به نام خدا

دانشگاه صنعتی شاهرود

Shahrood University of Technology

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

تمرین ششم یادگیری ماشین : ماشین بردار پشتیبان(SVM)

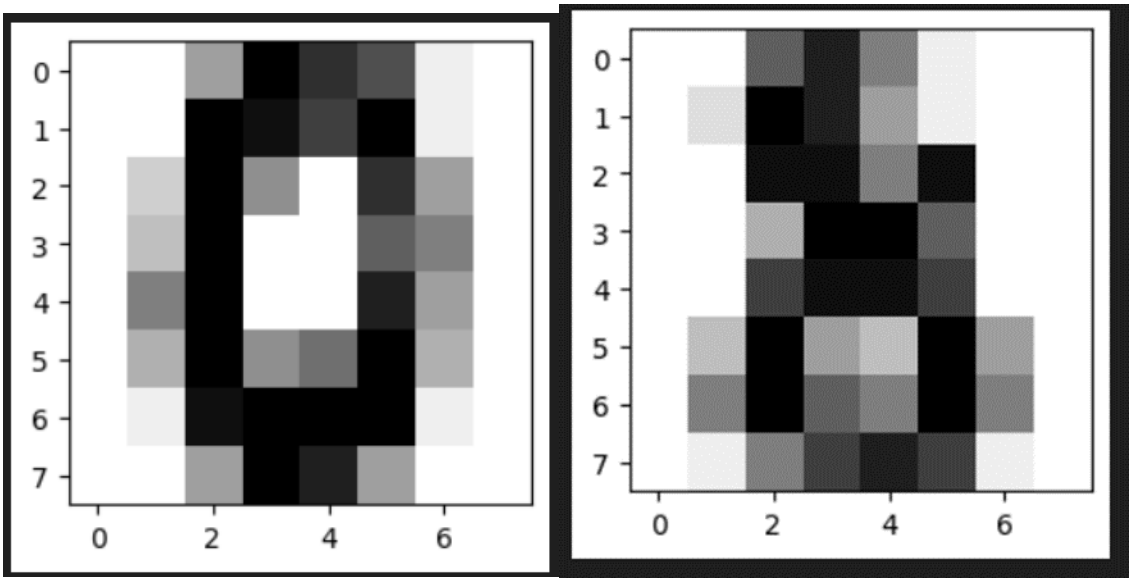
استاد: دکتر مرضیه رحیمی

نویسنده: مریم درویشیان

تاریخ: 1402/03/17

1) این مجموعه دادگان را معرفی کنید.

این مجموعه داده از 1797 تصویر هشت در هشت تشکیل شده است. هر تصویر، مانند تصویر زیر، یک رقم دست نویس است. برای استفاده از یک شکل هشت در هشت مانند این، ابتدا باید آن را به یک بردار ویژگی با طول 64 تبدیل کنیم.دو نمونه از مجموعه دیتا در شکل زیر نشان داده شده است.



2) دقتهای دوبخش آموزش و تست مدلهایSVCو رگرسیون لاجستیک را گزارش کنید.

```python
# svm model with kernel=linear and estimating train and test accuracies

import sklearn.svm as sv
from sklearn import model_selection as ms
from sklearn import preprocessing
from sklearn.svm import SVC

X = digits.data
Y = digits.target

Xtr, Xte, Ytr, Yte = ms.train_test_split(X, Y, train_size = 0.8, random_state = 21)


Classifier = sv.SVC(kernel = 'linear')
Classifier.fit(Xtr, Ytr)

trAcc=Classifier.score(Xtr,Ytr)
teAcc=Classifier.score(Xte,Yte)

print ('Train accuracy:',trAcc)
print('Test accuracy',teAcc)
```

✓ 0.0s

```
Train accuracy: 1.0
Test accuracy 0.9833333333333333
```

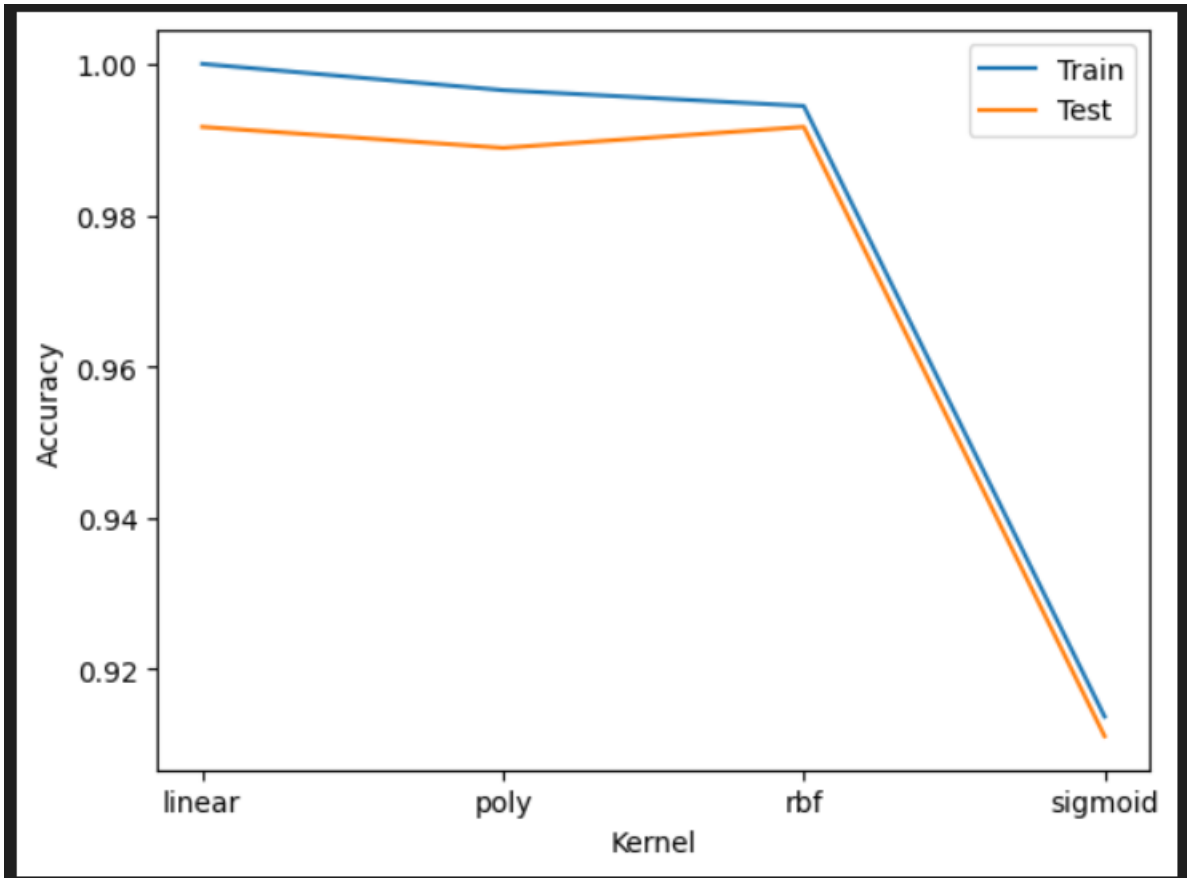
```python
#Logisic regression model

import sklearn.linear_model as lm


Xtr, Xte, Ytr, Yte = ms.train_test_split(X, Y, train_size = 0.8, random_state = 32)

LR = lm.LogisticRegression(max_iter = 5000)
LR.fit(Xtr, Ytr)
trAcc = LR.score(Xtr, Ytr)
teAcc = LR.score(Xte, Yte)

print('Train Acc.: ', trAcc)
print('Test Acc.: ', teAcc)

```

✓ 3.3s

```
Train Acc.:  1.0
Test Acc.:  0.975
```

```python
#svm model with kernels=linear, poly,rbf,sigmoid and plot

X = digits.data
Y = digits.target

Xtr, Xte, Ytr, Yte = ms.train_test_split(X, Y, train_size = 0.8, random_state = 1)

trAcc = []
teAcc = []
Kernel = []
for i in ['linear', 'poly', 'rbf', 'sigmoid']:
    Classifier = sv.SVC(kernel = i, degree = 2)
    Classifier.fit(Xtr, Ytr)
    trAcc.append(Classifier.score(Xtr, Ytr))
    teAcc.append(Classifier.score(Xte, Yte))
    Kernel.append(i)

plt.plot(trAcc, label = 'Train')
plt.plot(teAcc, label = 'Test')
plt.xticks([0, 1, 2, 3], Kernel)
plt.xlabel('Kernel')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

✓ 0.7s



```
[77]
```

```python
Classifier = sv.LinearSVC(dual=False,multi_class='crammer_singer', C=1e10)
Classifier.fit(Xtr, Ytr)
y_pred = Classifier.predict(Xte)

trAcc=Classifier.score(Xtr,Ytr)
teAcc=Classifier.score(Xte,Yte)

print ('Train accuracy:',trAcc)
print('Test accuracy',teAcc)

print(Classifier.intercept_)
print(Classifier.coef_)
pred=Classifier.predict(Xte)
predictions = pred.reshape(-1,1)

print('MSE : ', mean_squared_error(Yte,predictions))
print('RMSE : ', np.sqrt(mean_squared_error(Yte,predictions)))
```

✓ 9.9s

```
Train accuracy: 1.0
Test accuracy 0.9611111111111111
[ 0.00131951 -0.00995655  0.0012517   0.00055281  0.00843682 -0.00520923
 -0.00135077  0.00142365  0.01648682 -0.01296885]
[[ 0.00000000e+00  7.15316001e-04 -3.54021312e-04 -1.26552772e-02
   1.54221566e-02 -7.40842376e-03 -1.51937533e-02 -1.77106800e-03
   0.00000000e+00 -1.18773854e-02  1.95953381e-04  4.95316964e-02
  -6.24336980e-03  1.31646536e-02 -3.12959230e-03 -1.81932867e-03
   0.00000000e+00 -9.92865919e-03  2.22002619e-02 -2.11652196e-03
  -8.58927947e-02  4.69819017e-02  5.10700912e-03 -5.04458577e-04
   0.00000000e+00  2.31164413e-02  3.44014258e-02 -3.10237569e-02
  -9.02996677e-02  2.09942557e-03  2.45138467e-02  0.00000000e+00
   0.00000000e+00  5.20444368e-02  8.37674869e-03 -3.31250948e-02
  -8.52435743e-02  1.37906139e-02  2.20487168e-02  0.00000000e+00
   0.00000000e+00  1.74361397e-03  5.47637309e-02 -3.49481437e-02
  -4.43112098e-02  2.46597306e-02  2.18293824e-02  0.00000000e+00
   0.00000000e+00 -1.29026113e-02  2.09434232e-02  1.32897452e-02
   6.57019337e-02 -3.46376248e-03  3.44686738e-03 -2.69042065e-03
   0.00000000e+00 -1.71418540e-03 -9.84991729e-04  3.78712528e-02
  -4.49661531e-04  5.75030922e-04 -2.15956724e-03 -2.01781549e-03]
 [ 0.00000000e+00 -1.26809056e-03  1.18319184e-02  4.76268936e-02
  -1.95194561e-01  9.94103405e-02  3.59479367e-02  0.00000000e+00
   0.00000000e+00 -5.69716106e-02 -1.11346184e-01 -9.72307870e-02
   7.68706796e-02 -4.75874443e-03 -6.88368403e-02  1.30104261e-18
   1.67772896e-02  1.06654078e-01 -3.98463774e-02  2.12024916e-01
...
   0.00000000e+00 -4.84539856e-02 -1.25637838e-02  1.57835942e-02
  -2.56463612e-02  3.98257928e-02 -6.50507279e-02  1.59024113e-02]]
MSE :  0.4
RMSE :  0.6324555320336759
```

3) تحلیل خود را از دقتهای بدست آمده بنویسید.

مدل Svm به دقت مناسب 1 در زمان اموزش با کرنل خطی رسیده و برای داده تست 0.98 . مدل LogisticRegression به دقت مناسب 1 در زمان اموزش و برای داده تست 0.975 رسید. با مقایسه نمودار بطورکلی دقت SVM کرنلهای rbf و linear و sigmoid و polynomial متوجه میشویم که داده اموزش و تست در سیگموید بالاترین میزان همگرایی و کمترین دقت با کرنل sigmoid را دارد. بهترین دقت با کرنل خطی بدست امده است.

4) پارامترهای مدل و مقادیر ممکن برای هرپارامتردر مدلSVC را معرفی کنید.

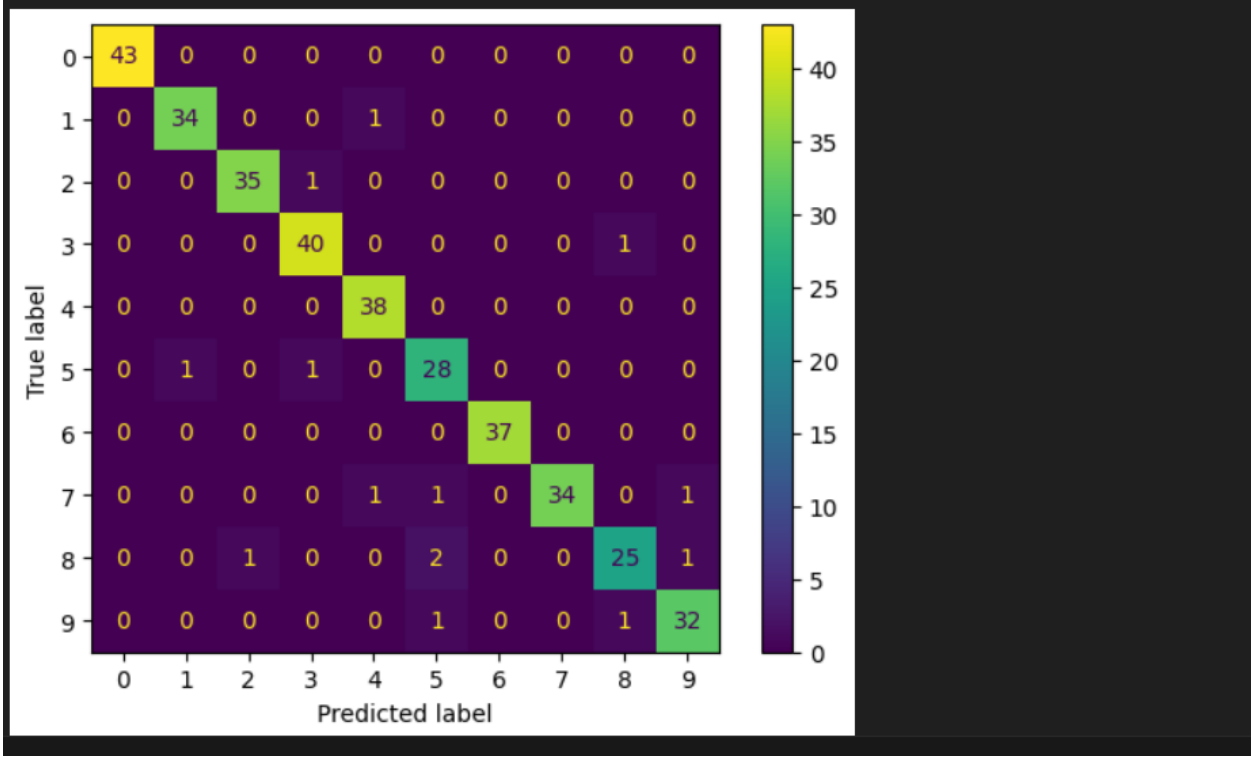
تابع کرنل مجموعه داده آموزشی را به ابعاد بالاتر تبدیل می کند تا به صورت خطی قابل تفکیک باشد.برای داده های جدا ناپذیر خطی برای اینکه مدل ما پیچیدگی کمتری داشته باشد با افزایش ابعاد با تابع کرنل مرز جداکننده خطی از تابع چند جمله ای به یک تابع خطی نگاشت میشود و از این طریق پیچیدگی تابع جداکننده کم میشود تا با مدل ساده تر داده ها توسط یک صفحه قابل تفکیک باشد. تابع کرنل پیش‌فرض برای پیاده‌سازی مدل طبقه‌بندی‌کننده بردار پشتیبان، تابع پایه شعاعی است که معمولاً به آن rbf می‌گویند. انواع تابع کرنل خطی و چندجمله ای و rbf و sigmoid است.

SVM همچنین دارای برخی فراپارامترها استمانند مقادیر C وگاما برای استفاده و یافتن فراپارامتر بهینه کار سختی به صورت دستی برای حل آن است. اما می‌توان آن را فقط با امتحان کردن همه ترکیب‌ها پیدا کرد و دید که کدام پارامترها بهترین عملکرد را دارند. ما مجبور نیستیم همه ترکیبات را صورت دستی انجام دهیم، چون Scikit-learn این قابلیت را با GridSearchCV فراهم کرده است.

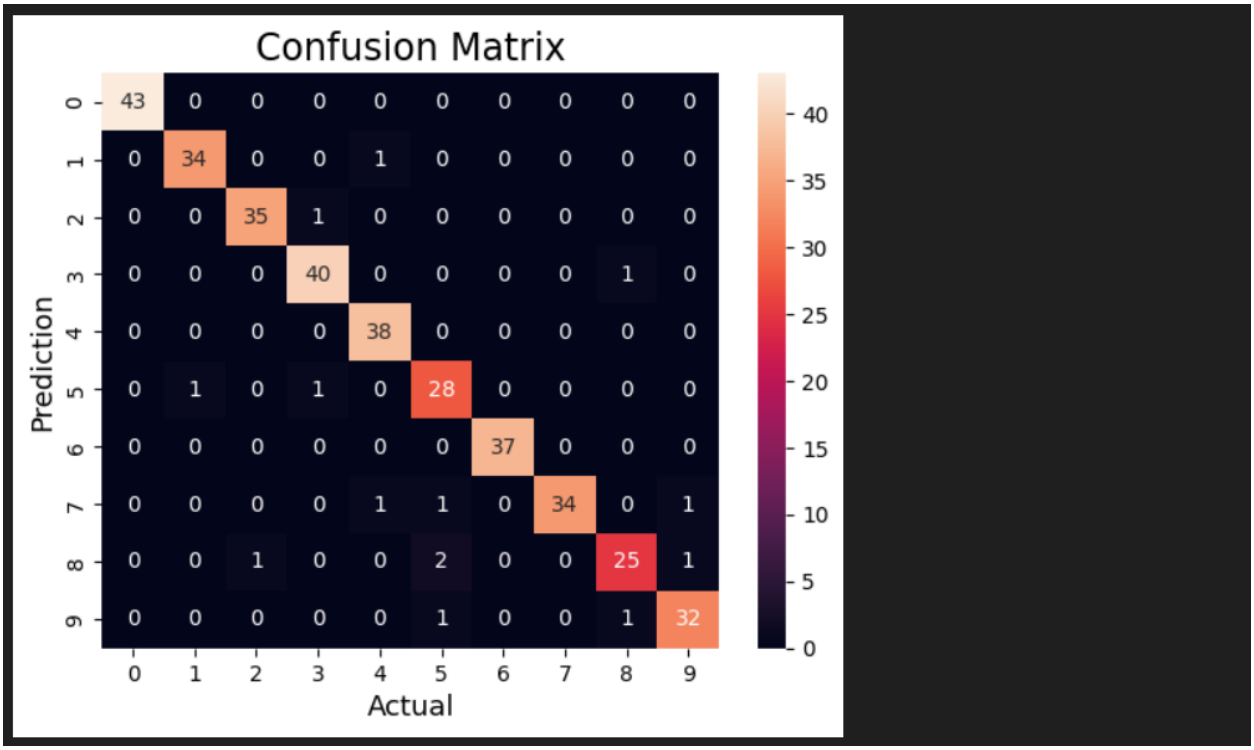
5) ماتریس درهمریختگی را برای حالت تست مدل LinearSVC محاسبه و رسم نمایید.

```python
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(Yte, y_pred, labels=Classifier.classes_)
color = 'white'
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=Classifier.classes_)
disp.plot()
plt.show()
```

✓ 0.5s



تعداد داده هایی که به درستی تشخیص داده شده بر روی قطر اصلی ماتریس قرار گرفته و تعداد خطا در هر سطر با توجه به تفاضل کل داده از مقدار قرارگرفته در قطر اصلی بدست می اید. برای اعداد 0 تا 9 بصورت ستونی جمع تعداد مقادیری که اشتباه تشخیص داده شده غیر از مقدار قرار گرفته در قطر اصلی برای هر نوع داده که در اینجا اعداد 0 تا 9 است قابل ارزیابی است.



6) مدل Svc را با مدل لاجستیک مقایسه کنید. کدام مدل دسته بندی بهتری را انجام داده است؟ علت آن چیست؟

با استفاده از روش gridsearch برای تنظیم بهیته هایپرپارامترها با بهترین ترکیب ممکن به دقت 1 در اموزش و به دقت 0.9972 برای داده های تست رسید.با مقدار

```
{'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
SVC(C=1, gamma=0.001)
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

مدل LogisticRegression به دقت مناسب 1 در زمان اموزش و برای داده تست 0.975 رسید.بنابراین روش svm میزان دقت بالاتری نسبت به لاجستیک دارد.

بخاطر استفاده از تابع کرنل دقت مدل افزایش پیدا کرده است چون مرز جداکننده چندجمله ای تبدیل به یک تابع خطی شده که حاشیه نرم قابل اطمینان براحتی برای داده ها قابل تشخیص و تفکیک شده است.

```python

Classifier = sv.SVC(kernel = 'rbf',C=1e10)
Classifier.fit(Xtr, Ytr)

trAcc=Classifier.score(Xtr,Ytr)
teAcc=Classifier.score(Xte,Yte)

print ('Train accuracy:',trAcc)
print('Test accuracy',teAcc)


param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

grid = GridSearchCV(Classifier, param_grid, refit = True, verbose = 3)

# فیت کردن مدل برای گرید سرچ
grid.fit(Xtr, Ytr)

# بهترین پارامتر بعد از تنظیم
print(grid.best_params_)

# بعد از تنظیم هایپرپارامتر
print(grid.best_estimator_)


```

✓ 22.6s

```
Train accuracy: 1.0
Test accuracy 0.9972222222222222
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.104 total time=   0.3s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.104 total time=   0.2s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.105 total time=   0.4s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.105 total time=   0.3s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.206 total time=   0.3s
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.104 total time=   0.1s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.104 total time=   0.1s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.105 total time=   0.1s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.105 total time=   0.1s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.111 total time=   0.1s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.104 total time=   0.1s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.104 total time=   0.1s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.105 total time=   0.1s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.105 total time=   0.1s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.171 total time=   0.1s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.958 total time=   0.0s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.965 total time=   0.0s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.955 total time=   0.0s
[CV 4/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.937 total time=   0.0s
[CV 5/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.972 total time=   0.0s
[CV 1/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.826 total time=   0.0s
[CV 2/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.868 total time=   0.0s
...
[CV 4/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.983 total time=   0.0s
[CV 5/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.983 total time=   0.0s
{'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
SVC(C=1, gamma=0.001)
```

7) مدل با کرنل خطی را با مدل با کرنلRBFو مقدار Cبهینه شده مقایسه نمایید.تفاوت عملکرد این دو کرنل در چیست؟ پارامترCچه نقشی درپیشبینی با مدلSVCدارد؟

```
import sklearn.svm as sv
from sklearn import model_selection as ms
from sklearn import preprocessing
from sklearn.svm import SVC

X = digits.data
Y = digits.target

Xtr, Xte, Ytr, Yte = ms.train_test_split(X, Y, train_size = 0.8, random_state = 21)


Classifier = sv.SVC(kernel = 'rbf')
Classifier.fit(Xtr, Ytr)

trAcc=Classifier.score(Xtr,Ytr)
teAcc=Classifier.score(Xte,Yte)

print ('Train accuracy:',trAcc)
print('Test accuracy',teAcc)
```

✓ 0.2s

```
Train accuracy: 0.9951287404314544
Test accuracy 0.9861111111111112
```

```
grid_predictions = grid.predict(Xte)

print(classification_report(Yte, grid_predictions))
```
✓ 0.1s

	precision	recall	f1-score	support
0	1.00	1.00	1.00	32
1	0.97	1.00	0.99	36
2	1.00	0.97	0.98	30
3	1.00	1.00	1.00	41
4	1.00	1.00	1.00	32
5	1.00	1.00	1.00	46
6	1.00	1.00	1.00	32
7	0.98	1.00	0.99	40
8	0.98	0.98	0.98	42
9	1.00	0.97	0.98	29
accuracy			0.99	360
macro avg	0.99	0.99	0.99	360
weighted avg	0.99	0.99	0.99	360

در rbf معمولی بدون تنظیم هایپرپارامترها به میزان دقت داده اموزش 0.99 درصد و برای داده های تست به دقت 0.98 درصد رسید و با تنظیم هایپرپارامترها با گرید سرچ توانست دقت مدل را افزایش دهد. با توجه به نتایج بدست امده از تنظیم هایپرپارامترهای روش svm با کرنل rbf با استفاده از روش گریدسرچ به بالاترین دقت برای داده های اموزش با مقدار یک و برای داده های به مقدار 0.99 درصد رسید.

پارامتر Regularization (C) : مقدار آن معمولاً باید بین 1 تا 10 باشد. پیش‌فرض 10 است. برای داده‌های آموزشی باعث افزایش مقدار دقت طبقه‌بندی میشود یا خطای رگرسیون را کاهش می‌دهد ، اما افزایش آن نیز ممکن است باعثoverfitting شود که با استفاده از گرید سرچ به مقدار بهینه 1 برای c رسیدیم.