In [1]:
```
import time
import seaborn as sns
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn import svm
```

In [ ]:
```
```

In [4]:
```
# load the data
# split the data into features and target variables
print ('#####...loading training data...####')
X_train = pd.read_excel('C:\\Users\\QuickPass\\Documents\\ML\\Sorted\\X_train.xlsx')
y_train = pd.read_excel('C:\\Users\\QuickPass\\Documents\\ML\\Sorted\\y_train.xlsx').values.ravel()

print ('#####...loading validation data...####')
X_valid = pd.read_excel('C:\\Users\\QuickPass\\Documents\\ML\\Sorted\\X_val.xlsx')
y_valid = pd.read_excel('C:\\Users\\QuickPass\\Documents\\ML\\Sorted\\y_val.xlsx').values.ravel()

print ('#####...loading training data...####')
X_test = pd.read_excel('C:\\Users\\QuickPass\\Documents\\ML\\Sorted\\X_test.xlsx')
y_test = pd.read_excel('C:\\Users\\QuickPass\\Documents\\ML\\Sorted\\y_test.xlsx').values.ravel()

print ("#####.... data subsets are ready for feature extraction...#####")
```

```
#####...loading training data...####
#####...loading validation data...####
#####...loading training data...####
#####.... data subsets are ready for feature extraction...#####
```

In [ ]:
```
```

In [5]:
```
# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_valid = scaler.transform(X_valid)
X_test = scaler.transform(X_test)

print("#####... data standardization finished! ...#####")
```

```
#####... data standardization finished! ...#####
```

In [ ]:

In [7]:
```
# Create an SVM classifier object
svc = svm.SVC(kernel='rbf')

# Define the parameters for the SVM model
C_values = [0.1, 1, 10, 100]
gamma_values = [0.1, 0.01, 0.001, 0.0001]
parameters = {'C': C_values, 'gamma': gamma_values}


# Create a GridSearchCV object
clf = GridSearchCV(svc, parameters)

# Fit the GridSearchCV object on the training set
clf.fit(X_train, y_train)
```

Out[7]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 1, 10, 100],
                         'gamma': [0.1, 0.01, 0.001, 0.0001]})
```

In [ ]:

In [10]:
```
# Print the best estimator and its score on the validation set
print('Best estimator:', clf.best_estimator_)
print('Best parameters:', clf.best_params_)
print('Accuracy on validation set:', clf.score(X_valid, y_valid))

# Calculate the score on the testing set
test_score = clf.score(X_test, y_test)
print('Accuracy on testing set:', test_score)
```

```
Best estimator: SVC(C=10, gamma=0.1)
Best parameters: {'C': 10, 'gamma': 0.1}
Accuracy on validation set: 0.9957058507783145
Accuracy on testing set: 0.994990159241367
```

In [11]:
```
# Extract the results of the grid search
cv_results = clf.cv_results_
print (cv_results)
```

```
{'mean_fit_time': array([  78.71642914,   30.53975382,   49.35579076,   49.84226031,
        119.34224544,   18.13125806,   49.84652009, 4458.75592742,
        124.58609715,   10.07971325,   27.31386528,   44.57811775,
        127.04149475,    8.61780567,   24.89608836,   39.22469931]), 'std_fit_time': array([3.31193060e-01, 2.55527152e
-01, 3.26104797e+00, 1.00408779e+00,
       5.66006205e+00, 1.65562374e+00, 4.77722487e+00, 8.80005525e+03,
       2.56032857e+00, 2.74370494e-01, 6.64489211e-01, 5.63556197e-01,
       4.29871919e+00, 4.45560613e-01, 9.35603874e-01, 3.63002688e-01]), 'mean_score_time': array([ 9.9548799 ,  8.8568
0532, 11.86098094, 11.70421886,  8.65075054,
        5.19948764, 13.02089415, 15.18981276,  7.97716942,  2.11670537,
        7.46807208, 11.55917978,  8.36757846,  1.62389131,  4.42425799,
       10.48122468]), 'std_score_time': array([0.30601711, 0.3026023 , 0.28636663, 0.87546882, 0.63610779,
       0.98940188, 0.4608174 , 0.18293152, 0.71654038, 0.09670918,
       0.28216297, 0.83215644, 0.49731784, 0.12470958, 0.27753646,
       0.17423262]), 'param_C': masked_array(data=[0.1, 0.1, 0.1, 0.1, 1, 1, 1, 1, 10, 10, 10, 10, 100,
                  100, 100, 100],
             mask=[False, False, False, False, False, False, False, False,
                  False, False, False, False, False, False, False, False],
       fill_value='?',
            dtype=object), 'param_gamma': masked_array(data=[0.1, 0.01, 0.001, 0.0001, 0.1, 0.01, 0.001, 0.0001,
                  0.1, 0.01, 0.001, 0.0001, 0.1, 0.01, 0.001, 0.0001],
             mask=[False, False, False, False, False, False, False, False,
                  False, False, False, False, False, False, False, False],
       fill_value='?',
            dtype=object), 'params': [{'C': 0.1, 'gamma': 0.1}, {'C': 0.1, 'gamma': 0.01}, {'C': 0.1, 'gamma': 0.001},
{'C': 0.1, 'gamma': 0.0001}, {'C': 1, 'gamma': 0.1}, {'C': 1, 'gamma': 0.01}, {'C': 1, 'gamma': 0.001}, {'C': 1, 'gamm
a': 0.0001}, {'C': 10, 'gamma': 0.1}, {'C': 10, 'gamma': 0.01}, {'C': 10, 'gamma': 0.001}, {'C': 10, 'gamma': 0.0001},
{'C': 100, 'gamma': 0.1}, {'C': 100, 'gamma': 0.01}, {'C': 100, 'gamma': 0.001}, {'C': 100, 'gamma': 0.0001}], 'split0_
test_score': array([0.95720889, 0.91680334, 0.78887729, 0.74996273, 0.99567616,
       0.98389742, 0.85626957, 0.77918593, 0.99657075, 0.99433428,
       0.93275682, 0.81839869, 0.99657075, 0.99537796, 0.98151185,
       0.85716416]), 'split1_test_score': array([0.95377963, 0.91859252, 0.78992098, 0.74996273, 0.99403608,
       0.98240644, 0.85343671, 0.77948412, 0.99507977, 0.9928433 ,
       0.9272402 , 0.81616222, 0.99507977, 0.99343969, 0.97733711,
       0.85776055]), 'split2_test_score': array([0.95929019, 0.91619445, 0.79182821, 0.75007456, 0.99552639,
       0.98285118, 0.85505517, 0.78317924, 0.99597375, 0.99254399,
       0.92663287, 0.81956457, 0.99597375, 0.99522815, 0.97837757,
       0.85833582]), 'split3_test_score': array([0.95407098, 0.92036982, 0.78988965, 0.75007456, 0.99358783,
       0.98404414, 0.8581867 , 0.78004772, 0.99478079, 0.99284223,
       0.93677304, 0.82180137, 0.99478079, 0.99492991, 0.97942141,
       0.85997614]), 'split4_test_score': array([0.95869371, 0.9224575 , 0.79063525, 0.74992544, 0.99358783,
       0.98180734, 0.86072174, 0.78198628, 0.99433343, 0.99179839,
       0.93304503, 0.82001193, 0.99433343, 0.99433343, 0.97793021,
       0.86370415]), 'mean_test_score': array([0.95660868, 0.91888353, 0.79023028, 0.75      , 0.99448286,
       0.9830013 , 0.85673398, 0.78077666, 0.9953477 , 0.99287244,
```

```
       0.93128959, 0.81918776, 0.9953477 , 0.99466183, 0.97891563,
       0.85938816]), 'std_test_score': array([2.29527737e-03, 2.30761074e-03, 9.75670557e-04, 6.23801228e-05,
       9.28944420e-04, 8.59358906e-04, 2.52660655e-03, 1.54712865e-03,
       8.13596670e-04, 8.24644730e-04, 3.83106326e-03, 1.86701598e-03,
       8.13596670e-04, 7.08083285e-04, 1.46618359e-03, 2.35270837e-03]), 'rank_test_score': array([ 8, 10, 14, 16,  4,
6, 12, 15,  1,  5,  9, 13,  1,  3,  7, 11])}
```

In [12]:
```
mean_test_scores = cv_results['mean_test_score']
params = cv_results['params']
C_values = [params[i]['C'] for i in range(len(params))]
gamma_values = [params[i]['gamma'] for i in range(len(params))]

print (mean_test_scores)
```

```
[0.95660868 0.91888353 0.79023028 0.75       0.99448286 0.9830013
 0.85673398 0.78077666 0.9953477  0.99287244 0.93128959 0.81918776
 0.9953477  0.99466183 0.97891563 0.85938816]
```

In [ ]:
```

```