Support Vector Machines (SVM) Task

1. Load and Prepare Dataset

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import numpy as np

data = load_breast_cancer()
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

2. Train SVM with Linear and RBF Kernel

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score

svm_linear = SVC(kernel='linear', C=1)
svm_linear.fit(X_train, y_train)
y_pred_linear = svm_linear.predict(X_test)

svm_rbf = SVC(kernel='rbf', C=1, gamma='scale')
svm_rbf.fit(X_train, y_train)
y_pred_rbf = svm_rbf.predict(X_test)
```

```
print("Linear Kernel Accuracy:", accuracy_score(y_test, y_pred_linear))
print("RBF Kernel Accuracy:", accuracy_score(y_test, y_pred_rbf))

3. Visualize Decision Boundary
------------------------------
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification

X_synthetic, y_synthetic = make_classification(n_features=2, n_redundant=0, n_informative=2,
                                               n_clusters_per_class=1, n_samples=100, random_state=42)

svm_vis = SVC(kernel='rbf', C=1, gamma='auto')
svm_vis.fit(X_synthetic, y_synthetic)

def plot_decision_boundary(model, X, y):
    h = .02
    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, h),
                         np.arange(y_min, y_max, h))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.8)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k')
    plt.title("SVM with RBF Kernel")
    plt.show()

plot_decision_boundary(svm_vis, X_synthetic, y_synthetic)
```

4. Tune Hyperparameters

```
from sklearn.model_selection import GridSearchCV

param_grid = {'C': [0.1, 1, 10],
              'gamma': ['scale', 0.01, 0.1, 1],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=1, cv=5)
grid.fit(X_train, y_train)

print("Best Parameters:", grid.best_params_)
print("Best Score:", grid.best_score_)
```