```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
from sklearn.linear_model import LogisticRegression

iris = load_iris()
X, y = iris.data, iris.target
feature_names, target_names = iris.feature_names, iris.target_names

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

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
X_scaled = scaler.fit_transform(X)

classifier = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=200)
classifier.fit(X_train_scaled, y_train)
y_pred = classifier.predict(X_test_scaled)

accuracy = accuracy_score(y_test, y_pred)
cv_scores = cross_val_score(classifier, X_scaled, y, cv=5)

print("Logistic Regression Performance:")
print(f"Test Accuracy: {accuracy:.4f}")
print(f"Cross-Validation Accuracy: {cv_scores.mean():.4f} (±{cv_scores.std():.4f})")
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=target_names))

plt.figure(figsize=(8, 6))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=target_names, yticklabels=target_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Logistic Regression')
plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 10))

plt.subplot(2, 3, 1)
x_min, x_max = X_scaled[:, 0].min() - 1, X_scaled[:, 0].max() + 1
y_min, y_max = X_scaled[:, 1].min() - 1, X_scaled[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max, 0.02))

features = [0, 1]
classifier_2d = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=200)
classifier_2d.fit(X_scaled[:, features], y)

Z = classifier_2d.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.RdYlBu)

for i, color in zip(range(3), ['blue', 'red', 'green']):
    idx = np.where(y == i)
    plt.scatter(X_scaled[idx, 0], X_scaled[idx, 1], c=color, label=target_names[i], edgecolor='k', alpha=0.7)

plt.xlabel(f'Scaled {feature_names[0]}')
plt.ylabel(f'Scaled {feature_names[1]}')
plt.title(f'Decision Boundary (features 0 & 1)')
plt.legend()

plt.subplot(2, 3, 2)
features = [0, 2]
classifier_2d = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=200)
classifier_2d.fit(X_scaled[:, features], y)

x_min, x_max = X_scaled[:, features[0]].min() - 1, X_scaled[:, features[0]].max() + 1
y_min, y_max = X_scaled[:, features[1]].min() - 1, X_scaled[:, features[1]].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max, 0.02))

Z = classifier_2d.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.RdYlBu)
```

```
for i, color in zip(range(3), ['blue', 'red', 'green']):
    idx = np.where(y == i)
    plt.scatter(X_scaled[idx, features[0]], X_scaled[idx, features[1]], c=color, label=target_names[i], edgecolor='k',
alpha=0.7)

plt.xlabel(f'Scaled {feature_names[features[0]]}')
plt.ylabel(f'Scaled {feature_names[features[1]]}')
plt.title(f'Decision Boundary (features 0 & 2)')

plt.subplot(2, 3, 3)
features = [0, 3]
classifier_2d = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=200)
classifier_2d.fit(X_scaled[:, features], y)

x_min, x_max = X_scaled[:, features[0]].min() - 1, X_scaled[:, features[0]].max() + 1
y_min, y_max = X_scaled[:, features[1]].min() - 1, X_scaled[:, features[1]].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max, 0.02))

Z = classifier_2d.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.RdYlBu)

for i, color in zip(range(3), ['blue', 'red', 'green']):
    idx = np.where(y == i)
    plt.scatter(X_scaled[idx, features[0]], X_scaled[idx, features[1]], c=color, label=target_names[i], edgecolor='k',
alpha=0.7)

plt.xlabel(f'Scaled {feature_names[features[0]]}')
plt.ylabel(f'Scaled {feature_names[features[1]]}')
plt.title(f'Decision Boundary (features 0 & 3)')

plt.subplot(2, 3, 4)
features = [1, 2]
classifier_2d = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=200)
classifier_2d.fit(X_scaled[:, features], y)

x_min, x_max = X_scaled[:, features[0]].min() - 1, X_scaled[:, features[0]].max() + 1
y_min, y_max = X_scaled[:, features[1]].min() - 1, X_scaled[:, features[1]].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max, 0.02))

Z = classifier_2d.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.RdYlBu)

for i, color in zip(range(3), ['blue', 'red', 'green']):
    idx = np.where(y == i)
    plt.scatter(X_scaled[idx, features[0]], X_scaled[idx, features[1]], c=color, label=target_names[i], edgecolor='k',
alpha=0.7)

plt.xlabel(f'Scaled {feature_names[features[0]]}')
plt.ylabel(f'Scaled {feature_names[features[1]]}')
plt.title(f'Decision Boundary (features 1 & 2)')

plt.subplot(2, 3, 5)
features = [1, 3]
classifier_2d = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=200)
classifier_2d.fit(X_scaled[:, features], y)

x_min, x_max = X_scaled[:, features[0]].min() - 1, X_scaled[:, features[0]].max() + 1
y_min, y_max = X_scaled[:, features[1]].min() - 1, X_scaled[:, features[1]].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max, 0.02))

Z = classifier_2d.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.RdYlBu)

for i, color in zip(range(3), ['blue', 'red', 'green']):
    idx = np.where(y == i)
    plt.scatter(X_scaled[idx, features[0]], X_scaled[idx, features[1]], c=color, label=target_names[i], edgecolor='k',
alpha=0.7)

plt.xlabel(f'Scaled {feature_names[features[0]]}')
plt.ylabel(f'Scaled {feature_names[features[1]]}')
plt.title(f'Decision Boundary (features 1 & 3)')

plt.subplot(2, 3, 6)
features = [2, 3]
```

```
classifier_2d = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=200)
classifier_2d.fit(X_scaled[:, features], y)

x_min, x_max = X_scaled[:, features[0]].min() - 1, X_scaled[:, features[0]].max() + 1
y_min, y_max = X_scaled[:, features[1]].min() - 1, X_scaled[:, features[1]].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max, 0.02))

Z = classifier_2d.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.RdYlBu)

for i, color in zip(range(3), ['blue', 'red', 'green']):
    idx = np.where(y == i)
    plt.scatter(X_scaled[idx, features[0]], X_scaled[idx, features[1]], c=color, label=target_names[i], edgecolor='k',
alpha=0.7)

plt.xlabel(f'Scaled {feature_names[features[0]]}')
plt.ylabel(f'Scaled {feature_names[features[1]]}')
plt.title(f'Decision Boundary (features 2 & 3)')

plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 6))
coef = classifier.coef_
feature_importance = np.abs(coef).sum(axis=0)
feature_importance = 100.0 * (feature_importance / feature_importance.sum())

indices = np.argsort(feature_importance)[::-1]
plt.bar(range(X.shape[1]), feature_importance[indices], align='center')
plt.xticks(range(X.shape[1]), [feature_names[i] for i in indices])
plt.xlabel('Features')
plt.ylabel('Relative Importance (%)')
plt.title('Logistic Regression Feature Importance')
plt.tight_layout()
plt.show()
```

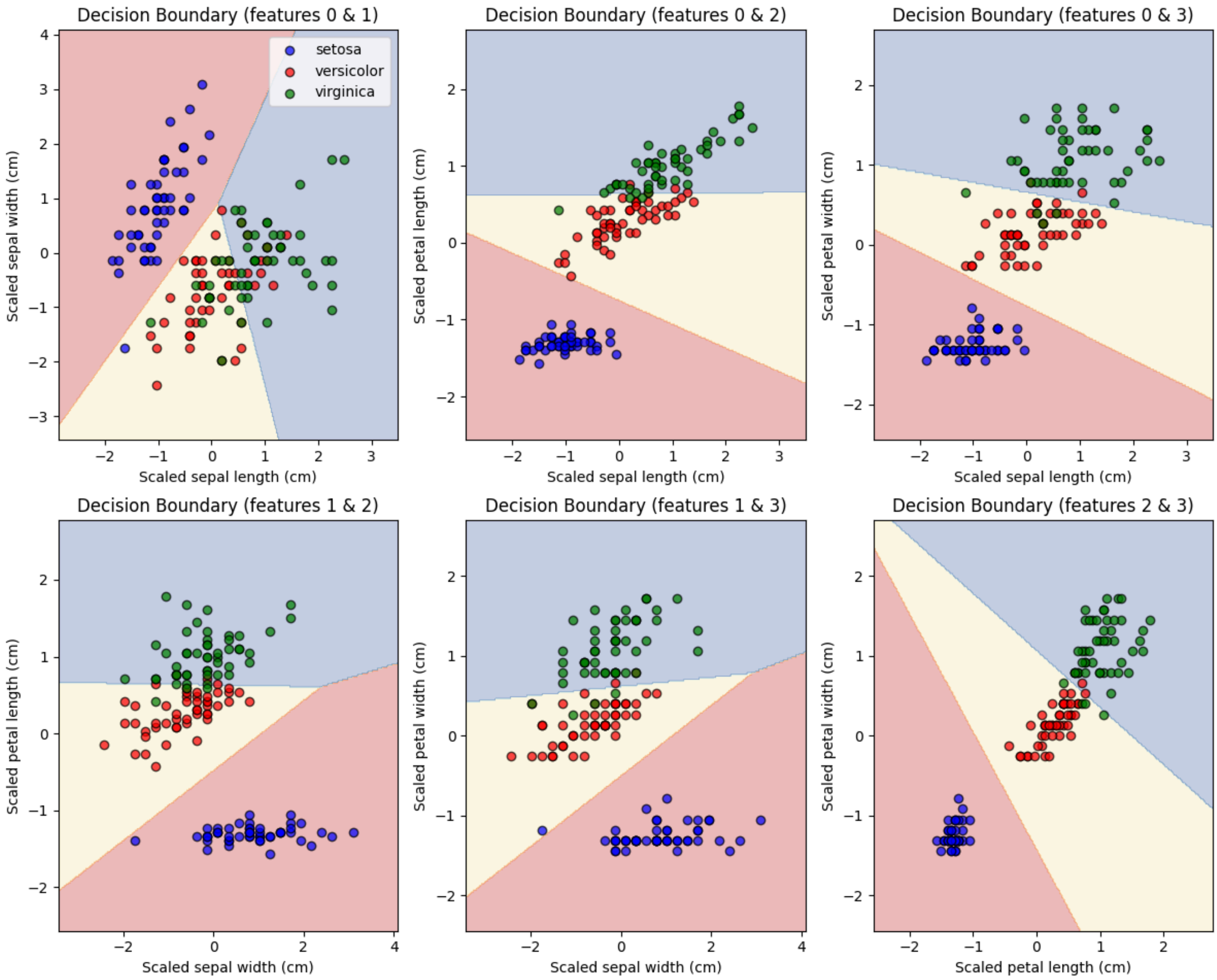

Figure_2-1.png

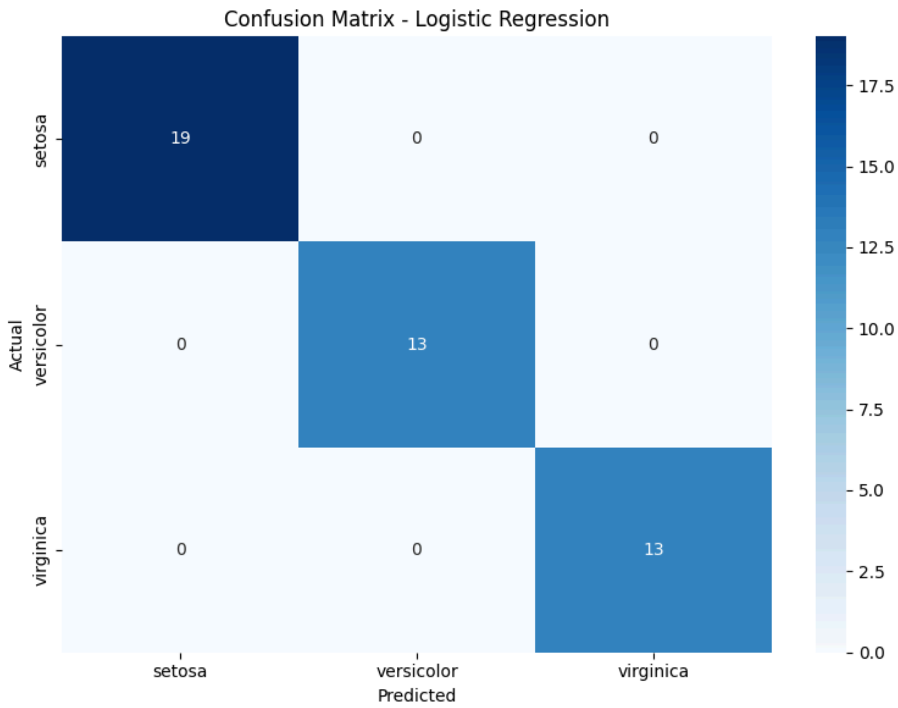

Figure_1-1.png

```
Logistic Regression Performance:
Test Accuracy: 1.0000
Cross-Validation Accuracy: 0.9600 (±0.0389)

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

image-3.png