Priciple of Data Science

Heart Disease Prediction using Data Science

Objective The primary goal of this venture is to broaden a dependable device getting to know version to are expecting heart disease through leveraging scientific and demographic data. The venture goals to investigate relationships and interactions amongst key threat factors, find hidden styles and subgroups for focused interventions, and make sure the version`s interpretability to facilitate actionable insights for clinical professionals. By combining superior analytics with strong validation, this paintings seeks to decorate decision-making in early analysis and customized healthcare strategies.

Step 1 : Data Loading and Exploration Loading the Data and Importing the necessary libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Loading data
df = pd.read_csv('heart.csv')


# Displaying the first few rows
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2

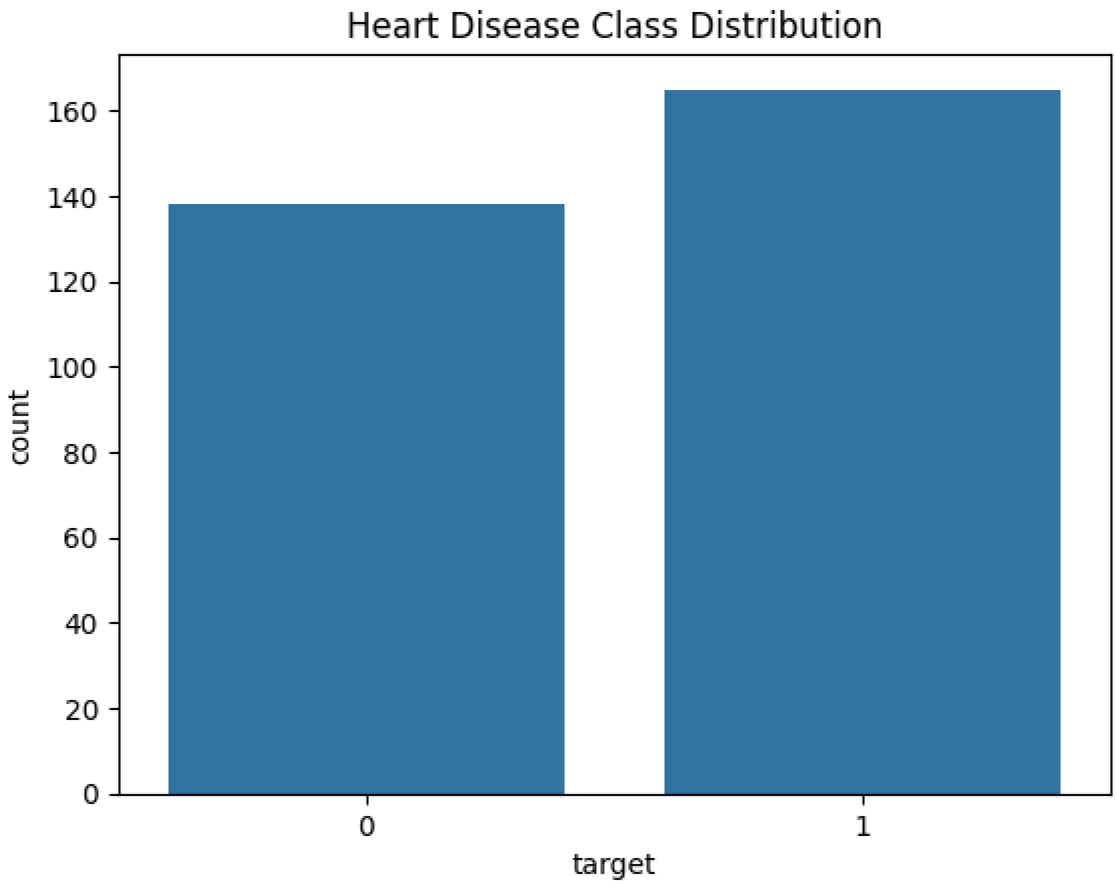
Some Basic Statistics

```
# Checking the data types and missing values
df.info()
df.isnull().sum()
```

```
<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   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

```
# Statistical summary
df.describe()

# Class distribution for the target variable
sns.countplot(x='target', data=df)
plt.title('Heart Disease Class Distribution')
```

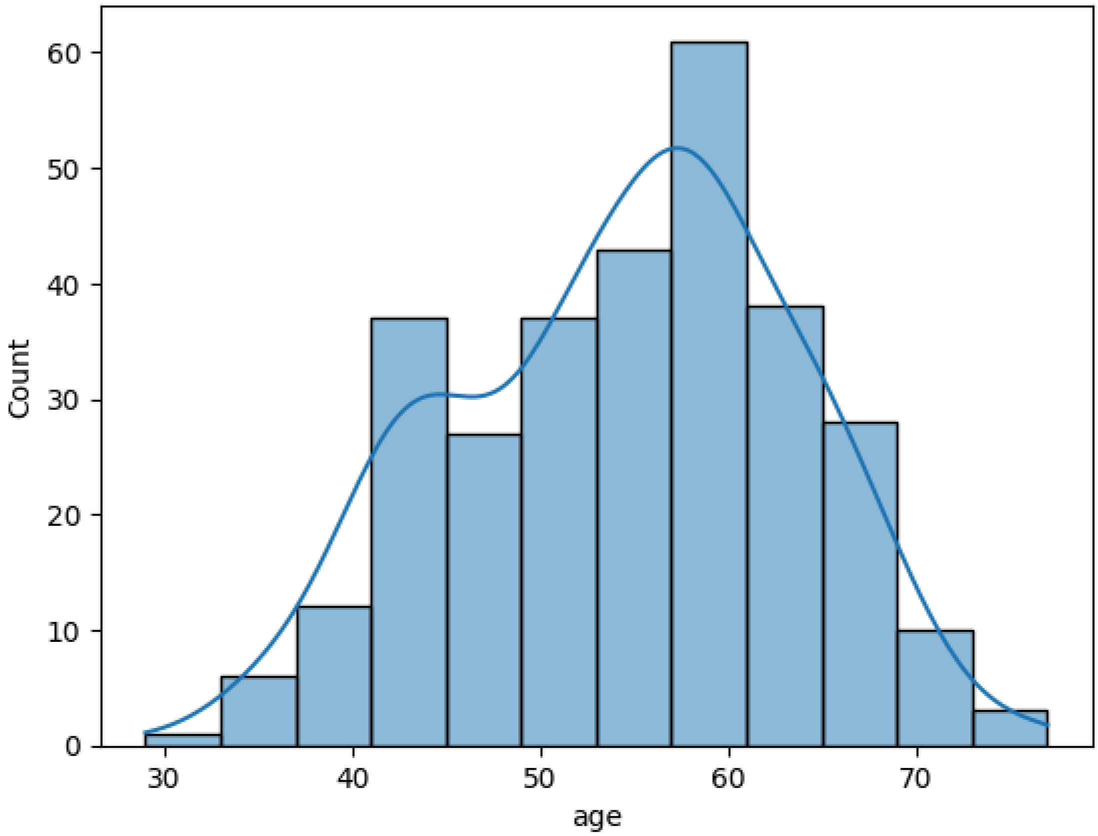
```
Text(0.5, 1.0, 'Heart Disease Class Distribution')
```



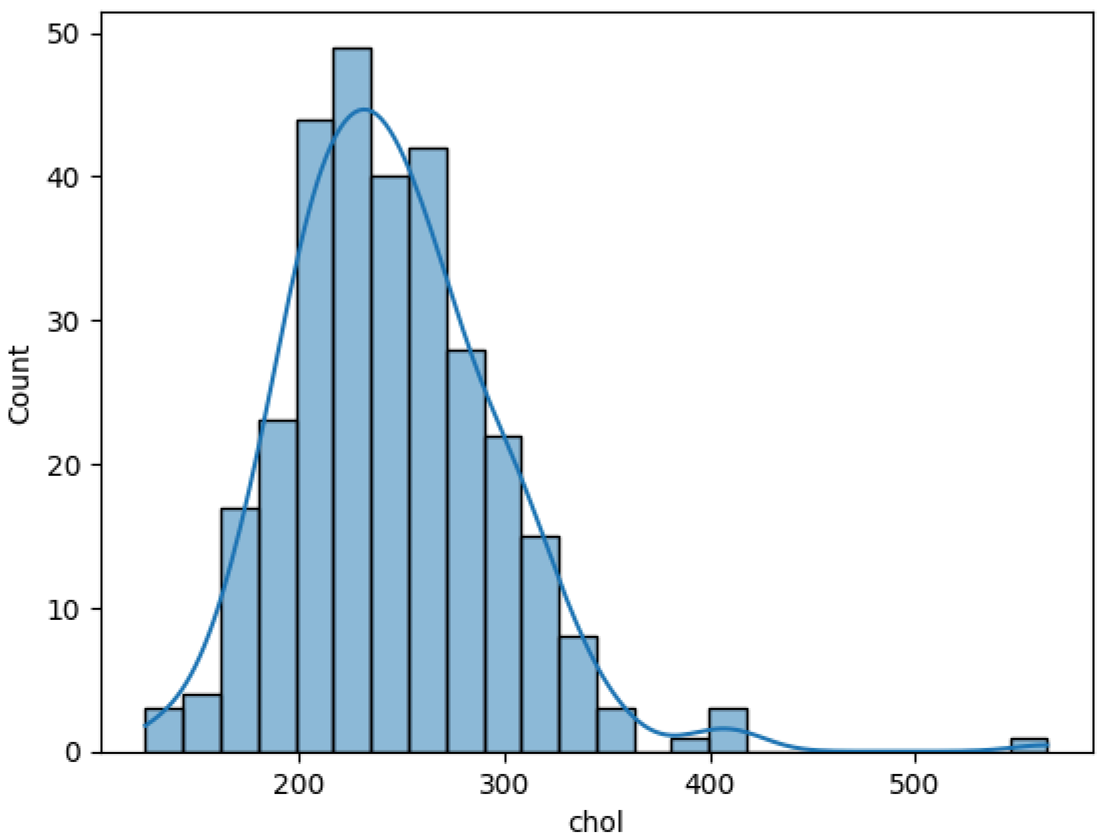
Step 2 : Exploratory Data Analysis (EDA) Univariate and Bivariate Analysis Visualize distributions and relationships. Analyze relationships between features using scatter plots, pairplots, or correlation heatmaps.

```
# Univariate analysis
for col in ['age', 'chol', 'thalach', 'trestbps']:
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribution of {col}')
    plt.show()
```

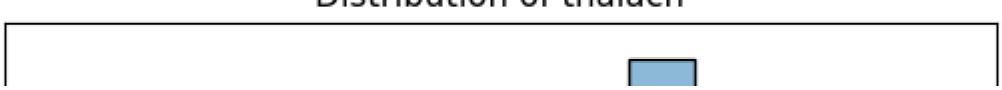
Distribution of age

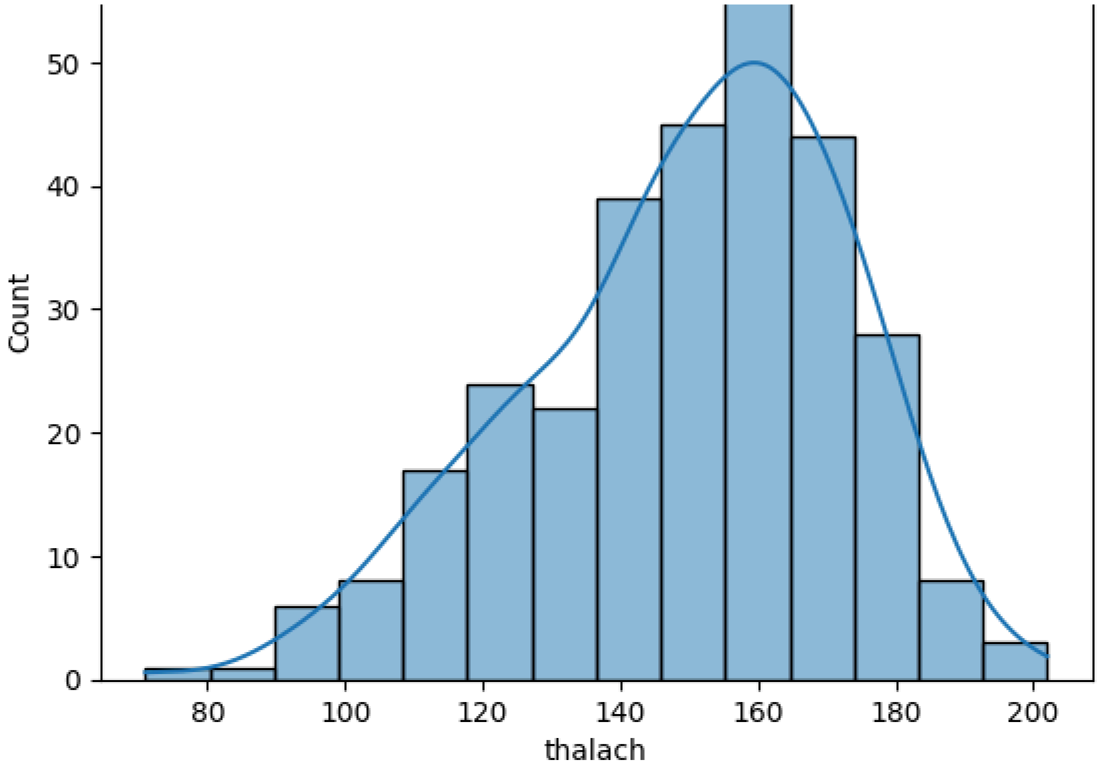


Distribution of chol

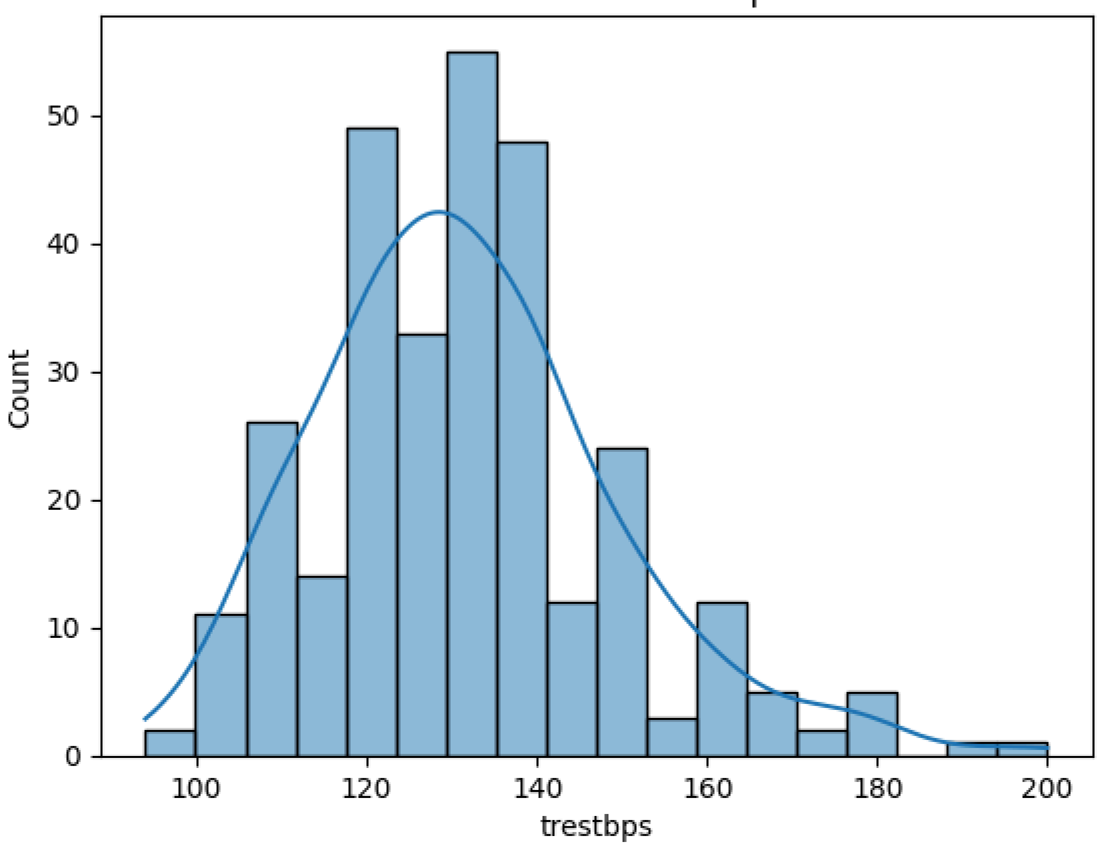


Distribution of thalach

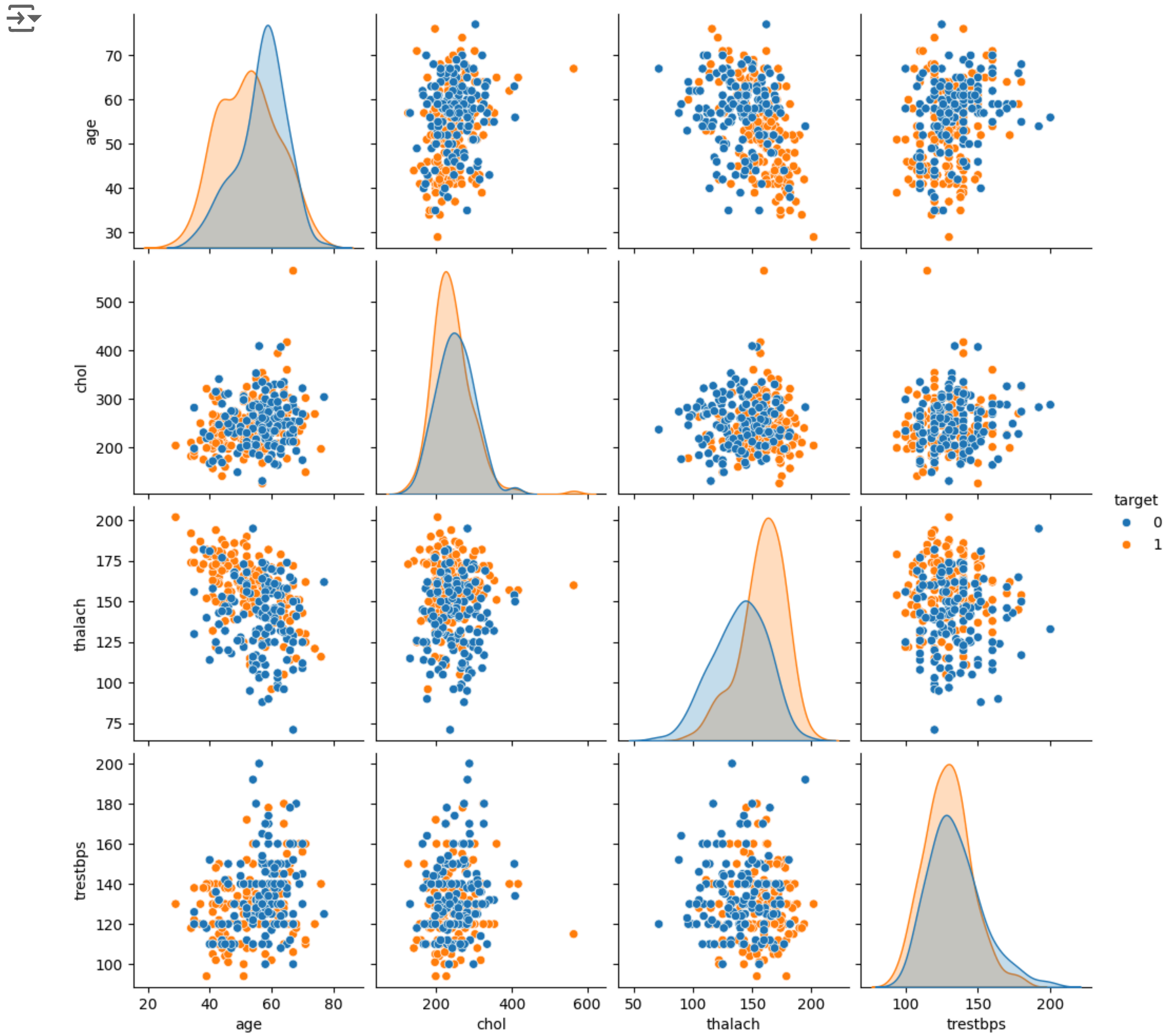




Distribution of trestbps



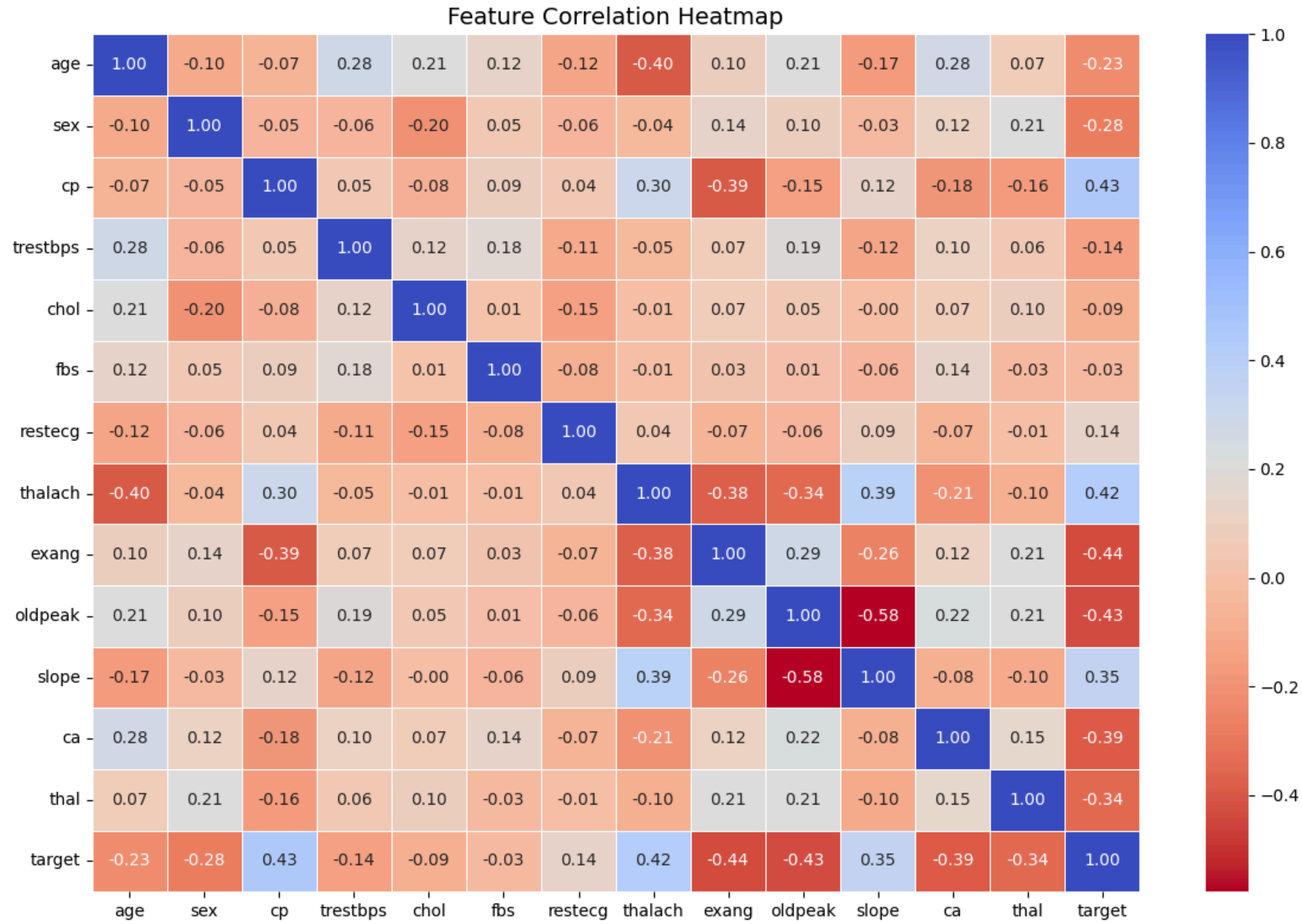
```
# Bivariate analysis
sns.pairplot(df, hue='target', vars=['age', 'chol', 'thalach', 'trestbps'])
plt.show()
```



age
chol
thalach
trestbps
target
0
1

Analysing features using correlation heatmap

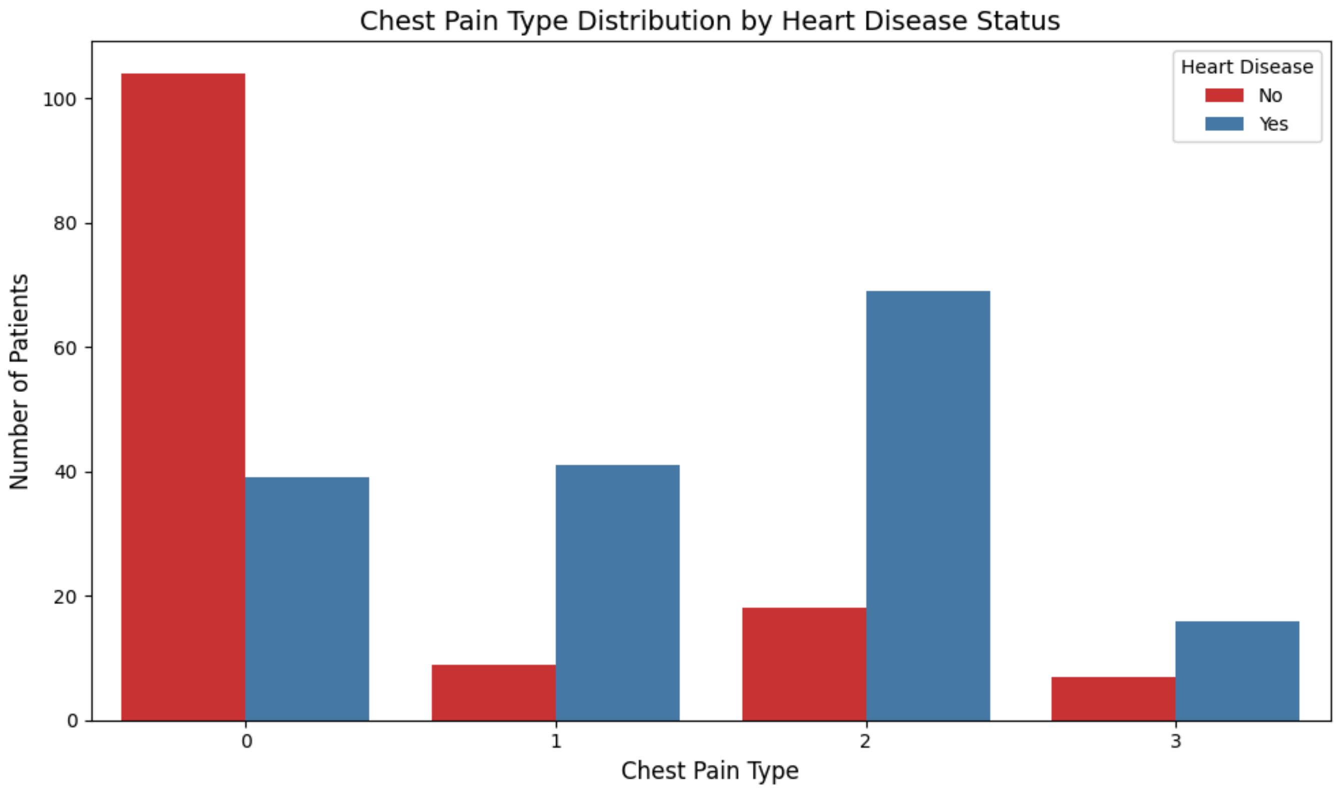
```
import matplotlib.pyplot as plt
import seaborn as sns


plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm_r', fmt='.2f', linewidths=0.5)
plt.title('Feature Correlation Heatmap', fontsize=14)
plt.tight_layout()
plt.show()
```



Sub group analysis

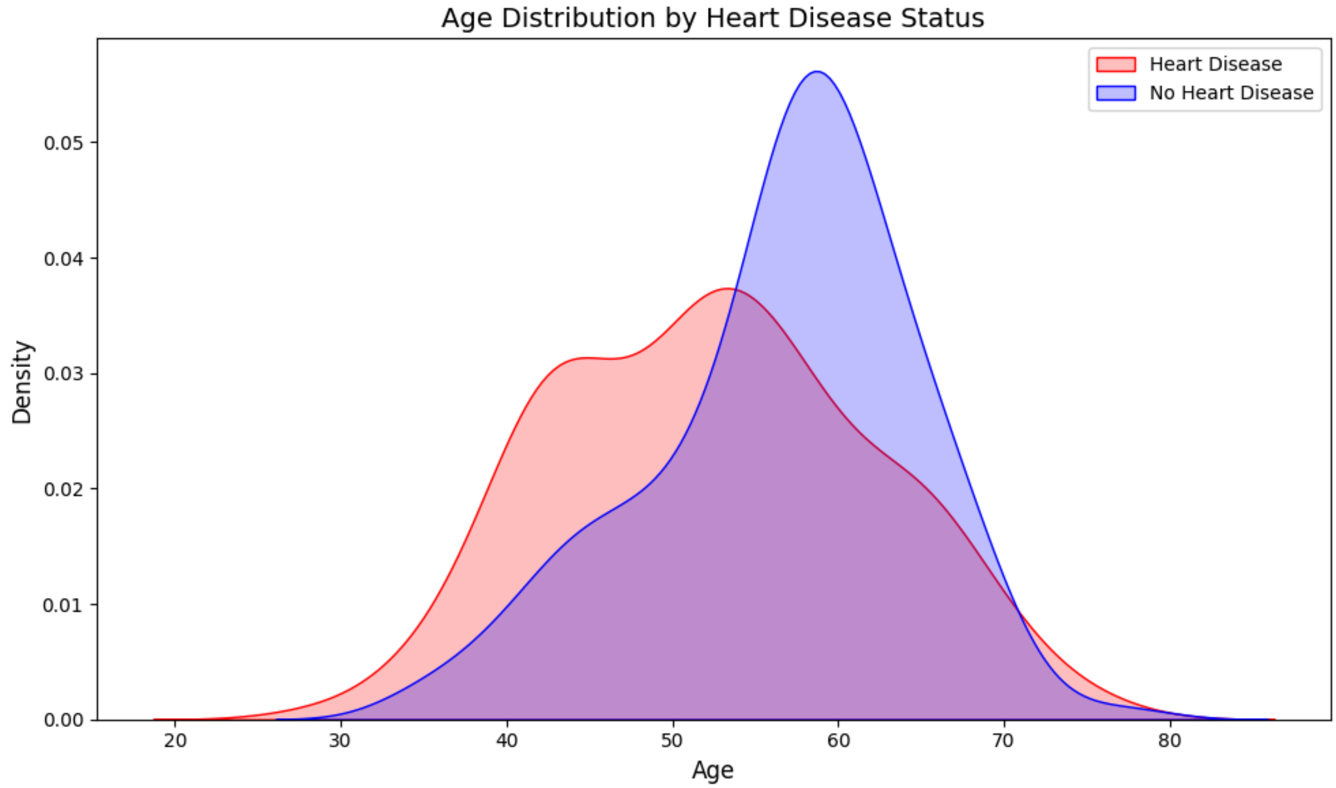
```
# Chest pain type vs target
plt.figure(figsize=(10, 6))
sns.countplot(x='cp', hue='target', data=data, palette='Set1')
plt.title('Chest Pain Type Distribution by Heart Disease Status', fontsize=14)
plt.xlabel('Chest Pain Type', fontsize=12)
```

```
plt.ylabel('Number of Patients', fontsize=12)
plt.legend(title='Heart Disease', labels=['No', 'Yes'])
plt.tight_layout()
plt.show()
```




```
# KDE plot for age distributions
plt.figure(figsize=(10, 6))
sns.kdeplot(data[data['target'] == 1]['age'], label='Heart Disease', shade=True, color='red'
sns.kdeplot(data[data['target'] == 0]['age'], label='No Heart Disease', shade=True, color='b
plt.title('Age Distribution by Heart Disease Status', fontsize=14)
plt.xlabel('Age', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.legend()
plt.tight_layout()
plt.show()
```

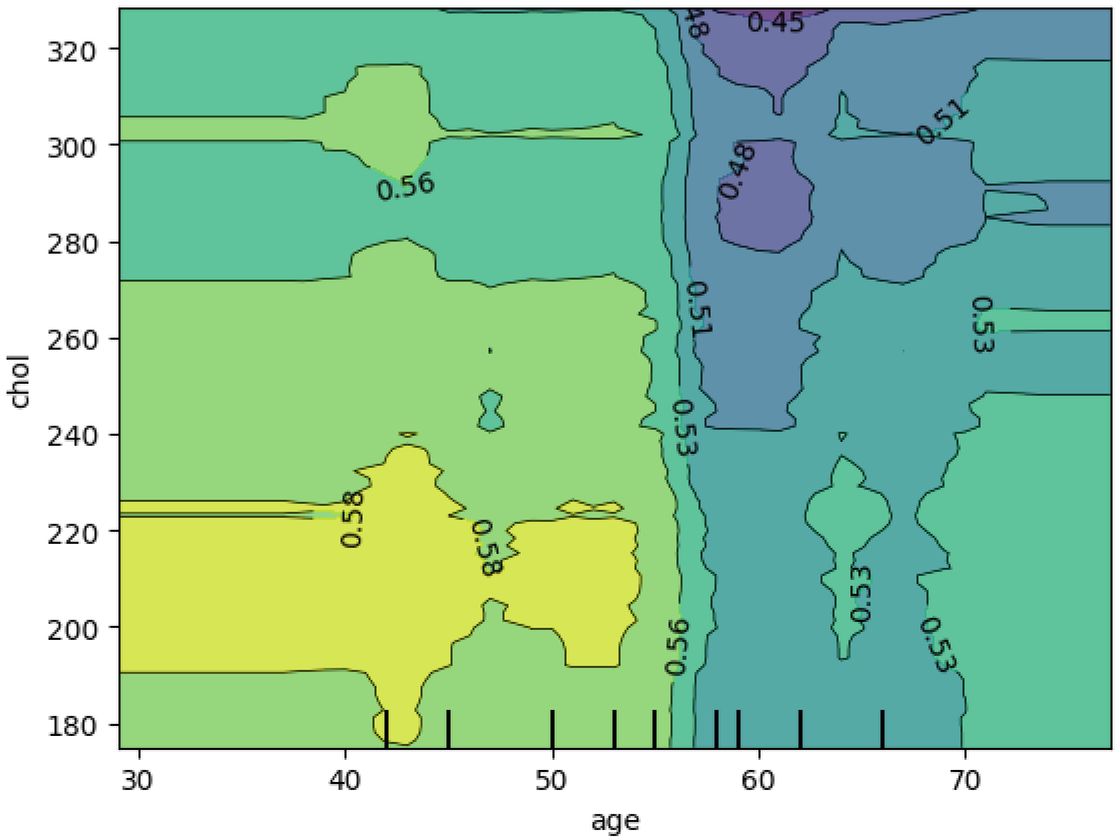
```
C:\Users\Sara Iqbal\AppData\Local\Temp\ipykernel_2296\500336527.py:3: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data[data['target'] == 1]['age'], label='Heart Disease', shade=True, color
C:\Users\Sara Iqbal\AppData\Local\Temp\ipykernel_2296\500336527.py:4: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data[data['target'] == 0]['age'], label='No Heart Disease', shade=True, cc
```



Feature Interactions by Creating Partial Dependence Plots (PDP).

```
from sklearn.inspection import PartialDependenceDisplay
from sklearn.ensemble import RandomForestClassifier

# Fit a basic model for PDP
```

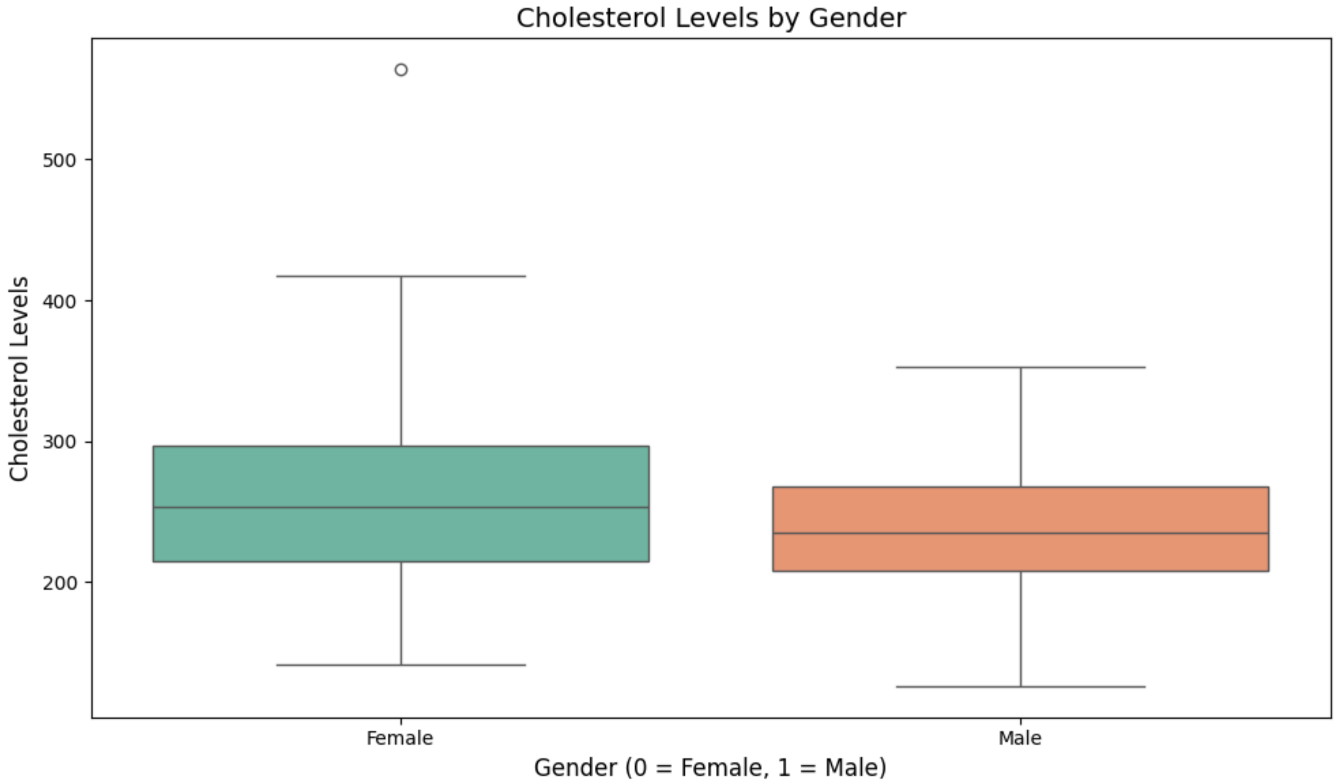
```
model = RandomForestClassifier(random_state=42)
X = df.drop(columns='target')
y = df['target']
model.fit(X, y)

# PDP for age and chol
PartialDependenceDisplay.from_estimator(model, X, features=[('age', 'chol')])
plt.show()
```



Outliers

```
# Boxplot for cholesterol levels by gender
plt.figure(figsize=(10, 6))
sns.boxplot(x='sex', y='chol', data=data, palette='Set2')
plt.title('Cholesterol Levels by Gender', fontsize=14)
plt.xlabel('Gender (0 = Female, 1 = Male)', fontsize=12)
plt.ylabel('Cholesterol Levels', fontsize=12)
plt.xticks(ticks=[0, 1], labels=['Female', 'Male'])
plt.tight_layout()
plt.show()
```

```
C:\Users\Sara Iqbal\AppData\Local\Temp\ipykernel_2296\1246271478.py:3: FutureWarning: Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
  sns.boxplot(x='sex', y='chol', data=data, palette='Set2')
```



The evaluation exhibits vast gender-precise variations in ldl cholesterol tiers. Females have a better median ldl cholesterol stage and extra variability, as indicated with the aid of using a much broader interquartile variety (IQR), as compared to males, who display decrease median tiers and a narrower IQR, reflecting greater regular values. An severe outlier withinside the girl group, exceeding 500 mg/dL, might also additionally imply a unprecedented clinical circumstance or anomaly requiring in addition investigation, even as no amazing outliers are gift withinside the male group. Cholesterol tiers variety from about one hundred fifty to 450 mg/dL for females (apart from the outlier) and one hundred eighty to 360 mg/dL for males, suggesting a narrower variety for males. These findings underscore the want for gender-precise fitness interventions, with ability affects from hormonal or way of life factors. Future steps encompass validating the girl outlier, carrying out speculation trying out to verify statistical variations in median ldl cholesterol tiers, and exploring correlations with different variables like BMI and blood stress to recognize broader influencing factors.

Step 3 : Feature Engineering 4.1 Composite Risk Index by Creating a composite feature.

```
df['risk_index'] = 0.3 * df['chol'] + 0.3 * df['trestbps'] + 0.2 * df['fbs'] + 0.2 * df['tha
```

Nonlinear Features : Engineer interaction and nonlinear terms.

```
df['oldpeak_squared'] = df['oldpeak'] ** 2
df['thalach_exang'] = df['thalach'] * df['exang']
```

Domain-Inspired Features : Creating medically meaningful thresholds.

```
df['high_chol'] = (df['chol'] > 240).astype(int)
```

Step 4 : Predictive Modeling

Baseline Models :Train Logistic Regression and Decision Trees.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, roc_auc_score

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Logistic Regression
lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)

print('Logistic Regression:')
print(classification_report(y_test, y_pred))
print('AUC:', roc_auc_score(y_test, lr.predict_proba(X_test)[:, 1]))
```

```
Logistic Regression:
              precision    recall  f1-score   support

           0       0.89      0.86      0.88        29
           1       0.88      0.91      0.89        32

    accuracy                           0.89        61
   macro avg       0.89      0.88      0.88        61
```

```
weighted avg       0.89      0.89      0.89        61

AUC: 0.927801724137931
C:\Users\Sara Iqbal\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\li
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

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

```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\sara iqbal\appdata\local\programs\pyt

Requirement already satisfied: numpy in c:\users\sara iqbal\appdata\local\programs\pytho
Requirement already satisfied: scipy in c:\users\sara iqbal\appdata\local\programs\pytho
```

```
!pip install optuna
```

```
Requirement already satisfied: optuna in c:\users\sara iqbal\appdata\local\programs\pyth
Requirement already satisfied: alembic>=1.5.0 in c:\users\sara iqbal\appdata\local\progr
Requirement already satisfied: colorlog in c:\users\sara iqbal\appdata\local\programs\py
Requirement already satisfied: numpy in c:\users\sara iqbal\appdata\local\programs\pytho
Requirement already satisfied: packaging>=20.0 in c:\users\sara iqbal\appdata\local\prog
Requirement already satisfied: sqlalchemy>=1.4.2 in c:\users\sara iqbal\appdata\local\pr
Requirement already satisfied: tqdm in c:\users\sara iqbal\appdata\local\programs\python
Requirement already satisfied: PyYAML in c:\users\sara iqbal\appdata\local\programs\pyth
Requirement already satisfied: Mako in c:\users\sara iqbal\appdata\local\programs\python
Requirement already satisfied: typing-extensions>=4 in c:\users\sara iqbal\appdata\local
Requirement already satisfied: greenlet!=0.4.17 in c:\users\sara iqbal\appdata\local\pro
Requirement already satisfied: colorama in c:\users\sara iqbal\appdata\local\programs\py
Requirement already satisfied: MarkupSafe>=0.9.2 in c:\users\sara iqbal\appdata\local\pr
```

Advanced Models

Use XGBoost and Hyperparameter Tuning with Optuna.

```
import xgboost as xgb
from optuna import create_study

# XGBoost model
xgb_model = xgb.XGBClassifier(random_state=42)
xgb_model.fit(X_train, y_train)
y_pred = xgb_model.predict(X_test)
```

```
print('XGBoost:')
print(classification_report(y_test, y_pred))
```

```
XGBoost:
              precision    recall  f1-score   support

           0       0.78      0.86      0.82        29
           1       0.86      0.78      0.82        32

    accuracy                           0.82        61
   macro avg       0.82      0.82      0.82        61
weighted avg       0.82      0.82      0.82        61
```

```
!pip install shap
```

```
Requirement already satisfied: shap in c:\users\sara iqbal\appdata\local\programs\pythor
Requirement already satisfied: numpy in c:\users\sara iqbal\appdata\local\programs\pythc
Requirement already satisfied: scipy in c:\users\sara iqbal\appdata\local\programs\pythc
Requirement already satisfied: scikit-learn in c:\users\sara iqbal\appdata\local\progran
Requirement already satisfied: pandas in c:\users\sara iqbal\appdata\local\programs\pytl
Requirement already satisfied: tqdm>=4.27.0 in c:\users\sara iqbal\appdata\local\progran
Requirement already satisfied: packaging>20.9 in c:\users\sara iqbal\appdata\local\progr
Requirement already satisfied: slicer==0.0.8 in c:\users\sara iqbal\appdata\local\progra
Requirement already satisfied: numba in c:\users\sara iqbal\appdata\local\programs\pythc
Requirement already satisfied: cloudpickle in c:\users\sara iqbal\appdata\local\programs
Requirement already satisfied: colorama in c:\users\sara iqbal\appdata\local\programs\py
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in c:\users\sara iqbal\appdata
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\sara iqbal\appdata\loc
Requirement already satisfied: pytz>=2020.1 in c:\users\sara iqbal\appdata\local\progran
Requirement already satisfied: tzdata>=2022.7 in c:\users\sara iqbal\appdata\local\progr
Requirement already satisfied: joblib>=1.2.0 in c:\users\sara iqbal\appdata\local\progra
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\sara iqbal\appdata\local
Requirement already satisfied: six>=1.5 in c:\users\sara iqbal\appdata\local\programs\py
```

```
!pip install numpy==2.0.0
```

```
Requirement already satisfied: numpy==2.0.0 in c:\users\sara iqbal\appdata\local\progran
```

Step 6 : Explainability

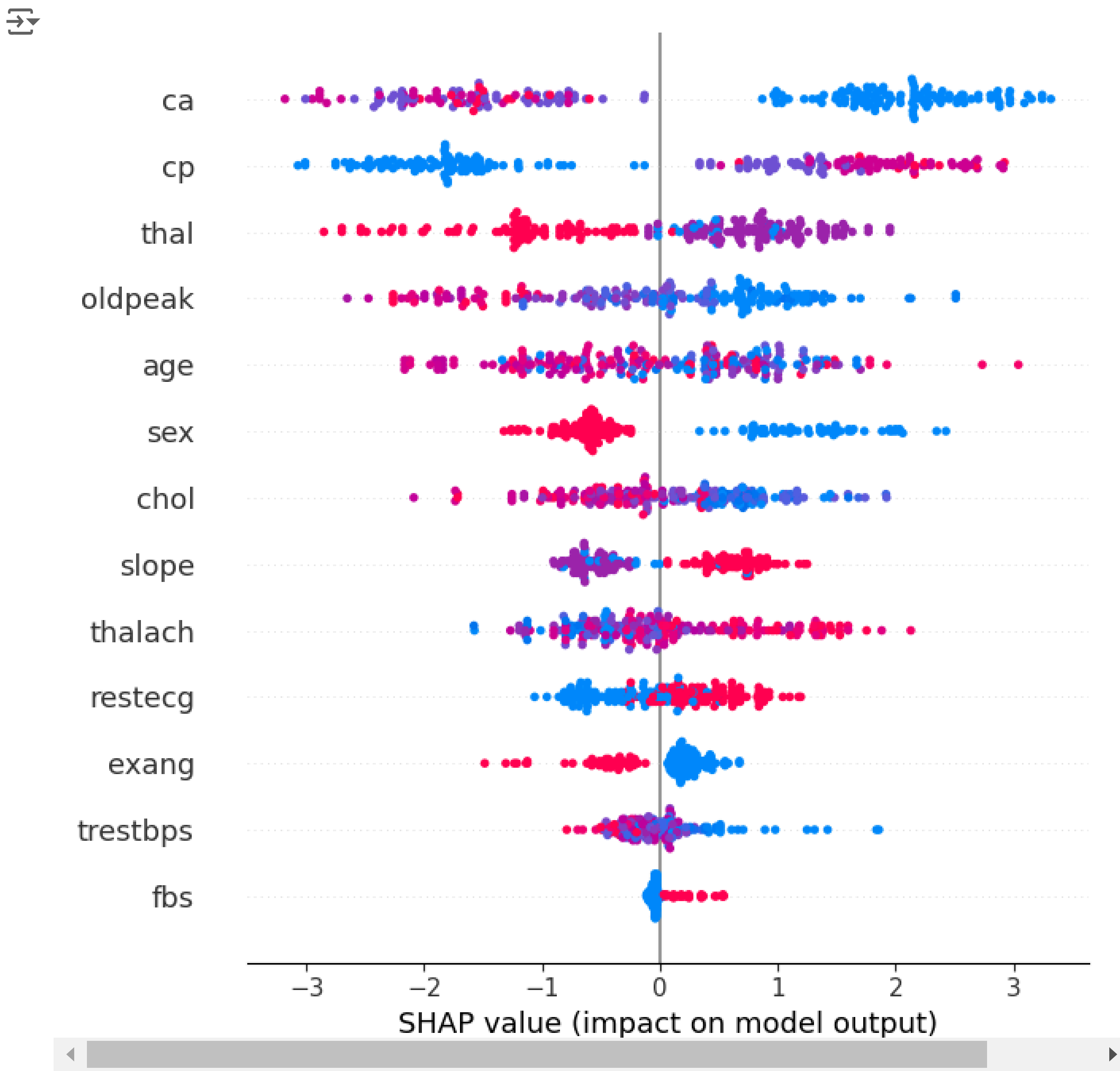
SHAP Analysis

```
import pandas as pd
import xgboost as xgb
import shap
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

```
data = pd.read_csv('heart (2).csv')  # Replace with your dataset file path
print(data.head())

X = data.drop(columns=['target'])  # Features
y = data['target']  # Target variable

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#Training the XGBoost model
xgb_model = xgb.XGBClassifier(random_state=42)
xgb_model.fit(X_train, y_train)

# Evaluating the model
y_pred = xgb_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   52    1   0       125   212    0        1      168      0      1.0      2
1   53    1   0       140   203    1        0      155      1      3.1      0
2   70    1   0       145   174    0        1      125      1      2.6      0
3   61    1   0       148   203    0        1      161      0      0.0      2
4   62    0   0       138   294    1        1      106      0      1.9      1

   ca  thal  target
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       102
           1       1.00      0.97      0.99       103

    accuracy                           0.99       205
   macro avg       0.99      0.99      0.99       205
weighted avg       0.99      0.99      0.99       205
```

```
explainer = shap.Explainer(xgb_model, X_test)
shap_values = explainer(X_test)
shap.summary_plot(shap_values, X_test)
```

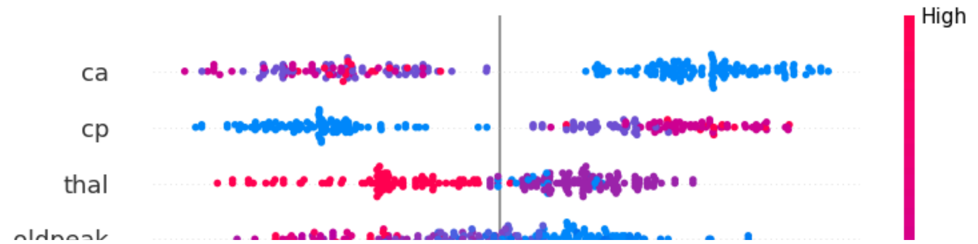


Step 6 : Clustering

K-Means : Clustering patients and analyze subgroups.

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

df['cluster'] = clusters
sns.pairplot(df, hue='cluster', vars=['age', 'chol', 'thalach'])
```

```
<seaborn.axisgrid.PairGrid at 0x1e84f34a5a0>
```



Step 6 : Clustering

K-Means : Clustering patients and analyze subgroups.


```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

df['cluster'] = clusters
sns.pairplot(df, hue='cluster', vars=['age', 'chol', 'thalach'])
```

```
<seaborn.axisgrid.PairGrid at 0x1e84f34a5a0>
```