KNN模型实践

助教：李兵

5222023330043@smail.nju.edu

作业介绍

1.了解KNN算法的原理，实现KNN模型，使用iris数据集进行训练与预测并可视化。

2.探索不同的距离算法对模型精度的影响。

3.探究k值对knn模型精度的影响。

导入相关包

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
from sklearn.datasets import load_iris
```

距离函数

提供了欧几里得距离

In [2]:
```
def euclidean_distance(point1, point2):
    return np.sqrt(np.sum((point1 - point2) ** 2))

def manhattan_distance():
    return 0

def minkowski_distance():
    return 0
```

模型的实现和数据集的准备

使用了iris数据集

In [3]:
```
class KNN:
    def __init__(self, k=3):
        self.k = k

    def fit(self, X_train, y_train):
        self.X_train = X_train
        self.y_train = y_train

    def predict(self, X_test):
        predictions = [self._predict(x) for x in X_test]
        return predictions

    def _predict(self, x):
```

```
        distances = [euclidean_distance(x, x_train) for x_train in self.X_train]
        k_indices = np.argsort(distances)[:self.k]
        k_nearest_labels = [self.y_train[i] for i in k_indices]
        most_common = Counter(k_nearest_labels).most_common(1)
        return most_common[0][0]

def load_iris_data():
    iris = load_iris()
    X = iris.data
    y = iris.target
    train_size = int(0.8 * len(X))
    X_train, X_test = X[:train_size], X[train_size:]
    y_train, y_test = y[:train_size], y[train_size:]
    return X_train, y_train, X_test, y_test
```

评估函数与数据集可视化函数

可视化两个特征

In [4]:
```
# 评估函数与数据的可视化（选取两个特征进行可视化）
def evaluate(y_true, y_pred):
    correct = np.sum(y_true == y_pred)
    accuracy = correct / len(y_true)
    return accuracy

def plot_data(X, y, title="Data"):
    plt.figure(figsize=(8, 6))
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.RdYlBu, edgecolor='k', s=50)
    plt.title(title)
    plt.xlabel("Feature 1 (Sepal Length)")
    plt.ylabel("Feature 2 (Sepal Width)")
    plt.colorbar()
    plt.show()
```

预测结果可视化函数

In [5]:
```
# 函数：可视化KNN预测的结果
def plot_knn_predictions(X_train, y_train, X_test, y_pred):
    plt.figure(figsize=(8, 6))
    plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=plt.cm.RdYlBu, edg
    plt.scatter(X_test[:, 0], X_test[:, 1], c=y_pred, cmap=plt.cm.Paired, edgecc
    plt.title("KNN Predictions vs Train Data")
    plt.xlabel("Feature 1 (Sepal Length)")
    plt.ylabel("Feature 2 (Sepal Width)")
    plt.legend()
    plt.colorbar()
    plt.show()
```

选取k值

可以尝试选取不同的k，探究一下k过大和过小对精度的影响

In [6]:
```
# 加载数据，创建模型
X_train, y_train, X_test, y_test = load_iris_data()
knn = KNN(k=5)
```

In [7]:
```
# 训练 KNN 模型，进行预测
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

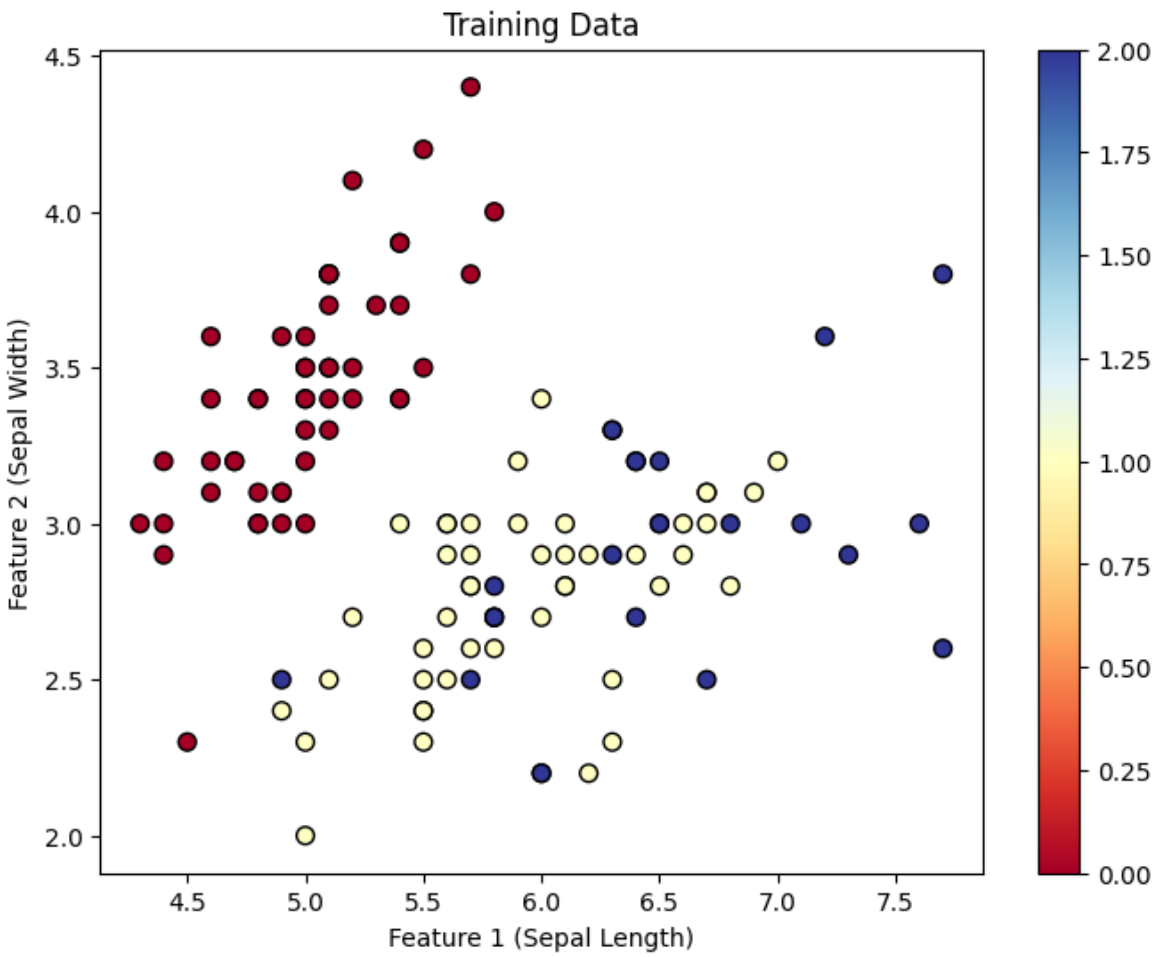
In [8]:
```
# 输出预测结果
print("Predictions: ", y_pred)
```

```
Predictions:  [np.int64(2), np.int64(2), np.int64(2), np.int64(1), np.int64(2), np.int64(2), np.int64(1), np.int64(1), np.int64(2), np.int64(2), np.int64(2), np.int64(2), np.int64(2), np.int64(1), np.int64(2), np.int64(2), np.int64(2), np.int64(2), np.int64(1), np.int64(2), np.int64(2), np.int64(2), np.int64(2), np.int64(2), np.int64(2), np.int64(2), np.int64(2), np.int64(2), np.int64(2), np.int64(1)]
```

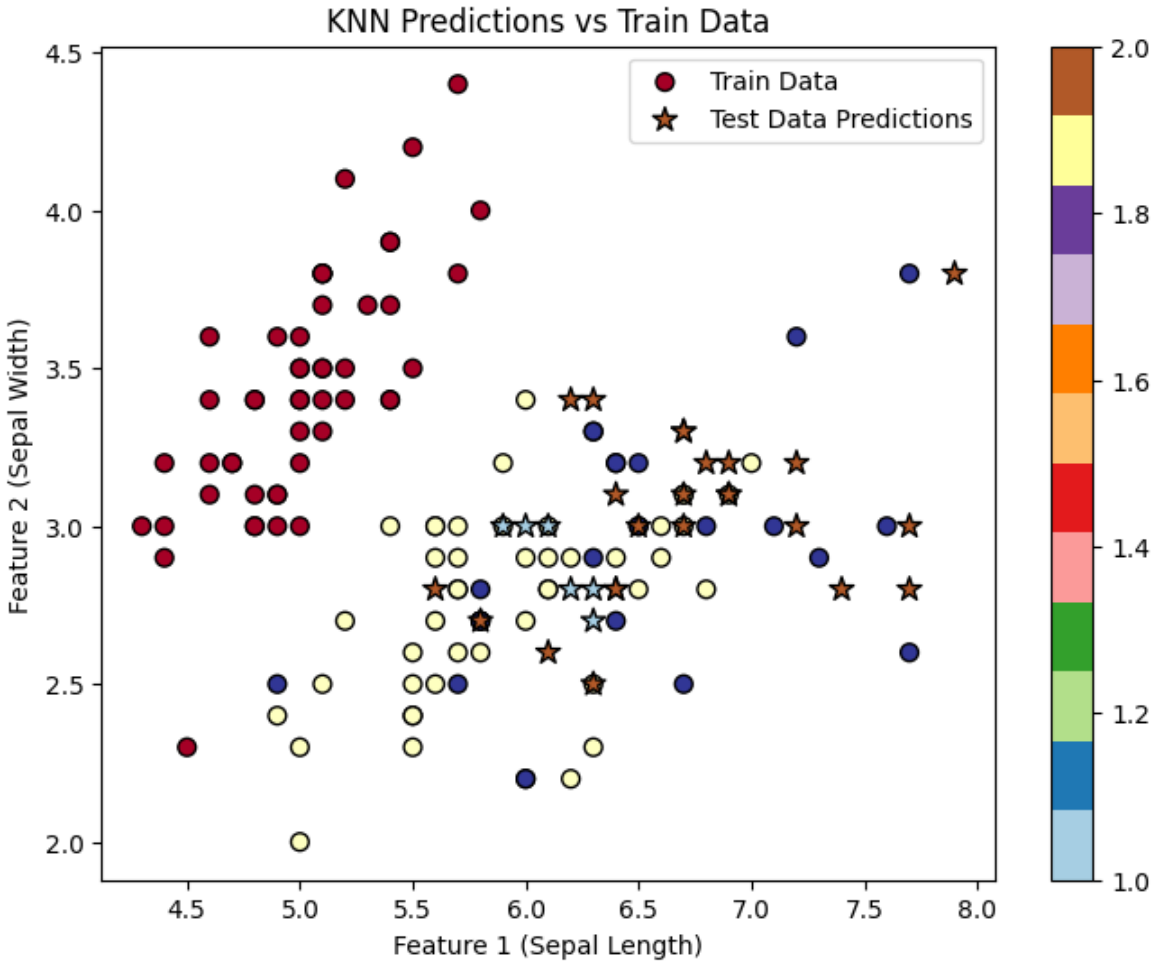
In [9]:
```
# 评估模型的准确率
accuracy = evaluate(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
```

```
Accuracy: 80.00%
```

In [10]:
```
# 可视化训练数据
plot_data(X_train, y_train, title="Training Data")
```



In [11]:
```
# 可视化KNN预测结果
plot_knn_predictions(X_train, y_train, X_test, y_pred)
```



思考

KNN 算法是一个基于实例的学习方法，尽管其简单且直观，但在大规模数据集和高维空间中，计算开销会迅速增加，且可能面临维度灾难等问题。为了提升其性能，可以考虑k值最优化，距离度量算法改进等。

（1）补全曼哈顿距离（ Manhattan Distance）和闵可夫斯基距离（Minkowski Distance）代码，并可视化最终结果（附上所取的k值）？

（2）在不同k值下比较得到的结果，回答探索最优k值有哪些方法并进行实现（一种即可）？

（3）距离的计算是否一定是确定的？即有没有可能对于不同的数据存在动态的距离度量方法（回答即可）？