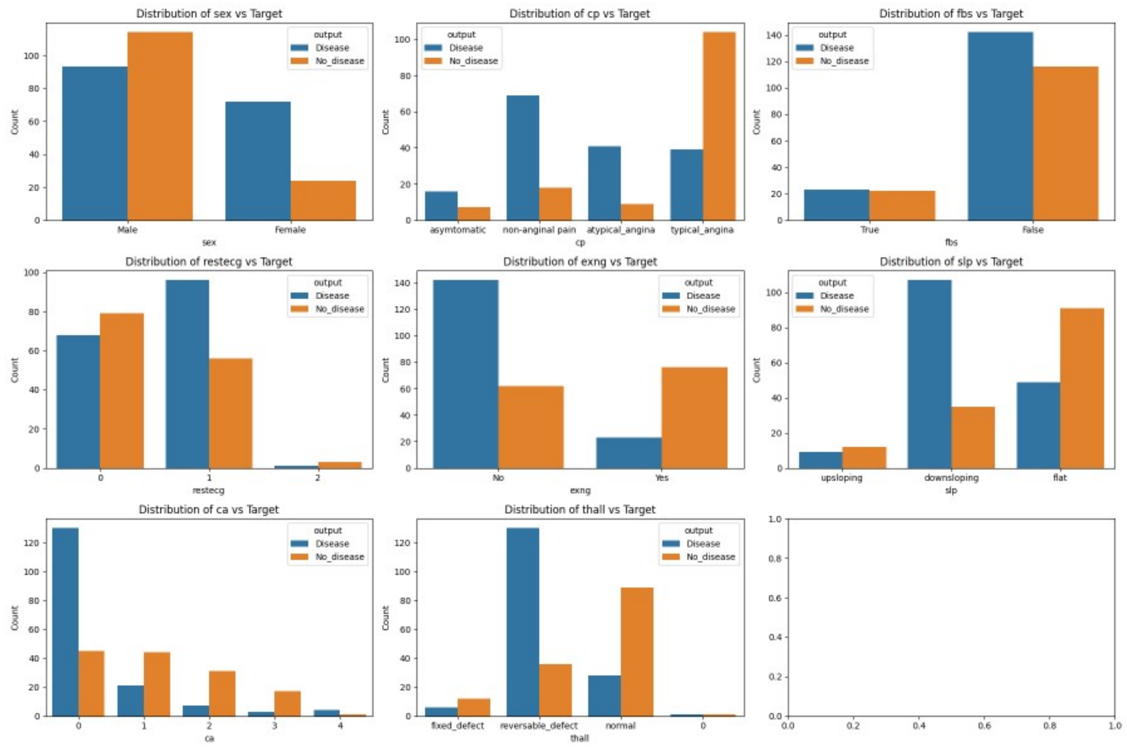
In terms of cholesterol levels, there's little variation between those with and without heart disease, although some outliers are present. The majority of individuals have cholesterol levels ranging between 200 and 300, adhering to a normal distribution.

When considering maximum heart rate achieved, people with heart disease generally have higher rates than those without. Several outliers exist, but the majority have heart rates between 150 and 175.

Lastly, the average oldpeak level for those with heart disease is lower than for those without. Despite some outliers, the distribution for oldpeak is right-skewed, with most individuals registering a value of zero.

0.11.3 Distribution of Categorical Variables

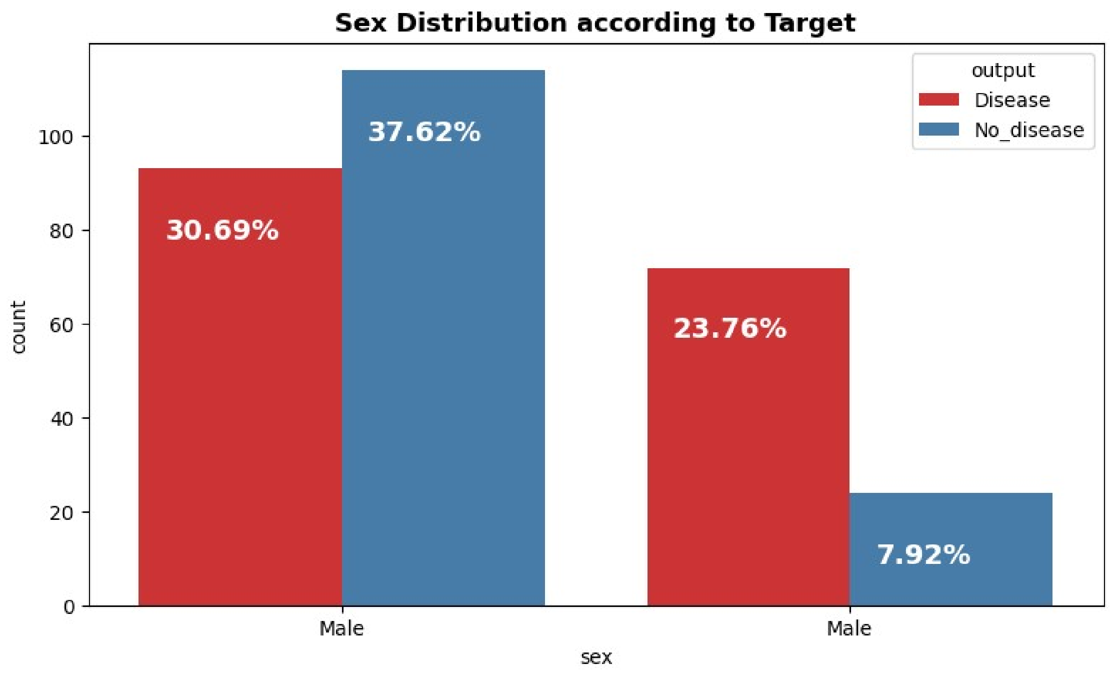
[19]:
```
# Create bar plots for categorical features
categorical_features = ['sex', 'cp', 'fbs', 'restecg', 'exng', 'slp', 'ca',␣
 ↪'thall']
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(18, 12))
fig.subplots_adjust(hspace=0.5)

for i, feature in enumerate(categorical_features):
    row, col = i // 3, i % 3
    ax = axes[row, col]
    sns.countplot(x=df[feature], hue=df['output'], ax=ax)
    ax.set_title(f'Distribution of {feature} vs Target')
    ax.set_xlabel(feature)
    ax.set_ylabel('Count')

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



0.11.4 Gender distribution according to target variable

```
[20]: fig, ax = plt.subplots(figsize=(8,5))
name = df['sex']
ax = sns.countplot(x='sex', hue='output', data=df, palette='Set1')
ax.set_title("Sex Distribution according to Target", fontsize = 13, weight =␣
 ↪'bold')
ax.set_xticklabels (name, rotation = 0)

totals = []
for i in ax.patches:
    totals.append(i.get_height())
total = sum(totals)
for i in ax.patches:
    ax.text(i.get_x()+.05, i.get_height()-15,
            str(round((i.get_height()/total)*100, 2))+'%', fontsize=14,
                color='white', weight = 'bold')

plt.tight_layout()
```

```
C:\Users\ajult\AppData\Local\Temp\ipykernel_19948\3742478429.py:5: UserWarning:
FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels (name, rotation = 0)
```



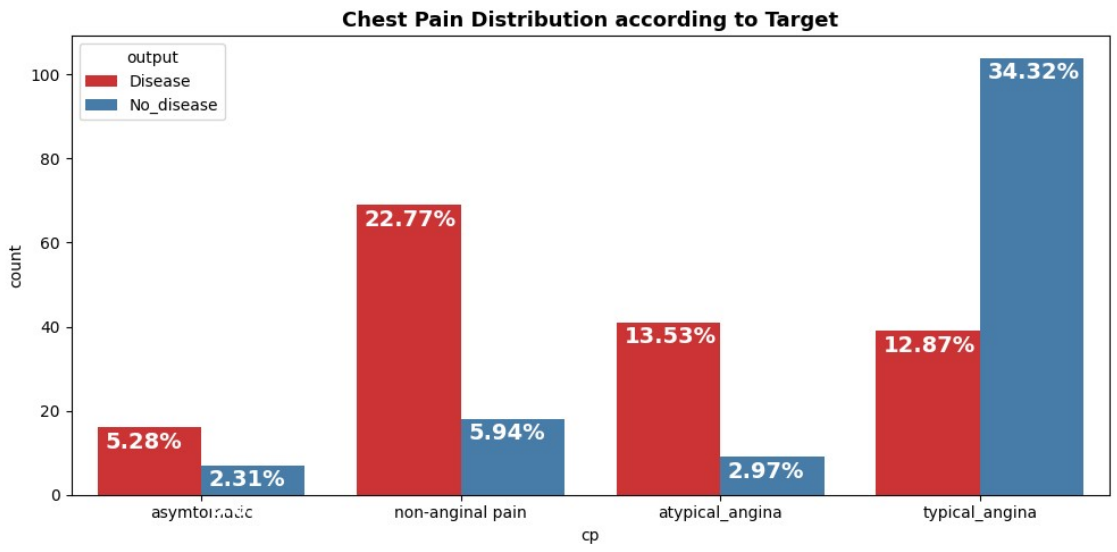
Chest pain distribution according to target variable

```
[21]: df.cp.value_counts()
```

```
[21]: cp
      typical_angina       143
      non-anginal pain      87
      atypical_angina       50
      asymtomatic           23
      Name: count, dtype: int64
```

```
[22]: fig, ax = plt.subplots(figsize=(10,5))
      name = df.cp.unique()
      ax = sns.countplot(x='cp', hue='output', data=df, palette='Set1')
      ax.set_title("Chest Pain Distribution according to Target", fontsize = 13,␣
       ↪weight = 'bold')
      ax.set_xticklabels (name, rotation = 0)

      totals = []
      for i in ax.patches:
          totals.append(i.get_height())
      total = sum(totals)
      for i in ax.patches:
          ax.text(i.get_x()+.03, i.get_height()-5,
                  str(round((i.get_height()/total)*100, 2))+'%', fontsize=14,
                color='white', weight = 'bold')
plt.tight_layout()
```

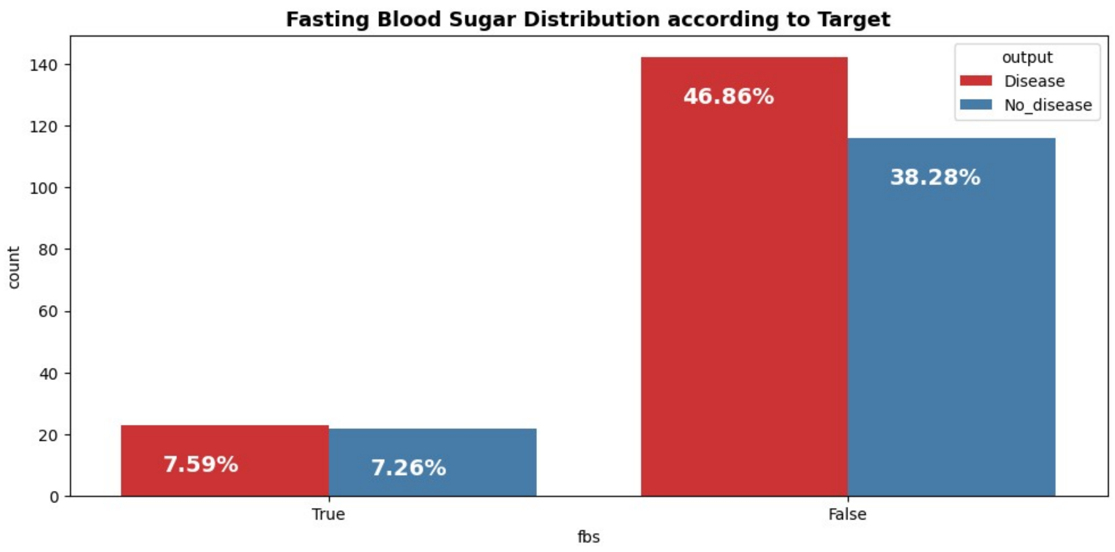
```
C:\Users\ajult\AppData\Local\Temp\ipykernel_19948\815883350.py:5: UserWarning:
FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels (name, rotation = 0)
```



Fasting blood sugar distribution according to target variable

[23]:
```
fig, ax = plt.subplots(figsize=(10,5))
name = df['fbs']
ax = sns.countplot(x='fbs', hue='output', data=df, palette='Set1')
ax.set_title("Fasting Blood Sugar Distribution according to Target", fontsize =␣
 ↪13, weight = 'bold')
ax.set_xticklabels (name, rotation = 0)

totals = []
for i in ax.patches:
    totals.append(i.get_height())
total = sum(totals)
for i in ax.patches:
    ax.text(i.get_x()+.08, i.get_height()-15,
            str(round((i.get_height()/total)*100, 2))+'%', fontsize=14,
                color='white', weight = 'bold')
plt.tight_layout()
```

```
C:\Users\ajult\AppData\Local\Temp\ipykernel_19948\2214451207.py:5: UserWarning:
FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels (name, rotation = 0)
```



Observations on distribution of Categorical Variables vs Target: Sex vs. Target: The data indicates that a higher number of women have heart disease compared to those who don't, while the opposite is true for men. Men make up 68.3% of the study population.

Chest Pain (cp) vs. Target: Among the four levels of chest pain, individuals at level 2 are more prone to heart disease. Conversely, those at level 0 are less likely to have heart disease and make up 47.2% of the dataset.

Fasting Blood Sugar (fbs) vs. Target: Individuals with an fbs under 120 are more susceptible to heart disease and constitute 85.1% of the dataset.

Resting ECG (restecg) vs. Target: Those with a restecg result of 1 are more likely to have heart disease compared to those with a result of 0. The majority have results categorized as 0 or 1.

Exercise-Induced Angina (exang) vs. Target: Individuals without exercise-induced angina are more likely to have heart disease. This group represents 67.3% of the study population.

Slope of Peak Exercise ST Segment vs. Target: Those with a downslope are more susceptible to heart disease. Most individuals display either a flat or downslope.

Number of Major Vessels Colored by Fluoroscopy (CA) vs. Target: Participants with zero major vessels colored are more prone to heart disease, making up 57.8% of the dataset.

Thallium Stress Result (thal) vs. Target: Individuals with a thal value of 2 are more likely to have heart disease, and they constitute 54.8% of the study population.

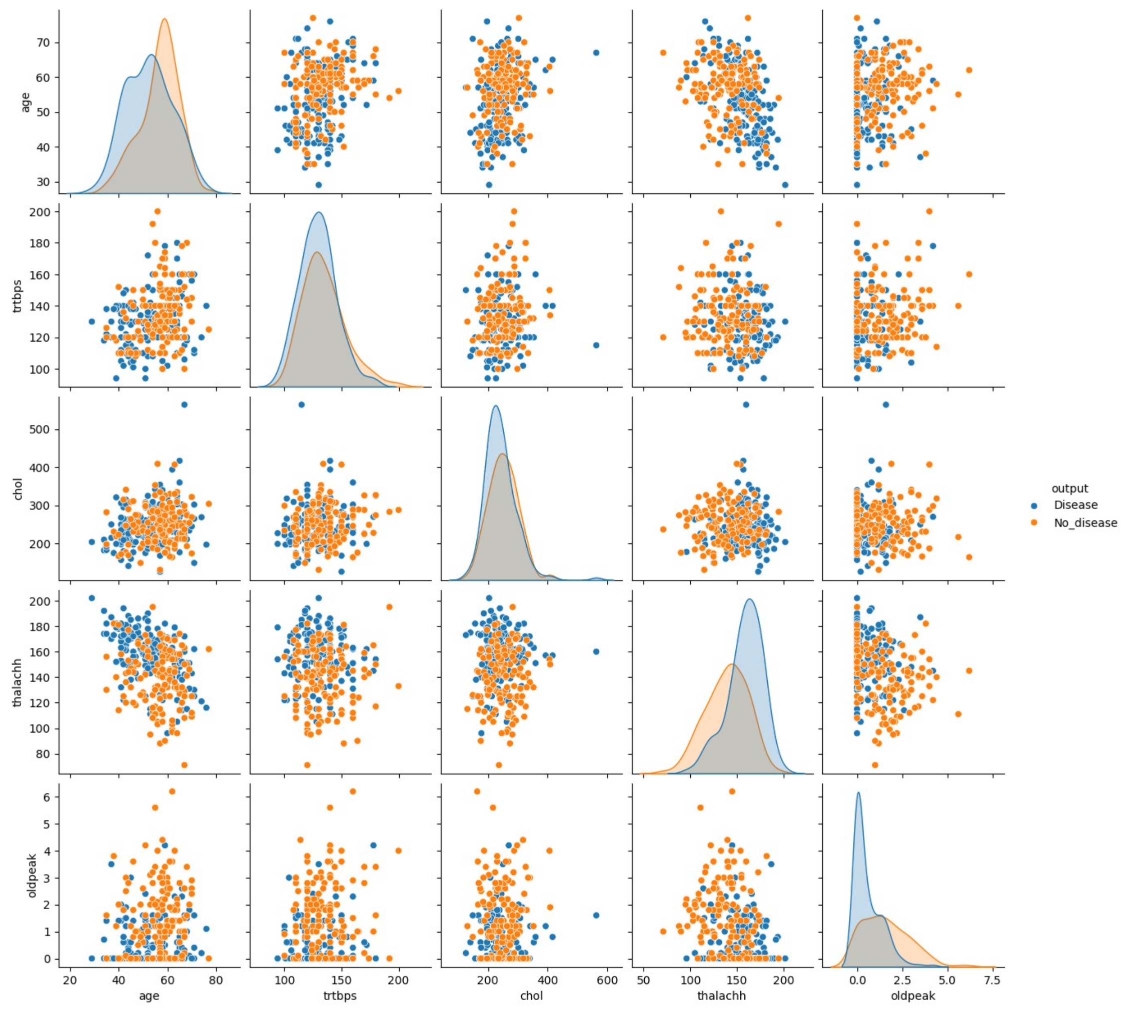
0.11.5 Visualize the distribution of continuous variable across target variable

[24]:
```
sns.pairplot(df[numerical_features + ['output']], hue='output')
```

```
C:\Users\ajult\AppData\Roaming\Python\Python311\site-
packages\seaborn\axisgrid.py:123: UserWarning: The figure layout has changed to
```

```
tight
  self._figure.tight_layout(*args, **kwargs)
```

[24]:
```
<seaborn.axisgrid.PairGrid at 0x248000ffc10>
```



Observation: A pair plot comparing heart disease to non-heart disease across numerical variables offers a comprehensive overview of the dataset, highlighting patterns and correlations among different metrics for both categories.

Age: The average age for individuals with heart disease is lower than for those without, suggesting age could inversely correlate with risk. The age distribution is mostly normal, centered around 50 to 70 years.

Resting Heart Rate (trtbps) & Cholesterol (chol): These variables show little variation between the heart disease and non-heart disease groups, indicating they may not be strong predictors. Outliers in these variables warrant further investigation.

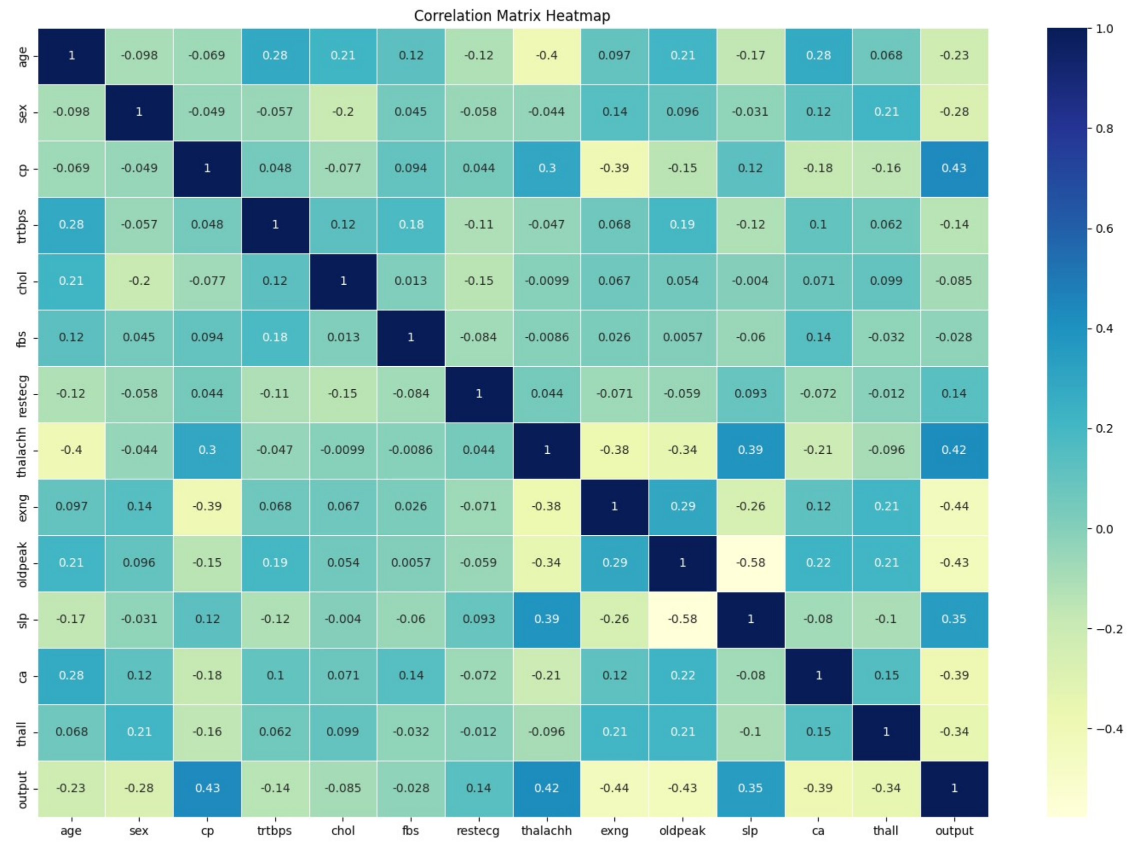
Maximum Heart Rate Achieved (thalachh): Individuals with heart disease generally achieve higher maximum heart rates, which could be an important variable for predictive modeling.

ST Depression Induced by Exercise Relative to Rest (oldpeak): A lower average oldpeak is observed among individuals with heart disease, potentially pointing to different stress responses between the two groups.

The pair plot serves as a valuable tool for visualizing interactions among these variables within the context of heart disease and non-heart disease categories

0.12 # Correlation Matrix

[25]:
```python
# Calculate the correlation matrix
correlation_matrix = data.corr()

# Visualize the correlation matrix as a heatmap
plt.figure(figsize=(18, 12))
sns.heatmap(correlation_matrix, annot=True, cmap='YlGnBu', linewidths=.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	ca	thall	output
age	1	-0.098	-0.069	0.28	0.21	0.12	-0.12	-0.4	0.097	0.21	-0.17	0.28	0.068	-0.23
sex	-0.098	1	-0.049	-0.057	-0.2	0.045	-0.058	-0.044	0.14	0.096	-0.031	0.12	0.21	-0.28
cp	-0.069	-0.049	1	0.048	-0.077	0.094	0.044	0.3	-0.39	-0.15	0.12	-0.18	-0.16	0.43
trtbps	0.28	-0.057	0.048	1	0.12	0.18	-0.11	-0.047	0.068	0.19	-0.12	0.1	0.062	-0.14
chol	0.21	-0.2	-0.077	0.12	1	0.013	-0.15	-0.0099	0.067	0.054	-0.004	0.071	0.099	-0.085
fbs	0.12	0.045	0.094	0.18	0.013	1	-0.084	-0.0086	0.026	0.0057	-0.06	0.14	-0.032	-0.028
restecg	-0.12	-0.058	0.044	-0.11	-0.15	-0.084	1	0.044	-0.071	-0.059	0.093	-0.072	-0.012	0.14
thalachh	-0.4	-0.044	0.3	-0.047	-0.0099	-0.0086	0.044	1	-0.38	-0.34	0.39	-0.21	-0.096	0.42
exng	0.097	0.14	-0.39	0.068	0.067	0.026	-0.071	-0.38	1	0.29	-0.26	0.12	0.21	-0.44
oldpeak	0.21	0.096	-0.15	0.19	0.054	0.0057	-0.059	-0.34	0.29	1	-0.58	0.22	0.21	-0.43
slp	-0.17	-0.031	0.12	-0.12	-0.004	-0.06	0.093	0.39	-0.26	-0.58	1	-0.08	-0.1	0.35
ca	0.28	0.12	-0.18	0.1	0.071	0.14	-0.072	-0.21	0.12	0.22	-0.08	1	0.15	-0.39
thall	0.068	0.21	-0.16	0.062	0.099	-0.032	-0.012	-0.096	0.21	0.21	-0.1	0.15	1	-0.34
output	-0.23	-0.28	0.43	-0.14	-0.085	-0.028	0.14	0.42	-0.44	-0.43	0.35	-0.39	-0.34	1

Observations: Lowest Correlation: Fasting Blood Sugar (fbs) and Cholesterol (chol) show the lowest correlation with the target variable. This aligns with earlier observations that these variables exhibited little variation between individuals with and without heart disease, suggesting they may not be strong predictors.

General Correlations: Most other variables are correlated with each other and with the target variable. For instance, age has an inverse correlation with the likelihood of having heart disease, while maximum heart rate achieved (thalachh) tends to be higher in individuals with heart disease.

The correlation matrix can serve as a statistical foundation for more in-depth analysis, helping to identify key variables that could be central to predictive modeling for heart disease.

0.13 Selecting PRML Algorithms

Selecting appropriate machine learning algorithms for a specific dataset involves considering factors like the nature of the data, the problem type, and the desired outcomes. In the case of heart disease dataset, which is a binary classification problem (predicting the presence or absence of heart disease), below mentioned algorithms can be suitable:

1. **Logistic Regression:**
   - Applicability: Binary classification problems.
   - Reasoning: Logistic Regression is a simple and interpretable algorithm that can serve as a baseline model. It works well when the relationship between features and the target variable is approximately linear. Given touryour dataset has both numerical and categorical features, logistic regression can handle both types effectively.
2. **Random Forest:**
   - Applicability: Classification problems, especially with structured data.
   - Reasoning: Random Forest is an ensemble algorithm known for its ability to handle both numerical and categorical features. It is robust, provides feature importance scores, and often works well "out of the box." Random Forests are also less prone to overfitting and can handle noiet3var3able.
3. **Support Vector Machines (SVM):**
   - Applicability: Binary classification problems.
   - Reasoning: SVM is effective for binary classification tasks, even when the data is not linearly separable. It works well when there is a clear margin of separation between classes. SVM can handle both numericalical features. s specific goals.

Reason for choosing the above algorithms

1) Data Type: Heart Disease dataset contains a mix of numerical and categorical features, which makes it suitable for algorithms like Random Forest, Gradient Boosting, and Logistic Regression that can handle both types of features effectively.

2) Binary Classification: Our task is binary classification (predicting the presence or absence of heart disease), making algorithms designed for classification tasks, like Logistic Regression, Random Forest, and Gradient Boosting, relevant.

3) Complexity:While Logistic Regression is simple and interpretable, Random Forest can capture more complex relationships in the data, which might be important for achieving high predictive accuracy.

4) Ensemble Methods: Random Forest and Gradient Boosting are both ensemble methods, which can help improve model performance by combining multiple weak learners.

0.14 Implementing Algorithms

[26]:
```
# import required modules
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

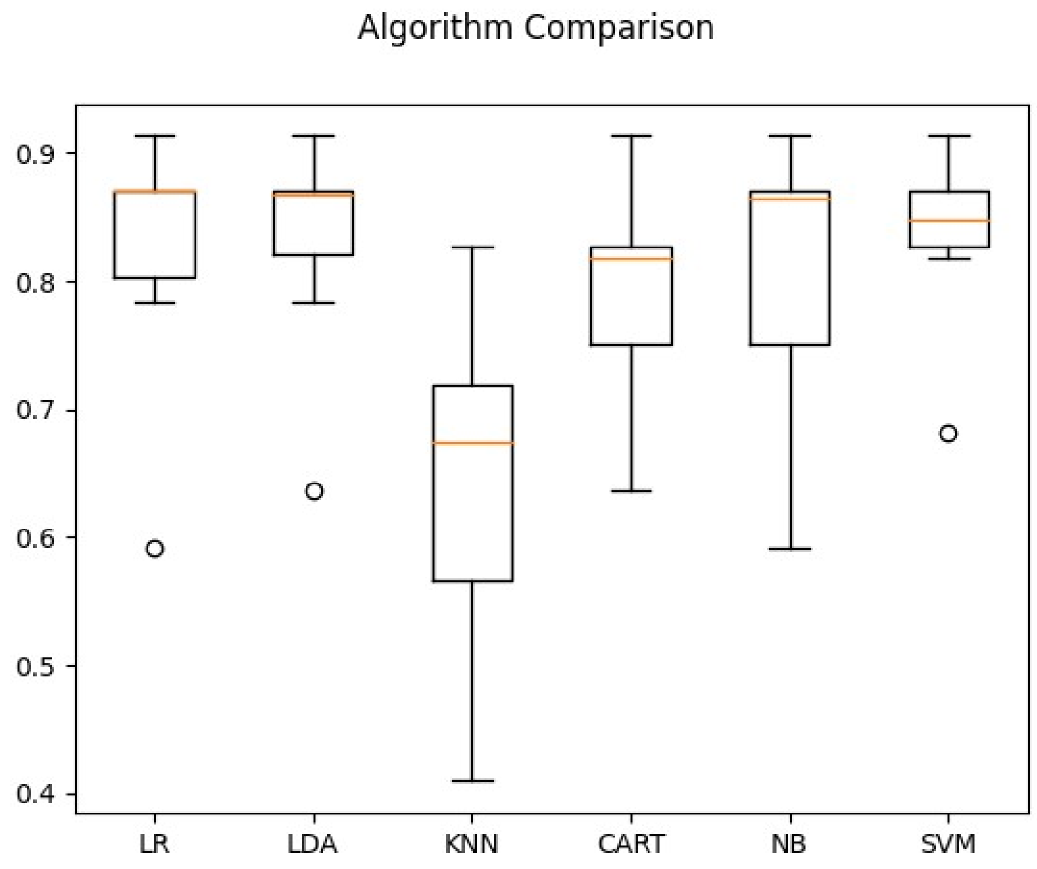
[27]:
```
data.shape

# features
X = data.iloc[:, 0:-1]

# target variable
y = data.iloc[:, -1]
```

[28]:
```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
  ↪random_state=16)
```

0.15 Spot Checking Algorithms with accuracy report

[29]:
```
# Load libraries
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

seed = 7

# Spot-Check Algorithms
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(kernel='linear')))
# evaluate each model in turn
results = []
names = []
for name, model in models:
        kfold = KFold(n_splits=10, random_state=seed, shuffle=True)
        cv_results = cross_val_score(model, X_train, y_train, cv=kfold,␣
 ↪scoring='accuracy')
        results.append(cv_results)
        names.append(name)
        msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
        print(msg)

# Compare Algorithms
fig = pyplot.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
pyplot.boxplot(results)
ax.set_xticklabels(names)
pyplot.show()
```

```
LR: 0.832016 (0.090642)
LDA: 0.831818 (0.073588)
KNN: 0.642095 (0.114810)
CART: 0.792490 (0.077942)
NB: 0.805731 (0.106107)
SVM: 0.840909 (0.062291)
```



0.15.1 Logistic Regression

[30]:
```
# import the class
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression(random_state=16, max_iter=1000)

# fit the model with data
logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)
```

[31]:
```
# import the metrics class
from sklearn import metrics

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

```
[31]: array([[31, 10],
             [ 4, 31]], dtype=int64)
```

```
[32]: from sklearn.metrics import classification_report

      report = classification_report(y_test, y_pred, target_names = ['Healthy',␣
       ↪'Heart Disease'])

      print(report)
```

```
               precision    recall  f1-score   support

      Healthy       0.89      0.76      0.82        41
Heart Disease       0.76      0.89      0.82        35

     accuracy                           0.82        76
    macro avg       0.82      0.82      0.82        76
 weighted avg       0.83      0.82      0.82        76
```

0.15.2 Support Vector Machines (SVM)

```
[33]: #Import svm model
      from sklearn import svm

      #Create a svm Classifier
      clf = svm.SVC(kernel='linear') # Linear Kernel

      #Train the model using the training sets
      clf.fit(X_train, y_train)

      #Predict the response for test dataset
      y_pred_svm = clf.predict(X_test)
```

```
[34]: # import the metrics class
      from sklearn import metrics

      cnf_matrix = metrics.confusion_matrix(y_test, y_pred_svm)
      cnf_matrix
```

```
[34]: array([[32,  9],
             [ 3, 32]], dtype=int64)
```

```
[35]: from sklearn.metrics import classification_report

      report = classification_report(y_test, y_pred_svm, target_names = ['Healthy',␣
       ↪'Heart Disease'])
```

```
print(report)
```

```
              precision    recall  f1-score   support

      Healthy       0.91      0.78      0.84        41
Heart Disease       0.78      0.91      0.84        35

     accuracy                           0.84        76
    macro avg       0.85      0.85      0.84        76
 weighted avg       0.85      0.84      0.84        76
```

0.15.3 Random Forest

[36]:
```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

[36]:
```
RandomForestClassifier()
```

[37]:
```
y_pred_rf = rf.predict(X_test)
```

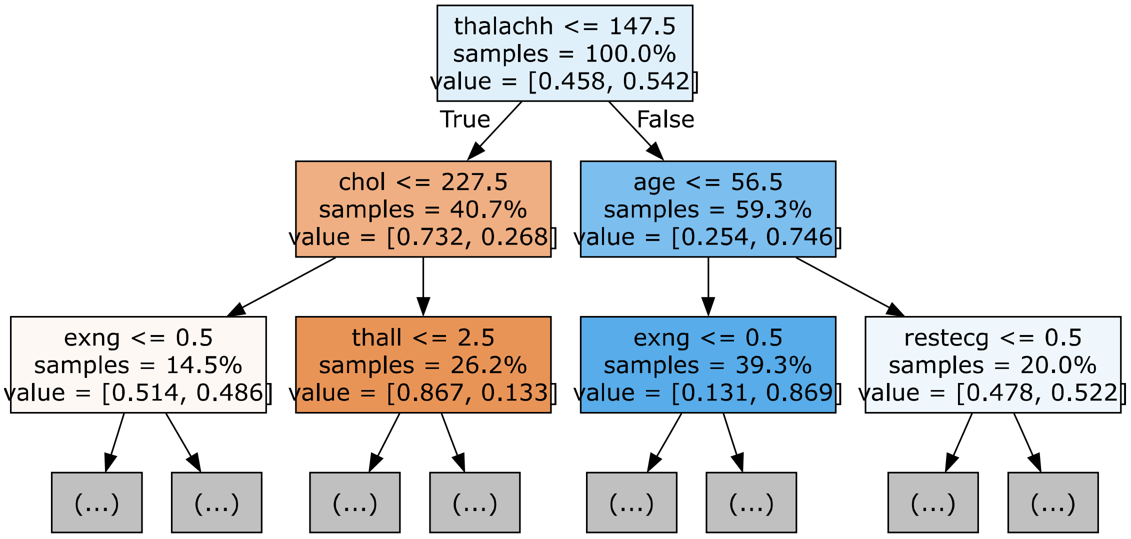
[38]:
```
# import the metrics class
from sklearn import metrics

cnf_matrix = metrics.confusion_matrix(y_test, y_pred_rf)
cnf_matrix
```

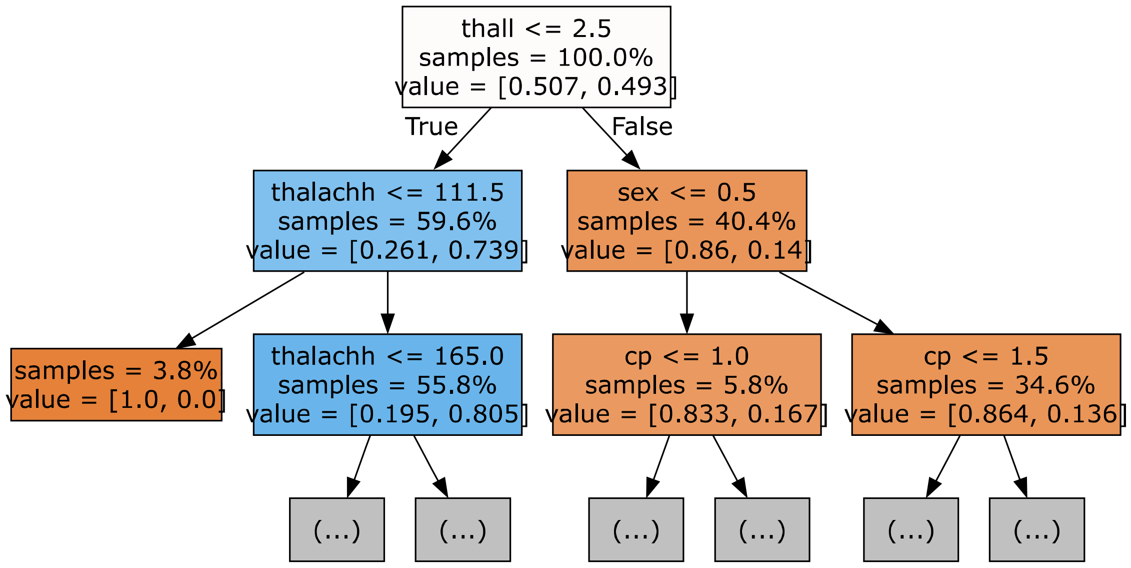
[38]:
```
array([[33,  8],
       [ 5, 30]], dtype=int64)
```

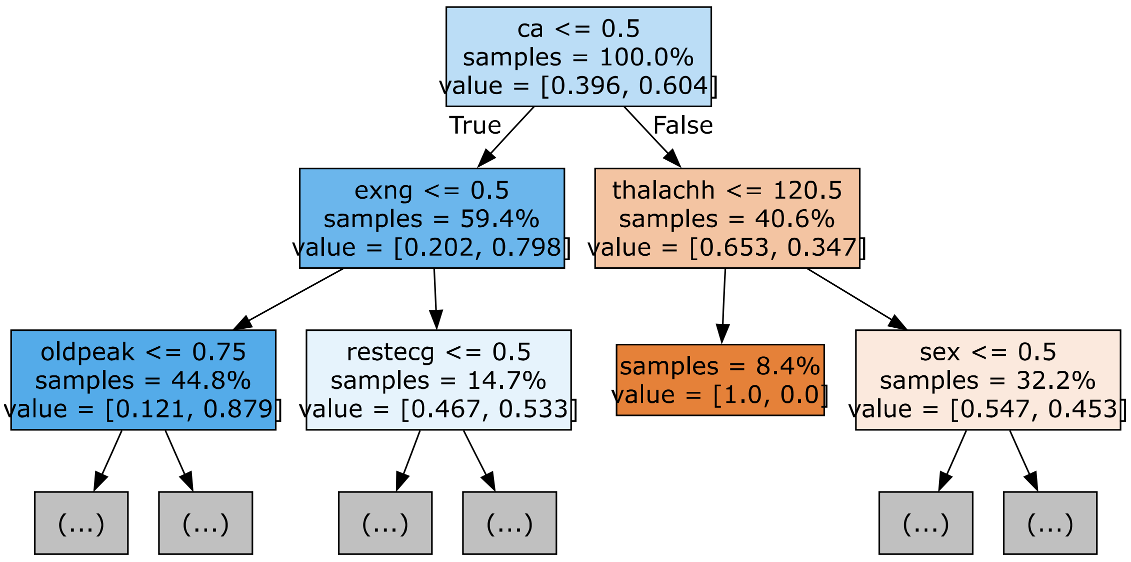
[39]:
```
from sklearn.metrics import classification_report

report = classification_report(y_test, y_pred_rf, target_names = ['Healthy',␣
 ↪'Heart Disease'])

print(report)
```

```
              precision    recall  f1-score   support

      Healthy       0.87      0.80      0.84        41
Heart Disease       0.79      0.86      0.82        35

     accuracy                           0.83        76
    macro avg       0.83      0.83      0.83        76
 weighted avg       0.83      0.83      0.83        76
```

```
[40]: # Export the first three decision trees from the forest
      from sklearn.tree import export_graphviz
      from IPython.display import Image
      import graphviz

      for i in range(3):
          tree = rf.estimators_[i]
          dot_data = export_graphviz(tree,
                                     feature_names=X_train.columns,
                                     filled=True,
                                     max_depth=2,
                                     impurity=False,
                                     proportion=True)
          graph = graphviz.Source(dot_data)
          display(graph)
```







0.16 ### References:

Author Unknown. (n.d.). Exploratory Data Analysis on Heart Disease UCI Data Set. Towards Data Science. Retrieved from https://towardsdatascience.com/exploratory-data-analysis-on-heart-disease-uci-data-set-ae129e47b323

Gaur, D. (n.d.). A Guide to any Classification Problem. Kaggle. Retrieved from https://www.kaggle.com/code/durgancegaur/a-guide-to-any-classification-problem

Rahmanpritom, R. (n.d.). Heart Attack Analysis & Prediction Dataset. Kaggle. Retrieved from https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset