HW6

name: Qianhong Yan

USCid: 8257959587

*background

In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

In multilabel classification, the Hamming loss is different from the subset zero-one loss. The zero-one loss considers the entire set of labels for a given sample incorrect if it does entirely match the true set of labels. Hamming loss is more forgiving in that it penalizes the individual labels.

1)select data randomly

In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split

Folder_Path = r'/Users/yqh/Desktop/'
SaveFile_Path = r'/Users/yqh/Desktop'
SaveFile_Name1 = r'training_data.csv'
SaveFile_Name2 = r'testing_data.csv'

data = pd.read_csv(Folder_Path + 'Frogs_MFCCs.csv', header=0)
X_train, X_test = train_test_split(data, test_size=0.3)

X_train.to_csv(Folder_Path + SaveFile_Name1, index=False, header=data.co
lumns)
X_test.to_csv(Folder_Path + SaveFile_Name2, index=False, header=data.col
umns)

print(X_train.info)
print(X_test.info)
print(data.columns)
```

2)normalize data

In [1]:
```
import pandas as pd


Folder_Path = r'/Users/yqh/Desktop/'
SaveFile_Path = r'/Users/yqh/Desktop'
SaveFile_Name1 = r'norm_training_data.csv'
SaveFile_Name2 = r'norm_testing_data.csv'

a = pd.read_csv(Folder_Path + 'testing_data.csv', usecols=range(0, 22),
header=0)
a1 = pd.read_csv(Folder_Path + 'testing_data.csv', usecols=[22], header=
0)
a2 = pd.read_csv(Folder_Path + 'testing_data.csv', usecols=[23], header=
0)
a3 = pd.read_csv(Folder_Path + 'testing_data.csv', usecols=[24], header=
0)
a4 = pd.read_csv(Folder_Path + 'testing_data.csv', usecols=[25], header=
0)
a = (a - a.min()) / (a.max() - a.min())
a.insert(22, 'Family', a1)
a.insert(23, 'Genus', a2)
a.insert(24, 'Species', a3)
a.insert(25, 'RecordID', a4)
a.to_csv(Folder_Path + SaveFile_Name2, index=False, header=a.columns)

b = pd.read_csv(Folder_Path + 'training_data.csv', usecols=range(0, 22),
 header=0)
b1 = pd.read_csv(Folder_Path + 'training_data.csv', usecols=[22], header
=0)
b2 = pd.read_csv(Folder_Path + 'training_data.csv', usecols=[23], header
=0)
b3 = pd.read_csv(Folder_Path + 'training_data.csv', usecols=[24], header
=0)
b4 = pd.read_csv(Folder_Path + 'training_data.csv', usecols=[25], header
=0)
b = (b - b.min()) / (b.max() - b.min())
b.insert(22, 'Family', b1)
b.insert(23, 'Genus', b2)
b.insert(24, 'Species', b3)
b.insert(25, 'RecordID', b4)
b.to_csv(Folder_Path + SaveFile_Name1, index=False, header=a.columns)
```

3)choose parameter using 10-fold cross validation

i)feature: Family--without normalization

In [3]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
import pandas as pd
import numpy as np
import warnings

warnings.filterwarnings('ignore')

Folder_Path = r'/Users/yqh/Desktop/'

a = pd.read_csv(Folder_Path + 'training_data.csv', usecols=range(0, 22),
 header=0)
b = pd.read_csv(Folder_Path + 'training_data.csv', usecols=[22], header=
0)

model = SVC(kernel='rbf', probability=True)
param_grid = {'C': range(1, 20), 'gamma': np.arange(0.1, 2, 0.1)}
grid_search = GridSearchCV(model, param_grid, scoring='accuracy', cv=10,
 n_jobs=1, verbose=1)
grid_search.fit(a, b)
best_parameters = grid_search.best_estimator_.get_params()

means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, grid_search.cv_results_['param
s']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))

for para, val in best_parameters.items():
    print(para, val)
```

```
Fitting 10 folds for each of 361 candidates, totalling 3610 fits
[Parallel(n_jobs=1)]: Done 3610 out of 3610 | elapsed: 37.2min finished
```

```
0.945 (+/-0.020) for {'C': 1, 'gamma': 0.1}
0.960 (+/-0.017) for {'C': 1, 'gamma': 0.2}
0.966 (+/-0.012) for {'C': 1, 'gamma': 0.30000000000000004}
0.970 (+/-0.013) for {'C': 1, 'gamma': 0.4}
0.972 (+/-0.013) for {'C': 1, 'gamma': 0.5}
0.977 (+/-0.013) for {'C': 1, 'gamma': 0.6}
0.981 (+/-0.014) for {'C': 1, 'gamma': 0.7000000000000001}
0.982 (+/-0.012) for {'C': 1, 'gamma': 0.8}
0.984 (+/-0.012) for {'C': 1, 'gamma': 0.9}
0.985 (+/-0.010) for {'C': 1, 'gamma': 1.0}
0.985 (+/-0.010) for {'C': 1, 'gamma': 1.1}
0.986 (+/-0.011) for {'C': 1, 'gamma': 1.2000000000000002}
0.987 (+/-0.011) for {'C': 1, 'gamma': 1.3000000000000003}
0.988 (+/-0.011) for {'C': 1, 'gamma': 1.4000000000000001}
0.988 (+/-0.010) for {'C': 1, 'gamma': 1.5000000000000002}
0.988 (+/-0.010) for {'C': 1, 'gamma': 1.6}
0.988 (+/-0.010) for {'C': 1, 'gamma': 1.7000000000000002}
0.989 (+/-0.010) for {'C': 1, 'gamma': 1.8000000000000003}
0.989 (+/-0.010) for {'C': 1, 'gamma': 1.9000000000000001}
0.956 (+/-0.017) for {'C': 2, 'gamma': 0.1}
0.966 (+/-0.012) for {'C': 2, 'gamma': 0.2}
0.971 (+/-0.012) for {'C': 2, 'gamma': 0.30000000000000004}
0.979 (+/-0.013) for {'C': 2, 'gamma': 0.4}
0.981 (+/-0.013) for {'C': 2, 'gamma': 0.5}
0.983 (+/-0.012) for {'C': 2, 'gamma': 0.6}
0.984 (+/-0.013) for {'C': 2, 'gamma': 0.7000000000000001}
0.985 (+/-0.012) for {'C': 2, 'gamma': 0.8}
0.987 (+/-0.012) for {'C': 2, 'gamma': 0.9}
0.987 (+/-0.011) for {'C': 2, 'gamma': 1.0}
0.988 (+/-0.011) for {'C': 2, 'gamma': 1.1}
0.988 (+/-0.011) for {'C': 2, 'gamma': 1.2000000000000002}
0.988 (+/-0.011) for {'C': 2, 'gamma': 1.3000000000000003}
0.988 (+/-0.011) for {'C': 2, 'gamma': 1.4000000000000001}
0.989 (+/-0.011) for {'C': 2, 'gamma': 1.5000000000000002}
0.989 (+/-0.011) for {'C': 2, 'gamma': 1.6}
0.989 (+/-0.011) for {'C': 2, 'gamma': 1.7000000000000002}
0.990 (+/-0.010) for {'C': 2, 'gamma': 1.8000000000000003}
0.990 (+/-0.009) for {'C': 2, 'gamma': 1.9000000000000001}
0.960 (+/-0.013) for {'C': 3, 'gamma': 0.1}
0.967 (+/-0.011) for {'C': 3, 'gamma': 0.2}
0.978 (+/-0.013) for {'C': 3, 'gamma': 0.30000000000000004}
0.981 (+/-0.012) for {'C': 3, 'gamma': 0.4}
0.983 (+/-0.012) for {'C': 3, 'gamma': 0.5}
0.984 (+/-0.012) for {'C': 3, 'gamma': 0.6}
0.986 (+/-0.010) for {'C': 3, 'gamma': 0.7000000000000001}
0.987 (+/-0.011) for {'C': 3, 'gamma': 0.8}
0.988 (+/-0.011) for {'C': 3, 'gamma': 0.9}
0.988 (+/-0.011) for {'C': 3, 'gamma': 1.0}
0.988 (+/-0.011) for {'C': 3, 'gamma': 1.1}
0.989 (+/-0.010) for {'C': 3, 'gamma': 1.2000000000000002}
0.989 (+/-0.010) for {'C': 3, 'gamma': 1.3000000000000003}
0.990 (+/-0.010) for {'C': 3, 'gamma': 1.4000000000000001}
0.990 (+/-0.010) for {'C': 3, 'gamma': 1.5000000000000002}
0.991 (+/-0.010) for {'C': 3, 'gamma': 1.6}
0.991 (+/-0.010) for {'C': 3, 'gamma': 1.7000000000000002}
0.991 (+/-0.009) for {'C': 3, 'gamma': 1.8000000000000003}
0.991 (+/-0.009) for {'C': 3, 'gamma': 1.9000000000000001}
```

```
0.962 (+/-0.011) for {'C': 4, 'gamma': 0.1}
0.974 (+/-0.013) for {'C': 4, 'gamma': 0.2}
0.979 (+/-0.012) for {'C': 4, 'gamma': 0.30000000000000004}
0.982 (+/-0.013) for {'C': 4, 'gamma': 0.4}
0.984 (+/-0.011) for {'C': 4, 'gamma': 0.5}
0.986 (+/-0.010) for {'C': 4, 'gamma': 0.6}
0.987 (+/-0.011) for {'C': 4, 'gamma': 0.7000000000000001}
0.988 (+/-0.010) for {'C': 4, 'gamma': 0.8}
0.989 (+/-0.011) for {'C': 4, 'gamma': 0.9}
0.989 (+/-0.010) for {'C': 4, 'gamma': 1.0}
0.989 (+/-0.010) for {'C': 4, 'gamma': 1.1}
0.990 (+/-0.009) for {'C': 4, 'gamma': 1.2000000000000002}
0.990 (+/-0.010) for {'C': 4, 'gamma': 1.3000000000000003}
0.991 (+/-0.010) for {'C': 4, 'gamma': 1.4000000000000001}
0.991 (+/-0.010) for {'C': 4, 'gamma': 1.5000000000000002}
0.992 (+/-0.010) for {'C': 4, 'gamma': 1.6}
0.992 (+/-0.010) for {'C': 4, 'gamma': 1.7000000000000002}
0.992 (+/-0.010) for {'C': 4, 'gamma': 1.8000000000000003}
0.992 (+/-0.009) for {'C': 4, 'gamma': 1.9000000000000001}
0.964 (+/-0.011) for {'C': 5, 'gamma': 0.1}
0.977 (+/-0.012) for {'C': 5, 'gamma': 0.2}
0.980 (+/-0.013) for {'C': 5, 'gamma': 0.30000000000000004}
0.983 (+/-0.010) for {'C': 5, 'gamma': 0.4}
0.985 (+/-0.010) for {'C': 5, 'gamma': 0.5}
0.987 (+/-0.010) for {'C': 5, 'gamma': 0.6}
0.988 (+/-0.010) for {'C': 5, 'gamma': 0.7000000000000001}
0.988 (+/-0.011) for {'C': 5, 'gamma': 0.8}
0.989 (+/-0.010) for {'C': 5, 'gamma': 0.9}
0.990 (+/-0.010) for {'C': 5, 'gamma': 1.0}
0.990 (+/-0.010) for {'C': 5, 'gamma': 1.1}
0.991 (+/-0.010) for {'C': 5, 'gamma': 1.2000000000000002}
0.991 (+/-0.010) for {'C': 5, 'gamma': 1.3000000000000003}
0.991 (+/-0.010) for {'C': 5, 'gamma': 1.4000000000000001}
0.992 (+/-0.009) for {'C': 5, 'gamma': 1.5000000000000002}
0.992 (+/-0.009) for {'C': 5, 'gamma': 1.6}
0.992 (+/-0.009) for {'C': 5, 'gamma': 1.7000000000000002}
0.992 (+/-0.009) for {'C': 5, 'gamma': 1.8000000000000003}
0.992 (+/-0.008) for {'C': 5, 'gamma': 1.9000000000000001}
0.964 (+/-0.010) for {'C': 6, 'gamma': 0.1}
0.978 (+/-0.013) for {'C': 6, 'gamma': 0.2}
0.980 (+/-0.012) for {'C': 6, 'gamma': 0.30000000000000004}
0.984 (+/-0.010) for {'C': 6, 'gamma': 0.4}
0.986 (+/-0.009) for {'C': 6, 'gamma': 0.5}
0.987 (+/-0.010) for {'C': 6, 'gamma': 0.6}
0.988 (+/-0.011) for {'C': 6, 'gamma': 0.7000000000000001}
0.989 (+/-0.011) for {'C': 6, 'gamma': 0.8}
0.989 (+/-0.009) for {'C': 6, 'gamma': 0.9}
0.990 (+/-0.010) for {'C': 6, 'gamma': 1.0}
0.991 (+/-0.010) for {'C': 6, 'gamma': 1.1}
0.991 (+/-0.010) for {'C': 6, 'gamma': 1.2000000000000002}
0.991 (+/-0.010) for {'C': 6, 'gamma': 1.3000000000000003}
0.992 (+/-0.009) for {'C': 6, 'gamma': 1.4000000000000001}
0.992 (+/-0.008) for {'C': 6, 'gamma': 1.5000000000000002}
0.992 (+/-0.009) for {'C': 6, 'gamma': 1.6}
0.992 (+/-0.008) for {'C': 6, 'gamma': 1.7000000000000002}
0.992 (+/-0.008) for {'C': 6, 'gamma': 1.8000000000000003}
0.992 (+/-0.008) for {'C': 6, 'gamma': 1.9000000000000001}
```

```
0.965 (+/-0.011) for {'C': 7, 'gamma': 0.1}
0.978 (+/-0.012) for {'C': 7, 'gamma': 0.2}
0.981 (+/-0.011) for {'C': 7, 'gamma': 0.30000000000000004}
0.984 (+/-0.010) for {'C': 7, 'gamma': 0.4}
0.986 (+/-0.009) for {'C': 7, 'gamma': 0.5}
0.988 (+/-0.011) for {'C': 7, 'gamma': 0.6}
0.988 (+/-0.011) for {'C': 7, 'gamma': 0.7000000000000001}
0.989 (+/-0.010) for {'C': 7, 'gamma': 0.8}
0.990 (+/-0.009) for {'C': 7, 'gamma': 0.9}
0.991 (+/-0.010) for {'C': 7, 'gamma': 1.0}
0.991 (+/-0.009) for {'C': 7, 'gamma': 1.1}
0.991 (+/-0.010) for {'C': 7, 'gamma': 1.2000000000000002}
0.991 (+/-0.009) for {'C': 7, 'gamma': 1.3000000000000003}
0.991 (+/-0.009) for {'C': 7, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 7, 'gamma': 1.5000000000000002}
0.992 (+/-0.008) for {'C': 7, 'gamma': 1.6}
0.992 (+/-0.008) for {'C': 7, 'gamma': 1.7000000000000002}
0.991 (+/-0.008) for {'C': 7, 'gamma': 1.8000000000000003}
0.992 (+/-0.008) for {'C': 7, 'gamma': 1.9000000000000001}
0.967 (+/-0.013) for {'C': 8, 'gamma': 0.1}
0.979 (+/-0.011) for {'C': 8, 'gamma': 0.2}
0.981 (+/-0.011) for {'C': 8, 'gamma': 0.30000000000000004}
0.985 (+/-0.011) for {'C': 8, 'gamma': 0.4}
0.987 (+/-0.010) for {'C': 8, 'gamma': 0.5}
0.988 (+/-0.011) for {'C': 8, 'gamma': 0.6}
0.988 (+/-0.011) for {'C': 8, 'gamma': 0.7000000000000001}
0.989 (+/-0.009) for {'C': 8, 'gamma': 0.8}
0.990 (+/-0.010) for {'C': 8, 'gamma': 0.9}
0.991 (+/-0.009) for {'C': 8, 'gamma': 1.0}
0.991 (+/-0.010) for {'C': 8, 'gamma': 1.1}
0.991 (+/-0.009) for {'C': 8, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 8, 'gamma': 1.3000000000000003}
0.991 (+/-0.009) for {'C': 8, 'gamma': 1.4000000000000001}
0.991 (+/-0.008) for {'C': 8, 'gamma': 1.5000000000000002}
0.991 (+/-0.008) for {'C': 8, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 8, 'gamma': 1.7000000000000002}
0.992 (+/-0.008) for {'C': 8, 'gamma': 1.8000000000000003}
0.992 (+/-0.008) for {'C': 8, 'gamma': 1.9000000000000001}
0.970 (+/-0.014) for {'C': 9, 'gamma': 0.1}
0.979 (+/-0.011) for {'C': 9, 'gamma': 0.2}
0.982 (+/-0.011) for {'C': 9, 'gamma': 0.30000000000000004}
0.985 (+/-0.009) for {'C': 9, 'gamma': 0.4}
0.987 (+/-0.011) for {'C': 9, 'gamma': 0.5}
0.987 (+/-0.011) for {'C': 9, 'gamma': 0.6}
0.989 (+/-0.009) for {'C': 9, 'gamma': 0.7000000000000001}
0.990 (+/-0.010) for {'C': 9, 'gamma': 0.8}
0.990 (+/-0.010) for {'C': 9, 'gamma': 0.9}
0.991 (+/-0.010) for {'C': 9, 'gamma': 1.0}
0.991 (+/-0.010) for {'C': 9, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 9, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 9, 'gamma': 1.3000000000000003}
0.991 (+/-0.007) for {'C': 9, 'gamma': 1.4000000000000001}
0.991 (+/-0.008) for {'C': 9, 'gamma': 1.5000000000000002}
0.991 (+/-0.008) for {'C': 9, 'gamma': 1.6}
0.992 (+/-0.008) for {'C': 9, 'gamma': 1.7000000000000002}
0.992 (+/-0.008) for {'C': 9, 'gamma': 1.8000000000000003}
0.992 (+/-0.008) for {'C': 9, 'gamma': 1.9000000000000001}
```

```
0.971 (+/-0.012) for {'C': 10, 'gamma': 0.1}
0.979 (+/-0.011) for {'C': 10, 'gamma': 0.2}
0.982 (+/-0.011) for {'C': 10, 'gamma': 0.30000000000000004}
0.986 (+/-0.009) for {'C': 10, 'gamma': 0.4}
0.987 (+/-0.010) for {'C': 10, 'gamma': 0.5}
0.988 (+/-0.011) for {'C': 10, 'gamma': 0.6}
0.989 (+/-0.009) for {'C': 10, 'gamma': 0.7000000000000001}
0.990 (+/-0.010) for {'C': 10, 'gamma': 0.8}
0.990 (+/-0.010) for {'C': 10, 'gamma': 0.9}
0.991 (+/-0.010) for {'C': 10, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 10, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 10, 'gamma': 1.2000000000000002}
0.991 (+/-0.007) for {'C': 10, 'gamma': 1.3000000000000003}
0.991 (+/-0.007) for {'C': 10, 'gamma': 1.4000000000000001}
0.991 (+/-0.007) for {'C': 10, 'gamma': 1.5000000000000002}
0.991 (+/-0.007) for {'C': 10, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 10, 'gamma': 1.7000000000000002}
0.992 (+/-0.008) for {'C': 10, 'gamma': 1.8000000000000003}
0.992 (+/-0.007) for {'C': 10, 'gamma': 1.9000000000000001}
0.973 (+/-0.010) for {'C': 11, 'gamma': 0.1}
0.980 (+/-0.012) for {'C': 11, 'gamma': 0.2}
0.983 (+/-0.011) for {'C': 11, 'gamma': 0.30000000000000004}
0.986 (+/-0.010) for {'C': 11, 'gamma': 0.4}
0.987 (+/-0.011) for {'C': 11, 'gamma': 0.5}
0.988 (+/-0.010) for {'C': 11, 'gamma': 0.6}
0.989 (+/-0.009) for {'C': 11, 'gamma': 0.7000000000000001}
0.990 (+/-0.010) for {'C': 11, 'gamma': 0.8}
0.990 (+/-0.010) for {'C': 11, 'gamma': 0.9}
0.991 (+/-0.009) for {'C': 11, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 11, 'gamma': 1.1}
0.991 (+/-0.007) for {'C': 11, 'gamma': 1.2000000000000002}
0.991 (+/-0.007) for {'C': 11, 'gamma': 1.3000000000000003}
0.991 (+/-0.007) for {'C': 11, 'gamma': 1.4000000000000001}
0.991 (+/-0.007) for {'C': 11, 'gamma': 1.5000000000000002}
0.991 (+/-0.007) for {'C': 11, 'gamma': 1.6}
0.992 (+/-0.008) for {'C': 11, 'gamma': 1.7000000000000002}
0.992 (+/-0.007) for {'C': 11, 'gamma': 1.8000000000000003}
0.992 (+/-0.007) for {'C': 11, 'gamma': 1.9000000000000001}
0.974 (+/-0.010) for {'C': 12, 'gamma': 0.1}
0.980 (+/-0.012) for {'C': 12, 'gamma': 0.2}
0.983 (+/-0.012) for {'C': 12, 'gamma': 0.30000000000000004}
0.987 (+/-0.010) for {'C': 12, 'gamma': 0.4}
0.987 (+/-0.011) for {'C': 12, 'gamma': 0.5}
0.989 (+/-0.010) for {'C': 12, 'gamma': 0.6}
0.990 (+/-0.010) for {'C': 12, 'gamma': 0.7000000000000001}
0.990 (+/-0.010) for {'C': 12, 'gamma': 0.8}
0.991 (+/-0.010) for {'C': 12, 'gamma': 0.9}
0.991 (+/-0.009) for {'C': 12, 'gamma': 1.0}
0.991 (+/-0.007) for {'C': 12, 'gamma': 1.1}
0.991 (+/-0.007) for {'C': 12, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.3000000000000003}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.4000000000000001}
0.991 (+/-0.007) for {'C': 12, 'gamma': 1.5000000000000002}
0.991 (+/-0.007) for {'C': 12, 'gamma': 1.6}
0.991 (+/-0.006) for {'C': 12, 'gamma': 1.7000000000000002}
0.992 (+/-0.007) for {'C': 12, 'gamma': 1.8000000000000003}
0.992 (+/-0.007) for {'C': 12, 'gamma': 1.9000000000000001}
```

```
0.975 (+/-0.010) for {'C': 13, 'gamma': 0.1}
0.980 (+/-0.011) for {'C': 13, 'gamma': 0.2}
0.984 (+/-0.011) for {'C': 13, 'gamma': 0.30000000000000004}
0.987 (+/-0.010) for {'C': 13, 'gamma': 0.4}
0.988 (+/-0.011) for {'C': 13, 'gamma': 0.5}
0.988 (+/-0.010) for {'C': 13, 'gamma': 0.6}
0.990 (+/-0.010) for {'C': 13, 'gamma': 0.7000000000000001}
0.990 (+/-0.010) for {'C': 13, 'gamma': 0.8}
0.991 (+/-0.010) for {'C': 13, 'gamma': 0.9}
0.991 (+/-0.009) for {'C': 13, 'gamma': 1.0}
0.991 (+/-0.007) for {'C': 13, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 13, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 13, 'gamma': 1.3000000000000003}
0.991 (+/-0.008) for {'C': 13, 'gamma': 1.4000000000000001}
0.991 (+/-0.007) for {'C': 13, 'gamma': 1.5000000000000002}
0.991 (+/-0.006) for {'C': 13, 'gamma': 1.6}
0.992 (+/-0.007) for {'C': 13, 'gamma': 1.7000000000000002}
0.992 (+/-0.007) for {'C': 13, 'gamma': 1.8000000000000003}
0.991 (+/-0.007) for {'C': 13, 'gamma': 1.9000000000000001}
0.975 (+/-0.009) for {'C': 14, 'gamma': 0.1}
0.980 (+/-0.011) for {'C': 14, 'gamma': 0.2}
0.984 (+/-0.011) for {'C': 14, 'gamma': 0.30000000000000004}
0.986 (+/-0.010) for {'C': 14, 'gamma': 0.4}
0.988 (+/-0.011) for {'C': 14, 'gamma': 0.5}
0.988 (+/-0.009) for {'C': 14, 'gamma': 0.6}
0.990 (+/-0.009) for {'C': 14, 'gamma': 0.7000000000000001}
0.990 (+/-0.010) for {'C': 14, 'gamma': 0.8}
0.991 (+/-0.010) for {'C': 14, 'gamma': 0.9}
0.991 (+/-0.007) for {'C': 14, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 14, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 14, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 14, 'gamma': 1.3000000000000003}
0.991 (+/-0.007) for {'C': 14, 'gamma': 1.4000000000000001}
0.991 (+/-0.007) for {'C': 14, 'gamma': 1.5000000000000002}
0.991 (+/-0.007) for {'C': 14, 'gamma': 1.6}
0.992 (+/-0.007) for {'C': 14, 'gamma': 1.7000000000000002}
0.992 (+/-0.007) for {'C': 14, 'gamma': 1.8000000000000003}
0.991 (+/-0.006) for {'C': 14, 'gamma': 1.9000000000000001}
0.975 (+/-0.009) for {'C': 15, 'gamma': 0.1}
0.980 (+/-0.011) for {'C': 15, 'gamma': 0.2}
0.985 (+/-0.011) for {'C': 15, 'gamma': 0.30000000000000004}
0.987 (+/-0.011) for {'C': 15, 'gamma': 0.4}
0.987 (+/-0.011) for {'C': 15, 'gamma': 0.5}
0.989 (+/-0.010) for {'C': 15, 'gamma': 0.6}
0.990 (+/-0.009) for {'C': 15, 'gamma': 0.7000000000000001}
0.990 (+/-0.010) for {'C': 15, 'gamma': 0.8}
0.991 (+/-0.009) for {'C': 15, 'gamma': 0.9}
0.991 (+/-0.007) for {'C': 15, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 15, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.3000000000000003}
0.991 (+/-0.007) for {'C': 15, 'gamma': 1.4000000000000001}
0.992 (+/-0.006) for {'C': 15, 'gamma': 1.5000000000000002}
0.991 (+/-0.007) for {'C': 15, 'gamma': 1.6}
0.992 (+/-0.007) for {'C': 15, 'gamma': 1.7000000000000002}
0.991 (+/-0.006) for {'C': 15, 'gamma': 1.8000000000000003}
0.991 (+/-0.006) for {'C': 15, 'gamma': 1.9000000000000001}
```

```
0.976 (+/-0.010) for {'C': 16, 'gamma': 0.1}
0.981 (+/-0.011) for {'C': 16, 'gamma': 0.2}
0.985 (+/-0.011) for {'C': 16, 'gamma': 0.30000000000000004}
0.987 (+/-0.011) for {'C': 16, 'gamma': 0.4}
0.988 (+/-0.010) for {'C': 16, 'gamma': 0.5}
0.989 (+/-0.009) for {'C': 16, 'gamma': 0.6}
0.990 (+/-0.009) for {'C': 16, 'gamma': 0.7000000000000001}
0.990 (+/-0.009) for {'C': 16, 'gamma': 0.8}
0.991 (+/-0.008) for {'C': 16, 'gamma': 0.9}
0.991 (+/-0.008) for {'C': 16, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.3000000000000003}
0.991 (+/-0.007) for {'C': 16, 'gamma': 1.4000000000000001}
0.991 (+/-0.006) for {'C': 16, 'gamma': 1.5000000000000002}
0.991 (+/-0.007) for {'C': 16, 'gamma': 1.6}
0.991 (+/-0.006) for {'C': 16, 'gamma': 1.7000000000000002}
0.991 (+/-0.006) for {'C': 16, 'gamma': 1.8000000000000003}
0.991 (+/-0.006) for {'C': 16, 'gamma': 1.9000000000000001}
0.976 (+/-0.009) for {'C': 17, 'gamma': 0.1}
0.981 (+/-0.011) for {'C': 17, 'gamma': 0.2}
0.985 (+/-0.011) for {'C': 17, 'gamma': 0.30000000000000004}
0.987 (+/-0.011) for {'C': 17, 'gamma': 0.4}
0.988 (+/-0.010) for {'C': 17, 'gamma': 0.5}
0.989 (+/-0.010) for {'C': 17, 'gamma': 0.6}
0.990 (+/-0.009) for {'C': 17, 'gamma': 0.7000000000000001}
0.990 (+/-0.010) for {'C': 17, 'gamma': 0.8}
0.991 (+/-0.007) for {'C': 17, 'gamma': 0.9}
0.991 (+/-0.008) for {'C': 17, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 17, 'gamma': 1.3000000000000003}
0.991 (+/-0.006) for {'C': 17, 'gamma': 1.4000000000000001}
0.991 (+/-0.006) for {'C': 17, 'gamma': 1.5000000000000002}
0.991 (+/-0.007) for {'C': 17, 'gamma': 1.6}
0.991 (+/-0.006) for {'C': 17, 'gamma': 1.7000000000000002}
0.991 (+/-0.006) for {'C': 17, 'gamma': 1.8000000000000003}
0.991 (+/-0.008) for {'C': 17, 'gamma': 1.9000000000000001}
0.976 (+/-0.009) for {'C': 18, 'gamma': 0.1}
0.981 (+/-0.011) for {'C': 18, 'gamma': 0.2}
0.986 (+/-0.011) for {'C': 18, 'gamma': 0.30000000000000004}
0.987 (+/-0.011) for {'C': 18, 'gamma': 0.4}
0.988 (+/-0.011) for {'C': 18, 'gamma': 0.5}
0.990 (+/-0.010) for {'C': 18, 'gamma': 0.6}
0.990 (+/-0.009) for {'C': 18, 'gamma': 0.7000000000000001}
0.990 (+/-0.009) for {'C': 18, 'gamma': 0.8}
0.991 (+/-0.008) for {'C': 18, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.2000000000000002}
0.991 (+/-0.009) for {'C': 18, 'gamma': 1.3000000000000003}
0.991 (+/-0.006) for {'C': 18, 'gamma': 1.4000000000000001}
0.991 (+/-0.006) for {'C': 18, 'gamma': 1.5000000000000002}
0.991 (+/-0.006) for {'C': 18, 'gamma': 1.6}
0.991 (+/-0.006) for {'C': 18, 'gamma': 1.7000000000000002}
0.991 (+/-0.007) for {'C': 18, 'gamma': 1.8000000000000003}
0.991 (+/-0.008) for {'C': 18, 'gamma': 1.9000000000000001}
```

```
0.977 (+/-0.010) for {'C': 19, 'gamma': 0.1}
0.982 (+/-0.012) for {'C': 19, 'gamma': 0.2}
0.986 (+/-0.011) for {'C': 19, 'gamma': 0.30000000000000004}
0.987 (+/-0.011) for {'C': 19, 'gamma': 0.4}
0.988 (+/-0.011) for {'C': 19, 'gamma': 0.5}
0.989 (+/-0.010) for {'C': 19, 'gamma': 0.6}
0.990 (+/-0.009) for {'C': 19, 'gamma': 0.7000000000000001}
0.990 (+/-0.009) for {'C': 19, 'gamma': 0.8}
0.991 (+/-0.008) for {'C': 19, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 19, 'gamma': 1.3000000000000003}
0.991 (+/-0.006) for {'C': 19, 'gamma': 1.4000000000000001}
0.991 (+/-0.006) for {'C': 19, 'gamma': 1.5000000000000002}
0.991 (+/-0.006) for {'C': 19, 'gamma': 1.6}
0.991 (+/-0.007) for {'C': 19, 'gamma': 1.7000000000000002}
0.991 (+/-0.007) for {'C': 19, 'gamma': 1.8000000000000003}
0.991 (+/-0.008) for {'C': 19, 'gamma': 1.9000000000000001}
C 4
cache_size 200
class_weight None
coef0 0.0
decision_function_shape ovr
degree 3
gamma 1.8000000000000003
kernel rbf
max_iter -1
probability True
random_state None
shrinking True
tol 0.001
verbose False
```

ii)feature: Genus-- without normalization

In [4]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
import pandas as pd
import numpy as np

Folder_Path = r'/Users/yqh/Desktop/'

b=[]
a = pd.read_csv(Folder_Path + 'training_data.csv', usecols=range(0, 22),
 header=0)
b = pd.read_csv(Folder_Path + 'training_data.csv', usecols=[23], header=
0)

model = SVC(kernel='rbf', probability=True)
param_grid = {'C': range(1, 20), 'gamma': np.arange(0.1, 2, 0.1)}
grid_search = GridSearchCV(model, param_grid, scoring='accuracy', cv=10,
 n_jobs=1, verbose=1)
grid_search.fit(a, b)
best_parameters = grid_search.best_estimator_.get_params()

means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, grid_search.cv_results_['param
s']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))

for para, val in best_parameters.items():
    print(para, val)
```

```
Fitting 10 folds for each of 361 candidates, totalling 3610 fits
[Parallel(n_jobs=1)]: Done 3610 out of 3610 | elapsed: 42.9min finished
```

```
0.939 (+/-0.020) for {'C': 1, 'gamma': 0.1}
0.954 (+/-0.024) for {'C': 1, 'gamma': 0.2}
0.969 (+/-0.014) for {'C': 1, 'gamma': 0.30000000000000004}
0.972 (+/-0.011) for {'C': 1, 'gamma': 0.4}
0.976 (+/-0.014) for {'C': 1, 'gamma': 0.5}
0.979 (+/-0.012) for {'C': 1, 'gamma': 0.6}
0.980 (+/-0.012) for {'C': 1, 'gamma': 0.7000000000000001}
0.981 (+/-0.011) for {'C': 1, 'gamma': 0.8}
0.983 (+/-0.011) for {'C': 1, 'gamma': 0.9}
0.985 (+/-0.009) for {'C': 1, 'gamma': 1.0}
0.985 (+/-0.009) for {'C': 1, 'gamma': 1.1}
0.986 (+/-0.009) for {'C': 1, 'gamma': 1.2000000000000002}
0.986 (+/-0.009) for {'C': 1, 'gamma': 1.3000000000000003}
0.987 (+/-0.009) for {'C': 1, 'gamma': 1.4000000000000001}
0.987 (+/-0.008) for {'C': 1, 'gamma': 1.5000000000000002}
0.986 (+/-0.008) for {'C': 1, 'gamma': 1.6}
0.987 (+/-0.007) for {'C': 1, 'gamma': 1.7000000000000002}
0.988 (+/-0.006) for {'C': 1, 'gamma': 1.8000000000000003}
0.988 (+/-0.006) for {'C': 1, 'gamma': 1.9000000000000001}
0.952 (+/-0.019) for {'C': 2, 'gamma': 0.1}
0.971 (+/-0.013) for {'C': 2, 'gamma': 0.2}
0.978 (+/-0.012) for {'C': 2, 'gamma': 0.30000000000000004}
0.981 (+/-0.011) for {'C': 2, 'gamma': 0.4}
0.982 (+/-0.010) for {'C': 2, 'gamma': 0.5}
0.984 (+/-0.011) for {'C': 2, 'gamma': 0.6}
0.985 (+/-0.009) for {'C': 2, 'gamma': 0.7000000000000001}
0.987 (+/-0.008) for {'C': 2, 'gamma': 0.8}
0.987 (+/-0.008) for {'C': 2, 'gamma': 0.9}
0.987 (+/-0.007) for {'C': 2, 'gamma': 1.0}
0.988 (+/-0.007) for {'C': 2, 'gamma': 1.1}
0.989 (+/-0.006) for {'C': 2, 'gamma': 1.2000000000000002}
0.989 (+/-0.006) for {'C': 2, 'gamma': 1.3000000000000003}
0.989 (+/-0.006) for {'C': 2, 'gamma': 1.4000000000000001}
0.989 (+/-0.006) for {'C': 2, 'gamma': 1.5000000000000002}
0.988 (+/-0.006) for {'C': 2, 'gamma': 1.6}
0.989 (+/-0.006) for {'C': 2, 'gamma': 1.7000000000000002}
0.988 (+/-0.007) for {'C': 2, 'gamma': 1.8000000000000003}
0.988 (+/-0.007) for {'C': 2, 'gamma': 1.9000000000000001}
0.958 (+/-0.018) for {'C': 3, 'gamma': 0.1}
0.975 (+/-0.010) for {'C': 3, 'gamma': 0.2}
0.981 (+/-0.010) for {'C': 3, 'gamma': 0.30000000000000004}
0.982 (+/-0.011) for {'C': 3, 'gamma': 0.4}
0.983 (+/-0.011) for {'C': 3, 'gamma': 0.5}
0.986 (+/-0.008) for {'C': 3, 'gamma': 0.6}
0.987 (+/-0.008) for {'C': 3, 'gamma': 0.7000000000000001}
0.988 (+/-0.007) for {'C': 3, 'gamma': 0.8}
0.988 (+/-0.007) for {'C': 3, 'gamma': 0.9}
0.988 (+/-0.007) for {'C': 3, 'gamma': 1.0}
0.988 (+/-0.007) for {'C': 3, 'gamma': 1.1}
0.989 (+/-0.007) for {'C': 3, 'gamma': 1.2000000000000002}
0.989 (+/-0.006) for {'C': 3, 'gamma': 1.3000000000000003}
0.989 (+/-0.006) for {'C': 3, 'gamma': 1.4000000000000001}
0.989 (+/-0.006) for {'C': 3, 'gamma': 1.5000000000000002}
0.989 (+/-0.007) for {'C': 3, 'gamma': 1.6}
0.989 (+/-0.007) for {'C': 3, 'gamma': 1.7000000000000002}
0.989 (+/-0.008) for {'C': 3, 'gamma': 1.8000000000000003}
0.989 (+/-0.008) for {'C': 3, 'gamma': 1.9000000000000001}
```

```
0.966 (+/-0.012) for {'C': 4, 'gamma': 0.1}
0.979 (+/-0.010) for {'C': 4, 'gamma': 0.2}
0.982 (+/-0.011) for {'C': 4, 'gamma': 0.30000000000000004}
0.983 (+/-0.012) for {'C': 4, 'gamma': 0.4}
0.985 (+/-0.009) for {'C': 4, 'gamma': 0.5}
0.987 (+/-0.009) for {'C': 4, 'gamma': 0.6}
0.988 (+/-0.007) for {'C': 4, 'gamma': 0.7000000000000001}
0.988 (+/-0.006) for {'C': 4, 'gamma': 0.8}
0.989 (+/-0.007) for {'C': 4, 'gamma': 0.9}
0.989 (+/-0.008) for {'C': 4, 'gamma': 1.0}
0.989 (+/-0.008) for {'C': 4, 'gamma': 1.1}
0.989 (+/-0.008) for {'C': 4, 'gamma': 1.2000000000000002}
0.990 (+/-0.007) for {'C': 4, 'gamma': 1.3000000000000003}
0.990 (+/-0.006) for {'C': 4, 'gamma': 1.4000000000000001}
0.990 (+/-0.006) for {'C': 4, 'gamma': 1.5000000000000002}
0.989 (+/-0.007) for {'C': 4, 'gamma': 1.6}
0.990 (+/-0.007) for {'C': 4, 'gamma': 1.7000000000000002}
0.990 (+/-0.007) for {'C': 4, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 4, 'gamma': 1.9000000000000001}
0.971 (+/-0.011) for {'C': 5, 'gamma': 0.1}
0.980 (+/-0.011) for {'C': 5, 'gamma': 0.2}
0.982 (+/-0.010) for {'C': 5, 'gamma': 0.30000000000000004}
0.984 (+/-0.012) for {'C': 5, 'gamma': 0.4}
0.987 (+/-0.009) for {'C': 5, 'gamma': 0.5}
0.988 (+/-0.008) for {'C': 5, 'gamma': 0.6}
0.989 (+/-0.007) for {'C': 5, 'gamma': 0.7000000000000001}
0.989 (+/-0.007) for {'C': 5, 'gamma': 0.8}
0.989 (+/-0.008) for {'C': 5, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 5, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 5, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 5, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 5, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 5, 'gamma': 1.4000000000000001}
0.990 (+/-0.007) for {'C': 5, 'gamma': 1.5000000000000002}
0.990 (+/-0.008) for {'C': 5, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 5, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 5, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 5, 'gamma': 1.9000000000000001}
0.973 (+/-0.009) for {'C': 6, 'gamma': 0.1}
0.980 (+/-0.011) for {'C': 6, 'gamma': 0.2}
0.983 (+/-0.012) for {'C': 6, 'gamma': 0.30000000000000004}
0.986 (+/-0.008) for {'C': 6, 'gamma': 0.4}
0.987 (+/-0.008) for {'C': 6, 'gamma': 0.5}
0.989 (+/-0.007) for {'C': 6, 'gamma': 0.6}
0.989 (+/-0.006) for {'C': 6, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 6, 'gamma': 0.8}
0.990 (+/-0.008) for {'C': 6, 'gamma': 0.9}
0.989 (+/-0.008) for {'C': 6, 'gamma': 1.0}
0.990 (+/-0.009) for {'C': 6, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 6, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 6, 'gamma': 1.3000000000000003}
0.991 (+/-0.008) for {'C': 6, 'gamma': 1.4000000000000001}
0.991 (+/-0.008) for {'C': 6, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 6, 'gamma': 1.6}
0.990 (+/-0.009) for {'C': 6, 'gamma': 1.7000000000000002}
0.990 (+/-0.009) for {'C': 6, 'gamma': 1.8000000000000003}
0.990 (+/-0.010) for {'C': 6, 'gamma': 1.9000000000000001}
```

```
0.975 (+/-0.008) for {'C': 7, 'gamma': 0.1}
0.981 (+/-0.011) for {'C': 7, 'gamma': 0.2}
0.983 (+/-0.012) for {'C': 7, 'gamma': 0.30000000000000004}
0.986 (+/-0.009) for {'C': 7, 'gamma': 0.4}
0.988 (+/-0.008) for {'C': 7, 'gamma': 0.5}
0.989 (+/-0.006) for {'C': 7, 'gamma': 0.6}
0.989 (+/-0.008) for {'C': 7, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 7, 'gamma': 0.8}
0.989 (+/-0.008) for {'C': 7, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 7, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 7, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 7, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 7, 'gamma': 1.3000000000000003}
0.991 (+/-0.009) for {'C': 7, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 7, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 7, 'gamma': 1.6}
0.990 (+/-0.010) for {'C': 7, 'gamma': 1.7000000000000002}
0.990 (+/-0.009) for {'C': 7, 'gamma': 1.8000000000000003}
0.990 (+/-0.009) for {'C': 7, 'gamma': 1.9000000000000001}
0.976 (+/-0.010) for {'C': 8, 'gamma': 0.1}
0.982 (+/-0.010) for {'C': 8, 'gamma': 0.2}
0.983 (+/-0.012) for {'C': 8, 'gamma': 0.30000000000000004}
0.987 (+/-0.009) for {'C': 8, 'gamma': 0.4}
0.989 (+/-0.007) for {'C': 8, 'gamma': 0.5}
0.988 (+/-0.007) for {'C': 8, 'gamma': 0.6}
0.989 (+/-0.009) for {'C': 8, 'gamma': 0.7000000000000001}
0.989 (+/-0.009) for {'C': 8, 'gamma': 0.8}
0.989 (+/-0.008) for {'C': 8, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 8, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 8, 'gamma': 1.1}
0.991 (+/-0.009) for {'C': 8, 'gamma': 1.2000000000000002}
0.991 (+/-0.009) for {'C': 8, 'gamma': 1.3000000000000003}
0.991 (+/-0.009) for {'C': 8, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 8, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 8, 'gamma': 1.6}
0.990 (+/-0.009) for {'C': 8, 'gamma': 1.7000000000000002}
0.990 (+/-0.009) for {'C': 8, 'gamma': 1.8000000000000003}
0.990 (+/-0.009) for {'C': 8, 'gamma': 1.9000000000000001}
0.977 (+/-0.012) for {'C': 9, 'gamma': 0.1}
0.983 (+/-0.011) for {'C': 9, 'gamma': 0.2}
0.984 (+/-0.010) for {'C': 9, 'gamma': 0.30000000000000004}
0.987 (+/-0.008) for {'C': 9, 'gamma': 0.4}
0.989 (+/-0.007) for {'C': 9, 'gamma': 0.5}
0.988 (+/-0.008) for {'C': 9, 'gamma': 0.6}
0.989 (+/-0.009) for {'C': 9, 'gamma': 0.7000000000000001}
0.989 (+/-0.009) for {'C': 9, 'gamma': 0.8}
0.990 (+/-0.008) for {'C': 9, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 9, 'gamma': 1.0}
0.990 (+/-0.009) for {'C': 9, 'gamma': 1.1}
0.991 (+/-0.009) for {'C': 9, 'gamma': 1.2000000000000002}
0.991 (+/-0.009) for {'C': 9, 'gamma': 1.3000000000000003}
0.990 (+/-0.010) for {'C': 9, 'gamma': 1.4000000000000001}
0.990 (+/-0.009) for {'C': 9, 'gamma': 1.5000000000000002}
0.990 (+/-0.010) for {'C': 9, 'gamma': 1.6}
0.990 (+/-0.009) for {'C': 9, 'gamma': 1.7000000000000002}
0.990 (+/-0.009) for {'C': 9, 'gamma': 1.8000000000000003}
0.990 (+/-0.009) for {'C': 9, 'gamma': 1.9000000000000001}
```

```
0.977 (+/-0.012) for {'C': 10, 'gamma': 0.1}
0.983 (+/-0.012) for {'C': 10, 'gamma': 0.2}
0.985 (+/-0.009) for {'C': 10, 'gamma': 0.30000000000000004}
0.987 (+/-0.008) for {'C': 10, 'gamma': 0.4}
0.988 (+/-0.006) for {'C': 10, 'gamma': 0.5}
0.988 (+/-0.007) for {'C': 10, 'gamma': 0.6}
0.988 (+/-0.008) for {'C': 10, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 10, 'gamma': 0.8}
0.990 (+/-0.008) for {'C': 10, 'gamma': 0.9}
0.991 (+/-0.008) for {'C': 10, 'gamma': 1.0}
0.991 (+/-0.009) for {'C': 10, 'gamma': 1.1}
0.991 (+/-0.009) for {'C': 10, 'gamma': 1.2000000000000002}
0.991 (+/-0.009) for {'C': 10, 'gamma': 1.3000000000000003}
0.990 (+/-0.009) for {'C': 10, 'gamma': 1.4000000000000001}
0.990 (+/-0.009) for {'C': 10, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 10, 'gamma': 1.6}
0.990 (+/-0.009) for {'C': 10, 'gamma': 1.7000000000000002}
0.990 (+/-0.009) for {'C': 10, 'gamma': 1.8000000000000003}
0.990 (+/-0.009) for {'C': 10, 'gamma': 1.9000000000000001}
0.977 (+/-0.012) for {'C': 11, 'gamma': 0.1}
0.982 (+/-0.012) for {'C': 11, 'gamma': 0.2}
0.985 (+/-0.009) for {'C': 11, 'gamma': 0.30000000000000004}
0.988 (+/-0.007) for {'C': 11, 'gamma': 0.4}
0.988 (+/-0.005) for {'C': 11, 'gamma': 0.5}
0.988 (+/-0.007) for {'C': 11, 'gamma': 0.6}
0.988 (+/-0.008) for {'C': 11, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 11, 'gamma': 0.8}
0.990 (+/-0.008) for {'C': 11, 'gamma': 0.9}
0.991 (+/-0.009) for {'C': 11, 'gamma': 1.0}
0.991 (+/-0.009) for {'C': 11, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 11, 'gamma': 1.2000000000000002}
0.990 (+/-0.009) for {'C': 11, 'gamma': 1.3000000000000003}
0.990 (+/-0.009) for {'C': 11, 'gamma': 1.4000000000000001}
0.990 (+/-0.009) for {'C': 11, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 11, 'gamma': 1.6}
0.990 (+/-0.009) for {'C': 11, 'gamma': 1.7000000000000002}
0.990 (+/-0.009) for {'C': 11, 'gamma': 1.8000000000000003}
0.990 (+/-0.009) for {'C': 11, 'gamma': 1.9000000000000001}
0.977 (+/-0.012) for {'C': 12, 'gamma': 0.1}
0.982 (+/-0.012) for {'C': 12, 'gamma': 0.2}
0.986 (+/-0.008) for {'C': 12, 'gamma': 0.30000000000000004}
0.988 (+/-0.006) for {'C': 12, 'gamma': 0.4}
0.988 (+/-0.007) for {'C': 12, 'gamma': 0.5}
0.988 (+/-0.008) for {'C': 12, 'gamma': 0.6}
0.989 (+/-0.007) for {'C': 12, 'gamma': 0.7000000000000001}
0.990 (+/-0.008) for {'C': 12, 'gamma': 0.8}
0.990 (+/-0.008) for {'C': 12, 'gamma': 0.9}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.1}
0.990 (+/-0.009) for {'C': 12, 'gamma': 1.2000000000000002}
0.990 (+/-0.009) for {'C': 12, 'gamma': 1.3000000000000003}
0.990 (+/-0.009) for {'C': 12, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 12, 'gamma': 1.5000000000000002}
0.990 (+/-0.008) for {'C': 12, 'gamma': 1.6}
0.990 (+/-0.009) for {'C': 12, 'gamma': 1.7000000000000002}
0.990 (+/-0.009) for {'C': 12, 'gamma': 1.8000000000000003}
0.990 (+/-0.009) for {'C': 12, 'gamma': 1.9000000000000001}
```

```
0.978 (+/-0.013) for {'C': 13, 'gamma': 0.1}
0.983 (+/-0.011) for {'C': 13, 'gamma': 0.2}
0.986 (+/-0.006) for {'C': 13, 'gamma': 0.30000000000000004}
0.987 (+/-0.006) for {'C': 13, 'gamma': 0.4}
0.988 (+/-0.007) for {'C': 13, 'gamma': 0.5}
0.988 (+/-0.008) for {'C': 13, 'gamma': 0.6}
0.989 (+/-0.007) for {'C': 13, 'gamma': 0.7000000000000001}
0.990 (+/-0.008) for {'C': 13, 'gamma': 0.8}
0.990 (+/-0.007) for {'C': 13, 'gamma': 0.9}
0.990 (+/-0.007) for {'C': 13, 'gamma': 1.0}
0.990 (+/-0.009) for {'C': 13, 'gamma': 1.1}
0.990 (+/-0.009) for {'C': 13, 'gamma': 1.2000000000000002}
0.990 (+/-0.009) for {'C': 13, 'gamma': 1.3000000000000003}
0.990 (+/-0.009) for {'C': 13, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 13, 'gamma': 1.5000000000000002}
0.990 (+/-0.008) for {'C': 13, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 13, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 13, 'gamma': 1.8000000000000003}
0.990 (+/-0.009) for {'C': 13, 'gamma': 1.9000000000000001}
0.979 (+/-0.013) for {'C': 14, 'gamma': 0.1}
0.983 (+/-0.012) for {'C': 14, 'gamma': 0.2}
0.986 (+/-0.006) for {'C': 14, 'gamma': 0.30000000000000004}
0.988 (+/-0.007) for {'C': 14, 'gamma': 0.4}
0.988 (+/-0.007) for {'C': 14, 'gamma': 0.5}
0.988 (+/-0.008) for {'C': 14, 'gamma': 0.6}
0.989 (+/-0.007) for {'C': 14, 'gamma': 0.7000000000000001}
0.990 (+/-0.007) for {'C': 14, 'gamma': 0.8}
0.990 (+/-0.007) for {'C': 14, 'gamma': 0.9}
0.990 (+/-0.007) for {'C': 14, 'gamma': 1.0}
0.990 (+/-0.009) for {'C': 14, 'gamma': 1.1}
0.990 (+/-0.009) for {'C': 14, 'gamma': 1.2000000000000002}
0.990 (+/-0.009) for {'C': 14, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 14, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 14, 'gamma': 1.5000000000000002}
0.990 (+/-0.008) for {'C': 14, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 14, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 14, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 14, 'gamma': 1.9000000000000001}
0.980 (+/-0.013) for {'C': 15, 'gamma': 0.1}
0.983 (+/-0.012) for {'C': 15, 'gamma': 0.2}
0.986 (+/-0.006) for {'C': 15, 'gamma': 0.30000000000000004}
0.988 (+/-0.007) for {'C': 15, 'gamma': 0.4}
0.988 (+/-0.008) for {'C': 15, 'gamma': 0.5}
0.988 (+/-0.007) for {'C': 15, 'gamma': 0.6}
0.989 (+/-0.006) for {'C': 15, 'gamma': 0.7000000000000001}
0.990 (+/-0.006) for {'C': 15, 'gamma': 0.8}
0.990 (+/-0.007) for {'C': 15, 'gamma': 0.9}
0.990 (+/-0.009) for {'C': 15, 'gamma': 1.0}
0.990 (+/-0.009) for {'C': 15, 'gamma': 1.1}
0.990 (+/-0.009) for {'C': 15, 'gamma': 1.2000000000000002}
0.990 (+/-0.009) for {'C': 15, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.5000000000000002}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.9000000000000001}
```

```
0.980 (+/-0.013) for {'C': 16, 'gamma': 0.1}
0.983 (+/-0.013) for {'C': 16, 'gamma': 0.2}
0.986 (+/-0.006) for {'C': 16, 'gamma': 0.30000000000000004}
0.987 (+/-0.007) for {'C': 16, 'gamma': 0.4}
0.988 (+/-0.008) for {'C': 16, 'gamma': 0.5}
0.989 (+/-0.006) for {'C': 16, 'gamma': 0.6}
0.989 (+/-0.006) for {'C': 16, 'gamma': 0.7000000000000001}
0.991 (+/-0.007) for {'C': 16, 'gamma': 0.8}
0.990 (+/-0.007) for {'C': 16, 'gamma': 0.9}
0.990 (+/-0.009) for {'C': 16, 'gamma': 1.0}
0.990 (+/-0.009) for {'C': 16, 'gamma': 1.1}
0.990 (+/-0.009) for {'C': 16, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.5000000000000002}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.9000000000000001}
0.980 (+/-0.013) for {'C': 17, 'gamma': 0.1}
0.984 (+/-0.011) for {'C': 17, 'gamma': 0.2}
0.987 (+/-0.006) for {'C': 17, 'gamma': 0.30000000000000004}
0.987 (+/-0.007) for {'C': 17, 'gamma': 0.4}
0.988 (+/-0.007) for {'C': 17, 'gamma': 0.5}
0.989 (+/-0.006) for {'C': 17, 'gamma': 0.6}
0.989 (+/-0.006) for {'C': 17, 'gamma': 0.7000000000000001}
0.990 (+/-0.007) for {'C': 17, 'gamma': 0.8}
0.990 (+/-0.009) for {'C': 17, 'gamma': 0.9}
0.990 (+/-0.009) for {'C': 17, 'gamma': 1.0}
0.990 (+/-0.009) for {'C': 17, 'gamma': 1.1}
0.990 (+/-0.009) for {'C': 17, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 17, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.9000000000000001}
0.980 (+/-0.013) for {'C': 18, 'gamma': 0.1}
0.984 (+/-0.012) for {'C': 18, 'gamma': 0.2}
0.988 (+/-0.007) for {'C': 18, 'gamma': 0.30000000000000004}
0.987 (+/-0.007) for {'C': 18, 'gamma': 0.4}
0.988 (+/-0.006) for {'C': 18, 'gamma': 0.5}
0.989 (+/-0.007) for {'C': 18, 'gamma': 0.6}
0.989 (+/-0.006) for {'C': 18, 'gamma': 0.7000000000000001}
0.990 (+/-0.007) for {'C': 18, 'gamma': 0.8}
0.990 (+/-0.009) for {'C': 18, 'gamma': 0.9}
0.990 (+/-0.009) for {'C': 18, 'gamma': 1.0}
0.990 (+/-0.009) for {'C': 18, 'gamma': 1.1}
0.990 (+/-0.009) for {'C': 18, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 18, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.9000000000000001}
```

```
0.980 (+/-0.013) for {'C': 19, 'gamma': 0.1}
0.984 (+/-0.012) for {'C': 19, 'gamma': 0.2}
0.988 (+/-0.007) for {'C': 19, 'gamma': 0.30000000000000004}
0.987 (+/-0.008) for {'C': 19, 'gamma': 0.4}
0.988 (+/-0.007) for {'C': 19, 'gamma': 0.5}
0.989 (+/-0.007) for {'C': 19, 'gamma': 0.6}
0.990 (+/-0.006) for {'C': 19, 'gamma': 0.7000000000000001}
0.990 (+/-0.007) for {'C': 19, 'gamma': 0.8}
0.990 (+/-0.009) for {'C': 19, 'gamma': 0.9}
0.990 (+/-0.009) for {'C': 19, 'gamma': 1.0}
0.990 (+/-0.009) for {'C': 19, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.5000000000000002}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.9000000000000001}
C 8
cache_size 200
class_weight None
coef0 0.0
decision_function_shape ovr
degree 3
gamma 1.2000000000000002
kernel rbf
max_iter -1
probability True
random_state None
shrinking True
tol 0.001
verbose False
```

iii)feature:Species--without normalization

In [5]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
import pandas as pd
import numpy as np

Folder_Path = r'/Users/yqh/Desktop/'

b=[]
a = pd.read_csv(Folder_Path + 'training_data.csv', usecols=range(0, 22),
 header=0)
b = pd.read_csv(Folder_Path + 'training_data.csv', usecols=[24], header=
0)

model = SVC(kernel='rbf', probability=True)
param_grid = {'C': range(1, 20), 'gamma': np.arange(0.1, 2, 0.1)}
grid_search = GridSearchCV(model, param_grid, scoring='accuracy', cv=10,
 n_jobs=1, verbose=1)
grid_search.fit(a, b)
best_parameters = grid_search.best_estimator_.get_params()

means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, grid_search.cv_results_['param
s']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))

for para, val in best_parameters.items():
    print(para, val)
```

```
Fitting 10 folds for each of 361 candidates, totalling 3610 fits
[Parallel(n_jobs=1)]: Done 3610 out of 3610 | elapsed: 41.0min finished
```

```
0.947 (+/-0.021) for {'C': 1, 'gamma': 0.1}
0.965 (+/-0.019) for {'C': 1, 'gamma': 0.2}
0.972 (+/-0.012) for {'C': 1, 'gamma': 0.30000000000000004}
0.976 (+/-0.014) for {'C': 1, 'gamma': 0.4}
0.978 (+/-0.011) for {'C': 1, 'gamma': 0.5}
0.981 (+/-0.011) for {'C': 1, 'gamma': 0.6}
0.982 (+/-0.010) for {'C': 1, 'gamma': 0.7000000000000001}
0.984 (+/-0.009) for {'C': 1, 'gamma': 0.8}
0.985 (+/-0.009) for {'C': 1, 'gamma': 0.9}
0.986 (+/-0.008) for {'C': 1, 'gamma': 1.0}
0.987 (+/-0.007) for {'C': 1, 'gamma': 1.1}
0.987 (+/-0.007) for {'C': 1, 'gamma': 1.2000000000000002}
0.987 (+/-0.007) for {'C': 1, 'gamma': 1.3000000000000003}
0.988 (+/-0.007) for {'C': 1, 'gamma': 1.4000000000000001}
0.988 (+/-0.006) for {'C': 1, 'gamma': 1.5000000000000002}
0.988 (+/-0.006) for {'C': 1, 'gamma': 1.6}
0.988 (+/-0.006) for {'C': 1, 'gamma': 1.7000000000000002}
0.987 (+/-0.006) for {'C': 1, 'gamma': 1.8000000000000003}
0.988 (+/-0.006) for {'C': 1, 'gamma': 1.9000000000000001}
0.962 (+/-0.017) for {'C': 2, 'gamma': 0.1}
0.975 (+/-0.012) for {'C': 2, 'gamma': 0.2}
0.980 (+/-0.010) for {'C': 2, 'gamma': 0.30000000000000004}
0.983 (+/-0.009) for {'C': 2, 'gamma': 0.4}
0.984 (+/-0.009) for {'C': 2, 'gamma': 0.5}
0.985 (+/-0.010) for {'C': 2, 'gamma': 0.6}
0.987 (+/-0.007) for {'C': 2, 'gamma': 0.7000000000000001}
0.988 (+/-0.007) for {'C': 2, 'gamma': 0.8}
0.988 (+/-0.007) for {'C': 2, 'gamma': 0.9}
0.989 (+/-0.009) for {'C': 2, 'gamma': 1.0}
0.989 (+/-0.008) for {'C': 2, 'gamma': 1.1}
0.989 (+/-0.008) for {'C': 2, 'gamma': 1.2000000000000002}
0.989 (+/-0.008) for {'C': 2, 'gamma': 1.3000000000000003}
0.989 (+/-0.007) for {'C': 2, 'gamma': 1.4000000000000001}
0.989 (+/-0.007) for {'C': 2, 'gamma': 1.5000000000000002}
0.989 (+/-0.007) for {'C': 2, 'gamma': 1.6}
0.989 (+/-0.007) for {'C': 2, 'gamma': 1.7000000000000002}
0.989 (+/-0.007) for {'C': 2, 'gamma': 1.8000000000000003}
0.989 (+/-0.007) for {'C': 2, 'gamma': 1.9000000000000001}
0.970 (+/-0.014) for {'C': 3, 'gamma': 0.1}
0.978 (+/-0.010) for {'C': 3, 'gamma': 0.2}
0.982 (+/-0.008) for {'C': 3, 'gamma': 0.30000000000000004}
0.984 (+/-0.010) for {'C': 3, 'gamma': 0.4}
0.986 (+/-0.008) for {'C': 3, 'gamma': 0.5}
0.987 (+/-0.006) for {'C': 3, 'gamma': 0.6}
0.988 (+/-0.007) for {'C': 3, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 3, 'gamma': 0.8}
0.989 (+/-0.007) for {'C': 3, 'gamma': 0.9}
0.988 (+/-0.007) for {'C': 3, 'gamma': 1.0}
0.989 (+/-0.007) for {'C': 3, 'gamma': 1.1}
0.989 (+/-0.008) for {'C': 3, 'gamma': 1.2000000000000002}
0.990 (+/-0.007) for {'C': 3, 'gamma': 1.3000000000000003}
0.989 (+/-0.007) for {'C': 3, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 3, 'gamma': 1.5000000000000002}
0.990 (+/-0.008) for {'C': 3, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 3, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 3, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 3, 'gamma': 1.9000000000000001}
```

```
0.973 (+/-0.010) for {'C': 4, 'gamma': 0.1}
0.980 (+/-0.008) for {'C': 4, 'gamma': 0.2}
0.983 (+/-0.009) for {'C': 4, 'gamma': 0.30000000000000004}
0.986 (+/-0.008) for {'C': 4, 'gamma': 0.4}
0.987 (+/-0.007) for {'C': 4, 'gamma': 0.5}
0.988 (+/-0.007) for {'C': 4, 'gamma': 0.6}
0.988 (+/-0.007) for {'C': 4, 'gamma': 0.7000000000000001}
0.988 (+/-0.008) for {'C': 4, 'gamma': 0.8}
0.989 (+/-0.008) for {'C': 4, 'gamma': 0.9}
0.989 (+/-0.008) for {'C': 4, 'gamma': 1.0}
0.989 (+/-0.008) for {'C': 4, 'gamma': 1.1}
0.989 (+/-0.008) for {'C': 4, 'gamma': 1.2000000000000002}
0.989 (+/-0.008) for {'C': 4, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 4, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 4, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 4, 'gamma': 1.6}
0.990 (+/-0.009) for {'C': 4, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 4, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 4, 'gamma': 1.9000000000000001}
0.974 (+/-0.012) for {'C': 5, 'gamma': 0.1}
0.982 (+/-0.010) for {'C': 5, 'gamma': 0.2}
0.984 (+/-0.009) for {'C': 5, 'gamma': 0.30000000000000004}
0.987 (+/-0.009) for {'C': 5, 'gamma': 0.4}
0.988 (+/-0.008) for {'C': 5, 'gamma': 0.5}
0.988 (+/-0.007) for {'C': 5, 'gamma': 0.6}
0.988 (+/-0.007) for {'C': 5, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 5, 'gamma': 0.8}
0.989 (+/-0.008) for {'C': 5, 'gamma': 0.9}
0.989 (+/-0.008) for {'C': 5, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 5, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 5, 'gamma': 1.2000000000000002}
0.989 (+/-0.009) for {'C': 5, 'gamma': 1.3000000000000003}
0.989 (+/-0.009) for {'C': 5, 'gamma': 1.4000000000000001}
0.990 (+/-0.009) for {'C': 5, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 5, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 5, 'gamma': 1.7000000000000002}
0.991 (+/-0.007) for {'C': 5, 'gamma': 1.8000000000000003}
0.991 (+/-0.007) for {'C': 5, 'gamma': 1.9000000000000001}
0.977 (+/-0.009) for {'C': 6, 'gamma': 0.1}
0.982 (+/-0.010) for {'C': 6, 'gamma': 0.2}
0.986 (+/-0.008) for {'C': 6, 'gamma': 0.30000000000000004}
0.988 (+/-0.007) for {'C': 6, 'gamma': 0.4}
0.988 (+/-0.007) for {'C': 6, 'gamma': 0.5}
0.988 (+/-0.007) for {'C': 6, 'gamma': 0.6}
0.989 (+/-0.007) for {'C': 6, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 6, 'gamma': 0.8}
0.989 (+/-0.008) for {'C': 6, 'gamma': 0.9}
0.989 (+/-0.008) for {'C': 6, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 6, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 6, 'gamma': 1.2000000000000002}
0.990 (+/-0.009) for {'C': 6, 'gamma': 1.3000000000000003}
0.990 (+/-0.009) for {'C': 6, 'gamma': 1.4000000000000001}
0.990 (+/-0.009) for {'C': 6, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 6, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 6, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 6, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 6, 'gamma': 1.9000000000000001}
```

```
0.978 (+/-0.012) for {'C': 7, 'gamma': 0.1}
0.983 (+/-0.009) for {'C': 7, 'gamma': 0.2}
0.986 (+/-0.008) for {'C': 7, 'gamma': 0.30000000000000004}
0.988 (+/-0.008) for {'C': 7, 'gamma': 0.4}
0.988 (+/-0.007) for {'C': 7, 'gamma': 0.5}
0.988 (+/-0.007) for {'C': 7, 'gamma': 0.6}
0.989 (+/-0.007) for {'C': 7, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 7, 'gamma': 0.8}
0.989 (+/-0.008) for {'C': 7, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 7, 'gamma': 1.0}
0.990 (+/-0.009) for {'C': 7, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 7, 'gamma': 1.2000000000000002}
0.990 (+/-0.009) for {'C': 7, 'gamma': 1.3000000000000003}
0.990 (+/-0.010) for {'C': 7, 'gamma': 1.4000000000000001}
0.990 (+/-0.009) for {'C': 7, 'gamma': 1.5000000000000002}
0.991 (+/-0.009) for {'C': 7, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 7, 'gamma': 1.7000000000000002}
0.991 (+/-0.008) for {'C': 7, 'gamma': 1.8000000000000003}
0.991 (+/-0.007) for {'C': 7, 'gamma': 1.9000000000000001}
0.980 (+/-0.012) for {'C': 8, 'gamma': 0.1}
0.984 (+/-0.009) for {'C': 8, 'gamma': 0.2}
0.987 (+/-0.008) for {'C': 8, 'gamma': 0.30000000000000004}
0.988 (+/-0.008) for {'C': 8, 'gamma': 0.4}
0.989 (+/-0.007) for {'C': 8, 'gamma': 0.5}
0.989 (+/-0.008) for {'C': 8, 'gamma': 0.6}
0.989 (+/-0.008) for {'C': 8, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 8, 'gamma': 0.8}
0.990 (+/-0.009) for {'C': 8, 'gamma': 0.9}
0.991 (+/-0.008) for {'C': 8, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 8, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 8, 'gamma': 1.2000000000000002}
0.990 (+/-0.010) for {'C': 8, 'gamma': 1.3000000000000003}
0.991 (+/-0.009) for {'C': 8, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 8, 'gamma': 1.5000000000000002}
0.991 (+/-0.009) for {'C': 8, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 8, 'gamma': 1.7000000000000002}
0.991 (+/-0.008) for {'C': 8, 'gamma': 1.8000000000000003}
0.991 (+/-0.008) for {'C': 8, 'gamma': 1.9000000000000001}
0.981 (+/-0.010) for {'C': 9, 'gamma': 0.1}
0.985 (+/-0.009) for {'C': 9, 'gamma': 0.2}
0.987 (+/-0.008) for {'C': 9, 'gamma': 0.30000000000000004}
0.988 (+/-0.008) for {'C': 9, 'gamma': 0.4}
0.989 (+/-0.008) for {'C': 9, 'gamma': 0.5}
0.989 (+/-0.009) for {'C': 9, 'gamma': 0.6}
0.989 (+/-0.008) for {'C': 9, 'gamma': 0.7000000000000001}
0.990 (+/-0.009) for {'C': 9, 'gamma': 0.8}
0.990 (+/-0.009) for {'C': 9, 'gamma': 0.9}
0.991 (+/-0.008) for {'C': 9, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 9, 'gamma': 1.1}
0.990 (+/-0.009) for {'C': 9, 'gamma': 1.2000000000000002}
0.991 (+/-0.010) for {'C': 9, 'gamma': 1.3000000000000003}
0.991 (+/-0.010) for {'C': 9, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 9, 'gamma': 1.5000000000000002}
0.991 (+/-0.009) for {'C': 9, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 9, 'gamma': 1.7000000000000002}
0.991 (+/-0.008) for {'C': 9, 'gamma': 1.8000000000000003}
0.991 (+/-0.008) for {'C': 9, 'gamma': 1.9000000000000001}
```

```
0.980 (+/-0.011) for {'C': 10, 'gamma': 0.1}
0.985 (+/-0.009) for {'C': 10, 'gamma': 0.2}
0.988 (+/-0.009) for {'C': 10, 'gamma': 0.30000000000000004}
0.988 (+/-0.007) for {'C': 10, 'gamma': 0.4}
0.989 (+/-0.008) for {'C': 10, 'gamma': 0.5}
0.989 (+/-0.009) for {'C': 10, 'gamma': 0.6}
0.989 (+/-0.008) for {'C': 10, 'gamma': 0.7000000000000001}
0.990 (+/-0.008) for {'C': 10, 'gamma': 0.8}
0.990 (+/-0.010) for {'C': 10, 'gamma': 0.9}
0.991 (+/-0.008) for {'C': 10, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 10, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 10, 'gamma': 1.2000000000000002}
0.991 (+/-0.009) for {'C': 10, 'gamma': 1.3000000000000003}
0.991 (+/-0.009) for {'C': 10, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 10, 'gamma': 1.5000000000000002}
0.991 (+/-0.009) for {'C': 10, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 10, 'gamma': 1.7000000000000002}
0.991 (+/-0.008) for {'C': 10, 'gamma': 1.8000000000000003}
0.991 (+/-0.008) for {'C': 10, 'gamma': 1.9000000000000001}
0.981 (+/-0.011) for {'C': 11, 'gamma': 0.1}
0.985 (+/-0.008) for {'C': 11, 'gamma': 0.2}
0.988 (+/-0.008) for {'C': 11, 'gamma': 0.30000000000000004}
0.988 (+/-0.008) for {'C': 11, 'gamma': 0.4}
0.989 (+/-0.008) for {'C': 11, 'gamma': 0.5}
0.989 (+/-0.009) for {'C': 11, 'gamma': 0.6}
0.989 (+/-0.008) for {'C': 11, 'gamma': 0.7000000000000001}
0.990 (+/-0.009) for {'C': 11, 'gamma': 0.8}
0.990 (+/-0.009) for {'C': 11, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 11, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 11, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 11, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 11, 'gamma': 1.3000000000000003}
0.991 (+/-0.009) for {'C': 11, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 11, 'gamma': 1.5000000000000002}
0.991 (+/-0.009) for {'C': 11, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 11, 'gamma': 1.7000000000000002}
0.991 (+/-0.008) for {'C': 11, 'gamma': 1.8000000000000003}
0.991 (+/-0.008) for {'C': 11, 'gamma': 1.9000000000000001}
0.981 (+/-0.012) for {'C': 12, 'gamma': 0.1}
0.985 (+/-0.008) for {'C': 12, 'gamma': 0.2}
0.988 (+/-0.007) for {'C': 12, 'gamma': 0.30000000000000004}
0.989 (+/-0.008) for {'C': 12, 'gamma': 0.4}
0.989 (+/-0.009) for {'C': 12, 'gamma': 0.5}
0.989 (+/-0.008) for {'C': 12, 'gamma': 0.6}
0.989 (+/-0.009) for {'C': 12, 'gamma': 0.7000000000000001}
0.990 (+/-0.009) for {'C': 12, 'gamma': 0.8}
0.990 (+/-0.009) for {'C': 12, 'gamma': 0.9}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.0}
0.991 (+/-0.007) for {'C': 12, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.3000000000000003}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 12, 'gamma': 1.5000000000000002}
0.991 (+/-0.009) for {'C': 12, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.7000000000000002}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.8000000000000003}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.9000000000000001}
```

```
0.982 (+/-0.011) for {'C': 13, 'gamma': 0.1}
0.986 (+/-0.007) for {'C': 13, 'gamma': 0.2}
0.988 (+/-0.007) for {'C': 13, 'gamma': 0.30000000000000004}
0.988 (+/-0.008) for {'C': 13, 'gamma': 0.4}
0.989 (+/-0.009) for {'C': 13, 'gamma': 0.5}
0.988 (+/-0.008) for {'C': 13, 'gamma': 0.6}
0.989 (+/-0.008) for {'C': 13, 'gamma': 0.7000000000000001}
0.990 (+/-0.008) for {'C': 13, 'gamma': 0.8}
0.990 (+/-0.009) for {'C': 13, 'gamma': 0.9}
0.991 (+/-0.008) for {'C': 13, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 13, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 13, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 13, 'gamma': 1.3000000000000003}
0.991 (+/-0.008) for {'C': 13, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 13, 'gamma': 1.5000000000000002}
0.991 (+/-0.009) for {'C': 13, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 13, 'gamma': 1.7000000000000002}
0.991 (+/-0.008) for {'C': 13, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 13, 'gamma': 1.9000000000000001}
0.982 (+/-0.011) for {'C': 14, 'gamma': 0.1}
0.987 (+/-0.008) for {'C': 14, 'gamma': 0.2}
0.988 (+/-0.008) for {'C': 14, 'gamma': 0.30000000000000004}
0.988 (+/-0.008) for {'C': 14, 'gamma': 0.4}
0.988 (+/-0.008) for {'C': 14, 'gamma': 0.5}
0.989 (+/-0.008) for {'C': 14, 'gamma': 0.6}
0.989 (+/-0.008) for {'C': 14, 'gamma': 0.7000000000000001}
0.990 (+/-0.008) for {'C': 14, 'gamma': 0.8}
0.990 (+/-0.010) for {'C': 14, 'gamma': 0.9}
0.991 (+/-0.007) for {'C': 14, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 14, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 14, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 14, 'gamma': 1.3000000000000003}
0.991 (+/-0.008) for {'C': 14, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 14, 'gamma': 1.5000000000000002}
0.991 (+/-0.009) for {'C': 14, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 14, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 14, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 14, 'gamma': 1.9000000000000001}
0.983 (+/-0.011) for {'C': 15, 'gamma': 0.1}
0.987 (+/-0.008) for {'C': 15, 'gamma': 0.2}
0.988 (+/-0.008) for {'C': 15, 'gamma': 0.30000000000000004}
0.989 (+/-0.008) for {'C': 15, 'gamma': 0.4}
0.988 (+/-0.007) for {'C': 15, 'gamma': 0.5}
0.989 (+/-0.008) for {'C': 15, 'gamma': 0.6}
0.989 (+/-0.008) for {'C': 15, 'gamma': 0.7000000000000001}
0.990 (+/-0.008) for {'C': 15, 'gamma': 0.8}
0.990 (+/-0.010) for {'C': 15, 'gamma': 0.9}
0.991 (+/-0.007) for {'C': 15, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 15, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 15, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.3000000000000003}
0.991 (+/-0.008) for {'C': 15, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 15, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 15, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.9000000000000001}
```

```
0.983 (+/-0.010) for {'C': 16, 'gamma': 0.1}
0.988 (+/-0.008) for {'C': 16, 'gamma': 0.2}
0.988 (+/-0.008) for {'C': 16, 'gamma': 0.30000000000000004}
0.989 (+/-0.008) for {'C': 16, 'gamma': 0.4}
0.988 (+/-0.007) for {'C': 16, 'gamma': 0.5}
0.989 (+/-0.008) for {'C': 16, 'gamma': 0.6}
0.989 (+/-0.008) for {'C': 16, 'gamma': 0.7000000000000001}
0.990 (+/-0.009) for {'C': 16, 'gamma': 0.8}
0.990 (+/-0.009) for {'C': 16, 'gamma': 0.9}
0.991 (+/-0.007) for {'C': 16, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 16, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 16, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 16, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.4000000000000001}
0.991 (+/-0.008) for {'C': 16, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 16, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.9000000000000001}
0.983 (+/-0.011) for {'C': 17, 'gamma': 0.1}
0.987 (+/-0.008) for {'C': 17, 'gamma': 0.2}
0.988 (+/-0.008) for {'C': 17, 'gamma': 0.30000000000000004}
0.989 (+/-0.008) for {'C': 17, 'gamma': 0.4}
0.988 (+/-0.007) for {'C': 17, 'gamma': 0.5}
0.989 (+/-0.007) for {'C': 17, 'gamma': 0.6}
0.989 (+/-0.008) for {'C': 17, 'gamma': 0.7000000000000001}
0.990 (+/-0.009) for {'C': 17, 'gamma': 0.8}
0.990 (+/-0.009) for {'C': 17, 'gamma': 0.9}
0.991 (+/-0.007) for {'C': 17, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 17, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 17, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 17, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.4000000000000001}
0.990 (+/-0.009) for {'C': 17, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 17, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.9000000000000001}
0.983 (+/-0.010) for {'C': 18, 'gamma': 0.1}
0.987 (+/-0.008) for {'C': 18, 'gamma': 0.2}
0.988 (+/-0.008) for {'C': 18, 'gamma': 0.30000000000000004}
0.988 (+/-0.008) for {'C': 18, 'gamma': 0.4}
0.989 (+/-0.006) for {'C': 18, 'gamma': 0.5}
0.990 (+/-0.007) for {'C': 18, 'gamma': 0.6}
0.989 (+/-0.007) for {'C': 18, 'gamma': 0.7000000000000001}
0.990 (+/-0.009) for {'C': 18, 'gamma': 0.8}
0.990 (+/-0.009) for {'C': 18, 'gamma': 0.9}
0.991 (+/-0.007) for {'C': 18, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 18, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 18, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 18, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.4000000000000001}
0.990 (+/-0.009) for {'C': 18, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 18, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.9000000000000001}
```

```
0.983 (+/-0.010) for {'C': 19, 'gamma': 0.1}
0.987 (+/-0.009) for {'C': 19, 'gamma': 0.2}
0.988 (+/-0.008) for {'C': 19, 'gamma': 0.30000000000000004}
0.988 (+/-0.007) for {'C': 19, 'gamma': 0.4}
0.989 (+/-0.007) for {'C': 19, 'gamma': 0.5}
0.990 (+/-0.007) for {'C': 19, 'gamma': 0.6}
0.989 (+/-0.007) for {'C': 19, 'gamma': 0.7000000000000001}
0.990 (+/-0.009) for {'C': 19, 'gamma': 0.8}
0.990 (+/-0.009) for {'C': 19, 'gamma': 0.9}
0.991 (+/-0.007) for {'C': 19, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 19, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 19, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.4000000000000001}
0.990 (+/-0.009) for {'C': 19, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 19, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.9000000000000001}
C 17
cache_size 200
class_weight None
coef0 0.0
decision_function_shape ovr
degree 3
gamma 1.0
kernel rbf
max_iter -1
probability True
random_state None
shrinking True
tol 0.001
verbose False
```

iv)feature: Family--with normalization

In [6]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
import pandas as pd
import numpy as np
import warnings

warnings.filterwarnings('ignore')

Folder_Path = r'/Users/yqh/Desktop/'

a = pd.read_csv(Folder_Path + 'norm_training_data.csv', usecols=range(0,
 22), header=0)
b = pd.read_csv(Folder_Path + 'norm_training_data.csv', usecols=[22], he
ader=0)

model = SVC(kernel='rbf', probability=True)
param_grid = {'C': range(1, 20), 'gamma': np.arange(0.1, 2, 0.1)}
grid_search = GridSearchCV(model, param_grid, scoring='accuracy', cv=10,
 n_jobs=1, verbose=1)
grid_search.fit(a, b)
best_parameters = grid_search.best_estimator_.get_params()

means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, grid_search.cv_results_['param
s']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))

for para, val in best_parameters.items():
    print(para, val)
```

```
Fitting 10 folds for each of 361 candidates, totalling 3610 fits
[Parallel(n_jobs=1)]: Done 3610 out of 3610 | elapsed: 40.9min finished
```

```
0.940 (+/-0.014) for {'C': 1, 'gamma': 0.1}
0.952 (+/-0.012) for {'C': 1, 'gamma': 0.2}
0.958 (+/-0.015) for {'C': 1, 'gamma': 0.30000000000000004}
0.963 (+/-0.011) for {'C': 1, 'gamma': 0.4}
0.966 (+/-0.010) for {'C': 1, 'gamma': 0.5}
0.970 (+/-0.011) for {'C': 1, 'gamma': 0.6}
0.971 (+/-0.013) for {'C': 1, 'gamma': 0.7000000000000001}
0.971 (+/-0.014) for {'C': 1, 'gamma': 0.8}
0.973 (+/-0.013) for {'C': 1, 'gamma': 0.9}
0.977 (+/-0.014) for {'C': 1, 'gamma': 1.0}
0.980 (+/-0.013) for {'C': 1, 'gamma': 1.1}
0.982 (+/-0.012) for {'C': 1, 'gamma': 1.2000000000000002}
0.983 (+/-0.012) for {'C': 1, 'gamma': 1.3000000000000003}
0.984 (+/-0.012) for {'C': 1, 'gamma': 1.4000000000000001}
0.985 (+/-0.011) for {'C': 1, 'gamma': 1.5000000000000002}
0.985 (+/-0.011) for {'C': 1, 'gamma': 1.6}
0.986 (+/-0.012) for {'C': 1, 'gamma': 1.7000000000000002}
0.987 (+/-0.012) for {'C': 1, 'gamma': 1.8000000000000003}
0.987 (+/-0.012) for {'C': 1, 'gamma': 1.9000000000000001}
0.948 (+/-0.013) for {'C': 2, 'gamma': 0.1}
0.959 (+/-0.012) for {'C': 2, 'gamma': 0.2}
0.964 (+/-0.010) for {'C': 2, 'gamma': 0.30000000000000004}
0.969 (+/-0.011) for {'C': 2, 'gamma': 0.4}
0.970 (+/-0.012) for {'C': 2, 'gamma': 0.5}
0.976 (+/-0.013) for {'C': 2, 'gamma': 0.6}
0.980 (+/-0.011) for {'C': 2, 'gamma': 0.7000000000000001}
0.983 (+/-0.011) for {'C': 2, 'gamma': 0.8}
0.982 (+/-0.011) for {'C': 2, 'gamma': 0.9}
0.984 (+/-0.011) for {'C': 2, 'gamma': 1.0}
0.985 (+/-0.011) for {'C': 2, 'gamma': 1.1}
0.986 (+/-0.012) for {'C': 2, 'gamma': 1.2000000000000002}
0.986 (+/-0.012) for {'C': 2, 'gamma': 1.3000000000000003}
0.986 (+/-0.011) for {'C': 2, 'gamma': 1.4000000000000001}
0.987 (+/-0.012) for {'C': 2, 'gamma': 1.5000000000000002}
0.987 (+/-0.013) for {'C': 2, 'gamma': 1.6}
0.988 (+/-0.013) for {'C': 2, 'gamma': 1.7000000000000002}
0.989 (+/-0.012) for {'C': 2, 'gamma': 1.8000000000000003}
0.989 (+/-0.012) for {'C': 2, 'gamma': 1.9000000000000001}
0.951 (+/-0.012) for {'C': 3, 'gamma': 0.1}
0.962 (+/-0.013) for {'C': 3, 'gamma': 0.2}
0.968 (+/-0.012) for {'C': 3, 'gamma': 0.30000000000000004}
0.971 (+/-0.013) for {'C': 3, 'gamma': 0.4}
0.979 (+/-0.010) for {'C': 3, 'gamma': 0.5}
0.981 (+/-0.011) for {'C': 3, 'gamma': 0.6}
0.982 (+/-0.011) for {'C': 3, 'gamma': 0.7000000000000001}
0.983 (+/-0.011) for {'C': 3, 'gamma': 0.8}
0.985 (+/-0.012) for {'C': 3, 'gamma': 0.9}
0.986 (+/-0.013) for {'C': 3, 'gamma': 1.0}
0.986 (+/-0.012) for {'C': 3, 'gamma': 1.1}
0.987 (+/-0.013) for {'C': 3, 'gamma': 1.2000000000000002}
0.988 (+/-0.012) for {'C': 3, 'gamma': 1.3000000000000003}
0.988 (+/-0.013) for {'C': 3, 'gamma': 1.4000000000000001}
0.989 (+/-0.012) for {'C': 3, 'gamma': 1.5000000000000002}
0.989 (+/-0.012) for {'C': 3, 'gamma': 1.6}
0.990 (+/-0.011) for {'C': 3, 'gamma': 1.7000000000000002}
0.991 (+/-0.011) for {'C': 3, 'gamma': 1.8000000000000003}
0.990 (+/-0.012) for {'C': 3, 'gamma': 1.9000000000000001}
```

```
0.954 (+/-0.010) for {'C': 4, 'gamma': 0.1}
0.963 (+/-0.013) for {'C': 4, 'gamma': 0.2}
0.969 (+/-0.011) for {'C': 4, 'gamma': 0.30000000000000004}
0.977 (+/-0.010) for {'C': 4, 'gamma': 0.4}
0.980 (+/-0.011) for {'C': 4, 'gamma': 0.5}
0.982 (+/-0.009) for {'C': 4, 'gamma': 0.6}
0.984 (+/-0.010) for {'C': 4, 'gamma': 0.7000000000000001}
0.984 (+/-0.011) for {'C': 4, 'gamma': 0.8}
0.986 (+/-0.012) for {'C': 4, 'gamma': 0.9}
0.986 (+/-0.013) for {'C': 4, 'gamma': 1.0}
0.987 (+/-0.013) for {'C': 4, 'gamma': 1.1}
0.988 (+/-0.013) for {'C': 4, 'gamma': 1.2000000000000002}
0.988 (+/-0.012) for {'C': 4, 'gamma': 1.3000000000000003}
0.989 (+/-0.012) for {'C': 4, 'gamma': 1.4000000000000001}
0.989 (+/-0.011) for {'C': 4, 'gamma': 1.5000000000000002}
0.989 (+/-0.010) for {'C': 4, 'gamma': 1.6}
0.989 (+/-0.010) for {'C': 4, 'gamma': 1.7000000000000002}
0.990 (+/-0.011) for {'C': 4, 'gamma': 1.8000000000000003}
0.990 (+/-0.011) for {'C': 4, 'gamma': 1.9000000000000001}
0.956 (+/-0.009) for {'C': 5, 'gamma': 0.1}
0.964 (+/-0.012) for {'C': 5, 'gamma': 0.2}
0.974 (+/-0.012) for {'C': 5, 'gamma': 0.30000000000000004}
0.979 (+/-0.011) for {'C': 5, 'gamma': 0.4}
0.981 (+/-0.010) for {'C': 5, 'gamma': 0.5}
0.983 (+/-0.010) for {'C': 5, 'gamma': 0.6}
0.984 (+/-0.010) for {'C': 5, 'gamma': 0.7000000000000001}
0.985 (+/-0.011) for {'C': 5, 'gamma': 0.8}
0.986 (+/-0.013) for {'C': 5, 'gamma': 0.9}
0.987 (+/-0.013) for {'C': 5, 'gamma': 1.0}
0.988 (+/-0.013) for {'C': 5, 'gamma': 1.1}
0.988 (+/-0.012) for {'C': 5, 'gamma': 1.2000000000000002}
0.989 (+/-0.011) for {'C': 5, 'gamma': 1.3000000000000003}
0.989 (+/-0.011) for {'C': 5, 'gamma': 1.4000000000000001}
0.989 (+/-0.011) for {'C': 5, 'gamma': 1.5000000000000002}
0.989 (+/-0.012) for {'C': 5, 'gamma': 1.6}
0.990 (+/-0.011) for {'C': 5, 'gamma': 1.7000000000000002}
0.990 (+/-0.010) for {'C': 5, 'gamma': 1.8000000000000003}
0.990 (+/-0.010) for {'C': 5, 'gamma': 1.9000000000000001}
0.957 (+/-0.011) for {'C': 6, 'gamma': 0.1}
0.966 (+/-0.011) for {'C': 6, 'gamma': 0.2}
0.976 (+/-0.011) for {'C': 6, 'gamma': 0.30000000000000004}
0.980 (+/-0.011) for {'C': 6, 'gamma': 0.4}
0.982 (+/-0.009) for {'C': 6, 'gamma': 0.5}
0.984 (+/-0.010) for {'C': 6, 'gamma': 0.6}
0.985 (+/-0.011) for {'C': 6, 'gamma': 0.7000000000000001}
0.986 (+/-0.013) for {'C': 6, 'gamma': 0.8}
0.987 (+/-0.013) for {'C': 6, 'gamma': 0.9}
0.987 (+/-0.013) for {'C': 6, 'gamma': 1.0}
0.988 (+/-0.012) for {'C': 6, 'gamma': 1.1}
0.989 (+/-0.012) for {'C': 6, 'gamma': 1.2000000000000002}
0.989 (+/-0.011) for {'C': 6, 'gamma': 1.3000000000000003}
0.989 (+/-0.011) for {'C': 6, 'gamma': 1.4000000000000001}
0.989 (+/-0.012) for {'C': 6, 'gamma': 1.5000000000000002}
0.990 (+/-0.011) for {'C': 6, 'gamma': 1.6}
0.990 (+/-0.011) for {'C': 6, 'gamma': 1.7000000000000002}
0.990 (+/-0.010) for {'C': 6, 'gamma': 1.8000000000000003}
0.990 (+/-0.012) for {'C': 6, 'gamma': 1.9000000000000001}
```

```
0.958 (+/-0.012) for {'C': 7, 'gamma': 0.1}
0.968 (+/-0.012) for {'C': 7, 'gamma': 0.2}
0.978 (+/-0.012) for {'C': 7, 'gamma': 0.30000000000000004}
0.980 (+/-0.011) for {'C': 7, 'gamma': 0.4}
0.982 (+/-0.009) for {'C': 7, 'gamma': 0.5}
0.984 (+/-0.010) for {'C': 7, 'gamma': 0.6}
0.985 (+/-0.012) for {'C': 7, 'gamma': 0.7000000000000001}
0.987 (+/-0.013) for {'C': 7, 'gamma': 0.8}
0.988 (+/-0.013) for {'C': 7, 'gamma': 0.9}
0.988 (+/-0.012) for {'C': 7, 'gamma': 1.0}
0.988 (+/-0.012) for {'C': 7, 'gamma': 1.1}
0.989 (+/-0.011) for {'C': 7, 'gamma': 1.2000000000000002}
0.988 (+/-0.011) for {'C': 7, 'gamma': 1.3000000000000003}
0.989 (+/-0.011) for {'C': 7, 'gamma': 1.4000000000000001}
0.990 (+/-0.011) for {'C': 7, 'gamma': 1.5000000000000002}
0.990 (+/-0.010) for {'C': 7, 'gamma': 1.6}
0.990 (+/-0.010) for {'C': 7, 'gamma': 1.7000000000000002}
0.990 (+/-0.011) for {'C': 7, 'gamma': 1.8000000000000003}
0.991 (+/-0.012) for {'C': 7, 'gamma': 1.9000000000000001}
0.959 (+/-0.012) for {'C': 8, 'gamma': 0.1}
0.971 (+/-0.013) for {'C': 8, 'gamma': 0.2}
0.978 (+/-0.011) for {'C': 8, 'gamma': 0.30000000000000004}
0.981 (+/-0.010) for {'C': 8, 'gamma': 0.4}
0.983 (+/-0.009) for {'C': 8, 'gamma': 0.5}
0.985 (+/-0.010) for {'C': 8, 'gamma': 0.6}
0.986 (+/-0.012) for {'C': 8, 'gamma': 0.7000000000000001}
0.987 (+/-0.013) for {'C': 8, 'gamma': 0.8}
0.987 (+/-0.012) for {'C': 8, 'gamma': 0.9}
0.988 (+/-0.012) for {'C': 8, 'gamma': 1.0}
0.988 (+/-0.011) for {'C': 8, 'gamma': 1.1}
0.988 (+/-0.011) for {'C': 8, 'gamma': 1.2000000000000002}
0.989 (+/-0.011) for {'C': 8, 'gamma': 1.3000000000000003}
0.990 (+/-0.011) for {'C': 8, 'gamma': 1.4000000000000001}
0.990 (+/-0.010) for {'C': 8, 'gamma': 1.5000000000000002}
0.990 (+/-0.010) for {'C': 8, 'gamma': 1.6}
0.991 (+/-0.011) for {'C': 8, 'gamma': 1.7000000000000002}
0.991 (+/-0.011) for {'C': 8, 'gamma': 1.8000000000000003}
0.991 (+/-0.011) for {'C': 8, 'gamma': 1.9000000000000001}
0.959 (+/-0.013) for {'C': 9, 'gamma': 0.1}
0.974 (+/-0.011) for {'C': 9, 'gamma': 0.2}
0.979 (+/-0.010) for {'C': 9, 'gamma': 0.30000000000000004}
0.981 (+/-0.009) for {'C': 9, 'gamma': 0.4}
0.984 (+/-0.010) for {'C': 9, 'gamma': 0.5}
0.985 (+/-0.011) for {'C': 9, 'gamma': 0.6}
0.987 (+/-0.012) for {'C': 9, 'gamma': 0.7000000000000001}
0.987 (+/-0.012) for {'C': 9, 'gamma': 0.8}
0.988 (+/-0.013) for {'C': 9, 'gamma': 0.9}
0.988 (+/-0.012) for {'C': 9, 'gamma': 1.0}
0.988 (+/-0.011) for {'C': 9, 'gamma': 1.1}
0.989 (+/-0.011) for {'C': 9, 'gamma': 1.2000000000000002}
0.989 (+/-0.011) for {'C': 9, 'gamma': 1.3000000000000003}
0.990 (+/-0.011) for {'C': 9, 'gamma': 1.4000000000000001}
0.991 (+/-0.010) for {'C': 9, 'gamma': 1.5000000000000002}
0.991 (+/-0.011) for {'C': 9, 'gamma': 1.6}
0.991 (+/-0.011) for {'C': 9, 'gamma': 1.7000000000000002}
0.991 (+/-0.011) for {'C': 9, 'gamma': 1.8000000000000003}
0.991 (+/-0.011) for {'C': 9, 'gamma': 1.9000000000000001}
```

```
0.960 (+/-0.013) for {'C': 10, 'gamma': 0.1}
0.975 (+/-0.009) for {'C': 10, 'gamma': 0.2}
0.979 (+/-0.010) for {'C': 10, 'gamma': 0.30000000000000004}
0.982 (+/-0.010) for {'C': 10, 'gamma': 0.4}
0.984 (+/-0.010) for {'C': 10, 'gamma': 0.5}
0.986 (+/-0.012) for {'C': 10, 'gamma': 0.6}
0.987 (+/-0.012) for {'C': 10, 'gamma': 0.7000000000000001}
0.987 (+/-0.012) for {'C': 10, 'gamma': 0.8}
0.988 (+/-0.012) for {'C': 10, 'gamma': 0.9}
0.988 (+/-0.012) for {'C': 10, 'gamma': 1.0}
0.989 (+/-0.011) for {'C': 10, 'gamma': 1.1}
0.989 (+/-0.011) for {'C': 10, 'gamma': 1.2000000000000002}
0.990 (+/-0.010) for {'C': 10, 'gamma': 1.3000000000000003}
0.990 (+/-0.010) for {'C': 10, 'gamma': 1.4000000000000001}
0.990 (+/-0.011) for {'C': 10, 'gamma': 1.5000000000000002}
0.991 (+/-0.011) for {'C': 10, 'gamma': 1.6}
0.991 (+/-0.011) for {'C': 10, 'gamma': 1.7000000000000002}
0.991 (+/-0.011) for {'C': 10, 'gamma': 1.8000000000000003}
0.991 (+/-0.011) for {'C': 10, 'gamma': 1.9000000000000001}
0.961 (+/-0.013) for {'C': 11, 'gamma': 0.1}
0.976 (+/-0.011) for {'C': 11, 'gamma': 0.2}
0.979 (+/-0.011) for {'C': 11, 'gamma': 0.30000000000000004}
0.982 (+/-0.010) for {'C': 11, 'gamma': 0.4}
0.984 (+/-0.011) for {'C': 11, 'gamma': 0.5}
0.986 (+/-0.011) for {'C': 11, 'gamma': 0.6}
0.987 (+/-0.012) for {'C': 11, 'gamma': 0.7000000000000001}
0.987 (+/-0.012) for {'C': 11, 'gamma': 0.8}
0.988 (+/-0.012) for {'C': 11, 'gamma': 0.9}
0.988 (+/-0.011) for {'C': 11, 'gamma': 1.0}
0.989 (+/-0.011) for {'C': 11, 'gamma': 1.1}
0.989 (+/-0.011) for {'C': 11, 'gamma': 1.2000000000000002}
0.990 (+/-0.010) for {'C': 11, 'gamma': 1.3000000000000003}
0.990 (+/-0.011) for {'C': 11, 'gamma': 1.4000000000000001}
0.991 (+/-0.011) for {'C': 11, 'gamma': 1.5000000000000002}
0.991 (+/-0.011) for {'C': 11, 'gamma': 1.6}
0.991 (+/-0.011) for {'C': 11, 'gamma': 1.7000000000000002}
0.991 (+/-0.011) for {'C': 11, 'gamma': 1.8000000000000003}
0.991 (+/-0.011) for {'C': 11, 'gamma': 1.9000000000000001}
0.962 (+/-0.014) for {'C': 12, 'gamma': 0.1}
0.976 (+/-0.011) for {'C': 12, 'gamma': 0.2}
0.980 (+/-0.010) for {'C': 12, 'gamma': 0.30000000000000004}
0.982 (+/-0.010) for {'C': 12, 'gamma': 0.4}
0.984 (+/-0.010) for {'C': 12, 'gamma': 0.5}
0.987 (+/-0.012) for {'C': 12, 'gamma': 0.6}
0.987 (+/-0.012) for {'C': 12, 'gamma': 0.7000000000000001}
0.987 (+/-0.012) for {'C': 12, 'gamma': 0.8}
0.988 (+/-0.012) for {'C': 12, 'gamma': 0.9}
0.988 (+/-0.011) for {'C': 12, 'gamma': 1.0}
0.989 (+/-0.011) for {'C': 12, 'gamma': 1.1}
0.990 (+/-0.010) for {'C': 12, 'gamma': 1.2000000000000002}
0.990 (+/-0.010) for {'C': 12, 'gamma': 1.3000000000000003}
0.991 (+/-0.011) for {'C': 12, 'gamma': 1.4000000000000001}
0.991 (+/-0.011) for {'C': 12, 'gamma': 1.5000000000000002}
0.991 (+/-0.011) for {'C': 12, 'gamma': 1.6}
0.991 (+/-0.011) for {'C': 12, 'gamma': 1.7000000000000002}
0.991 (+/-0.011) for {'C': 12, 'gamma': 1.8000000000000003}
0.991 (+/-0.010) for {'C': 12, 'gamma': 1.9000000000000001}
```

```
0.963 (+/-0.015) for {'C': 13, 'gamma': 0.1}
0.977 (+/-0.011) for {'C': 13, 'gamma': 0.2}
0.980 (+/-0.009) for {'C': 13, 'gamma': 0.30000000000000004}
0.983 (+/-0.009) for {'C': 13, 'gamma': 0.4}
0.985 (+/-0.010) for {'C': 13, 'gamma': 0.5}
0.987 (+/-0.012) for {'C': 13, 'gamma': 0.6}
0.987 (+/-0.012) for {'C': 13, 'gamma': 0.7000000000000001}
0.987 (+/-0.012) for {'C': 13, 'gamma': 0.8}
0.988 (+/-0.013) for {'C': 13, 'gamma': 0.9}
0.988 (+/-0.011) for {'C': 13, 'gamma': 1.0}
0.989 (+/-0.011) for {'C': 13, 'gamma': 1.1}
0.990 (+/-0.010) for {'C': 13, 'gamma': 1.2000000000000002}
0.990 (+/-0.010) for {'C': 13, 'gamma': 1.3000000000000003}
0.991 (+/-0.010) for {'C': 13, 'gamma': 1.4000000000000001}
0.991 (+/-0.011) for {'C': 13, 'gamma': 1.5000000000000002}
0.991 (+/-0.011) for {'C': 13, 'gamma': 1.6}
0.991 (+/-0.011) for {'C': 13, 'gamma': 1.7000000000000002}
0.991 (+/-0.010) for {'C': 13, 'gamma': 1.8000000000000003}
0.991 (+/-0.010) for {'C': 13, 'gamma': 1.9000000000000001}
0.963 (+/-0.015) for {'C': 14, 'gamma': 0.1}
0.976 (+/-0.011) for {'C': 14, 'gamma': 0.2}
0.981 (+/-0.009) for {'C': 14, 'gamma': 0.30000000000000004}
0.983 (+/-0.009) for {'C': 14, 'gamma': 0.4}
0.985 (+/-0.010) for {'C': 14, 'gamma': 0.5}
0.987 (+/-0.012) for {'C': 14, 'gamma': 0.6}
0.987 (+/-0.012) for {'C': 14, 'gamma': 0.7000000000000001}
0.987 (+/-0.011) for {'C': 14, 'gamma': 0.8}
0.988 (+/-0.012) for {'C': 14, 'gamma': 0.9}
0.988 (+/-0.011) for {'C': 14, 'gamma': 1.0}
0.990 (+/-0.011) for {'C': 14, 'gamma': 1.1}
0.990 (+/-0.009) for {'C': 14, 'gamma': 1.2000000000000002}
0.990 (+/-0.010) for {'C': 14, 'gamma': 1.3000000000000003}
0.991 (+/-0.010) for {'C': 14, 'gamma': 1.4000000000000001}
0.991 (+/-0.011) for {'C': 14, 'gamma': 1.5000000000000002}
0.991 (+/-0.010) for {'C': 14, 'gamma': 1.6}
0.991 (+/-0.010) for {'C': 14, 'gamma': 1.7000000000000002}
0.991 (+/-0.010) for {'C': 14, 'gamma': 1.8000000000000003}
0.991 (+/-0.010) for {'C': 14, 'gamma': 1.9000000000000001}
0.964 (+/-0.013) for {'C': 15, 'gamma': 0.1}
0.977 (+/-0.011) for {'C': 15, 'gamma': 0.2}
0.981 (+/-0.009) for {'C': 15, 'gamma': 0.30000000000000004}
0.983 (+/-0.010) for {'C': 15, 'gamma': 0.4}
0.986 (+/-0.011) for {'C': 15, 'gamma': 0.5}
0.987 (+/-0.012) for {'C': 15, 'gamma': 0.6}
0.987 (+/-0.012) for {'C': 15, 'gamma': 0.7000000000000001}
0.988 (+/-0.012) for {'C': 15, 'gamma': 0.8}
0.988 (+/-0.012) for {'C': 15, 'gamma': 0.9}
0.989 (+/-0.010) for {'C': 15, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.1}
0.990 (+/-0.010) for {'C': 15, 'gamma': 1.2000000000000002}
0.991 (+/-0.010) for {'C': 15, 'gamma': 1.3000000000000003}
0.991 (+/-0.010) for {'C': 15, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 15, 'gamma': 1.5000000000000002}
0.991 (+/-0.010) for {'C': 15, 'gamma': 1.6}
0.991 (+/-0.009) for {'C': 15, 'gamma': 1.7000000000000002}
0.991 (+/-0.009) for {'C': 15, 'gamma': 1.8000000000000003}
0.991 (+/-0.010) for {'C': 15, 'gamma': 1.9000000000000001}
```

```
0.964 (+/-0.013) for {'C': 16, 'gamma': 0.1}
0.978 (+/-0.010) for {'C': 16, 'gamma': 0.2}
0.981 (+/-0.009) for {'C': 16, 'gamma': 0.30000000000000004}
0.984 (+/-0.010) for {'C': 16, 'gamma': 0.4}
0.986 (+/-0.012) for {'C': 16, 'gamma': 0.5}
0.987 (+/-0.012) for {'C': 16, 'gamma': 0.6}
0.987 (+/-0.012) for {'C': 16, 'gamma': 0.7000000000000001}
0.987 (+/-0.012) for {'C': 16, 'gamma': 0.8}
0.988 (+/-0.011) for {'C': 16, 'gamma': 0.9}
0.989 (+/-0.010) for {'C': 16, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.1}
0.990 (+/-0.010) for {'C': 16, 'gamma': 1.2000000000000002}
0.991 (+/-0.009) for {'C': 16, 'gamma': 1.3000000000000003}
0.991 (+/-0.009) for {'C': 16, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 16, 'gamma': 1.5000000000000002}
0.991 (+/-0.009) for {'C': 16, 'gamma': 1.6}
0.991 (+/-0.009) for {'C': 16, 'gamma': 1.7000000000000002}
0.991 (+/-0.010) for {'C': 16, 'gamma': 1.8000000000000003}
0.991 (+/-0.010) for {'C': 16, 'gamma': 1.9000000000000001}
0.965 (+/-0.012) for {'C': 17, 'gamma': 0.1}
0.978 (+/-0.010) for {'C': 17, 'gamma': 0.2}
0.981 (+/-0.009) for {'C': 17, 'gamma': 0.30000000000000004}
0.984 (+/-0.010) for {'C': 17, 'gamma': 0.4}
0.986 (+/-0.012) for {'C': 17, 'gamma': 0.5}
0.987 (+/-0.012) for {'C': 17, 'gamma': 0.6}
0.987 (+/-0.012) for {'C': 17, 'gamma': 0.7000000000000001}
0.987 (+/-0.013) for {'C': 17, 'gamma': 0.8}
0.989 (+/-0.010) for {'C': 17, 'gamma': 0.9}
0.989 (+/-0.010) for {'C': 17, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.1}
0.990 (+/-0.010) for {'C': 17, 'gamma': 1.2000000000000002}
0.991 (+/-0.009) for {'C': 17, 'gamma': 1.3000000000000003}
0.991 (+/-0.009) for {'C': 17, 'gamma': 1.4000000000000001}
0.991 (+/-0.009) for {'C': 17, 'gamma': 1.5000000000000002}
0.991 (+/-0.008) for {'C': 17, 'gamma': 1.6}
0.990 (+/-0.010) for {'C': 17, 'gamma': 1.7000000000000002}
0.991 (+/-0.010) for {'C': 17, 'gamma': 1.8000000000000003}
0.991 (+/-0.009) for {'C': 17, 'gamma': 1.9000000000000001}
0.966 (+/-0.012) for {'C': 18, 'gamma': 0.1}
0.978 (+/-0.009) for {'C': 18, 'gamma': 0.2}
0.981 (+/-0.009) for {'C': 18, 'gamma': 0.30000000000000004}
0.984 (+/-0.010) for {'C': 18, 'gamma': 0.4}
0.987 (+/-0.012) for {'C': 18, 'gamma': 0.5}
0.987 (+/-0.012) for {'C': 18, 'gamma': 0.6}
0.987 (+/-0.012) for {'C': 18, 'gamma': 0.7000000000000001}
0.988 (+/-0.012) for {'C': 18, 'gamma': 0.8}
0.989 (+/-0.011) for {'C': 18, 'gamma': 0.9}
0.989 (+/-0.010) for {'C': 18, 'gamma': 1.0}
0.990 (+/-0.009) for {'C': 18, 'gamma': 1.1}
0.991 (+/-0.009) for {'C': 18, 'gamma': 1.2000000000000002}
0.991 (+/-0.009) for {'C': 18, 'gamma': 1.3000000000000003}
0.991 (+/-0.009) for {'C': 18, 'gamma': 1.4000000000000001}
0.991 (+/-0.008) for {'C': 18, 'gamma': 1.5000000000000002}
0.990 (+/-0.010) for {'C': 18, 'gamma': 1.6}
0.990 (+/-0.010) for {'C': 18, 'gamma': 1.7000000000000002}
0.991 (+/-0.009) for {'C': 18, 'gamma': 1.8000000000000003}
0.991 (+/-0.009) for {'C': 18, 'gamma': 1.9000000000000001}
```

```
0.967 (+/-0.012) for {'C': 19, 'gamma': 0.1}
0.978 (+/-0.009) for {'C': 19, 'gamma': 0.2}
0.982 (+/-0.009) for {'C': 19, 'gamma': 0.30000000000000004}
0.984 (+/-0.010) for {'C': 19, 'gamma': 0.4}
0.986 (+/-0.012) for {'C': 19, 'gamma': 0.5}
0.987 (+/-0.012) for {'C': 19, 'gamma': 0.6}
0.987 (+/-0.012) for {'C': 19, 'gamma': 0.7000000000000001}
0.988 (+/-0.011) for {'C': 19, 'gamma': 0.8}
0.989 (+/-0.010) for {'C': 19, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.0}
0.990 (+/-0.009) for {'C': 19, 'gamma': 1.1}
0.991 (+/-0.009) for {'C': 19, 'gamma': 1.2000000000000002}
0.991 (+/-0.009) for {'C': 19, 'gamma': 1.3000000000000003}
0.991 (+/-0.009) for {'C': 19, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.5000000000000002}
0.990 (+/-0.010) for {'C': 19, 'gamma': 1.6}
0.990 (+/-0.010) for {'C': 19, 'gamma': 1.7000000000000002}
0.991 (+/-0.009) for {'C': 19, 'gamma': 1.8000000000000003}
0.992 (+/-0.008) for {'C': 19, 'gamma': 1.9000000000000001}
C 19
cache_size 200
class_weight None
coef0 0.0
decision_function_shape ovr
degree 3
gamma 1.9000000000000001
kernel rbf
max_iter -1
probability True
random_state None
shrinking True
tol 0.001
verbose False
```

v)feature: Genus--with normalization

In [7]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
import pandas as pd
import numpy as np
import warnings

warnings.filterwarnings('ignore')

Folder_Path = r'/Users/yqh/Desktop/'

a = pd.read_csv(Folder_Path + 'norm_training_data.csv', usecols=range(0,
 22), header=0)
b = pd.read_csv(Folder_Path + 'norm_training_data.csv', usecols=[23], he
ader=0)

model = SVC(kernel='rbf', probability=True)
param_grid = {'C': range(1, 20), 'gamma': np.arange(0.1, 2, 0.1)}
grid_search = GridSearchCV(model, param_grid, scoring='accuracy', cv=10,
 n_jobs=1, verbose=1)
grid_search.fit(a, b)
best_parameters = grid_search.best_estimator_.get_params()

means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, grid_search.cv_results_['param
s']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))

for para, val in best_parameters.items():
    print(para, val)
```

```
Fitting 10 folds for each of 361 candidates, totalling 3610 fits
[Parallel(n_jobs=1)]: Done 3610 out of 3610 | elapsed: 44.9min finished
```

```
0.929 (+/-0.014) for {'C': 1, 'gamma': 0.1}
0.945 (+/-0.018) for {'C': 1, 'gamma': 0.2}
0.954 (+/-0.018) for {'C': 1, 'gamma': 0.30000000000000004}
0.963 (+/-0.016) for {'C': 1, 'gamma': 0.4}
0.969 (+/-0.013) for {'C': 1, 'gamma': 0.5}
0.972 (+/-0.013) for {'C': 1, 'gamma': 0.6}
0.975 (+/-0.012) for {'C': 1, 'gamma': 0.7000000000000001}
0.977 (+/-0.012) for {'C': 1, 'gamma': 0.8}
0.979 (+/-0.011) for {'C': 1, 'gamma': 0.9}
0.980 (+/-0.012) for {'C': 1, 'gamma': 1.0}
0.981 (+/-0.012) for {'C': 1, 'gamma': 1.1}
0.982 (+/-0.012) for {'C': 1, 'gamma': 1.2000000000000002}
0.983 (+/-0.011) for {'C': 1, 'gamma': 1.3000000000000003}
0.983 (+/-0.011) for {'C': 1, 'gamma': 1.4000000000000001}
0.984 (+/-0.010) for {'C': 1, 'gamma': 1.5000000000000002}
0.985 (+/-0.010) for {'C': 1, 'gamma': 1.6}
0.985 (+/-0.011) for {'C': 1, 'gamma': 1.7000000000000002}
0.985 (+/-0.010) for {'C': 1, 'gamma': 1.8000000000000003}
0.986 (+/-0.009) for {'C': 1, 'gamma': 1.9000000000000001}
0.944 (+/-0.019) for {'C': 2, 'gamma': 0.1}
0.958 (+/-0.016) for {'C': 2, 'gamma': 0.2}
0.971 (+/-0.012) for {'C': 2, 'gamma': 0.30000000000000004}
0.975 (+/-0.011) for {'C': 2, 'gamma': 0.4}
0.978 (+/-0.010) for {'C': 2, 'gamma': 0.5}
0.980 (+/-0.012) for {'C': 2, 'gamma': 0.6}
0.983 (+/-0.013) for {'C': 2, 'gamma': 0.7000000000000001}
0.983 (+/-0.011) for {'C': 2, 'gamma': 0.8}
0.984 (+/-0.011) for {'C': 2, 'gamma': 0.9}
0.985 (+/-0.011) for {'C': 2, 'gamma': 1.0}
0.986 (+/-0.010) for {'C': 2, 'gamma': 1.1}
0.986 (+/-0.010) for {'C': 2, 'gamma': 1.2000000000000002}
0.987 (+/-0.010) for {'C': 2, 'gamma': 1.3000000000000003}
0.988 (+/-0.009) for {'C': 2, 'gamma': 1.4000000000000001}
0.988 (+/-0.008) for {'C': 2, 'gamma': 1.5000000000000002}
0.989 (+/-0.008) for {'C': 2, 'gamma': 1.6}
0.989 (+/-0.007) for {'C': 2, 'gamma': 1.7000000000000002}
0.989 (+/-0.008) for {'C': 2, 'gamma': 1.8000000000000003}
0.989 (+/-0.007) for {'C': 2, 'gamma': 1.9000000000000001}
0.952 (+/-0.021) for {'C': 3, 'gamma': 0.1}
0.968 (+/-0.012) for {'C': 3, 'gamma': 0.2}
0.975 (+/-0.011) for {'C': 3, 'gamma': 0.30000000000000004}
0.979 (+/-0.013) for {'C': 3, 'gamma': 0.4}
0.981 (+/-0.012) for {'C': 3, 'gamma': 0.5}
0.982 (+/-0.012) for {'C': 3, 'gamma': 0.6}
0.983 (+/-0.011) for {'C': 3, 'gamma': 0.7000000000000001}
0.985 (+/-0.011) for {'C': 3, 'gamma': 0.8}
0.986 (+/-0.011) for {'C': 3, 'gamma': 0.9}
0.986 (+/-0.011) for {'C': 3, 'gamma': 1.0}
0.987 (+/-0.009) for {'C': 3, 'gamma': 1.1}
0.988 (+/-0.008) for {'C': 3, 'gamma': 1.2000000000000002}
0.988 (+/-0.008) for {'C': 3, 'gamma': 1.3000000000000003}
0.989 (+/-0.008) for {'C': 3, 'gamma': 1.4000000000000001}
0.989 (+/-0.008) for {'C': 3, 'gamma': 1.5000000000000002}
0.989 (+/-0.008) for {'C': 3, 'gamma': 1.6}
0.990 (+/-0.007) for {'C': 3, 'gamma': 1.7000000000000002}
0.990 (+/-0.007) for {'C': 3, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 3, 'gamma': 1.9000000000000001}
```

```
0.956 (+/-0.017) for {'C': 4, 'gamma': 0.1}
0.972 (+/-0.010) for {'C': 4, 'gamma': 0.2}
0.977 (+/-0.013) for {'C': 4, 'gamma': 0.30000000000000004}
0.979 (+/-0.012) for {'C': 4, 'gamma': 0.4}
0.982 (+/-0.013) for {'C': 4, 'gamma': 0.5}
0.983 (+/-0.012) for {'C': 4, 'gamma': 0.6}
0.984 (+/-0.011) for {'C': 4, 'gamma': 0.7000000000000001}
0.985 (+/-0.011) for {'C': 4, 'gamma': 0.8}
0.987 (+/-0.010) for {'C': 4, 'gamma': 0.9}
0.987 (+/-0.009) for {'C': 4, 'gamma': 1.0}
0.988 (+/-0.008) for {'C': 4, 'gamma': 1.1}
0.989 (+/-0.008) for {'C': 4, 'gamma': 1.2000000000000002}
0.989 (+/-0.009) for {'C': 4, 'gamma': 1.3000000000000003}
0.989 (+/-0.008) for {'C': 4, 'gamma': 1.4000000000000001}
0.990 (+/-0.007) for {'C': 4, 'gamma': 1.5000000000000002}
0.990 (+/-0.007) for {'C': 4, 'gamma': 1.6}
0.990 (+/-0.007) for {'C': 4, 'gamma': 1.7000000000000002}
0.990 (+/-0.007) for {'C': 4, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 4, 'gamma': 1.9000000000000001}
0.959 (+/-0.016) for {'C': 5, 'gamma': 0.1}
0.974 (+/-0.012) for {'C': 5, 'gamma': 0.2}
0.978 (+/-0.013) for {'C': 5, 'gamma': 0.30000000000000004}
0.981 (+/-0.013) for {'C': 5, 'gamma': 0.4}
0.982 (+/-0.013) for {'C': 5, 'gamma': 0.5}
0.984 (+/-0.011) for {'C': 5, 'gamma': 0.6}
0.985 (+/-0.011) for {'C': 5, 'gamma': 0.7000000000000001}
0.987 (+/-0.010) for {'C': 5, 'gamma': 0.8}
0.988 (+/-0.009) for {'C': 5, 'gamma': 0.9}
0.988 (+/-0.007) for {'C': 5, 'gamma': 1.0}
0.989 (+/-0.008) for {'C': 5, 'gamma': 1.1}
0.989 (+/-0.009) for {'C': 5, 'gamma': 1.2000000000000002}
0.989 (+/-0.008) for {'C': 5, 'gamma': 1.3000000000000003}
0.989 (+/-0.008) for {'C': 5, 'gamma': 1.4000000000000001}
0.989 (+/-0.008) for {'C': 5, 'gamma': 1.5000000000000002}
0.989 (+/-0.008) for {'C': 5, 'gamma': 1.6}
0.990 (+/-0.007) for {'C': 5, 'gamma': 1.7000000000000002}
0.990 (+/-0.007) for {'C': 5, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 5, 'gamma': 1.9000000000000001}
0.961 (+/-0.014) for {'C': 6, 'gamma': 0.1}
0.976 (+/-0.013) for {'C': 6, 'gamma': 0.2}
0.979 (+/-0.013) for {'C': 6, 'gamma': 0.30000000000000004}
0.981 (+/-0.013) for {'C': 6, 'gamma': 0.4}
0.982 (+/-0.013) for {'C': 6, 'gamma': 0.5}
0.984 (+/-0.011) for {'C': 6, 'gamma': 0.6}
0.986 (+/-0.009) for {'C': 6, 'gamma': 0.7000000000000001}
0.988 (+/-0.009) for {'C': 6, 'gamma': 0.8}
0.989 (+/-0.007) for {'C': 6, 'gamma': 0.9}
0.989 (+/-0.008) for {'C': 6, 'gamma': 1.0}
0.988 (+/-0.009) for {'C': 6, 'gamma': 1.1}
0.989 (+/-0.008) for {'C': 6, 'gamma': 1.2000000000000002}
0.989 (+/-0.008) for {'C': 6, 'gamma': 1.3000000000000003}
0.989 (+/-0.008) for {'C': 6, 'gamma': 1.4000000000000001}
0.989 (+/-0.007) for {'C': 6, 'gamma': 1.5000000000000002}
0.990 (+/-0.007) for {'C': 6, 'gamma': 1.6}
0.990 (+/-0.007) for {'C': 6, 'gamma': 1.7000000000000002}
0.990 (+/-0.007) for {'C': 6, 'gamma': 1.8000000000000003}
0.991 (+/-0.007) for {'C': 6, 'gamma': 1.9000000000000001}
```

```
0.964 (+/-0.010) for {'C': 7, 'gamma': 0.1}
0.977 (+/-0.013) for {'C': 7, 'gamma': 0.2}
0.979 (+/-0.014) for {'C': 7, 'gamma': 0.30000000000000004}
0.981 (+/-0.013) for {'C': 7, 'gamma': 0.4}
0.983 (+/-0.011) for {'C': 7, 'gamma': 0.5}
0.986 (+/-0.008) for {'C': 7, 'gamma': 0.6}
0.987 (+/-0.008) for {'C': 7, 'gamma': 0.7000000000000001}
0.988 (+/-0.007) for {'C': 7, 'gamma': 0.8}
0.988 (+/-0.008) for {'C': 7, 'gamma': 0.9}
0.988 (+/-0.009) for {'C': 7, 'gamma': 1.0}
0.989 (+/-0.008) for {'C': 7, 'gamma': 1.1}
0.989 (+/-0.008) for {'C': 7, 'gamma': 1.2000000000000002}
0.989 (+/-0.008) for {'C': 7, 'gamma': 1.3000000000000003}
0.989 (+/-0.007) for {'C': 7, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 7, 'gamma': 1.5000000000000002}
0.990 (+/-0.007) for {'C': 7, 'gamma': 1.6}
0.991 (+/-0.007) for {'C': 7, 'gamma': 1.7000000000000002}
0.991 (+/-0.007) for {'C': 7, 'gamma': 1.8000000000000003}
0.991 (+/-0.007) for {'C': 7, 'gamma': 1.9000000000000001}
0.969 (+/-0.009) for {'C': 8, 'gamma': 0.1}
0.976 (+/-0.014) for {'C': 8, 'gamma': 0.2}
0.979 (+/-0.012) for {'C': 8, 'gamma': 0.30000000000000004}
0.982 (+/-0.014) for {'C': 8, 'gamma': 0.4}
0.984 (+/-0.009) for {'C': 8, 'gamma': 0.5}
0.986 (+/-0.008) for {'C': 8, 'gamma': 0.6}
0.988 (+/-0.007) for {'C': 8, 'gamma': 0.7000000000000001}
0.988 (+/-0.007) for {'C': 8, 'gamma': 0.8}
0.988 (+/-0.008) for {'C': 8, 'gamma': 0.9}
0.988 (+/-0.008) for {'C': 8, 'gamma': 1.0}
0.989 (+/-0.008) for {'C': 8, 'gamma': 1.1}
0.989 (+/-0.008) for {'C': 8, 'gamma': 1.2000000000000002}
0.989 (+/-0.007) for {'C': 8, 'gamma': 1.3000000000000003}
0.990 (+/-0.007) for {'C': 8, 'gamma': 1.4000000000000001}
0.990 (+/-0.007) for {'C': 8, 'gamma': 1.5000000000000002}
0.991 (+/-0.007) for {'C': 8, 'gamma': 1.6}
0.991 (+/-0.007) for {'C': 8, 'gamma': 1.7000000000000002}
0.991 (+/-0.007) for {'C': 8, 'gamma': 1.8000000000000003}
0.992 (+/-0.007) for {'C': 8, 'gamma': 1.9000000000000001}
0.972 (+/-0.012) for {'C': 9, 'gamma': 0.1}
0.977 (+/-0.014) for {'C': 9, 'gamma': 0.2}
0.981 (+/-0.013) for {'C': 9, 'gamma': 0.30000000000000004}
0.982 (+/-0.014) for {'C': 9, 'gamma': 0.4}
0.985 (+/-0.009) for {'C': 9, 'gamma': 0.5}
0.987 (+/-0.009) for {'C': 9, 'gamma': 0.6}
0.987 (+/-0.007) for {'C': 9, 'gamma': 0.7000000000000001}
0.987 (+/-0.007) for {'C': 9, 'gamma': 0.8}
0.988 (+/-0.007) for {'C': 9, 'gamma': 0.9}
0.988 (+/-0.007) for {'C': 9, 'gamma': 1.0}
0.989 (+/-0.007) for {'C': 9, 'gamma': 1.1}
0.990 (+/-0.007) for {'C': 9, 'gamma': 1.2000000000000002}
0.990 (+/-0.007) for {'C': 9, 'gamma': 1.3000000000000003}
0.990 (+/-0.007) for {'C': 9, 'gamma': 1.4000000000000001}
0.991 (+/-0.007) for {'C': 9, 'gamma': 1.5000000000000002}
0.991 (+/-0.008) for {'C': 9, 'gamma': 1.6}
0.991 (+/-0.007) for {'C': 9, 'gamma': 1.7000000000000002}
0.991 (+/-0.007) for {'C': 9, 'gamma': 1.8000000000000003}
0.991 (+/-0.007) for {'C': 9, 'gamma': 1.9000000000000001}
```

```
0.973 (+/-0.011) for {'C': 10, 'gamma': 0.1}
0.978 (+/-0.013) for {'C': 10, 'gamma': 0.2}
0.981 (+/-0.013) for {'C': 10, 'gamma': 0.30000000000000004}
0.983 (+/-0.012) for {'C': 10, 'gamma': 0.4}
0.986 (+/-0.008) for {'C': 10, 'gamma': 0.5}
0.988 (+/-0.007) for {'C': 10, 'gamma': 0.6}
0.987 (+/-0.007) for {'C': 10, 'gamma': 0.7000000000000001}
0.988 (+/-0.007) for {'C': 10, 'gamma': 0.8}
0.988 (+/-0.007) for {'C': 10, 'gamma': 0.9}
0.989 (+/-0.006) for {'C': 10, 'gamma': 1.0}
0.990 (+/-0.007) for {'C': 10, 'gamma': 1.1}
0.990 (+/-0.007) for {'C': 10, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 10, 'gamma': 1.3000000000000003}
0.991 (+/-0.008) for {'C': 10, 'gamma': 1.4000000000000001}
0.991 (+/-0.008) for {'C': 10, 'gamma': 1.5000000000000002}
0.991 (+/-0.007) for {'C': 10, 'gamma': 1.6}
0.992 (+/-0.006) for {'C': 10, 'gamma': 1.7000000000000002}
0.992 (+/-0.007) for {'C': 10, 'gamma': 1.8000000000000003}
0.991 (+/-0.007) for {'C': 10, 'gamma': 1.9000000000000001}
0.973 (+/-0.011) for {'C': 11, 'gamma': 0.1}
0.977 (+/-0.014) for {'C': 11, 'gamma': 0.2}
0.981 (+/-0.014) for {'C': 11, 'gamma': 0.30000000000000004}
0.984 (+/-0.009) for {'C': 11, 'gamma': 0.4}
0.986 (+/-0.008) for {'C': 11, 'gamma': 0.5}
0.988 (+/-0.006) for {'C': 11, 'gamma': 0.6}
0.987 (+/-0.007) for {'C': 11, 'gamma': 0.7000000000000001}
0.987 (+/-0.007) for {'C': 11, 'gamma': 0.8}
0.988 (+/-0.007) for {'C': 11, 'gamma': 0.9}
0.989 (+/-0.006) for {'C': 11, 'gamma': 1.0}
0.990 (+/-0.007) for {'C': 11, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 11, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 11, 'gamma': 1.3000000000000003}
0.991 (+/-0.007) for {'C': 11, 'gamma': 1.4000000000000001}
0.991 (+/-0.007) for {'C': 11, 'gamma': 1.5000000000000002}
0.991 (+/-0.006) for {'C': 11, 'gamma': 1.6}
0.991 (+/-0.007) for {'C': 11, 'gamma': 1.7000000000000002}
0.991 (+/-0.008) for {'C': 11, 'gamma': 1.8000000000000003}
0.991 (+/-0.006) for {'C': 11, 'gamma': 1.9000000000000001}
0.973 (+/-0.013) for {'C': 12, 'gamma': 0.1}
0.978 (+/-0.013) for {'C': 12, 'gamma': 0.2}
0.981 (+/-0.014) for {'C': 12, 'gamma': 0.30000000000000004}
0.985 (+/-0.009) for {'C': 12, 'gamma': 0.4}
0.986 (+/-0.006) for {'C': 12, 'gamma': 0.5}
0.987 (+/-0.007) for {'C': 12, 'gamma': 0.6}
0.988 (+/-0.007) for {'C': 12, 'gamma': 0.7000000000000001}
0.987 (+/-0.007) for {'C': 12, 'gamma': 0.8}
0.989 (+/-0.008) for {'C': 12, 'gamma': 0.9}
0.989 (+/-0.007) for {'C': 12, 'gamma': 1.0}
0.990 (+/-0.007) for {'C': 12, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.2000000000000002}
0.991 (+/-0.006) for {'C': 12, 'gamma': 1.3000000000000003}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.4000000000000001}
0.991 (+/-0.007) for {'C': 12, 'gamma': 1.5000000000000002}
0.991 (+/-0.007) for {'C': 12, 'gamma': 1.6}
0.991 (+/-0.007) for {'C': 12, 'gamma': 1.7000000000000002}
0.991 (+/-0.006) for {'C': 12, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 12, 'gamma': 1.9000000000000001}
```

```
0.974 (+/-0.014) for {'C': 13, 'gamma': 0.1}
0.979 (+/-0.013) for {'C': 13, 'gamma': 0.2}
0.982 (+/-0.014) for {'C': 13, 'gamma': 0.30000000000000004}
0.985 (+/-0.009) for {'C': 13, 'gamma': 0.4}
0.986 (+/-0.007) for {'C': 13, 'gamma': 0.5}
0.987 (+/-0.007) for {'C': 13, 'gamma': 0.6}
0.988 (+/-0.007) for {'C': 13, 'gamma': 0.7000000000000001}
0.988 (+/-0.008) for {'C': 13, 'gamma': 0.8}
0.989 (+/-0.007) for {'C': 13, 'gamma': 0.9}
0.990 (+/-0.007) for {'C': 13, 'gamma': 1.0}
0.990 (+/-0.007) for {'C': 13, 'gamma': 1.1}
0.990 (+/-0.007) for {'C': 13, 'gamma': 1.2000000000000002}
0.991 (+/-0.007) for {'C': 13, 'gamma': 1.3000000000000003}
0.991 (+/-0.007) for {'C': 13, 'gamma': 1.4000000000000001}
0.990 (+/-0.006) for {'C': 13, 'gamma': 1.5000000000000002}
0.991 (+/-0.006) for {'C': 13, 'gamma': 1.6}
0.991 (+/-0.006) for {'C': 13, 'gamma': 1.7000000000000002}
0.991 (+/-0.007) for {'C': 13, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 13, 'gamma': 1.9000000000000001}
0.975 (+/-0.015) for {'C': 14, 'gamma': 0.1}
0.979 (+/-0.012) for {'C': 14, 'gamma': 0.2}
0.982 (+/-0.013) for {'C': 14, 'gamma': 0.30000000000000004}
0.985 (+/-0.009) for {'C': 14, 'gamma': 0.4}
0.986 (+/-0.007) for {'C': 14, 'gamma': 0.5}
0.988 (+/-0.008) for {'C': 14, 'gamma': 0.6}
0.988 (+/-0.008) for {'C': 14, 'gamma': 0.7000000000000001}
0.989 (+/-0.009) for {'C': 14, 'gamma': 0.8}
0.990 (+/-0.007) for {'C': 14, 'gamma': 0.9}
0.990 (+/-0.007) for {'C': 14, 'gamma': 1.0}
0.990 (+/-0.007) for {'C': 14, 'gamma': 1.1}
0.990 (+/-0.007) for {'C': 14, 'gamma': 1.2000000000000002}
0.991 (+/-0.007) for {'C': 14, 'gamma': 1.3000000000000003}
0.990 (+/-0.007) for {'C': 14, 'gamma': 1.4000000000000001}
0.991 (+/-0.006) for {'C': 14, 'gamma': 1.5000000000000002}
0.991 (+/-0.006) for {'C': 14, 'gamma': 1.6}
0.991 (+/-0.007) for {'C': 14, 'gamma': 1.7000000000000002}
0.991 (+/-0.008) for {'C': 14, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 14, 'gamma': 1.9000000000000001}
0.975 (+/-0.015) for {'C': 15, 'gamma': 0.1}
0.979 (+/-0.012) for {'C': 15, 'gamma': 0.2}
0.983 (+/-0.012) for {'C': 15, 'gamma': 0.30000000000000004}
0.986 (+/-0.009) for {'C': 15, 'gamma': 0.4}
0.986 (+/-0.006) for {'C': 15, 'gamma': 0.5}
0.988 (+/-0.007) for {'C': 15, 'gamma': 0.6}
0.988 (+/-0.008) for {'C': 15, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 15, 'gamma': 0.8}
0.990 (+/-0.007) for {'C': 15, 'gamma': 0.9}
0.990 (+/-0.007) for {'C': 15, 'gamma': 1.0}
0.990 (+/-0.007) for {'C': 15, 'gamma': 1.1}
0.991 (+/-0.007) for {'C': 15, 'gamma': 1.2000000000000002}
0.990 (+/-0.007) for {'C': 15, 'gamma': 1.3000000000000003}
0.991 (+/-0.007) for {'C': 15, 'gamma': 1.4000000000000001}
0.990 (+/-0.007) for {'C': 15, 'gamma': 1.5000000000000002}
0.991 (+/-0.007) for {'C': 15, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 15, 'gamma': 1.7000000000000002}
0.991 (+/-0.008) for {'C': 15, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.9000000000000001}
```

```
0.975 (+/-0.016) for {'C': 16, 'gamma': 0.1}
0.979 (+/-0.012) for {'C': 16, 'gamma': 0.2}
0.984 (+/-0.010) for {'C': 16, 'gamma': 0.30000000000000004}
0.986 (+/-0.009) for {'C': 16, 'gamma': 0.4}
0.986 (+/-0.006) for {'C': 16, 'gamma': 0.5}
0.988 (+/-0.007) for {'C': 16, 'gamma': 0.6}
0.988 (+/-0.008) for {'C': 16, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 16, 'gamma': 0.8}
0.990 (+/-0.008) for {'C': 16, 'gamma': 0.9}
0.990 (+/-0.007) for {'C': 16, 'gamma': 1.0}
0.990 (+/-0.007) for {'C': 16, 'gamma': 1.1}
0.990 (+/-0.006) for {'C': 16, 'gamma': 1.2000000000000002}
0.991 (+/-0.007) for {'C': 16, 'gamma': 1.3000000000000003}
0.991 (+/-0.007) for {'C': 16, 'gamma': 1.4000000000000001}
0.990 (+/-0.007) for {'C': 16, 'gamma': 1.5000000000000002}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.6}
0.991 (+/-0.008) for {'C': 16, 'gamma': 1.7000000000000002}
0.991 (+/-0.008) for {'C': 16, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.9000000000000001}
0.975 (+/-0.017) for {'C': 17, 'gamma': 0.1}
0.979 (+/-0.014) for {'C': 17, 'gamma': 0.2}
0.985 (+/-0.009) for {'C': 17, 'gamma': 0.30000000000000004}
0.985 (+/-0.007) for {'C': 17, 'gamma': 0.4}
0.986 (+/-0.007) for {'C': 17, 'gamma': 0.5}
0.988 (+/-0.008) for {'C': 17, 'gamma': 0.6}
0.988 (+/-0.009) for {'C': 17, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 17, 'gamma': 0.8}
0.990 (+/-0.007) for {'C': 17, 'gamma': 0.9}
0.990 (+/-0.007) for {'C': 17, 'gamma': 1.0}
0.990 (+/-0.006) for {'C': 17, 'gamma': 1.1}
0.990 (+/-0.006) for {'C': 17, 'gamma': 1.2000000000000002}
0.990 (+/-0.007) for {'C': 17, 'gamma': 1.3000000000000003}
0.991 (+/-0.007) for {'C': 17, 'gamma': 1.4000000000000001}
0.990 (+/-0.007) for {'C': 17, 'gamma': 1.5000000000000002}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.9000000000000001}
0.975 (+/-0.017) for {'C': 18, 'gamma': 0.1}
0.979 (+/-0.014) for {'C': 18, 'gamma': 0.2}
0.985 (+/-0.010) for {'C': 18, 'gamma': 0.30000000000000004}
0.986 (+/-0.008) for {'C': 18, 'gamma': 0.4}
0.987 (+/-0.007) for {'C': 18, 'gamma': 0.5}
0.987 (+/-0.008) for {'C': 18, 'gamma': 0.6}
0.989 (+/-0.009) for {'C': 18, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 18, 'gamma': 0.8}
0.990 (+/-0.007) for {'C': 18, 'gamma': 0.9}
0.990 (+/-0.007) for {'C': 18, 'gamma': 1.0}
0.990 (+/-0.006) for {'C': 18, 'gamma': 1.1}
0.990 (+/-0.006) for {'C': 18, 'gamma': 1.2000000000000002}
0.990 (+/-0.007) for {'C': 18, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.5000000000000002}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.7000000000000002}
0.990 (+/-0.009) for {'C': 18, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.9000000000000001}
```

```
0.975 (+/-0.016) for {'C': 19, 'gamma': 0.1}
0.980 (+/-0.014) for {'C': 19, 'gamma': 0.2}
0.985 (+/-0.010) for {'C': 19, 'gamma': 0.30000000000000004}
0.985 (+/-0.007) for {'C': 19, 'gamma': 0.4}
0.987 (+/-0.008) for {'C': 19, 'gamma': 0.5}
0.987 (+/-0.009) for {'C': 19, 'gamma': 0.6}
0.989 (+/-0.009) for {'C': 19, 'gamma': 0.7000000000000001}
0.990 (+/-0.008) for {'C': 19, 'gamma': 0.8}
0.990 (+/-0.007) for {'C': 19, 'gamma': 0.9}
0.990 (+/-0.006) for {'C': 19, 'gamma': 1.0}
0.990 (+/-0.006) for {'C': 19, 'gamma': 1.1}
0.990 (+/-0.006) for {'C': 19, 'gamma': 1.2000000000000002}
0.990 (+/-0.007) for {'C': 19, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.5000000000000002}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.6}
0.990 (+/-0.009) for {'C': 19, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.9000000000000001}
C 8
cache_size 200
class_weight None
coef0 0.0
decision_function_shape ovr
degree 3
gamma 1.9000000000000001
kernel rbf
max_iter -1
probability True
random_state None
shrinking True
tol 0.001
verbose False
```

vi)feature:Species--with normalization

In [9]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
import pandas as pd
import numpy as np

Folder_Path = r'/Users/yqh/Desktop/'

b=[]
a = pd.read_csv(Folder_Path + 'norm_training_data.csv', usecols=range(0,
 22), header=0)
b = pd.read_csv(Folder_Path + 'norm_training_data.csv', usecols=[24], he
ader=0)

model = SVC(kernel='rbf', probability=True)
param_grid = {'C': range(1, 20), 'gamma': np.arange(0.1, 2, 0.1)}
grid_search = GridSearchCV(model, param_grid, scoring='accuracy', cv=10,
 n_jobs=1, verbose=1)
grid_search.fit(a, b)
best_parameters = grid_search.best_estimator_.get_params()

means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, grid_search.cv_results_['param
s']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))

for para, val in best_parameters.items():
    print(para, val)
```

```
Fitting 10 folds for each of 361 candidates, totalling 3610 fits
[Parallel(n_jobs=1)]: Done 3610 out of 3610 | elapsed: 37.6min finished
```

```
0.936 (+/-0.019) for {'C': 1, 'gamma': 0.1}
0.954 (+/-0.016) for {'C': 1, 'gamma': 0.2}
0.965 (+/-0.017) for {'C': 1, 'gamma': 0.30000000000000004}
0.972 (+/-0.011) for {'C': 1, 'gamma': 0.4}
0.974 (+/-0.012) for {'C': 1, 'gamma': 0.5}
0.978 (+/-0.014) for {'C': 1, 'gamma': 0.6}
0.978 (+/-0.012) for {'C': 1, 'gamma': 0.7000000000000001}
0.979 (+/-0.011) for {'C': 1, 'gamma': 0.8}
0.980 (+/-0.010) for {'C': 1, 'gamma': 0.9}
0.981 (+/-0.009) for {'C': 1, 'gamma': 1.0}
0.982 (+/-0.007) for {'C': 1, 'gamma': 1.1}
0.984 (+/-0.007) for {'C': 1, 'gamma': 1.2000000000000002}
0.985 (+/-0.008) for {'C': 1, 'gamma': 1.3000000000000003}
0.985 (+/-0.007) for {'C': 1, 'gamma': 1.4000000000000001}
0.986 (+/-0.007) for {'C': 1, 'gamma': 1.5000000000000002}
0.987 (+/-0.008) for {'C': 1, 'gamma': 1.6}
0.987 (+/-0.007) for {'C': 1, 'gamma': 1.7000000000000002}
0.987 (+/-0.007) for {'C': 1, 'gamma': 1.8000000000000003}
0.987 (+/-0.006) for {'C': 1, 'gamma': 1.9000000000000001}
0.953 (+/-0.017) for {'C': 2, 'gamma': 0.1}
0.970 (+/-0.013) for {'C': 2, 'gamma': 0.2}
0.976 (+/-0.016) for {'C': 2, 'gamma': 0.30000000000000004}
0.978 (+/-0.012) for {'C': 2, 'gamma': 0.4}
0.980 (+/-0.009) for {'C': 2, 'gamma': 0.5}
0.982 (+/-0.007) for {'C': 2, 'gamma': 0.6}
0.983 (+/-0.008) for {'C': 2, 'gamma': 0.7000000000000001}
0.984 (+/-0.008) for {'C': 2, 'gamma': 0.8}
0.985 (+/-0.007) for {'C': 2, 'gamma': 0.9}
0.987 (+/-0.006) for {'C': 2, 'gamma': 1.0}
0.987 (+/-0.007) for {'C': 2, 'gamma': 1.1}
0.988 (+/-0.007) for {'C': 2, 'gamma': 1.2000000000000002}
0.988 (+/-0.006) for {'C': 2, 'gamma': 1.3000000000000003}
0.988 (+/-0.005) for {'C': 2, 'gamma': 1.4000000000000001}
0.988 (+/-0.006) for {'C': 2, 'gamma': 1.5000000000000002}
0.989 (+/-0.007) for {'C': 2, 'gamma': 1.6}
0.989 (+/-0.007) for {'C': 2, 'gamma': 1.7000000000000002}
0.989 (+/-0.006) for {'C': 2, 'gamma': 1.8000000000000003}
0.989 (+/-0.006) for {'C': 2, 'gamma': 1.9000000000000001}
0.962 (+/-0.014) for {'C': 3, 'gamma': 0.1}
0.975 (+/-0.014) for {'C': 3, 'gamma': 0.2}
0.978 (+/-0.012) for {'C': 3, 'gamma': 0.30000000000000004}
0.981 (+/-0.009) for {'C': 3, 'gamma': 0.4}
0.983 (+/-0.009) for {'C': 3, 'gamma': 0.5}
0.983 (+/-0.010) for {'C': 3, 'gamma': 0.6}
0.985 (+/-0.008) for {'C': 3, 'gamma': 0.7000000000000001}
0.987 (+/-0.007) for {'C': 3, 'gamma': 0.8}
0.987 (+/-0.009) for {'C': 3, 'gamma': 0.9}
0.988 (+/-0.008) for {'C': 3, 'gamma': 1.0}
0.989 (+/-0.007) for {'C': 3, 'gamma': 1.1}
0.989 (+/-0.007) for {'C': 3, 'gamma': 1.2000000000000002}
0.989 (+/-0.007) for {'C': 3, 'gamma': 1.3000000000000003}
0.989 (+/-0.008) for {'C': 3, 'gamma': 1.4000000000000001}
0.989 (+/-0.007) for {'C': 3, 'gamma': 1.5000000000000002}
0.989 (+/-0.007) for {'C': 3, 'gamma': 1.6}
0.989 (+/-0.006) for {'C': 3, 'gamma': 1.7000000000000002}
0.989 (+/-0.006) for {'C': 3, 'gamma': 1.8000000000000003}
0.989 (+/-0.007) for {'C': 3, 'gamma': 1.9000000000000001}
```

```
0.970 (+/-0.015) for {'C': 4, 'gamma': 0.1}
0.977 (+/-0.013) for {'C': 4, 'gamma': 0.2}
0.980 (+/-0.010) for {'C': 4, 'gamma': 0.30000000000000004}
0.982 (+/-0.009) for {'C': 4, 'gamma': 0.4}
0.984 (+/-0.010) for {'C': 4, 'gamma': 0.5}
0.985 (+/-0.010) for {'C': 4, 'gamma': 0.6}
0.986 (+/-0.008) for {'C': 4, 'gamma': 0.7000000000000001}
0.988 (+/-0.008) for {'C': 4, 'gamma': 0.8}
0.988 (+/-0.008) for {'C': 4, 'gamma': 0.9}
0.989 (+/-0.007) for {'C': 4, 'gamma': 1.0}
0.989 (+/-0.007) for {'C': 4, 'gamma': 1.1}
0.989 (+/-0.007) for {'C': 4, 'gamma': 1.2000000000000002}
0.990 (+/-0.007) for {'C': 4, 'gamma': 1.3000000000000003}
0.990 (+/-0.007) for {'C': 4, 'gamma': 1.4000000000000001}
0.989 (+/-0.006) for {'C': 4, 'gamma': 1.5000000000000002}
0.989 (+/-0.006) for {'C': 4, 'gamma': 1.6}
0.989 (+/-0.006) for {'C': 4, 'gamma': 1.7000000000000002}
0.989 (+/-0.006) for {'C': 4, 'gamma': 1.8000000000000003}
0.989 (+/-0.007) for {'C': 4, 'gamma': 1.9000000000000001}
0.972 (+/-0.013) for {'C': 5, 'gamma': 0.1}
0.978 (+/-0.011) for {'C': 5, 'gamma': 0.2}
0.982 (+/-0.008) for {'C': 5, 'gamma': 0.30000000000000004}
0.984 (+/-0.010) for {'C': 5, 'gamma': 0.4}
0.985 (+/-0.011) for {'C': 5, 'gamma': 0.5}
0.986 (+/-0.009) for {'C': 5, 'gamma': 0.6}
0.988 (+/-0.009) for {'C': 5, 'gamma': 0.7000000000000001}
0.988 (+/-0.007) for {'C': 5, 'gamma': 0.8}
0.988 (+/-0.007) for {'C': 5, 'gamma': 0.9}
0.989 (+/-0.006) for {'C': 5, 'gamma': 1.0}
0.989 (+/-0.006) for {'C': 5, 'gamma': 1.1}
0.989 (+/-0.005) for {'C': 5, 'gamma': 1.2000000000000002}
0.989 (+/-0.007) for {'C': 5, 'gamma': 1.3000000000000003}
0.989 (+/-0.007) for {'C': 5, 'gamma': 1.4000000000000001}
0.989 (+/-0.007) for {'C': 5, 'gamma': 1.5000000000000002}
0.989 (+/-0.007) for {'C': 5, 'gamma': 1.6}
0.989 (+/-0.007) for {'C': 5, 'gamma': 1.7000000000000002}
0.990 (+/-0.007) for {'C': 5, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 5, 'gamma': 1.9000000000000001}
0.973 (+/-0.015) for {'C': 6, 'gamma': 0.1}
0.980 (+/-0.011) for {'C': 6, 'gamma': 0.2}
0.983 (+/-0.010) for {'C': 6, 'gamma': 0.30000000000000004}
0.984 (+/-0.011) for {'C': 6, 'gamma': 0.4}
0.986 (+/-0.010) for {'C': 6, 'gamma': 0.5}
0.988 (+/-0.010) for {'C': 6, 'gamma': 0.6}
0.988 (+/-0.008) for {'C': 6, 'gamma': 0.7000000000000001}
0.988 (+/-0.007) for {'C': 6, 'gamma': 0.8}
0.989 (+/-0.007) for {'C': 6, 'gamma': 0.9}
0.989 (+/-0.006) for {'C': 6, 'gamma': 1.0}
0.989 (+/-0.007) for {'C': 6, 'gamma': 1.1}
0.989 (+/-0.007) for {'C': 6, 'gamma': 1.2000000000000002}
0.989 (+/-0.007) for {'C': 6, 'gamma': 1.3000000000000003}
0.989 (+/-0.007) for {'C': 6, 'gamma': 1.4000000000000001}
0.989 (+/-0.007) for {'C': 6, 'gamma': 1.5000000000000002}
0.989 (+/-0.006) for {'C': 6, 'gamma': 1.6}
0.990 (+/-0.006) for {'C': 6, 'gamma': 1.7000000000000002}
0.990 (+/-0.006) for {'C': 6, 'gamma': 1.8000000000000003}
0.991 (+/-0.006) for {'C': 6, 'gamma': 1.9000000000000001}
```

```
0.975 (+/-0.013) for {'C': 7, 'gamma': 0.1}
0.981 (+/-0.009) for {'C': 7, 'gamma': 0.2}
0.983 (+/-0.009) for {'C': 7, 'gamma': 0.30000000000000004}
0.985 (+/-0.011) for {'C': 7, 'gamma': 0.4}
0.987 (+/-0.010) for {'C': 7, 'gamma': 0.5}
0.988 (+/-0.008) for {'C': 7, 'gamma': 0.6}
0.988 (+/-0.007) for {'C': 7, 'gamma': 0.7000000000000001}
0.988 (+/-0.007) for {'C': 7, 'gamma': 0.8}
0.989 (+/-0.007) for {'C': 7, 'gamma': 0.9}
0.989 (+/-0.007) for {'C': 7, 'gamma': 1.0}
0.989 (+/-0.006) for {'C': 7, 'gamma': 1.1}
0.989 (+/-0.007) for {'C': 7, 'gamma': 1.2000000000000002}
0.989 (+/-0.007) for {'C': 7, 'gamma': 1.3000000000000003}
0.989 (+/-0.007) for {'C': 7, 'gamma': 1.4000000000000001}
0.990 (+/-0.007) for {'C': 7, 'gamma': 1.5000000000000002}
0.990 (+/-0.006) for {'C': 7, 'gamma': 1.6}
0.990 (+/-0.006) for {'C': 7, 'gamma': 1.7000000000000002}
0.991 (+/-0.005) for {'C': 7, 'gamma': 1.8000000000000003}
0.991 (+/-0.005) for {'C': 7, 'gamma': 1.9000000000000001}
0.976 (+/-0.011) for {'C': 8, 'gamma': 0.1}
0.982 (+/-0.009) for {'C': 8, 'gamma': 0.2}
0.984 (+/-0.010) for {'C': 8, 'gamma': 0.30000000000000004}
0.985 (+/-0.010) for {'C': 8, 'gamma': 0.4}
0.987 (+/-0.009) for {'C': 8, 'gamma': 0.5}
0.988 (+/-0.007) for {'C': 8, 'gamma': 0.6}
0.988 (+/-0.007) for {'C': 8, 'gamma': 0.7000000000000001}
0.989 (+/-0.007) for {'C': 8, 'gamma': 0.8}
0.988 (+/-0.007) for {'C': 8, 'gamma': 0.9}
0.989 (+/-0.007) for {'C': 8, 'gamma': 1.0}
0.989 (+/-0.007) for {'C': 8, 'gamma': 1.1}
0.989 (+/-0.007) for {'C': 8, 'gamma': 1.2000000000000002}
0.989 (+/-0.008) for {'C': 8, 'gamma': 1.3000000000000003}
0.990 (+/-0.006) for {'C': 8, 'gamma': 1.4000000000000001}
0.990 (+/-0.006) for {'C': 8, 'gamma': 1.5000000000000002}
0.990 (+/-0.007) for {'C': 8, 'gamma': 1.6}
0.990 (+/-0.007) for {'C': 8, 'gamma': 1.7000000000000002}
0.991 (+/-0.005) for {'C': 8, 'gamma': 1.8000000000000003}
0.991 (+/-0.005) for {'C': 8, 'gamma': 1.9000000000000001}
0.976 (+/-0.011) for {'C': 9, 'gamma': 0.1}
0.983 (+/-0.010) for {'C': 9, 'gamma': 0.2}
0.984 (+/-0.011) for {'C': 9, 'gamma': 0.30000000000000004}
0.987 (+/-0.010) for {'C': 9, 'gamma': 0.4}
0.987 (+/-0.009) for {'C': 9, 'gamma': 0.5}
0.988 (+/-0.007) for {'C': 9, 'gamma': 0.6}
0.988 (+/-0.007) for {'C': 9, 'gamma': 0.7000000000000001}
0.988 (+/-0.007) for {'C': 9, 'gamma': 0.8}
0.988 (+/-0.007) for {'C': 9, 'gamma': 0.9}
0.989 (+/-0.007) for {'C': 9, 'gamma': 1.0}
0.989 (+/-0.007) for {'C': 9, 'gamma': 1.1}
0.990 (+/-0.007) for {'C': 9, 'gamma': 1.2000000000000002}
0.989 (+/-0.007) for {'C': 9, 'gamma': 1.3000000000000003}
0.990 (+/-0.007) for {'C': 9, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 9, 'gamma': 1.5000000000000002}
0.990 (+/-0.007) for {'C': 9, 'gamma': 1.6}
0.990 (+/-0.006) for {'C': 9, 'gamma': 1.7000000000000002}
0.991 (+/-0.006) for {'C': 9, 'gamma': 1.8000000000000003}
0.991 (+/-0.006) for {'C': 9, 'gamma': 1.9000000000000001}
```

```
0.977 (+/-0.011) for {'C': 10, 'gamma': 0.1}
0.983 (+/-0.010) for {'C': 10, 'gamma': 0.2}
0.984 (+/-0.010) for {'C': 10, 'gamma': 0.30000000000000004}
0.987 (+/-0.009) for {'C': 10, 'gamma': 0.4}
0.988 (+/-0.007) for {'C': 10, 'gamma': 0.5}
0.989 (+/-0.006) for {'C': 10, 'gamma': 0.6}
0.989 (+/-0.006) for {'C': 10, 'gamma': 0.7000000000000001}
0.989 (+/-0.006) for {'C': 10, 'gamma': 0.8}
0.989 (+/-0.007) for {'C': 10, 'gamma': 0.9}
0.989 (+/-0.006) for {'C': 10, 'gamma': 1.0}
0.990 (+/-0.007) for {'C': 10, 'gamma': 1.1}
0.989 (+/-0.007) for {'C': 10, 'gamma': 1.2000000000000002}
0.990 (+/-0.007) for {'C': 10, 'gamma': 1.3000000000000003}
0.991 (+/-0.008) for {'C': 10, 'gamma': 1.4000000000000001}
0.991 (+/-0.008) for {'C': 10, 'gamma': 1.5000000000000002}
0.990 (+/-0.007) for {'C': 10, 'gamma': 1.6}
0.991 (+/-0.006) for {'C': 10, 'gamma': 1.7000000000000002}
0.991 (+/-0.006) for {'C': 10, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 10, 'gamma': 1.9000000000000001}
0.977 (+/-0.010) for {'C': 11, 'gamma': 0.1}
0.983 (+/-0.011) for {'C': 11, 'gamma': 0.2}
0.985 (+/-0.010) for {'C': 11, 'gamma': 0.30000000000000004}
0.987 (+/-0.008) for {'C': 11, 'gamma': 0.4}
0.988 (+/-0.006) for {'C': 11, 'gamma': 0.5}
0.989 (+/-0.006) for {'C': 11, 'gamma': 0.6}
0.988 (+/-0.007) for {'C': 11, 'gamma': 0.7000000000000001}
0.989 (+/-0.007) for {'C': 11, 'gamma': 0.8}
0.989 (+/-0.006) for {'C': 11, 'gamma': 0.9}
0.990 (+/-0.007) for {'C': 11, 'gamma': 1.0}
0.990 (+/-0.007) for {'C': 11, 'gamma': 1.1}
0.990 (+/-0.007) for {'C': 11, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 11, 'gamma': 1.3000000000000003}
0.991 (+/-0.008) for {'C': 11, 'gamma': 1.4000000000000001}
0.991 (+/-0.008) for {'C': 11, 'gamma': 1.5000000000000002}
0.990 (+/-0.007) for {'C': 11, 'gamma': 1.6}
0.990 (+/-0.007) for {'C': 11, 'gamma': 1.7000000000000002}
0.990 (+/-0.007) for {'C': 11, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 11, 'gamma': 1.9000000000000001}
0.979 (+/-0.009) for {'C': 12, 'gamma': 0.1}
0.983 (+/-0.011) for {'C': 12, 'gamma': 0.2}
0.986 (+/-0.010) for {'C': 12, 'gamma': 0.30000000000000004}
0.987 (+/-0.007) for {'C': 12, 'gamma': 0.4}
0.988 (+/-0.006) for {'C': 12, 'gamma': 0.5}
0.989 (+/-0.007) for {'C': 12, 'gamma': 0.6}
0.988 (+/-0.008) for {'C': 12, 'gamma': 0.7000000000000001}
0.989 (+/-0.007) for {'C': 12, 'gamma': 0.8}
0.989 (+/-0.006) for {'C': 12, 'gamma': 0.9}
0.990 (+/-0.007) for {'C': 12, 'gamma': 1.0}
0.990 (+/-0.007) for {'C': 12, 'gamma': 1.1}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.2000000000000002}
0.991 (+/-0.008) for {'C': 12, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 12, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 12, 'gamma': 1.5000000000000002}
0.990 (+/-0.008) for {'C': 12, 'gamma': 1.6}
0.990 (+/-0.008) for {'C': 12, 'gamma': 1.7000000000000002}
0.990 (+/-0.007) for {'C': 12, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 12, 'gamma': 1.9000000000000001}
```

```
0.980 (+/-0.009) for {'C': 13, 'gamma': 0.1}
0.984 (+/-0.011) for {'C': 13, 'gamma': 0.2}
0.986 (+/-0.010) for {'C': 13, 'gamma': 0.30000000000000004}
0.988 (+/-0.006) for {'C': 13, 'gamma': 0.4}
0.989 (+/-0.006) for {'C': 13, 'gamma': 0.5}
0.989 (+/-0.007) for {'C': 13, 'gamma': 0.6}
0.989 (+/-0.008) for {'C': 13, 'gamma': 0.7000000000000001}
0.989 (+/-0.007) for {'C': 13, 'gamma': 0.8}
0.990 (+/-0.007) for {'C': 13, 'gamma': 0.9}
0.990 (+/-0.007) for {'C': 13, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 13, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 13, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 13, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 13, 'gamma': 1.4000000000000001}
0.990 (+/-0.008) for {'C': 13, 'gamma': 1.5000000000000002}
0.990 (+/-0.009) for {'C': 13, 'gamma': 1.6}
0.989 (+/-0.008) for {'C': 13, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 13, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 13, 'gamma': 1.9000000000000001}
0.979 (+/-0.009) for {'C': 14, 'gamma': 0.1}
0.984 (+/-0.012) for {'C': 14, 'gamma': 0.2}
0.986 (+/-0.010) for {'C': 14, 'gamma': 0.30000000000000004}
0.988 (+/-0.007) for {'C': 14, 'gamma': 0.4}
0.988 (+/-0.006) for {'C': 14, 'gamma': 0.5}
0.989 (+/-0.007) for {'C': 14, 'gamma': 0.6}
0.989 (+/-0.007) for {'C': 14, 'gamma': 0.7000000000000001}
0.989 (+/-0.007) for {'C': 14, 'gamma': 0.8}
0.990 (+/-0.007) for {'C': 14, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 14, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 14, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 14, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 14, 'gamma': 1.3000000000000003}
0.990 (+/-0.009) for {'C': 14, 'gamma': 1.4000000000000001}
0.989 (+/-0.009) for {'C': 14, 'gamma': 1.5000000000000002}
0.989 (+/-0.010) for {'C': 14, 'gamma': 1.6}
0.989 (+/-0.009) for {'C': 14, 'gamma': 1.7000000000000002}
0.990 (+/-0.007) for {'C': 14, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 14, 'gamma': 1.9000000000000001}
0.980 (+/-0.008) for {'C': 15, 'gamma': 0.1}
0.984 (+/-0.012) for {'C': 15, 'gamma': 0.2}
0.986 (+/-0.009) for {'C': 15, 'gamma': 0.30000000000000004}
0.988 (+/-0.007) for {'C': 15, 'gamma': 0.4}
0.988 (+/-0.006) for {'C': 15, 'gamma': 0.5}
0.989 (+/-0.008) for {'C': 15, 'gamma': 0.6}
0.989 (+/-0.007) for {'C': 15, 'gamma': 0.7000000000000001}
0.989 (+/-0.007) for {'C': 15, 'gamma': 0.8}
0.990 (+/-0.007) for {'C': 15, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.0}
0.991 (+/-0.008) for {'C': 15, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.2000000000000002}
0.990 (+/-0.009) for {'C': 15, 'gamma': 1.3000000000000003}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.4000000000000001}
0.989 (+/-0.010) for {'C': 15, 'gamma': 1.5000000000000002}
0.989 (+/-0.010) for {'C': 15, 'gamma': 1.6}
0.989 (+/-0.009) for {'C': 15, 'gamma': 1.7000000000000002}
0.990 (+/-0.007) for {'C': 15, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 15, 'gamma': 1.9000000000000001}
```

```
0.980 (+/-0.008) for {'C': 16, 'gamma': 0.1}
0.984 (+/-0.011) for {'C': 16, 'gamma': 0.2}
0.987 (+/-0.008) for {'C': 16, 'gamma': 0.30000000000000004}
0.988 (+/-0.007) for {'C': 16, 'gamma': 0.4}
0.988 (+/-0.007) for {'C': 16, 'gamma': 0.5}
0.989 (+/-0.008) for {'C': 16, 'gamma': 0.6}
0.989 (+/-0.007) for {'C': 16, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 16, 'gamma': 0.8}
0.990 (+/-0.008) for {'C': 16, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.1}
0.990 (+/-0.009) for {'C': 16, 'gamma': 1.2000000000000002}
0.990 (+/-0.008) for {'C': 16, 'gamma': 1.3000000000000003}
0.989 (+/-0.009) for {'C': 16, 'gamma': 1.4000000000000001}
0.989 (+/-0.010) for {'C': 16, 'gamma': 1.5000000000000002}
0.989 (+/-0.010) for {'C': 16, 'gamma': 1.6}
0.989 (+/-0.009) for {'C': 16, 'gamma': 1.7000000000000002}
0.989 (+/-0.008) for {'C': 16, 'gamma': 1.8000000000000003}
0.990 (+/-0.007) for {'C': 16, 'gamma': 1.9000000000000001}
0.981 (+/-0.009) for {'C': 17, 'gamma': 0.1}
0.985 (+/-0.011) for {'C': 17, 'gamma': 0.2}
0.987 (+/-0.007) for {'C': 17, 'gamma': 0.30000000000000004}
0.987 (+/-0.007) for {'C': 17, 'gamma': 0.4}
0.989 (+/-0.007) for {'C': 17, 'gamma': 0.5}
0.989 (+/-0.008) for {'C': 17, 'gamma': 0.6}
0.989 (+/-0.006) for {'C': 17, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 17, 'gamma': 0.8}
0.990 (+/-0.008) for {'C': 17, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.0}
0.990 (+/-0.009) for {'C': 17, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.2000000000000002}
0.990 (+/-0.009) for {'C': 17, 'gamma': 1.3000000000000003}
0.989 (+/-0.009) for {'C': 17, 'gamma': 1.4000000000000001}
0.989 (+/-0.010) for {'C': 17, 'gamma': 1.5000000000000002}
0.989 (+/-0.010) for {'C': 17, 'gamma': 1.6}
0.989 (+/-0.009) for {'C': 17, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 17, 'gamma': 1.9000000000000001}
0.981 (+/-0.009) for {'C': 18, 'gamma': 0.1}
0.985 (+/-0.010) for {'C': 18, 'gamma': 0.2}
0.986 (+/-0.009) for {'C': 18, 'gamma': 0.30000000000000004}
0.988 (+/-0.007) for {'C': 18, 'gamma': 0.4}
0.989 (+/-0.007) for {'C': 18, 'gamma': 0.5}
0.989 (+/-0.007) for {'C': 18, 'gamma': 0.6}
0.989 (+/-0.006) for {'C': 18, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 18, 'gamma': 0.8}
0.990 (+/-0.008) for {'C': 18, 'gamma': 0.9}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.1}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.2000000000000002}
0.989 (+/-0.009) for {'C': 18, 'gamma': 1.3000000000000003}
0.989 (+/-0.009) for {'C': 18, 'gamma': 1.4000000000000001}
0.989 (+/-0.010) for {'C': 18, 'gamma': 1.5000000000000002}
0.989 (+/-0.010) for {'C': 18, 'gamma': 1.6}
0.990 (+/-0.009) for {'C': 18, 'gamma': 1.7000000000000002}
0.990 (+/-0.009) for {'C': 18, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 18, 'gamma': 1.9000000000000001}
```

```
0.981 (+/-0.009) for {'C': 19, 'gamma': 0.1}
0.986 (+/-0.009) for {'C': 19, 'gamma': 0.2}
0.987 (+/-0.008) for {'C': 19, 'gamma': 0.30000000000000004}
0.988 (+/-0.007) for {'C': 19, 'gamma': 0.4}
0.989 (+/-0.008) for {'C': 19, 'gamma': 0.5}
0.989 (+/-0.007) for {'C': 19, 'gamma': 0.6}
0.989 (+/-0.007) for {'C': 19, 'gamma': 0.7000000000000001}
0.989 (+/-0.008) for {'C': 19, 'gamma': 0.8}
0.990 (+/-0.008) for {'C': 19, 'gamma': 0.9}
0.990 (+/-0.009) for {'C': 19, 'gamma': 1.0}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.1}
0.989 (+/-0.009) for {'C': 19, 'gamma': 1.2000000000000002}
0.990 (+/-0.009) for {'C': 19, 'gamma': 1.3000000000000003}
0.989 (+/-0.009) for {'C': 19, 'gamma': 1.4000000000000001}
0.989 (+/-0.010) for {'C': 19, 'gamma': 1.5000000000000002}
0.989 (+/-0.009) for {'C': 19, 'gamma': 1.6}
0.990 (+/-0.009) for {'C': 19, 'gamma': 1.7000000000000002}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.8000000000000003}
0.990 (+/-0.008) for {'C': 19, 'gamma': 1.9000000000000001}
C 7
cache_size 200
class_weight None
coef0 0.0
decision_function_shape ovr
degree 3
gamma 1.9000000000000001
kernel rbf
max_iter -1
probability True
random_state None
shrinking True
tol 0.001
verbose False
```

4)Using L1 penalty

i)feature:Family--with normalization

In [10]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import LinearSVC
import pandas as pd

Folder_Path = r'/Users/yqh/Desktop/'

a = pd.read_csv(Folder_Path + 'norm_training_data.csv', usecols=range(0,
 22), header=0)
b = pd.read_csv(Folder_Path + 'norm_training_data.csv', usecols=[22], he
ader=0)

model = LinearSVC(penalty='l1', dual=False)
param_grid = {'C': range(1, 20)}
grid_search = GridSearchCV(model, param_grid, scoring='accuracy', cv=10,
 n_jobs=1, verbose=1)
grid_search.fit(a, b)
best_parameters = grid_search.best_estimator_.get_params()

means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, grid_search.cv_results_['param
s']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))

for para, val in best_parameters.items():
    print(para, val)
```

```
Fitting 10 folds for each of 19 candidates, totalling 190 fits

[Parallel(n_jobs=1)]: Done 190 out of 190 | elapsed:  6.2min finished

0.937 (+/-0.010) for {'C': 1}
0.938 (+/-0.009) for {'C': 2}
0.937 (+/-0.007) for {'C': 3}
0.937 (+/-0.008) for {'C': 4}
0.937 (+/-0.007) for {'C': 5}
0.937 (+/-0.007) for {'C': 6}
0.937 (+/-0.008) for {'C': 7}
0.937 (+/-0.007) for {'C': 8}
0.937 (+/-0.007) for {'C': 9}
0.937 (+/-0.008) for {'C': 10}
0.937 (+/-0.007) for {'C': 11}
0.937 (+/-0.007) for {'C': 12}
0.937 (+/-0.007) for {'C': 13}
0.937 (+/-0.007) for {'C': 14}
0.937 (+/-0.007) for {'C': 15}
0.937 (+/-0.007) for {'C': 16}
0.937 (+/-0.007) for {'C': 17}
0.937 (+/-0.008) for {'C': 18}
0.937 (+/-0.008) for {'C': 19}
C 2
class_weight None
dual False
fit_intercept True
intercept_scaling 1
loss squared_hinge
max_iter 1000
multi_class ovr
penalty l1
random_state None
tol 0.0001
verbose 0
```

ii)feature:Genus--with normalization

In [11]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import LinearSVC
import pandas as pd

Folder_Path = r'/Users/yqh/Desktop/'

a = pd.read_csv(Folder_Path + 'norm_training_data.csv', usecols=range(0,
 22), header=0)
b = pd.read_csv(Folder_Path + 'norm_training_data.csv', usecols=[23], he
ader=0)

model = LinearSVC(penalty='l1', dual=False)
param_grid = {'C': range(1, 20)}
grid_search = GridSearchCV(model, param_grid, scoring='accuracy', cv=10,
 n_jobs=1, verbose=1)
grid_search.fit(a, b)
best_parameters = grid_search.best_estimator_.get_params()

means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, grid_search.cv_results_['param
s']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))

for para, val in best_parameters.items():
    print(para, val)
```

```
Fitting 10 folds for each of 19 candidates, totalling 190 fits

[Parallel(n_jobs=1)]: Done 190 out of 190 | elapsed:  8.9min finished

0.947 (+/-0.012) for {'C': 1}
0.951 (+/-0.010) for {'C': 2}
0.954 (+/-0.009) for {'C': 3}
0.954 (+/-0.011) for {'C': 4}
0.954 (+/-0.010) for {'C': 5}
0.955 (+/-0.010) for {'C': 6}
0.955 (+/-0.011) for {'C': 7}
0.955 (+/-0.011) for {'C': 8}
0.955 (+/-0.012) for {'C': 9}
0.955 (+/-0.012) for {'C': 10}
0.955 (+/-0.011) for {'C': 11}
0.956 (+/-0.012) for {'C': 12}
0.955 (+/-0.012) for {'C': 13}
0.955 (+/-0.011) for {'C': 14}
0.956 (+/-0.012) for {'C': 15}
0.955 (+/-0.012) for {'C': 16}
0.955 (+/-0.011) for {'C': 17}
0.956 (+/-0.011) for {'C': 18}
0.955 (+/-0.012) for {'C': 19}
C 12
class_weight None
dual False
fit_intercept True
intercept_scaling 1
loss squared_hinge
max_iter 1000
multi_class ovr
penalty l1
random_state None
tol 0.0001
verbose 0
```

iii)feature:Species--with normalization

In [12]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import LinearSVC
import pandas as pd

Folder_Path = r'/Users/yqh/Desktop/'

a = pd.read_csv(Folder_Path + 'norm_training_data.csv', usecols=range(0,
 22), header=0)
b = pd.read_csv(Folder_Path + 'norm_training_data.csv', usecols=[24], he
ader=0)

model = LinearSVC(penalty='l1', dual=False)
param_grid = {'C': range(1, 20)}
grid_search = GridSearchCV(model, param_grid, scoring='accuracy', cv=10,
 n_jobs=1, verbose=1)
grid_search.fit(a, b)
best_parameters = grid_search.best_estimator_.get_params()

means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, grid_search.cv_results_['param
s']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))

for para, val in best_parameters.items():
    print(para, val)
```

```
Fitting 10 folds for each of 19 candidates, totalling 190 fits

[Parallel(n_jobs=1)]: Done 190 out of 190 | elapsed:  9.1min finished

0.959 (+/-0.010) for {'C': 1}
0.962 (+/-0.008) for {'C': 2}
0.963 (+/-0.010) for {'C': 3}
0.964 (+/-0.009) for {'C': 4}
0.964 (+/-0.010) for {'C': 5}
0.964 (+/-0.010) for {'C': 6}
0.964 (+/-0.009) for {'C': 7}
0.964 (+/-0.009) for {'C': 8}
0.965 (+/-0.010) for {'C': 9}
0.965 (+/-0.010) for {'C': 10}
0.964 (+/-0.010) for {'C': 11}
0.965 (+/-0.010) for {'C': 12}
0.964 (+/-0.010) for {'C': 13}
0.965 (+/-0.010) for {'C': 14}
0.964 (+/-0.010) for {'C': 15}
0.964 (+/-0.009) for {'C': 16}
0.964 (+/-0.009) for {'C': 17}
0.964 (+/-0.009) for {'C': 18}
0.965 (+/-0.010) for {'C': 19}
C 14
class_weight None
dual False
fit_intercept True
intercept_scaling 1
loss squared_hinge
max_iter 1000
multi_class ovr
penalty l1
random_state None
tol 0.0001
verbose 0
```

5)After SMOTE

i)feature:Family

In [6]:
```
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import GridSearchCV
from sklearn.svm import LinearSVC
import pandas as pd
import warnings

warnings.filterwarnings('ignore')

file_name = r'/Users/yqh/Desktop/norm_training_data.csv'

X = pd.read_csv(file_name, usecols=range(0, 22), header=0)
y = pd.read_csv(file_name, usecols=[22], header=0)
X_resampled, y_resampled = SMOTE().fit_sample(X, y)

model = LinearSVC(penalty='l1', dual=False)
param_grid = {'C': range(1, 20)}
grid_search = GridSearchCV(model, param_grid, scoring='accuracy', cv=10,
 n_jobs=1, verbose=1)
grid_search.fit(X_resampled, y_resampled)
best_parameters = grid_search.best_estimator_.get_params()

means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, grid_search.cv_results_['param
s']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))

for para, val in best_parameters.items():
    print(para, val)
```

```
Fitting 10 folds for each of 19 candidates, totalling 190 fits

[Parallel(n_jobs=1)]: Done 190 out of 190 | elapsed: 19.1min finished

0.948 (+/-0.009) for {'C': 1}
0.948 (+/-0.009) for {'C': 2}
0.948 (+/-0.009) for {'C': 3}
0.948 (+/-0.009) for {'C': 4}
0.948 (+/-0.010) for {'C': 5}
0.949 (+/-0.009) for {'C': 6}
0.949 (+/-0.010) for {'C': 7}
0.949 (+/-0.010) for {'C': 8}
0.948 (+/-0.009) for {'C': 9}
0.948 (+/-0.009) for {'C': 10}
0.949 (+/-0.009) for {'C': 11}
0.949 (+/-0.009) for {'C': 12}
0.949 (+/-0.009) for {'C': 13}
0.949 (+/-0.009) for {'C': 14}
0.948 (+/-0.009) for {'C': 15}
0.948 (+/-0.010) for {'C': 16}
0.948 (+/-0.010) for {'C': 17}
0.949 (+/-0.009) for {'C': 18}
0.949 (+/-0.009) for {'C': 19}
C 11
class_weight None
dual False
fit_intercept True
intercept_scaling 1
loss squared_hinge
max_iter 1000
multi_class ovr
penalty l1
random_state None
tol 0.0001
verbose 0
```

ii)feature:Genus

In [1]:
```
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import GridSearchCV
from sklearn.svm import LinearSVC
import pandas as pd
import warnings

warnings.filterwarnings('ignore')

file_name = r'/Users/yqh/Desktop/norm_training_data.csv'

X = pd.read_csv(file_name, usecols=range(0, 22), header=0)
y = pd.read_csv(file_name, usecols=[23], header=0)
X_resampled, y_resampled = SMOTE().fit_sample(X, y)

model = LinearSVC(penalty='l1', dual=False)
param_grid = {'C': range(5, 15)}
grid_search = GridSearchCV(model, param_grid, scoring='accuracy', cv=10,
 n_jobs=1, verbose=1)
grid_search.fit(X_resampled, y_resampled)
best_parameters = grid_search.best_estimator_.get_params()

means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, grid_search.cv_results_['param
s']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))

for para, val in best_parameters.items():
    print(para, val)
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits

[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed: 34.5min finished

0.955 (+/-0.008) for {'C': 5}
0.955 (+/-0.008) for {'C': 6}
0.955 (+/-0.008) for {'C': 7}
0.955 (+/-0.008) for {'C': 8}
0.955 (+/-0.008) for {'C': 9}
0.955 (+/-0.008) for {'C': 10}
0.955 (+/-0.008) for {'C': 11}
0.955 (+/-0.008) for {'C': 12}
0.955 (+/-0.008) for {'C': 13}
0.955 (+/-0.008) for {'C': 14}
C 6
class_weight None
dual False
fit_intercept True
intercept_scaling 1
loss squared_hinge
max_iter 1000
multi_class ovr
penalty l1
random_state None
tol 0.0001
verbose 0
```

iii)feature:Species

In [2]:
```
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import GridSearchCV
from sklearn.svm import LinearSVC
import pandas as pd
import warnings

warnings.filterwarnings('ignore')

file_name = r'/Users/yqh/Desktop/norm_training_data.csv'

X = pd.read_csv(file_name, usecols=range(0, 22), header=0)
y = pd.read_csv(file_name, usecols=[24], header=0)
X_resampled, y_resampled = SMOTE().fit_sample(X, y)

model = LinearSVC(penalty='l1', dual=False)
param_grid = {'C': range(5, 15)}
grid_search = GridSearchCV(model, param_grid, scoring='accuracy', cv=10,
 n_jobs=1, verbose=1)
grid_search.fit(X_resampled, y_resampled)
best_parameters = grid_search.best_estimator_.get_params()

means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, grid_search.cv_results_['param
s']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))

for para, val in best_parameters.items():
    print(para, val)
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits

[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed: 43.6min finished

0.958 (+/-0.007) for {'C': 5}
0.959 (+/-0.007) for {'C': 6}
0.958 (+/-0.008) for {'C': 7}
0.959 (+/-0.007) for {'C': 8}
0.959 (+/-0.007) for {'C': 9}
0.958 (+/-0.007) for {'C': 10}
0.959 (+/-0.007) for {'C': 11}
0.959 (+/-0.007) for {'C': 12}
0.958 (+/-0.007) for {'C': 13}
0.959 (+/-0.007) for {'C': 14}
C 8
class_weight None
dual False
fit_intercept True
intercept_scaling 1
loss squared_hinge
max_iter 1000
multi_class ovr
penalty l1
random_state None
tol 0.0001
verbose 0
```

6)Classifier Chain

add one label to training data in each time, we can see the exact match accuracy is improving

In [2]:
```
import numpy as np
import pandas as pd
from sklearn.multioutput import ClassifierChain
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import jaccard_similarity_score
from sklearn.linear_model import LogisticRegression
from sklearn import preprocessing
from sklearn.svm import LinearSVC


file_name1 = r'/Users/yqh/Desktop/norm_training_data.csv'
file_name2 = r'/Users/yqh/Desktop/norm_testing_data.csv'

X_train = pd.read_csv(file_name1, usecols=range(0, 22), header=0)
y_train1 = pd.read_csv(file_name1, usecols=[22], header=0)
y_train2 = pd.read_csv(file_name1, usecols=[23], header=0)
y_train3 = pd.read_csv(file_name1, usecols=[24], header=0)
y11 = preprocessing.LabelEncoder().fit_transform(y_train1)
y12 = preprocessing.LabelEncoder().fit_transform(y_train2)
y13 = preprocessing.LabelEncoder().fit_transform(y_train3)
y11 = pd.DataFrame(y11, columns=['Family'])
y12 = pd.DataFrame(y12, columns=['Genus'])
y13 = pd.DataFrame(y13, columns=['Species'])


X_test = pd.read_csv(file_name2, usecols=range(0, 22), header=0)
y_test1 = pd.read_csv(file_name2, usecols=[22], header=0)
y_test2 = pd.read_csv(file_name2, usecols=[23], header=0)
y_test3 = pd.read_csv(file_name2, usecols=[24], header=0)
y21 = preprocessing.LabelEncoder().fit_transform(y_test1)
y22 = preprocessing.LabelEncoder().fit_transform(y_test2)
y23 = preprocessing.LabelEncoder().fit_transform(y_test3)
y21 = pd.DataFrame(y21, columns=['Family'])
y22 = pd.DataFrame(y22, columns=['Genus'])
y23 = pd.DataFrame(y23, columns=['Species'])

y_train = pd.DataFrame(index=range(0, 5036))
y_train.insert(0, 'Family', y11)
# y_train.insert(1, 'Genus', y12)
# y_train.insert(2, 'Species', y13)
# X_train.insert(22, 'Family', y11)
# X_train.insert(23, 'Genus', y12)


y_test = pd.DataFrame(index=range(0, 2159))
y_test.insert(0, 'Family', y21)
# y_test.insert(1, 'Genus', y22)
# y_test.insert(2, 'Species', y23)
# X_test.insert(0, 'Family', y21)
# X_test.insert(23, 'Genus', y22)

# Fit an independent logistic regression model for each class using the
# OneVsRestClassifier wrapper.
ovr = OneVsRestClassifier(LinearSVC())
ovr.fit(X_train, y_train)
y_pred_ovr = ovr.predict(X_test)
```

```
ovr_jaccard_score = jaccard_similarity_score(y_test, y_pred_ovr)

# Fit an ensemble of logistic regression classifier chains and take the
# take the average prediction of all the chains.
chains = [ClassifierChain(LogisticRegression(), order='random', cv=10, r
andom_state=None)]#,for i in range(4)]
for chain in chains:
    chain.fit(X_train, y_train)

y_pred_chains = np.array([chain.predict(X_test) for chain in
                          chains])
chain_jaccard_scores = [jaccard_similarity_score(y_test, y_pred_chain >=
 .5)
                        for y_pred_chain in y_pred_chains]

y_pred_ensemble = y_pred_chains.mean(axis=0)
ensemble_jaccard_score = jaccard_similarity_score(y_test, y_pred_ensembl
e >= .5)

model_scores = [ovr_jaccard_score] + chain_jaccard_scores
model_scores.append(ensemble_jaccard_score)

model_names = ('Independent', 'Ensemble')

x_pos = np.arange(len(model_names))

print(model_scores)
print(ovr_jaccard_score)
print(ensemble_jaccard_score)
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-pa
ckages/sklearn/preprocessing/label.py:111: DataConversionWarning: A col
umn-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
[0.4891153311718388, 0.07364520611394164, 0.07364520611394164]
0.4891153311718388
0.07364520611394164
```

In [3]:
```
import numpy as np
import pandas as pd
from sklearn.multioutput import ClassifierChain
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import jaccard_similarity_score
from sklearn.linear_model import LogisticRegression
from sklearn import preprocessing
from sklearn.svm import LinearSVC


file_name1 = r'/Users/yqh/Desktop/norm_training_data.csv'
file_name2 = r'/Users/yqh/Desktop/norm_testing_data.csv'

X_train = pd.read_csv(file_name1, usecols=range(0, 22), header=0)
y_train1 = pd.read_csv(file_name1, usecols=[22], header=0)
y_train2 = pd.read_csv(file_name1, usecols=[23], header=0)
y_train3 = pd.read_csv(file_name1, usecols=[24], header=0)
y11 = preprocessing.LabelEncoder().fit_transform(y_train1)
y12 = preprocessing.LabelEncoder().fit_transform(y_train2)
y13 = preprocessing.LabelEncoder().fit_transform(y_train3)
y11 = pd.DataFrame(y11, columns=['Family'])
y12 = pd.DataFrame(y12, columns=['Genus'])
y13 = pd.DataFrame(y13, columns=['Species'])


X_test = pd.read_csv(file_name2, usecols=range(0, 22), header=0)
y_test1 = pd.read_csv(file_name2, usecols=[22], header=0)
y_test2 = pd.read_csv(file_name2, usecols=[23], header=0)
y_test3 = pd.read_csv(file_name2, usecols=[24], header=0)
y21 = preprocessing.LabelEncoder().fit_transform(y_test1)
y22 = preprocessing.LabelEncoder().fit_transform(y_test2)
y23 = preprocessing.LabelEncoder().fit_transform(y_test3)
y21 = pd.DataFrame(y21, columns=['Family'])
y22 = pd.DataFrame(y22, columns=['Genus'])
y23 = pd.DataFrame(y23, columns=['Species'])

y_train = pd.DataFrame(index=range(0, 5036))
y_train.insert(0, 'Genus', y12)
# y_train.insert(1, 'Genus', y12)
# y_train.insert(2, 'Species', y13)
X_train.insert(22, 'Family', y11)
# X_train.insert(23, 'Genus', y12)


y_test = pd.DataFrame(index=range(0, 2159))
y_test.insert(0, 'Genus', y22)
# y_test.insert(1, 'Genus', y22)
# y_test.insert(2, 'Species', y23)
X_test.insert(22, 'Family', y21)
# X_test.insert(23, 'Genus', y22)

# Fit an independent logistic regression model for each class using the
# OneVsRestClassifier wrapper.
ovr = OneVsRestClassifier(LinearSVC())
ovr.fit(X_train, y_train)
y_pred_ovr = ovr.predict(X_test)
```

```
ovr_jaccard_score = jaccard_similarity_score(y_test, y_pred_ovr)

# Fit an ensemble of logistic regression classifier chains and take the
# take the average prediction of all the chains.
chains = [ClassifierChain(LogisticRegression(), order='random', cv=10, r
andom_state=None)]#,for i in range(4)]
for chain in chains:
    chain.fit(X_train, y_train)

y_pred_chains = np.array([chain.predict(X_test) for chain in
                          chains])
chain_jaccard_scores = [jaccard_similarity_score(y_test, y_pred_chain >=
 .5)
                        for y_pred_chain in y_pred_chains]

y_pred_ensemble = y_pred_chains.mean(axis=0)
ensemble_jaccard_score = jaccard_similarity_score(y_test, y_pred_ensembl
e >= .5)

model_scores = [ovr_jaccard_score] + chain_jaccard_scores
model_scores.append(ensemble_jaccard_score)

model_names = ('Independent', 'Ensemble')

x_pos = np.arange(len(model_names))

print(model_scores)
print(ovr_jaccard_score)
print(ensemble_jaccard_score)
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-pa
ckages/sklearn/preprocessing/label.py:111: DataConversionWarning: A col
umn-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
[0.9064381658175081, 0.6419638721630384, 0.6419638721630384]
0.9064381658175081
0.6419638721630384
```

In [4]:
```
import numpy as np
import pandas as pd
from sklearn.multioutput import ClassifierChain
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import jaccard_similarity_score
from sklearn.linear_model import LogisticRegression
from sklearn import preprocessing
from sklearn.svm import LinearSVC


file_name1 = r'/Users/yqh/Desktop/norm_training_data.csv'
file_name2 = r'/Users/yqh/Desktop/norm_testing_data.csv'

X_train = pd.read_csv(file_name1, usecols=range(0, 22), header=0)
y_train1 = pd.read_csv(file_name1, usecols=[22], header=0)
y_train2 = pd.read_csv(file_name1, usecols=[23], header=0)
y_train3 = pd.read_csv(file_name1, usecols=[24], header=0)
y11 = preprocessing.LabelEncoder().fit_transform(y_train1)
y12 = preprocessing.LabelEncoder().fit_transform(y_train2)
y13 = preprocessing.LabelEncoder().fit_transform(y_train3)
y11 = pd.DataFrame(y11, columns=['Family'])
y12 = pd.DataFrame(y12, columns=['Genus'])
y13 = pd.DataFrame(y13, columns=['Species'])


X_test = pd.read_csv(file_name2, usecols=range(0, 22), header=0)
y_test1 = pd.read_csv(file_name2, usecols=[22], header=0)
y_test2 = pd.read_csv(file_name2, usecols=[23], header=0)
y_test3 = pd.read_csv(file_name2, usecols=[24], header=0)
y21 = preprocessing.LabelEncoder().fit_transform(y_test1)
y22 = preprocessing.LabelEncoder().fit_transform(y_test2)
y23 = preprocessing.LabelEncoder().fit_transform(y_test3)
y21 = pd.DataFrame(y21, columns=['Family'])
y22 = pd.DataFrame(y22, columns=['Genus'])
y23 = pd.DataFrame(y23, columns=['Species'])

y_train = pd.DataFrame(index=range(0, 5036))
y_train.insert(0, 'Species', y13)
# y_train.insert(1, 'Genus', y12)
# y_train.insert(2, 'Species', y13)
X_train.insert(22, 'Family', y11)
X_train.insert(23, 'Genus', y12)


y_test = pd.DataFrame(index=range(0, 2159))
y_test.insert(0, 'Species', y23)
# y_test.insert(1, 'Genus', y22)
# y_test.insert(2, 'Species', y23)
X_test.insert(22, 'Family', y21)
X_test.insert(23, 'Genus', y22)

# Fit an independent logistic regression model for each class using the
# OneVsRestClassifier wrapper.
ovr = OneVsRestClassifier(LinearSVC())
ovr.fit(X_train, y_train)
y_pred_ovr = ovr.predict(X_test)
```

```
ovr_jaccard_score = jaccard_similarity_score(y_test, y_pred_ovr)

# Fit an ensemble of logistic regression classifier chains and take the
# take the average prediction of all the chains.
chains = [ClassifierChain(LogisticRegression(), order='random', cv=10, r
andom_state=None)]#,for i in range(4)]
for chain in chains:
    chain.fit(X_train, y_train)

y_pred_chains = np.array([chain.predict(X_test) for chain in
                          chains])
chain_jaccard_scores = [jaccard_similarity_score(y_test, y_pred_chain >=
 .5)
                        for y_pred_chain in y_pred_chains]

y_pred_ensemble = y_pred_chains.mean(axis=0)
ensemble_jaccard_score = jaccard_similarity_score(y_test, y_pred_ensembl
e >= .5)

model_scores = [ovr_jaccard_score] + chain_jaccard_scores
model_scores.append(ensemble_jaccard_score)

model_names = ('Independent', 'Ensemble')

x_pos = np.arange(len(model_names))

print(model_scores)
print(ovr_jaccard_score)
print(ensemble_jaccard_score)
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-pa
ckages/sklearn/preprocessing/label.py:111: DataConversionWarning: A col
umn-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
[0.9606299212598425, 0.5687818434460399, 0.5687818434460399]
0.9606299212598425
0.5687818434460399
```

we can see that after adding one more label to training data at each time, the score of the model increase from 0.489 to 0.961

7)Evaluation of multi-label classification

i)n multiclass and multilabel classification task, the notions of precision, recall, and F-measures can be applied to each label independently. There are a few ways to combine results across labels, specified by the "average" argument to the average_precision_score.

Evaluation metrics for multi-label classification performance are inherently different from those used in multi-class (or binary) classification, due to the inherent differences of the classification problem. If T denotes the true set of labels for a given sample, and P the predicted set of labels, then the following metrics can be defined on that sample: Hamming loss: the fraction of the wrong labels to the total number of labels, i.e. $\frac{1}{|N| \cdot |L|} \sum_{i=1}^{|N|} \sum_{j=1}^{|L|} \text{xor}(y_{i,j}, z_{i,j}) \frac{1}{|N| \cdot |L|} \sum_{i=1}^{|N|} \sum_{j=1}^{|L|} \text{xor}(y_{i,j}, z_{i,j})$, where $y_{i,j} y_{i,j}$ is the target and $z_{i,j} z_{i,j}$ is the prediction. This is a loss function, so the optimal value is zero.

The closely related Jaccard index, also called Intersection over Union in the multi-label setting, is defined as the number of correctly predicted labels divided by the union of predicted and true labels, $\frac{|T \cap P|}{|T \cup P|}$

Precision, recall and F_1 score: precision is $\frac{|T \cap P|}{|P|}$, recall is $\frac{|T \cap P|}{|T|}$, and F_1 is their harmonic mean.

ROC_AUC: is almost the same as multi-class classification.