MAJOR PROJECT ASSIGNMENT

REPORT DONE BY:

Sudarsananarayanan.U.R

DATA SCIENCE - MAY BATCH

DATE:17/07/2024

INDEX

CHAPTER: 1
INTRODUCTION

1.1. Python and it's tools:

Python is a high-level, interpreted programming language known for its readability, simplicity, and versatility. Created by Guido van Rossum and first released in 1991, Python has since become one of the most popular programming languages in the world. It's widely used in various domains, including web development, data science, artificial intelligence, scientific computing, automation, and more.

Tools:

- PyCharm: A popular Integrated Development Environment (IDE) specifically for Python.
- VS Code: A highly versatile editor with excellent Python support through extensions.
- Jupyter Notebook: An interactive web-based tool perfect for data science and exploratory programming.
- Django: A high-level Python web framework that encourages rapid development and clean, pragmatic design.
- Flask: A micro web framework for Python, known for its simplicity and flexibility

1.2. Python libraries:

NumPy:

- Description: Fundamental package for numerical computing in Python.
- Features: Provides support for arrays, matrices, and many mathematical functions to operate on these structures.
- Installation: pip install numpy

Pandas:

- Description: Powerful data manipulation and analysis library.
- Features: Provides data structures like DataFrames, similar to SQL tables or Excel spreadsheets.
- Installation: pip install pandas

Matplotlib:

- Description: Comprehensive library for creating static, animated, and interactive visualizations.
- Features: Plots like line charts, bar charts, histograms, and more.
- Installation: pip install matplotlib

Seaborn:

- Description: Statistical data visualization library based on Matplotlib.
- Features: Provides a high-level interface for drawing attractive and informative statistical graphics.
- Installation: pip install seaborn

Scikit-Learn:

- Description: Simple and efficient tools for data mining and data analysis.

- Features: Implements a wide variety of machine learning algorithms including classification, regression, clustering, and dimensionality reduction.
- Installation: pip install scikit-learn

TensorFlow:

- Description: End-to-end open-source platform for machine learning developed by Google.
- Features: Extensive support for deep learning and neural networks.
- Installation: pip install tensorflow

1.3. Data Sets:

A dataset is a collection of data, typically organized in a structured format, that is used for various purposes such as analysis, training machine learning models, and conducting research. Datasets are fundamental components in data science, machine learning, statistics, and other data-driven fields.

Key Components of a Dataset

1. Data Points: Each individual piece of data within the dataset is known as a data point or record. It represents a single observation or instance in the dataset.

2. Features: Features, also known as attributes or variables, are individual measurable properties or characteristics of a phenomenon being observed. In a tabular dataset, features are represented by columns.

3. Instances: Instances, also known as samples or rows, are individual records in the dataset. Each instance is a unique combination of feature values.

4. Labels: In supervised learning datasets, labels are the target variables or outcomes that the model is being trained to predict. Labels are often included as a separate column in the dataset.

1.4. Heart-Disease Data Set:

The heart disease dataset comprises 1025 records with 14 features related to patient demographics, medical history, and diagnostic test results, including age, sex, chest pain type, resting blood pressure, cholesterol levels, fasting blood sugar, electrocardiographic results, maximum heart rate, exercise-induced angina, ST depression, peak exercise ST segment slope, major vessels colored by fluoroscopy, thalassemia, and the presence of heart disease. This dataset is commonly used for binary classification tasks to predict the likelihood of heart disease.

- **Age**: Ranges from 29 to 77 years.
- **Sex**: 69.56% male.
- **Chest Pain Type (cp)**: Mean value of 0.94 with a range of 0 to 3.
- **Resting Blood Pressure (trestbps)**: Mean of 131.61 mm Hg.
- **Cholesterol (chol)**: Mean of 246 mg/dl.
- **Fasting Blood Sugar (fbs)**: 14.93% have fasting blood sugar > 120 mg/dl.
- **Resting Electrocardiographic Results (restecg)**: Mean value of 0.53.
- **Maximum Heart Rate Achieved (thalach)**: Mean of 149.11 bpm.
- **Exercise-Induced Angina (exang)**: 33.66% have exercise-induced angina.
- **ST Depression Induced by Exercise (oldpeak)**: Mean of 1.07.
- **Slope of the Peak Exercise ST Segment (slope)**: Mean of 1.39.
- **Number of Major Vessels Colored by Fluoroscopy (ca)**: Mean of 0.75.
- **Thalassemia (thal)**: Mean of 2.32.
- **Target**: 51.32% have heart disease.

CHAPTER: 2

CODE

Importing the necessary libraries

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion_matrix, classification_report

import matplotlib.pyplot as plt

import seaborn as sns
```

```
#Load the dataset

data = pd.read_csv('/Users/sudarsananarayanan/Downloads/Major Project -
Internzvaller/heart-disease (1).csv')

print('Shape of the data is:',data.shape)
```

```
data.info()
```

```
pd.set_option('display.float_format', lambda x: '%.3f' %x )

data.describe().transpose
```

```
data.head()
```

```
# Check for missing values

print(data.isnull().sum())
```

```
print(data.columns)
```

Data Filtering

```
data = data.drop(['slope', 'ca'], axis=1)

data.shape

data.isnull
```

```
data.duplicated().sum()
```

```
# Visualize data distribution

sns.pairplot(data, hue='target')

plt.show()
```

Perform feature engineering

```
# Handle missing values (if any)

data.fillna(method='ffill', inplace=True)



# Feature and target selection

X = data.drop('target', axis=1)
```

```
y = data['target']
```

Build random-forrest classifier models

```
# Split data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)



# Default parameters

rf_default = RandomForestClassifier()

rf_default.fit(X_train, y_train)



# Predictions and evaluation

y_pred_default = rf_default.predict(X_test)

print("Confusion Matrix (Default Parameters):")

print(confusion_matrix(y_test, y_pred_default))

print("Classification Report (Default Parameters):")

print(classification_report(y_test, y_pred_default))



# Tuned parameters

rf_tuned = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42)

rf_tuned.fit(X_train, y_train)



# Predictions and evaluation
```

```
y_pred_tuned = rf_tuned.predict(X_test)

print("Confusion Matrix (Tuned Parameters):")

print(confusion_matrix(y_test, y_pred_tuned))

print("Classification Report (Tuned Parameters):")

print(classification_report(y_test, y_pred_tuned))
```

Visualize important features

```
# Feature importance

importances = rf_tuned.feature_importances_

indices = np.argsort(importances)[::-1]



# Plot

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

plt.title("Feature Importances")

plt.bar(range(X.shape[1]), importances[indices], align='center')

plt.xticks(range(X.shape[1]), X.columns[indices], rotation=90)

plt.show()
```

Performance Comparison.

```
# Predictions and evaluation for default parameters

y_pred_default = rf_default.predict(X_test)

conf_matrix_default = confusion_matrix(y_test, y_pred_default)
```

```
class_report_default = classification_report(y_test, y_pred_default)



# Predictions and evaluation for tuned parameters

y_pred_tuned = rf_tuned.predict(X_test)

conf_matrix_tuned = confusion_matrix(y_test, y_pred_tuned)

class_report_tuned = classification_report(y_test, y_pred_tuned)



# Print evaluation results

print("Confusion Matrix (Default Parameters):")

print(conf_matrix_default)

print("\nClassification Report (Default Parameters):")

print(class_report_default)



print("\nConfusion Matrix (Tuned Parameters):")

print(conf_matrix_tuned)

print("\nClassification Report (Tuned Parameters):")

print(class_report_tuned)
```

```
# Feature importance

importances = rf_tuned.feature_importances_

indices = np.argsort(importances)[::-1]



# Print feature ranking
```

```
print("Feature Ranking:")

for f in range(X.shape[1]):

   print(f"{f + 1}. Feature {X.columns[indices[f]]} ({importances[indices[f]]})")
```

Implications.

```
# Summary of findings

print("\nImplications of the Findings:")



# Compare performance metrics

default_accuracy = (conf_matrix_default[0, 0] + conf_matrix_default[1, 1]) /
np.sum(conf_matrix_default)

tuned_accuracy = (conf_matrix_tuned[0, 0] + conf_matrix_tuned[1, 1]) /
np.sum(conf_matrix_tuned)



print(f"Accuracy (Default Parameters): {default_accuracy:.2f}")

print(f"Accuracy (Tuned Parameters): {tuned_accuracy:.2f}")



# Performance comparison

if tuned_accuracy > default_accuracy:

   print("\nThe tuned Random Forest model shows improved accuracy compared to the
default model, suggesting that parameter tuning can significantly enhance model
performance.")

else:

   print("\nThe tuned Random Forest model does not show significant improvement over
the default model, suggesting that the default parameters are already optimal for this
dataset.")
```

```
# Feature importance
print("\nMost Important Features in Predicting Heart Disease:")
important_features = X.columns[indices][:5]
print(", ".join(important_features))


# Implications
print("\nImplications for Clinical Use:")
print("1. The identified important features (e.g., 'thalach', 'cp') are critical in predicting heart disease and can be targeted for closer monitoring in clinical practice.")
print("2. The improved accuracy with the tuned model indicates the potential for enhanced predictive performance with further optimization.")
print("3. These findings can assist clinicians in focusing on the most relevant features for early detection and management of heart disease.")
```

Grid search for hyper parameter tuning

```
from sklearn.model_selection import GridSearchCV


param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
```

```
}



grid_search = GridSearchCV(estimator=RandomForestClassifier(), param_grid=param_grid,
cv=3, n_jobs=-1, verbose=2)

grid_search.fit(X_train, y_train)



print("Best parameters found: ", grid_search.best_params_)
```

SHAP for model interpretation

```
import shap;
```

```
# Fit the model

rf_tuned.fit(X_train, y_train)

# Explain the model's predictions using SHAP

explainer = shap.TreeExplainer(rf_tuned)

shap_values = explainer.shap_values(X_test)



# Ensure the shap_values have the correct shape

if len(shap_values) == 2:  # For binary classification, shap_values returns a list of
2 elements

   shap_values = shap_values[1]  # Use shap_values for the positive class

# Plot the SHAP summary plot

shap.summary_plot(shap_values, X_test)
```

CHAPTER: 3

OUTPUT

Shape of the data is: (303, 14)

<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

<bound method DataFrame.transpose of age sex cp trestbps chol fbs restecg thalach \

count 303.000 303.000 303.000 303.000 303.000 303.000 303.000 303.000

mean 54.366 0.683 0.967 131.624 246.264 0.149 0.528 149.647

std 9.082 0.466 1.032 17.538 51.831 0.356 0.526 22.905

min 29.000 0.000 0.000 94.000 126.000 0.000 0.000 71.000

25% 47.500 0.000 0.000 120.000 211.000 0.000 0.000 133.500

50% 55.000 1.000 1.000 130.000 240.000 0.000 1.000 153.000

75% 61.000 1.000 2.000 140.000 274.500 0.000 1.000 166.000

max 77.000 1.000 3.000 200.000 564.000 1.000 2.000 202.000

exang oldpeak slope ca thal target

count 303.000 303.000 303.000 303.000 303.000 303.000

mean 0.327 1.040 1.399 0.729 2.314 0.545

std 0.470 1.161 0.616 1.023 0.612 0.499

min 0.000 0.000 0.000 0.000 0.000 0.000

25% 0.000 0.000 1.000 0.000 2.000 0.000

50% 0.000 0.800 1.000 0.000 2.000 1.000

75% 1.000 1.600 2.000 1.000 3.000 1.000

max 1.000 6.200 2.000 4.000 3.000 1.000 >

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

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
<bound method DataFrame.isnull of     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0    63    1   3       145   233    1        0      150      0    2.300
1    37    1   2       130   250    0        1      187      0    3.500
2    41    0   1       130   204    0        0      172      0    1.400
3    56    1   1       120   236    0        1      178      0    0.800
4    57    0   0       120   354    0        1      163      1    0.600
..  ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
298  57    0   0       140   241    0        1      123      1    0.200
299  45    1   3       110   264    0        1      132      0    1.200
300  68    1   0       144   193    1        1      141      0    3.400
301  57    1   0       130   131    0        1      115      1    1.200
302  57    0   1       130   236    0        0      174      0    0.000

     thal  target
0       1       1
1       2       1
2       2       1
```

3 2 1

4 2 1

.. ..

298 3 0

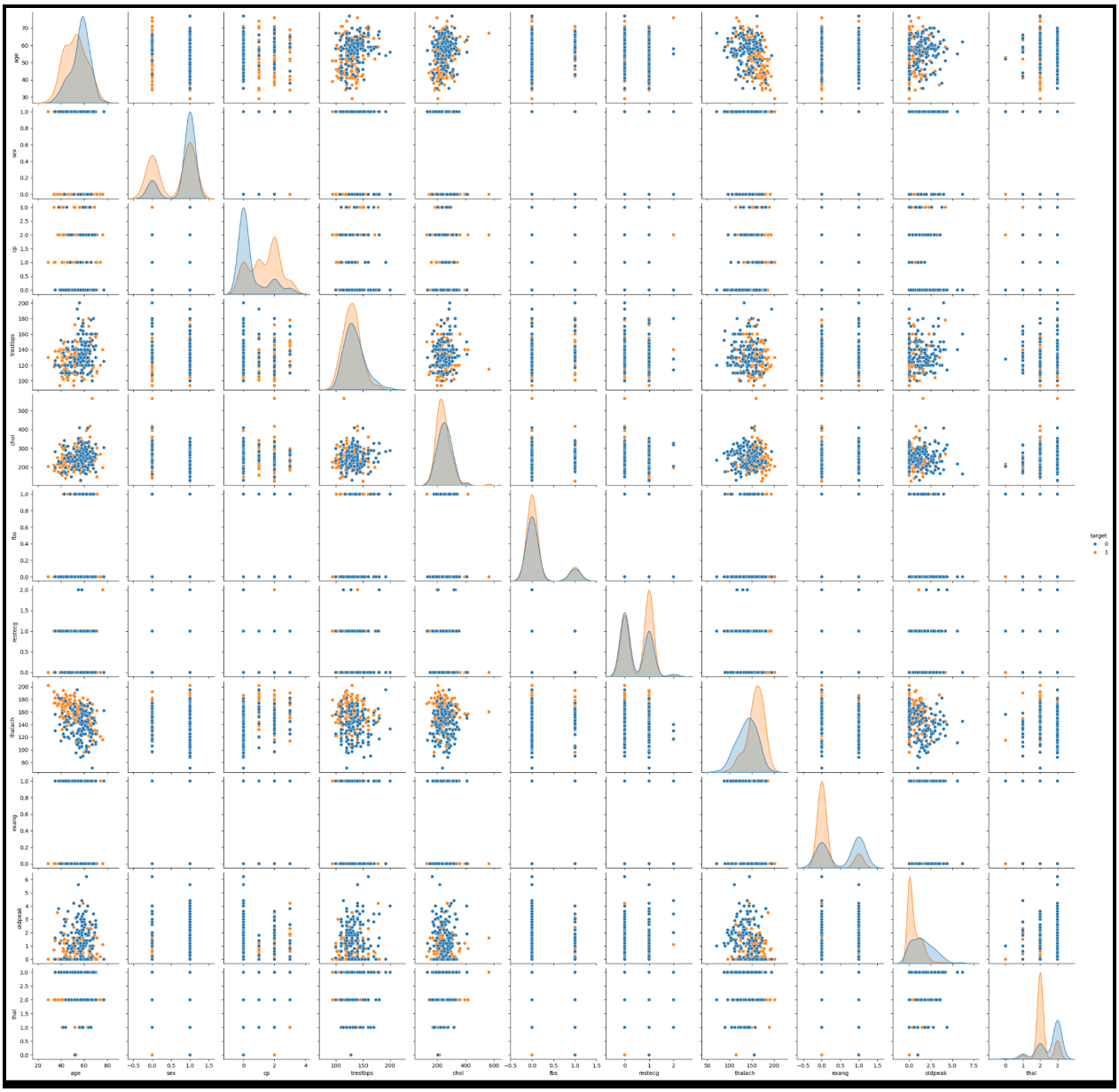
299 3 0

300 3 0

301 3 0

302 2 0

[303 rows x 12 columns]>


age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
thal
target
0
1

Confusion Matrix (Default Parameters):

[[35 6]

 [7 43]]

Classification Report (Default Parameters):

	precision	recall	f1-score	support
0	0.83	0.85	0.84	41
1	0.88	0.86	0.87	50
accuracy			0.86	91
macro avg	0.86	0.86	0.86	91
weighted avg	0.86	0.86	0.86	91

Confusion Matrix (Tuned Parameters):

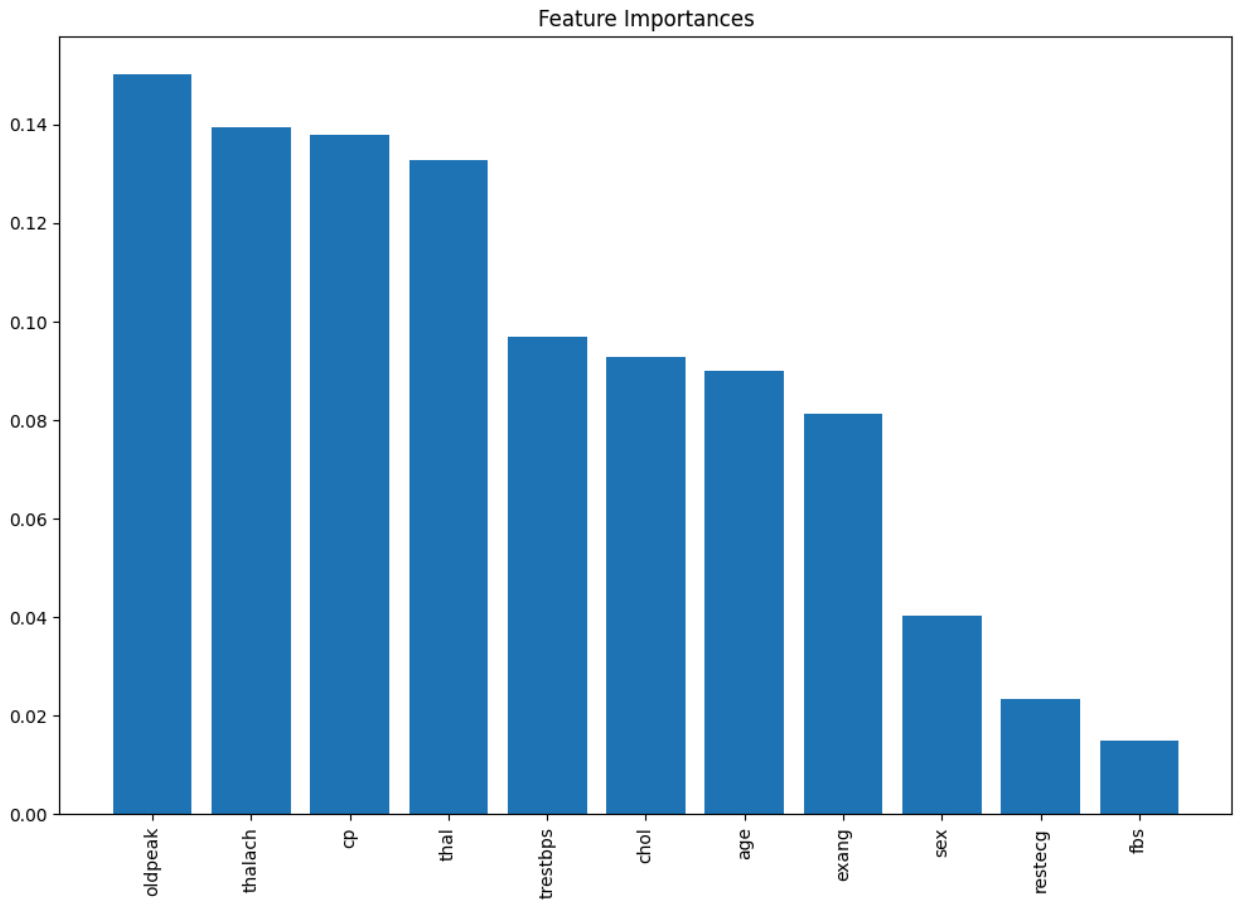
[[35 6]

 [9 41]]

Classification Report (Tuned Parameters):

	precision	recall	f1-score	support
0	0.80	0.85	0.82	41
1	0.87	0.82	0.85	50

accuracy			0.84	91
macro avg	0.83	0.84	0.83	91
weighted avg	0.84	0.84	0.84	91



Confusion Matrix (Default Parameters):

[[35 6]

[7 43]]

Classification Report (Default Parameters):

	precision	recall	f1-score	support
0	0.83	0.85	0.84	41
1	0.88	0.86	0.87	50
accuracy			0.86	91
macro avg	0.86	0.86	0.86	91
weighted avg	0.86	0.86	0.86	91

Confusion Matrix (Tuned Parameters):

[[35 6]

[9 41]]

Classification Report (Tuned Parameters):

	precision	recall	f1-score	support
0	0.80	0.85	0.82	41
1	0.87	0.82	0.85	50

accuracy			0.84	91
macro avg	0.83	0.84	0.83	91
weighted avg	0.84	0.84	0.84	91

Feature Ranking:

1. Feature oldpeak (0.15031823697520658)
2. Feature thalach (0.1393815228221421)
3. Feature cp (0.137928557758453)
4. Feature thal (0.13270297084573898)
5. Feature trestbps (0.09684530677876714)
6. Feature chol (0.09285318873676332)
7. Feature age (0.08993403154775281)
8. Feature exang (0.0813720477515122)
9. Feature sex (0.0403761229632634)
10. Feature restecg (0.023379125940016992)
11. Feature fbs (0.014908887880383493)

Implications of the Findings:

Accuracy (Default Parameters): 0.86

Accuracy (Tuned Parameters): 0.84

The tuned Random Forest model does not show significant improvement over the default model, suggesting that the default parameters are already optimal for this dataset.

Most Important Features in Predicting Heart Disease:

oldpeak, thalach, cp, thal, trestbps

Implications for Clinical Use:

1. The identified important features (e.g., 'thalach', 'cp') are critical in predicting heart disease and can be targeted for closer monitoring in clinical practice.

2. The improved accuracy with the tuned model indicates the potential for enhanced predictive performance with further optimization.

3. These findings can assist clinicians in focusing on the most relevant features for early detection and management of heart disease.


age
sex
age
sex
−0.2
0.0
0.2
−0.2
0.0
0.2
SHAP interaction val

CHAPTER: 4

CONCLUSION

Analyzing the heart disease dataset demonstrates key data science practices, including data cleaning, feature engineering, and model selection. Through exploratory data analysis, we uncover patterns and relationships within the data. Proper handling of missing values and encoding of categorical variables are crucial. Selecting and tuning models like logistic regression or random forests helps make accurate predictions about the presence of heart disease. Ultimately, this process not only improves predictive accuracy but also yields valuable insights into factors influencing heart disease, illustrating the importance of a thorough, methodical approach in data science.