2.0数据准备

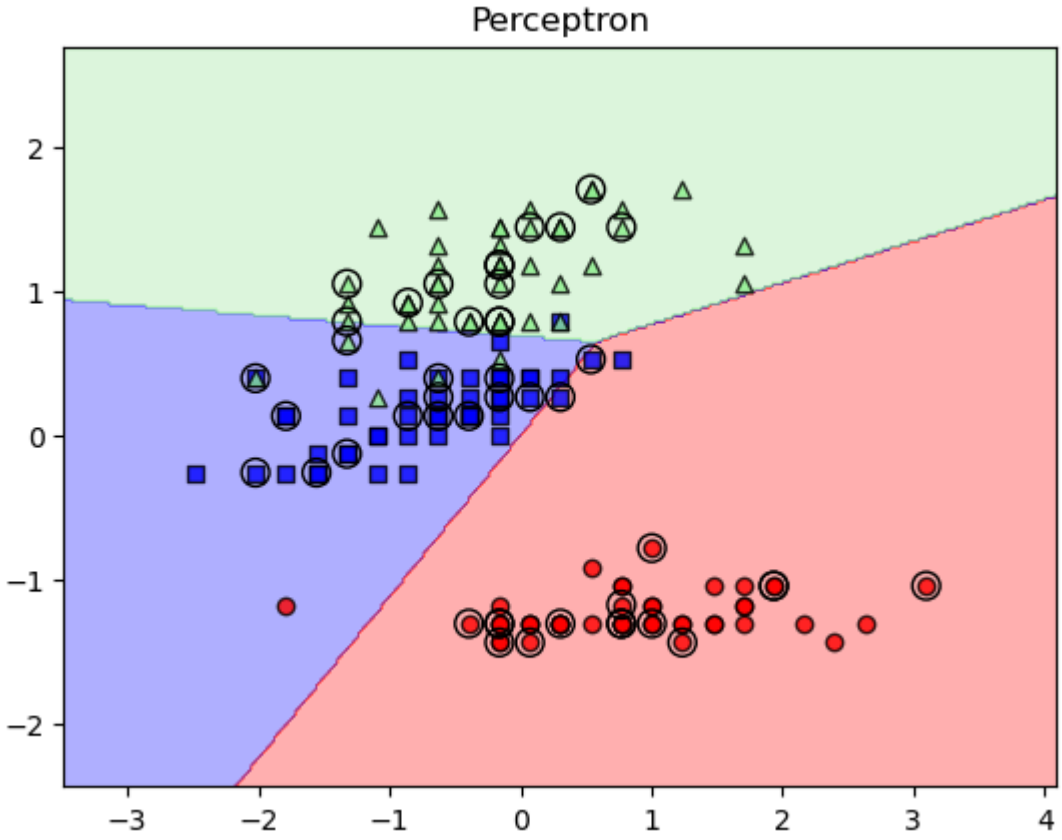
In [9]:
```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# 1. 加载数据 [1]
iris = datasets.load_iris()
X = iris.data[:, [1, 3]]
y = iris.target

# 2. 分割训练集和测试集 (30%测试) [3, 4]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=1, stratify=y)

# 3. 特征标准化（对感知器、逻辑回归、SVM和KNN至关重要） [5, 6]
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

# 可视化工具函数：绘制决策区域 [2, 7]
from matplotlib.colors import ListedColormap
def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):
    markers = ('o', 's', '^', 'v', '<')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    lab = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    lab = lab.reshape(xx1.shape)
    plt.contourf(xx1, xx2, lab, alpha=0.3, cmap=cmap)
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1], alpha=0.8, c=colors[idx],
                    marker=markers[idx], label=f'Class {cl}', edgecolor='black')
    if test_idx:
        X_test, y_test = X[test_idx, :], y[test_idx]
        plt.scatter(X_test[:, 0], X_test[:, 1], c='none', edgecolor='black',
                    alpha=1.0, linewidth=1, marker='o', s=100, label='Test set')

X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
```

2.1sk-learn 感知机

In [10]:
```
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score

# 训练感知器 [8]
```

```
ppn = Perceptron(eta0=0.1, random_state=1)
ppn.fit(X_train_std, y_train)

# 预测与评估 [8, 10]
y_pred = ppn.predict(X_test_std)
print(f'Accuracy: {accuracy_score(y_test, y_pred):.3f}')

# 可视化 [7]
plot_decision_regions(X_combined_std, y_combined, classifier=ppn, test_idx=range
plt.title('Perceptron')
plt.show()
```

```
Accuracy: 0.911
```

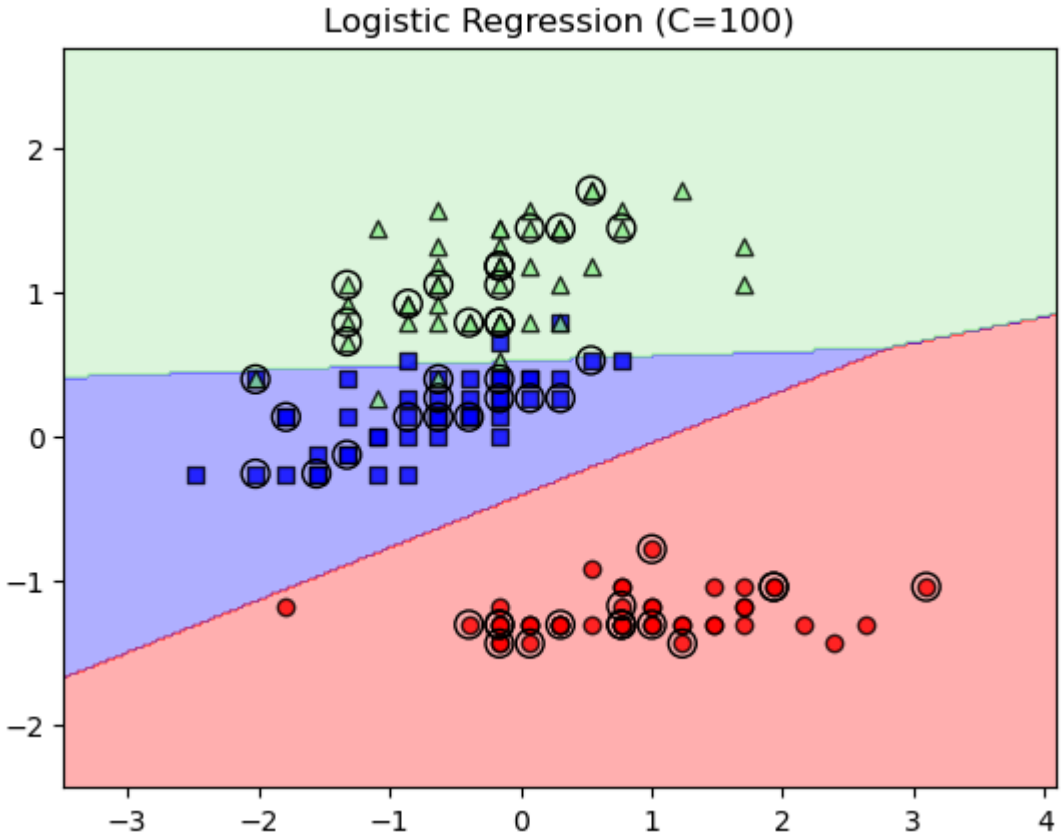


2.2逻辑回归和正则化

In [11]:

```
from sklearn.linear_model import LogisticRegression

# 训练逻辑回归（使用 OvR 处理多分类） [14]
lr = LogisticRegression(C=100.0, solver='lbfgs', multi_class='ovr')
lr.fit(X_train_std, y_train)

# 预测概率示例 [15]
prob = lr.predict_proba(X_test_std[:3, :])
print(f'Probabilities of first 3 samples:\n{prob}')

# 可视化决策区域 [14]
plot_decision_regions(X_combined_std, y_combined, classifier=lr, test_idx=range(
plt.title('Logistic Regression (C=100)')
plt.show()
```

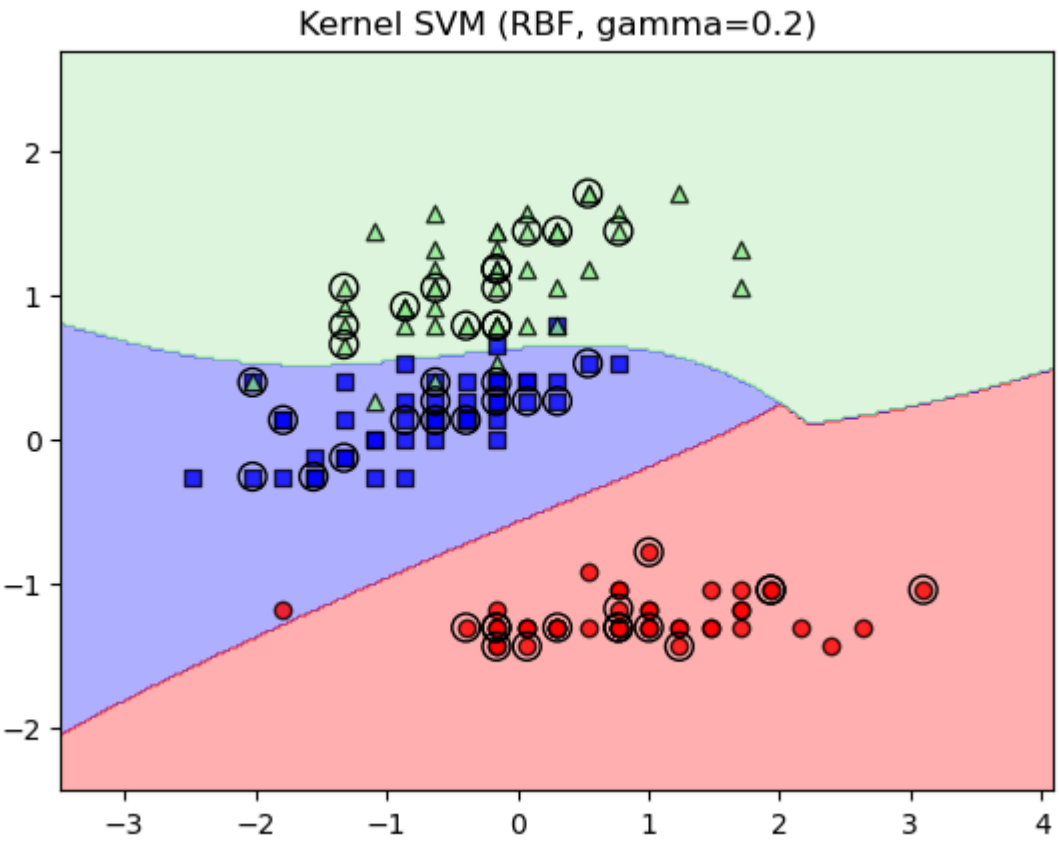
```
d:\anaconda\envs\torch\Lib\site-packages\sklearn\linear_model\_logistic.py:1256:
FutureWarning: 'multi_class' was deprecated in version 1.5 and will be removed in
1.7. Use OneVsRestClassifier(LogisticRegression(..)) instead. Leave it to its def
ault value to avoid this warning.
  warnings.warn(
Probabilities of first 3 samples:
[[1.36950504e-07 8.53967091e-02 9.14603154e-01]
 [9.74933685e-01 2.50663137e-02 1.03471668e-09]
 [8.86527759e-01 1.13472239e-01 1.98375424e-09]]
```



2.3支持向量机和核方法

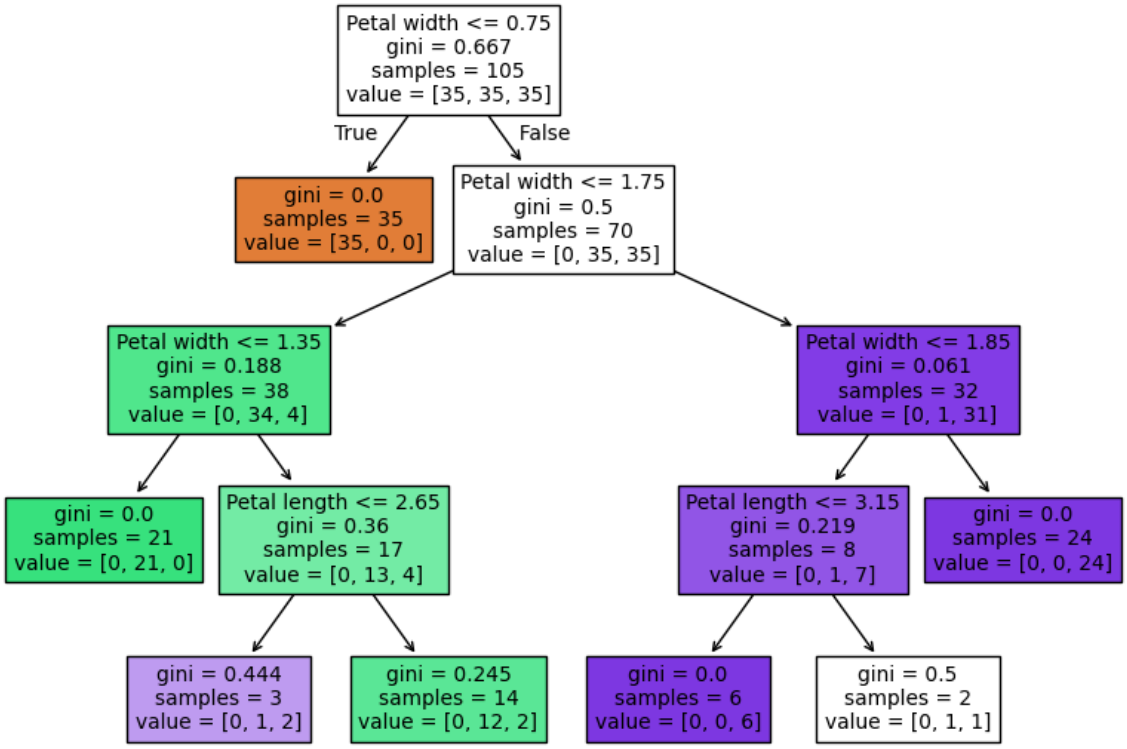
In [12]:

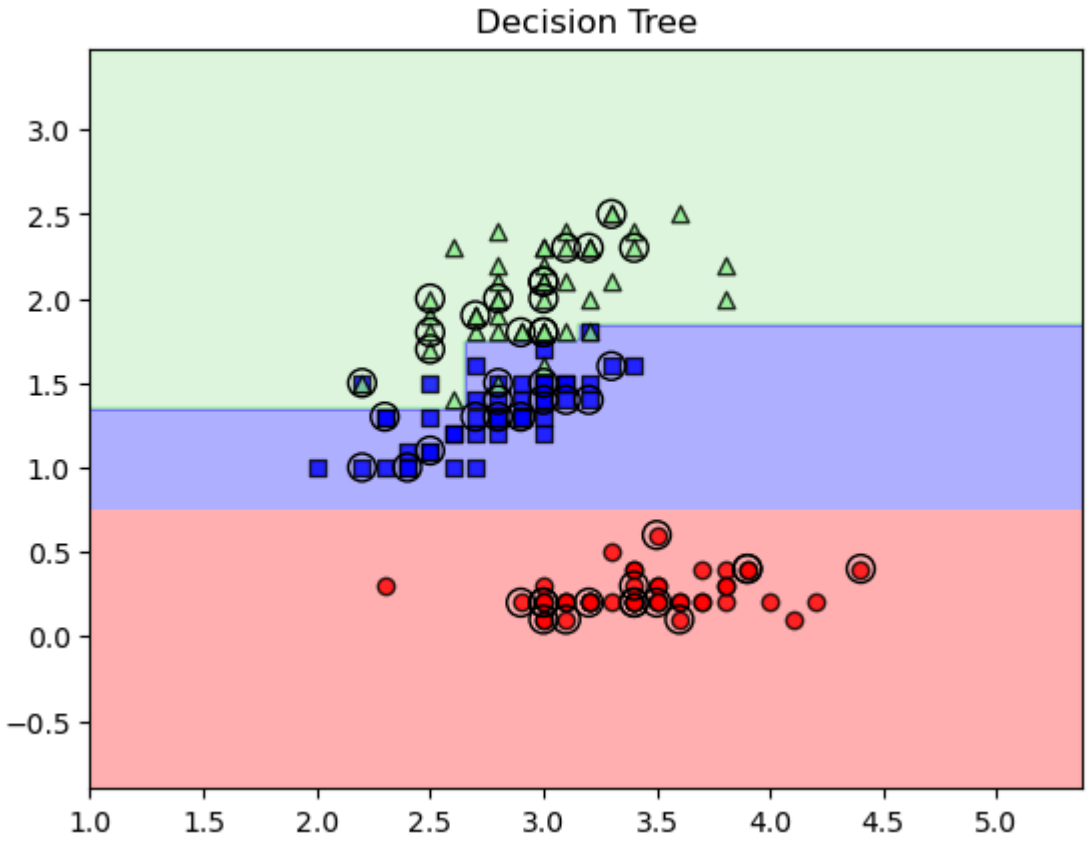
```python
from sklearn.svm import SVC
# 1. 线性 SVM [19]
svm_linear = SVC(kernel='linear', C=1.0, random_state=1)
svm_linear.fit(X_train_std, y_train)

# 2. 核 SVM (使用 RBF 核处理非线性) [18, 20]
# gamma 参数控制“高斯球”的影响范围，gamma 越大，决策边界越紧凑 [20, 21]
svm_rbf = SVC(kernel='rbf', random_state=1, gamma=0.2, C=1.0)
svm_rbf.fit(X_train_std, y_train)

# 可视化核 SVM 结果 [20]
plot_decision_regions(X_combined_std, y_combined, classifier=svm_rbf, test_idx=r
plt.title('Kernel SVM (RBF, gamma=0.2)')
plt.show()
```



2.4决策树

In [13]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

# 训练决策树（限制最大深度为4以防过拟合） [23]
tree_model = DecisionTreeClassifier(criterion='gini', max_depth=4, random_state=
tree_model.fit(X_train, y_train) # 决策树可以使用非标准化数据

# 可视化决策树结构 [24]
plt.figure(figsize=(10, 7))
tree.plot_tree(tree_model, filled=True,
               feature_names=['Petal length', 'Petal width'])
plt.show()

# 绘制决策区域 [23]
X_combined_orig = np.vstack((X_train, X_test))
plot_decision_regions(X_combined_orig, y_combined, classifier=tree_model, test_i
plt.title('Decision Tree')
plt.show()
```





2.5KNN

In [14]:

```
from sklearn.neighbors import KNeighborsClassifier

# 训练 KNN (选择 5 个邻居, p=2 表示欧式距离) [6, 27]
knn = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski')
knn.fit(X_train_std, y_train)
```

```
# 可视化 [27]
plot_decision_regions(X_combined_std, y_combined, classifier=knn, test_idx=range
plt.title('KNN (k=5)')
plt.show()
```