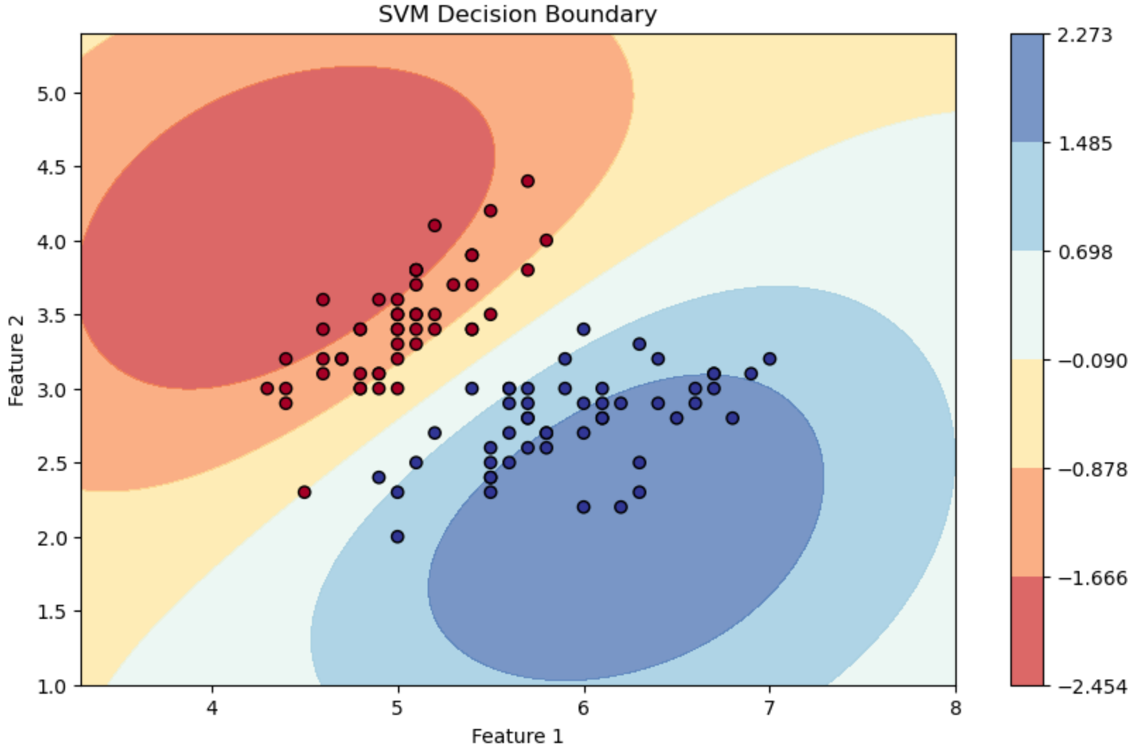
question 2

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the Iris dataset as an example (two classes: setosa and versicolor)
data = datasets.load_iris()
X = data.data[:100, :2]  # Use only the first two features for simplicity
y = data.target[:100]     # Two classes: 0 (setosa) and 1 (versicolor)

# Split the dataset into a training set and a testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Create an SVM classifier with an RBF kernel
svm_classifier = SVC(kernel='rbf', C=1.0)

# Fit the classifier on the training data
svm_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = svm_classifier.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Plot the decision boundary
plt.figure(figsize=(10, 6))

# Create a grid of points to plot the decision boundary
xx, yy = np.meshgrid(np.linspace(X[:, 0].min() - 1, X[:, 0].max() + 1, 100),
                     np.linspace(X[:, 1].min() - 1, X[:, 1].max() + 1, 100))

# Get decision boundary values for each point in the grid
Z = svm_classifier.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot the contour lines and the support vectors
plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), Z.max(), 7), cmap=plt.cr
plt.colorbar()
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.RdYlBu, edgecolor='k')
plt.title("SVM Decision Boundary")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

```
Accuracy: 1.00
```


SVM Decision Boundary
Feature 1
Feature 2
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8
2.273
1.485
0.698
−0.090
−0.878
−1.666
−2.454

In [2]:
```
import numpy as np
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import accuracy_score

# Load a dataset (you can replace this with your own dataset)
data = datasets.load_iris()
X = data.data
y = data.target

# Split the dataset into a training set and a testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Define a parameter grid with different kernel functions and C values
param_grid = {
    'kernel': ['linear', 'rbf', 'poly'],
    'C': [0.1, 1.0, 10.0]
}

# Create an SVM classifier
svm_classifier = SVC()

# Use GridSearchCV to find the best combination of hyperparameters
grid_search = GridSearchCV(svm_classifier, param_grid, cv=5, scoring='accura
grid_search.fit(X_train, y_train)

# Get the best hyperparameters and best accuracy
best_params = grid_search.best_params_
best_accuracy = grid_search.best_score_

print("Best Hyperparameters:")
print(best_params)
print(f"Best Accuracy: {best_accuracy:.2f}")

# Train an SVM classifier with the best hyperparameters
best_svm_classifier = SVC(kernel=best_params['kernel'], C=best_params['C'])
best_svm_classifier.fit(X_train, y_train)

# Evaluate the classifier on the test set
y_pred = best_svm_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on Test Set: {accuracy:.2f}")
```

```
Best Hyperparameters:
{'C': 1.0, 'kernel': 'linear'}
Best Accuracy: 0.96
Accuracy on Test Set: 1.00
```

In [3]:

```
import numpy as np
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix

# Load the Iris dataset as an example (three classes)
data = datasets.load_iris()
data
```

```
ive attributes and the class\n    :Attribute Information:\n        - sep
al length in cm\n        - sepal width in cm\n        - petal length in
cm\n        - petal width in cm\n        - class:\n                - Iri
s-Setosa\n                - Iris-Versicolour\n                - Iris-Vir
ginica\n                \n    :Summary Statistics:\n\n    ==============
==== ==== ======= ===== ====================\n                    Min  M
ax   Mean    SD   Class Correlation\n    ============== ==== ==== ======
= ===== ====================\n    sepal length:   4.3  7.9   5.84   0.83
0.7826\n    sepal width:    2.0  4.4   3.05   0.43   -0.4194\n    petal
length:   1.0  6.9   3.76   1.76    0.9490  (high!)\n    petal width:
0.1  2.5   1.20   0.76    0.9565  (high!)\n    ============== ==== ====
======= ===== ====================\n\n    :Missing Attribute Values: Non
e\n    :Class Distribution: 33.3% for each of 3 classes.\n    :Creator:
R.A. Fisher\n    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
\n    :Date: July, 1988\n\nThe famous Iris database, first used by Sir
R.A. Fisher. The dataset is taken\nfrom Fisher\'s paper. Note that it\'s
the same as in R, but not as in the UCI\nMachine Learning Repository, wh
ich has two wrong data points.\n\nThis is perhaps the best known databas
e to be found in the\npattern recognition literature.  Fisher\'s paper i
```

In [4]:

```
X = data.data
y = data.target

# Split the dataset into a training set and a testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

# Create an SVM classifier with the one-vs-one (OvO) strategy
svm_classifier = SVC(kernel='linear', decision_function_shape='ovo')

# Train the classifier on the training data
svm_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = svm_classifier.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Calculate and print the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 1.00
Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

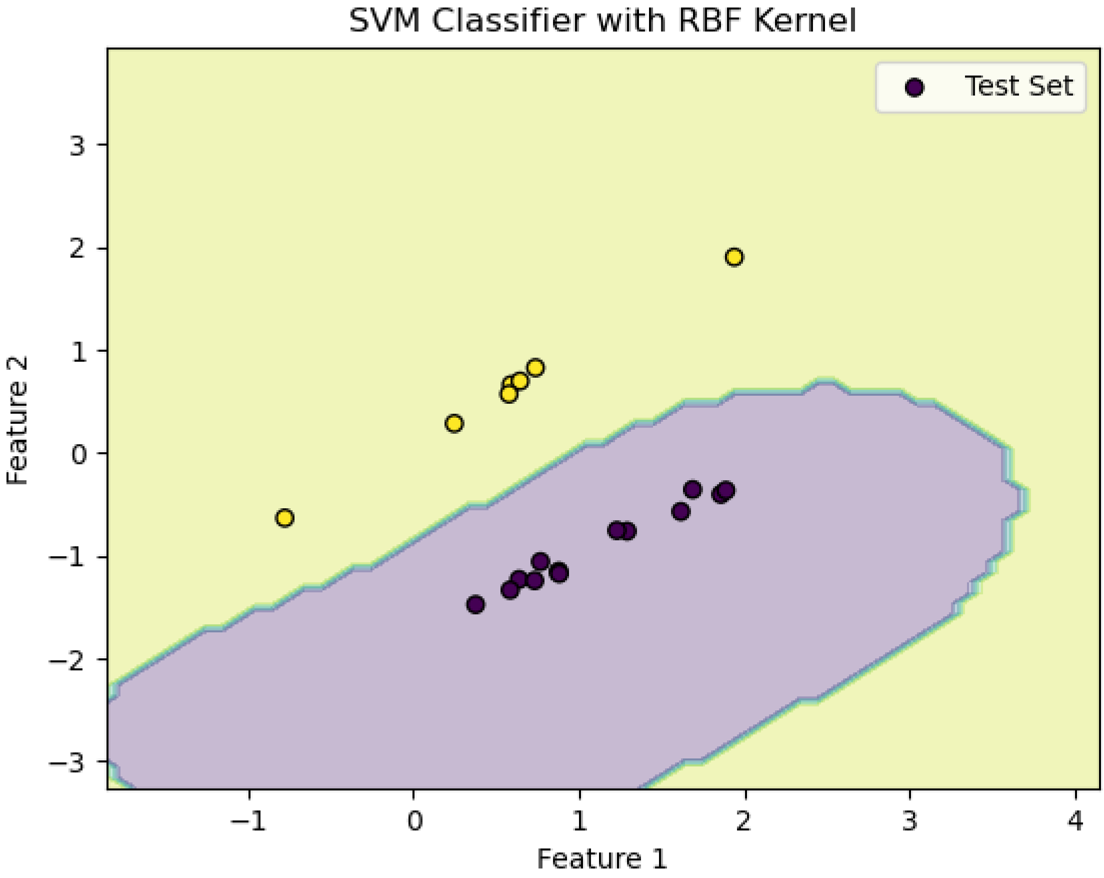
question 1

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Generate a dataset
X, y = datasets.make_classification(n_samples=100, n_features=2, n_classes=

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

# Initialize SVM classifier with RBF kernel
svm_classifier = SVC(kernel='rbf', gamma='auto')

# Train the classifier
svm_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = svm_classifier.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 100.00%
```

In [7]:

```
# Plot decision boundary
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1), np.arange(y_min, y_max, (

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

plt.contourf(xx, yy, Z, alpha=0.3)
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, marker='o', edgecolors='k
plt.title('SVM Classifier with RBF Kernel')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.show()
```



In [ ]: