ASSIGNMENT/ TASK 12

GO_STP_12574
SHWETA JHA

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 numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

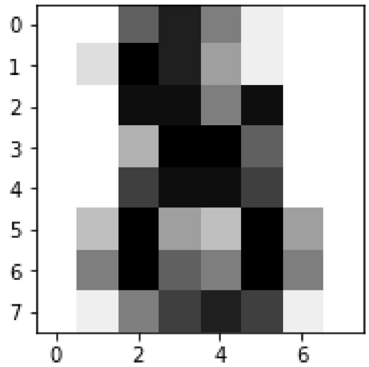
Imprting Data Set

```
from sklearn.datasets import load_digits
digits = load_digits()
print("Image Data Shape:",digits.data.shape)
print("Label Data Shape:",digits.target.shape)
```

```
     Image Data Shape: (1797, 64)
     Label Data Shape: (1797,)
```

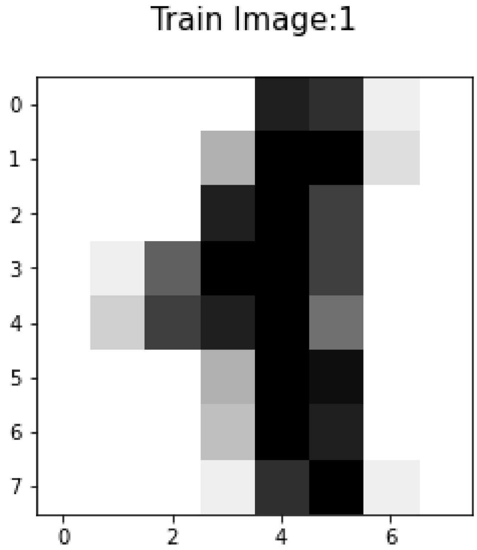
#Display the first digit

```
plt.figure(1, figsize=(3, 3))
plt.imshow(digits.images[-1], cmap=plt.cm.gray_r, interpolation='nearest')
plt.show()
```

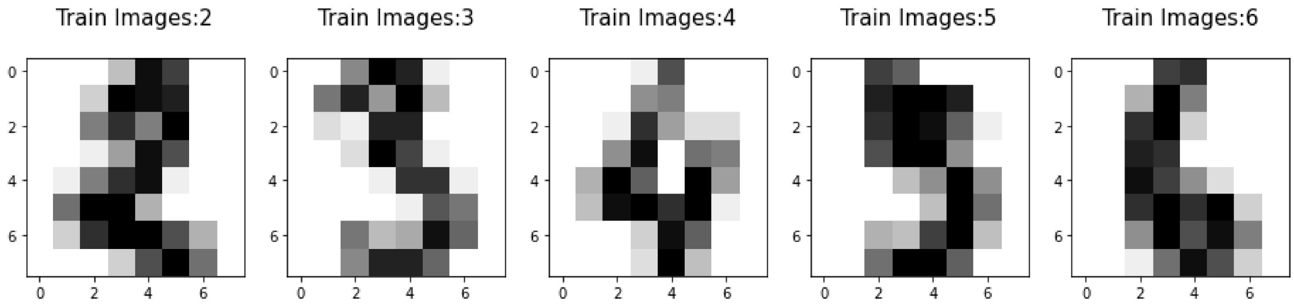


Number Of Digits..

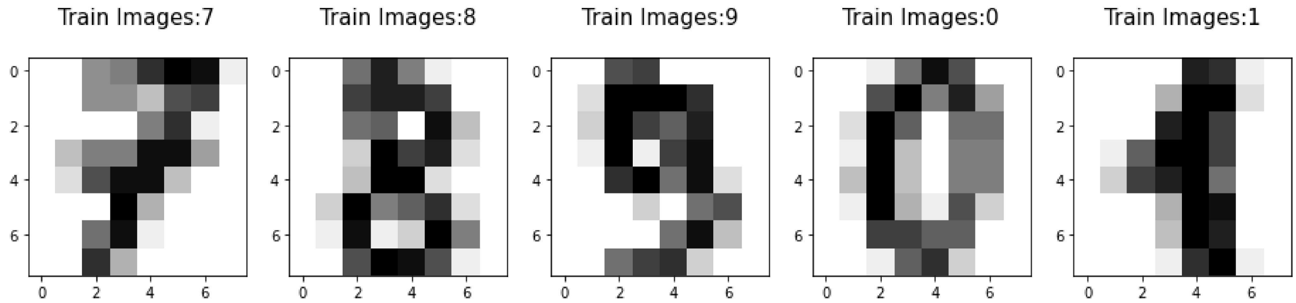
```
image=digits.data[11]
plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.binary,interpolation="nearest")
plt.title("Train Image:%i\n"%digits.target[11],fontsize=15)
plt.show()
```



```
plt.figure(figsize=(16,6))
for i,(image,label) in enumerate(zip(digits.data[2:7],digits.target[2:7])):
  plt.subplot(1,5,i+1)
  plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.binary,interpolation='nearest')
  plt.title("Train Images:%i\n"% label,fontsize=15)
```



```
plt.figure(figsize=(16,6))
for i,(image,label) in enumerate(zip(digits.data[7:12],digits.target[7:12])):
  plt.subplot(1,5,i+1)
  plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.binary,interpolation='nearest')
  plt.title("Train Images:%i\n"% label,fontsize=15)
```



```
import pandas as pd
df=pd.DataFrame(digits.data)
df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
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
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
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
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
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

SPLITING THE DATASET INTO TRANING AND TEST

```
from sklearn.model_selection import train_test_split
```

```
xtrain,xtest,ytrain,ytest=train_test_split(digits.data,digits.target,random_state=12,test_
```

```
print("X Train Shape:",xtrain.shape)
print("y Train Shape:",ytrain.shape)
print("X Test Shape:",xtest.shape)
print("y Test Shape:",ytest.shape)
```

```
X Train Shape: (1437, 64)
y Train Shape: (1437,)
X Test Shape: (360, 64)
y Test Shape: (360,)
```

For RBF

```
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
X, y = load_digits(return_X_y=True)
kernel = 1.0 * RBF(1.0)
gpc = GaussianProcessClassifier(kernel=kernel,
                                random_state=0).fit(X, y)
gpc.score(X, y)
gpc.predict_proba(X[:2,:])
```

```
array([[0.10000045, 0.09999995, 0.09999995, 0.09999995, 0.09999995,
        0.09999995, 0.09999995, 0.09999995, 0.09999995, 0.09999995],
       [0.09999995, 0.10000045, 0.09999995, 0.09999995, 0.09999995,
        0.09999995, 0.09999995, 0.09999995, 0.09999995, 0.09999995]])
```

```
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
model1=SVC(kernel='rbf',random_state=0,probability=True)
model1.fit(xtrain,ytrain)
y_pred1=model1.predict(xtest)
print("Model Score of Kernal(rbf):",model1.score(xtest,ytest))
```

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

For Linear

```
model2=SVC(kernel='linear',random_state=0,probability=True)
model2.fit(xtrain,ytrain)
y_pred2=model2.predict(xtest)
print("Model Score of Kernal(linear):",model2.score(xtest,ytest))
```

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

For Poly

```
model3=SVC(kernel='poly',random_state=0,probability=True)
model3.fit(xtrain,ytrain)
y_pred3=model3.predict(xtest)
print("Model Score of Kernal(poly):",model3.score(xtest,ytest))
```

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

Accuracy

```
accuracy=accuracy_score(ytest,y_pred3)
print("Accuracy is",accuracy)
```

```
Accuracy is 0.9944444444444445
```

```
from sklearn.metrics import  confusion_matrix
```

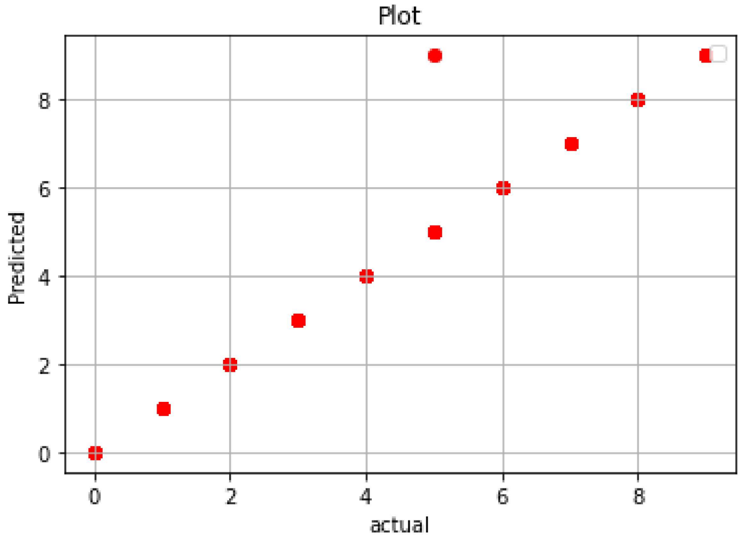
```
cm=np.array(confusion_matrix(ytest,y_pred3))
```

```
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]])
```

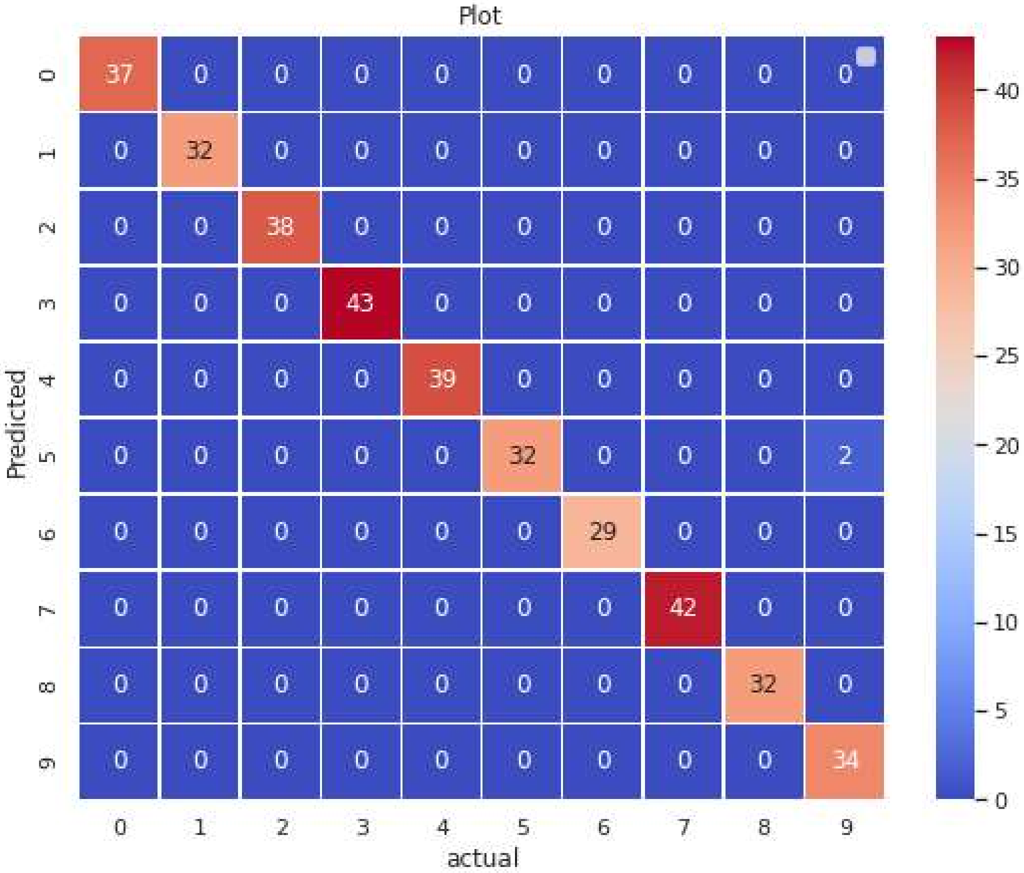
```
plt.scatter(ytest,y_pred3,color='r')
plt.xlabel('actual')
plt.ylabel('Predicted')
plt.title("Plot")
plt.grid()
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



```
sns.heatmap(cm, cmap='coolwarm', annot=True, linewidths=0.30)
plt.xlabel('actual')
plt.ylabel('Predicted')
plt.title("Plot")
plt.grid()
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



Plot

	0	1	2	3	4	5	6	7	8	9
0	37	0	0	0	0	0	0	0	0	0
1	0	32	0	0	0	0	0	0	0	0
2	0	0	38	0	0	0	0	0	0	0
3	0	0	0	43	0	0	0	0	0	0
4	0	0	0	0	39	0	0	0	0	0
5	0	0	0	0	0	32	0	0	0	2
6	0	0	0	0	0	0	29	0	0	0
7	0	0	0	0	0	0	0	42	0	0
8	0	0	0	0	0	0	0	0	32	0
9	0	0	0	0	0	0	0	0	0	34

✓ 1s completed at 5:27 AM