SVM实例

运行结果

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
{'C': 5} 0.698
              precision    recall  f1-score   support

           A       0.85      0.87      0.86       207
           B       0.63      0.72      0.67       190
           C       0.66      0.81      0.73       187
           D       0.71      0.78      0.74       186
           E       0.67      0.48      0.56       192
           F       0.81      0.71      0.75       194
           G       0.53      0.26      0.35       184
           H       0.43      0.40      0.42       181
           I       0.87      0.86      0.86       204
           J       0.89      0.78      0.83       192
           K       0.66      0.57      0.61       192
           L       0.76      0.81      0.79       179
           M       0.80      0.91      0.85       196
           N       0.74      0.74      0.74       176
           O       0.58      0.40      0.47       173
           P       0.81      0.84      0.82       201
           Q       0.61      0.78      0.68       195
           R       0.62      0.79      0.69       189
           S       0.47      0.46      0.47       178
           T       0.82      0.89      0.85       211
           U       0.85      0.82      0.83       202
           V       0.80      0.77      0.79       190
           W       0.87      0.86      0.86       198
           X       0.66      0.75      0.70       207
           Y       0.74      0.73      0.74       211
           Z       0.73      0.77      0.75       185

    accuracy                           0.72      5000
   macro avg       0.71      0.71      0.71      5000
weighted avg       0.72      0.72      0.71      5000
```

代码分析

这段代码用于支持向量机（SVM）分类任务的建模和评估。

```
predictors = letters.columns[1:]
X_train, X_test, y_train, y_test =
model_selection.train_test_split(letters[predictors], letters.letter,

 test_size=0.25, random_state=1234)
```

将数据集拆分为输入特征X和目标变量y，其中X包含除'letter'列之外的所有列，y是'letter'列。然后，使用train_test_split函数将数据集划分为训练集和测试集，其中测试集占比为0.25。

```
C = [0.05, 0.1, 0.5, 1, 2, 5]
parameters = {'C': C}
grid_linear_svc = model_selection.GridSearchCV(estimator=svm.LinearSVC(
    dual=False), param_grid=parameters, scoring='accuracy', cv=5, verbose=1)
```

使用GridSearchCV函数进行网格搜索，以选择线性可分SVM模型中最佳的C值（正则化参数）。参数C的候选值为[0.05, 0.1, 0.5, 1, 2, 5]。使用accuracy作为评分指标进行交叉验证（cv=5），同时设置verbose参数为1以显示搜索过程的详细信息。

```
grid_linear_svc.fit(X_train, y_train)
```

对网格搜索对象grid_linear_svc进行训练，通过拟合训练数据集X_train和y_train来搜索最佳的C值。

```
grid_linear_svc.best_params_, grid_linear_svc.best_score_
```

返回交叉验证后得到的最佳参数值和相应的评分。

```
pred_linear_svc = grid_linear_svc.predict(X_test)
```

使用训练好的模型对测试集X_test进行预测，将预测结果存储在pred_linear_svc中。

```
report = metrics.classification_report(y_test, pred_linear_svc)
print(report)
```

这段代码使用classification_report函数计算模型的预测准确率，并将结果打印出来。