

```
# svm_classification.py

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.decomposition import PCA
from sklearn.metrics import classification_report, confusion_matrix
```

```
# Load data
df = pd.read_csv("breast-cancer.csv")
```

```
# Drop unnecessary column
df = df.drop(columns=["id"])
```

```
# Encode target
le = LabelEncoder()
df['diagnosis'] = le.fit_transform(df['diagnosis']) # M=1, B=0
```





```
# Split features and target
X = df.drop(columns=['diagnosis'])
y = df['diagnosis']
```

```
# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# PCA for 2D visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```





```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```
# Train SVM - Linear Kernel
svc_linear = SVC(kernel='linear')
svc_linear.fit(X_train, y_train)
```

  SVC  


```
SVC(kernel='linear')
```

```
# Train SVM - RBF Kernel
svc_rbf = SVC(kernel='rbf')
svc_rbf.fit(X_train, y_train)
```

  SVC  

```
SVC()
```

```
# Evaluate
print("Linear SVM Report:\n", classification_report(y_test, svc_linear.predict(X_test)))
print("RBF SVM Report:\n", classification_report(y_test, svc_rbf.predict(X_test)))
```

 Linear SVM Report:

	precision	recall	f1-score	support
0	0.97	0.96	0.96	71
1	0.93	0.95	0.94	43
accuracy			0.96	114

macro avg	0.95	0.96	0.95	114
weighted avg	0.96	0.96	0.96	114

RBF SVM Report:

	precision	recall	f1-score	support
0	0.97	0.99	0.98	71
1	0.98	0.95	0.96	43

accuracy			0.97	114
macro avg	0.97	0.97	0.97	114
weighted avg	0.97	0.97	0.97	114

```
# Grid search for hyperparameter tuning
params = {'C': [0.1, 1, 10], 'gamma': ['scale', 0.01, 0.1]}
grid = GridSearchCV(SVC(kernel='rbf'), param_grid=params, cv=5)
grid.fit(X_scaled, y)
print("Best Params from Grid Search:", grid.best_params_)
```

Best Params from Grid Search: {'C': 10, 'gamma': 0.01}

```
# Cross-validation score
scores = cross_val_score(SVC(kernel='rbf', **grid.best_params_), X_scaled, y, cv=5)
print("Cross-Validation Accuracy: %.2f%%" % (scores.mean() * 100))
```

Cross-Validation Accuracy: 97.89%

```
def plot_decision_boundary(model, X, y, title):
    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))
    model.fit(X, y)
    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', cmap=plt.cm.coolwarm)
    plt.xlabel('PCA 1')
    plt.ylabel('PCA 2')
    plt.title(title)
    plt.show()

plot_decision_boundary(SVC(kernel='linear'), X_pca, y, "Linear SVM Decision Boundary (PCA)")
plot_decision_boundary(SVC(kernel='rbf'), X_pca, y, "RBF SVM Decision Boundary (PCA)")
```

