Train SVM classifier using sklearn digits dataset (i.e. from sklearn.datasets import load_digits) and then,

1. Measure accuracy of your model using different kernels such as rbf and linear.
2. Tune your model further using regularization and gamma parameters and try to come up with highest accurancy score
3. Use 80% of samples as training data size

```
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
import seaborn as sns
```

```
digits=load_digits()
digits
```

```
{'DESCR': ".. _digits_dataset:\n\nOptical recognition of handwritten digits dataset\n
 'data': array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  2., ..., 12.,  0.,  0.],
        [ 0.,  0., 10., ..., 12.,  1.,  0.]]),
 'images': array([[[ 0.,  0.,  5., ...,  1.,  0.,  0.],
         [ 0.,  0., 13., ..., 15.,  5.,  0.],
         [ 0.,  3., 15., ..., 11.,  8.,  0.],
         ...,
         [ 0.,  4., 11., ..., 12.,  7.,  0.],
         [ 0.,  2., 14., ..., 12.,  0.,  0.],
         [ 0.,  0.,  6., ...,  0.,  0.,  0.]],

        [[ 0.,  0.,  0., ...,  5.,  0.,  0.],
         [ 0.,  0.,  0., ...,  9.,  0.,  0.],
         [ 0.,  0.,  3., ...,  6.,  0.,  0.],
         ...,
         [ 0.,  0.,  1., ...,  6.,  0.,  0.],
         [ 0.,  0.,  1., ...,  6.,  0.,  0.],
         [ 0.,  0.,  0., ..., 10.,  0.,  0.]],

        [[ 0.,  0.,  0., ..., 12.,  0.,  0.],
         [ 0.,  0.,  3., ..., 14.,  0.,  0.],
         [ 0.,  0.,  8., ..., 16.,  0.,  0.],
         ...,
         [ 0.,  9., 16., ...,  0.,  0.,  0.],
         [ 0.,  3., 13., ..., 11.,  5.,  0.],
         [ 0.,  0.,  0., ..., 16.,  9.,  0.]],
```

```
        ...,

        [[ 0.,  0.,  1., ...,  1.,  0.,  0.],
         [ 0.,  0., 13., ...,  2.,  1.,  0.],
         [ 0.,  0., 16., ..., 16.,  5.,  0.],
         ...,
         [ 0.,  0., 16., ..., 15.,  0.,  0.],
         [ 0.,  0., 15., ..., 16.,  0.,  0.],
         [ 0.,  0.,  2., ...,  6.,  0.,  0.]],

        [[ 0.,  0.,  2., ...,  0.,  0.,  0.],
         [ 0.,  0., 14., ..., 15.,  1.,  0.],
         [ 0.,  4., 16., ..., 16.,  7.,  0.],
         ...,
         [ 0.,  0.,  0., ..., 16.,  2.,  0.],
         [ 0.,  0.,  4., ..., 16.,  2.,  0.],
         [ 0.,  0.,  5., ..., 12.,  0.,  0.]],

        [[ 0.,  0., 10., ...,  1.,  0.,  0.],
         [ 0.,  2., 16., ...,  1.,  0.,  0.],
         [ 0.,  0., 15., ..., 15.,  0.,  0.],
         ...,
         [ 0.,  4., 16., ..., 16.,  6.,  0.],
         [ 0.,  8., 16., ..., 16.,  8.,  0.],
         [ 0.,  1.,  8., ..., 12.,  1.,  0.]]]),
 'target': array([0, 1, 2, ..., 8, 9, 8]),
```

```
digits.keys()
```

```
dict_keys(['data', 'target', 'target_names', 'images', 'DESCR'])
```

```
df=pd.DataFrame(digits.data)
df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	1
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	13.0	15.0	10.0	15.0	5.0	0.0	0.0	3
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	0.0	11.0	16.0	9.0	0.0	0.0	0.0	0
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	3.0	16.0	15.0	14.0	0.0	0.0	0.0	0
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	13.0	6.0	15.0	4.0	0.0	0.0	0.0	2
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	0.0	7.0	8.0	0.0	0.0	0.0	0.0	0

```
df.shape
```

```
(1797, 64)
```

```
df.columns
```

```
RangeIndex(start=0, stop=64, step=1)
```

```
df['target']=digits.target
df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	1
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	13.0	15.0	10.0	15.0	5.0	0.0	0.0	3
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	0.0	11.0	16.0	9.0	0.0	0.0	0.0	0
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	3.0	16.0	15.0	14.0	0.0	0.0	0.0	0
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	13.0	6.0	15.0	4.0	0.0	0.0	0.0	2
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	0.0	7.0	8.0	0.0	0.0	0.0	0.0	0

```
print(digits.data.shape)
print(digits.target.shape)
```

```
(1797, 64)
(1797,)
```

```
df.target
```

```
0       0
1       1
2       2
3       3
4       4
       ..
1792    9
1793    0
1794    8
1795    9
1796    8
Name: target, Length: 1797, dtype: int64
```

```
df.values
```

```
array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  1.],
       [ 0.,  0.,  0., ...,  9.,  0.,  2.],
       ...,
       [ 0.,  0.,  1., ...,  0.,  0.,  8.],
       [ 0.,  0.,  2., ...,  0.,  0.,  9.],
       [ 0.,  0., 10., ...,  1.,  0.,  8.]])
```

```
from sklearn.model_selection import train_test_split
x=df.drop(['target'],axis='columns')
```

```
y=df.target
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_state=12)


print(len(x_train), len(x_test))
```

```
1437 360
```

```
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
model1=SVC(kernel='rbf',random_state=0, probability=True)
model1.fit(x_train,y_train)
y_pred_1=model1.predict(x_test)
print("Model Score of Kernal(rbf) :", model1.score(x_test,y_test))
```

```
Model Score of Kernal(rbf) : 0.9916666666666667
```

```
model2=SVC(kernel='linear',random_state=0, probability=True)
model2.fit(x_train,y_train)
y_pred_2=model2.predict(x_test)
print("Model Score of Kernal(linear) :", model2.score(x_test,y_test))
```

```
Model Score of Kernal(linear) : 0.975
```

```
model3=SVC(kernel='poly',random_state=0, probability=True)
model3.fit(x_train,y_train)
y_pred_3=model3.predict(x_test)
print("Model Score of Kernal(poly) :", model3.score(x_test,y_test))
```

```
Model Score of Kernal(poly) : 0.9944444444444445
```

```
accuracy=accuracy_score(y_test,y_pred_3)
print('ACCURACY is',accuracy)
```

```
ACCURACY is 0.9944444444444445
```

```
from sklearn.metrics import confusion_matrix
cm=np.array(confusion_matrix(y_test,y_pred_3))
cm
```

```
array([[37,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 32,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 38,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 43,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 39,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0, 32,  0,  0,  0,  2],
       [ 0,  0,  0,  0,  0,  0, 29,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 42,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0, 32,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0, 34]])
```

```
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(y_test,y_pred_3)
mse
```

```
0.08888888888888889
```

```
model1_C=SVC(C=3)
model1_C.fit(x_train,y_train)
model1_C.score(x_test,y_test)
```

```
0.9944444444444445
```

```
model2_C=SVC(C=3)
model2_C.fit(x_train,y_train)
model2_C.score(x_test,y_test)
```

```
0.9944444444444445
```

```
model3_C=SVC(C=3)
model3_C.fit(x_train,y_train)
model3_C.score(x_test,y_test)
```

```
0.9944444444444445
```

```
plt.figure(figsize=(5,5))
sns.heatmap(cm, annot=True, fmt=".2f", linewidths=.5, square = True, cmap = 'Blues_r')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
A=f'Accuracy Score :{accuracy:.2f}'
plt.title(A)
plt.show()
```