第八章

1.实践案例

①决策树代码

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.tree import plot\_tree

import matplotlib.pyplot as plt

# 加载鸢尾花数据集

iris = load\_iris()

X = iris.data

y = iris.target

# 将数据集分为训练集和测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# 初始化决策树分类器

clf = DecisionTreeClassifier(random\_state=42)

# 训练模型

clf.fit(X\_train, y\_train)

# 使用模型进行预测

y\_pred = clf.predict(X\_test)

# 评估模型

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.2f}")

# 打印分类报告

print("Classification Report:")

print(classification\_report(y\_test, y\_pred, target\_names=iris.target\_names))

# 打印混淆矩阵

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

plt.figure(figsize=(20,10))

plot\_tree(clf, filled=True, feature\_names=iris.feature\_names, class\_names=iris.target\_names, rounded=True)

plt.show()

②SVM案例代码

import numpy as np

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import classification\_report, confusion\_matrix

# 加载鸢尾花数据集

iris = datasets.load\_iris()

X = iris.data # 特征

y = iris.target # 标签

# 将数据集分为训练集和测试集并进行特征缩放

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# 特征缩放（标准化）

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# 创建SVM分类器

svm\_classifier = SVC(kernel='linear') # 这里使用线性核，你也可以尝试其他核函数，如'rbf'

# 训练模型

svm\_classifier.fit(X\_train, y\_train)

# 进行预测

y\_pred = svm\_classifier.predict(X\_test)

# 评估模型性能

print("混淆矩阵:")

print(confusion\_matrix(y\_test, y\_pred))

print("\n分类报告:")

print(classification\_report(y\_test, y\_pred))

2.实验一

# 导入必要库

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# 忽略全局警告

import warnings

warnings.filterwarnings("ignore")

# 设置显示负号

import matplotlib

matplotlib.rcParams['axes.unicode\_minus'] = False

# 读取数据

file\_path = "Obesity\_Dataset.xlsx"

data = pd.read\_excel(file\_path)

# 数据预处理

# 将目标变量"Class"编码为数字

label\_encoder = LabelEncoder()

data['Class'] = label\_encoder.fit\_transform(data['Class'])

# 检查是否有缺失值

if data.isnull().sum().sum() > 0:

print("数据包含缺失值，请处理！")

else:

print("数据无缺失值。")

# 特征和目标分离

X = data.drop(columns=['Class'])

y = data['Class']

# 将分类变量编码为数值

categorical\_columns = X.select\_dtypes(include=['object']).columns

for col in categorical\_columns:

X[col] = label\_encoder.fit\_transform(X[col])

# 数据标准化

scaler = StandardScaler()

X = scaler.fit\_transform(X)

# 划分训练集和测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# 支持向量机分类

svm\_model = SVC(kernel='linear', random\_state=42)

svm\_model.fit(X\_train, y\_train)

svm\_predictions = svm\_model.predict(X\_test)

# 决策树分类

tree\_model = DecisionTreeClassifier(random\_state=42)

tree\_model.fit(X\_train, y\_train)

tree\_predictions = tree\_model.predict(X\_test)

# 评估支持向量机

print("支持向量机分类结果：")

print("混淆矩阵：")

print(confusion\_matrix(y\_test, svm\_predictions))

print("\n分类报告：")

print(classification\_report(y\_test, svm\_predictions))

# 评估决策树

print("决策树分类结果：")

print("混淆矩阵：")

print(confusion\_matrix(y\_test, tree\_predictions))

print("\n分类报告：")

print(classification\_report(y\_test, tree\_predictions))

# 绘制混淆矩阵热图函数

def plot\_confusion\_matrix(y\_true, y\_pred, title):

cm = confusion\_matrix(y\_true, y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=label\_encoder.classes\_, yticklabels=label\_encoder.classes\_)

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title(title)

plt.show()

# SVM混淆矩阵可视化

plot\_confusion\_matrix(y\_test, svm\_predictions, "SVM Confusion Matrix")

# 决策树混淆矩阵可视化

plot\_confusion\_matrix(y\_test, tree\_predictions, "Decision Tree Confusion Matrix")

# 绘制特征重要性函数

def plot\_feature\_importance(feature\_names, importances, title):

plt.figure(figsize=(10, 6))

sorted\_indices = np.argsort(importances)[::-1]

plt.bar(range(len(importances)), importances[sorted\_indices], align="center")

plt.xticks(range(len(importances)), np.array(feature\_names)[sorted\_indices], rotation=90)

plt.title(title)

plt.tight\_layout()

plt.show()

# 决策树特征重要性可视化

if hasattr(tree\_model, "feature\_importances\_"):

feature\_names = data.drop(columns=['Class']).columns

plot\_feature\_importance(feature\_names, tree\_model.feature\_importances\_,

"Decision Tree Feature Importance")

3.实验二

# 导入必要库

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# 忽略全局警告

import warnings

warnings.filterwarnings("ignore")

# 设置显示负号

import matplotlib

matplotlib.rcParams['axes.unicode\_minus'] = False

# 读取数据

file\_path = "Obesity\_Dataset.xlsx"

data = pd.read\_excel(file\_path)

# 数据预处理

# 将目标变量"Class"编码为数字

label\_encoder = LabelEncoder()

data['Class'] = label\_encoder.fit\_transform(data['Class'])

# 检查是否有缺失值

if data.isnull().sum().sum() > 0:

print("数据包含缺失值，请处理！")

else:

print("数据无缺失值。")

# 特征和目标分离

X = data.drop(columns=['Class'])

y = data['Class']

# 将分类变量编码为数值

categorical\_columns = X.select\_dtypes(include=['object']).columns

for col in categorical\_columns:

X[col] = label\_encoder.fit\_transform(X[col])

# 数据标准化

scaler = StandardScaler()

X = scaler.fit\_transform(X)

# 划分训练集和测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# 支持向量机分类

svm\_model = SVC(kernel='linear', random\_state=42)

svm\_model.fit(X\_train, y\_train)

svm\_predictions = svm\_model.predict(X\_test)

# 决策树分类

tree\_model = DecisionTreeClassifier(random\_state=42)

tree\_model.fit(X\_train, y\_train)

tree\_predictions = tree\_model.predict(X\_test)

# 评估支持向量机

print("支持向量机分类结果：")

print("混淆矩阵：")

print(confusion\_matrix(y\_test, svm\_predictions))

print("\n分类报告：")

print(classification\_report(y\_test, svm\_predictions))

# 评估决策树

print("决策树分类结果：")

print("混淆矩阵：")

print(confusion\_matrix(y\_test, tree\_predictions))

print("\n分类报告：")

print(classification\_report(y\_test, tree\_predictions))

# 绘制混淆矩阵热图函数

def plot\_confusion\_matrix(y\_true, y\_pred, title):

cm = confusion\_matrix(y\_true, y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=label\_encoder.classes\_, yticklabels=label\_encoder.classes\_)

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title(title)

plt.show()

# SVM混淆矩阵可视化

plot\_confusion\_matrix(y\_test, svm\_predictions, "SVM Confusion Matrix")

# 决策树混淆矩阵可视化

plot\_confusion\_matrix(y\_test, tree\_predictions, "Decision Tree Confusion Matrix")

# 绘制特征重要性函数

def plot\_feature\_importance(feature\_names, importances, title):

plt.figure(figsize=(10, 6))

sorted\_indices = np.argsort(importances)[::-1]

plt.bar(range(len(importances)), importances[sorted\_indices], align="center")

plt.xticks(range(len(importances)), np.array(feature\_names)[sorted\_indices], rotation=90)

plt.title(title)

plt.tight\_layout()

plt.show()

# 决策树特征重要性可视化

if hasattr(tree\_model, "feature\_importances\_"):

feature\_names = data.drop(columns=['Class']).columns

plot\_feature\_importance(feature\_names, tree\_model.feature\_importances\_,

"Decision Tree Feature Importance")

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

# 忽略全局警告

import warnings

warnings.filterwarnings("ignore")

# 设置显示负号

import matplotlib

matplotlib.rcParams['axes.unicode\_minus'] = False

# 读取数据

file\_path = "cleaned\_merged\_heart\_dataset.csv"

data = pd.read\_csv(file\_path)

# 数据预处理

# 将目标变量"target"编码为数字

label\_encoder = LabelEncoder()

data['target'] = label\_encoder.fit\_transform(data['target'])

# 检查是否有缺失值

if data.isnull().sum().sum() > 0:

print("数据包含缺失值，请处理！")

else:

print("数据无缺失值。")

# 分割数据集为特征和目标变量

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

y = data['target']

# 特征缩放

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# 分割数据集为训练集和测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 决策树分类器

dt\_classifier = DecisionTreeClassifier(random\_state=42)

dt\_classifier.fit(X\_train\_scaled, y\_train)

y\_pred\_dt = dt\_classifier.predict(X\_test\_scaled)

# SVM分类器

svm\_classifier = SVC(kernel='linear', random\_state=42)

svm\_classifier.fit(X\_train\_scaled, y\_train)

y\_pred\_svm = svm\_classifier.predict(X\_test\_scaled)

# 评估模型

print("决策树分类器评估报告：")

print(classification\_report(y\_test, y\_pred\_dt))

print("决策树分类器混淆矩阵：")

print(confusion\_matrix(y\_test, y\_pred\_dt))

print("\nSVM分类器评估报告：")

print(classification\_report(y\_test, y\_pred\_svm))

print("SVM分类器混淆矩阵：")

print(confusion\_matrix(y\_test, y\_pred\_svm))

# 可视化混淆矩阵

def plot\_confusion\_matrix(cm, title='Confusion Matrix', cmap=plt.cm.Blues):

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(set(y)))

plt.xticks(tick\_marks, set(y), rotation=45)

plt.yticks(tick\_marks, set(y))

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

plt.show()

plot\_confusion\_matrix(confusion\_matrix(y\_test, y\_pred\_dt), title='Decision Tree Confusion Matrix')

plot\_confusion\_matrix(confusion\_matrix(y\_test, y\_pred\_svm), title='SVM Confusion Matrix')

# 提取特征重要性

feature\_importances = pd.Series(dt\_classifier.feature\_importances\_, index=X.columns).sort\_values(ascending=False)

# 可视化特征重要性

plt.figure(figsize=(10, 6))

sns.barplot(x=feature\_importances, y=feature\_importances.index)

plt.title('Feature Importances from Decision Tree')

plt.xlabel('Importance')

plt.ylabel('Feature')

plt.show()