Assignment:

Try SVM classifier on MNIST dataset, compare the preformance of linear, polynomial and RBF kernels.

```
import sys, os
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.svm import SVC
import pandas as pd
```

```
train_data=pd.read_csv('/content/mnist_train.csv')
test_data=pd.read_csv('/content/mnist_test.csv')
```

```
train_data.shape
```

```
(60000, 785)
```

```
test_data.shape
```

```
(10000, 785)
```

```
train_data.head()
```

	label	1x1	1x2	1x3	1x4	1x5	1x6	1x7	1x8	1x9	1x10	1x11	1x12	1x13	1x14	1x1
0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

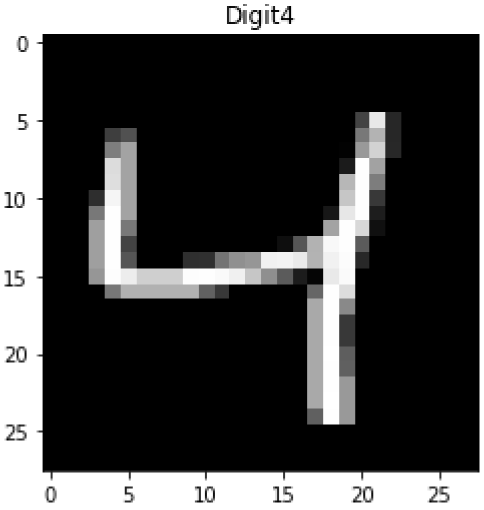
5 rows × 785 columns

```
test_data.head()
```

	label	1x1	1x2	1x3	1x4	1x5	1x6	1x7	1x8	1x9	1x10	1x11	1x12	1x13	1x14	1x1
0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

```
x = train_data.iloc[2, 1:]
x.shape
x = x.values.reshape(28,28)
plt.imshow(x, cmap='gray')
plt.title("Digit"+str(train_data.iloc[2, 0]))
```

```
Text(0.5, 1.0, 'Digit4')
```



```
# data types
print(train_data.info())
print(test_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60000 entries, 0 to 59999
Columns: 785 entries, label to 28x28
dtypes: int64(785)
memory usage: 359.3 MB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Columns: 785 entries, label to 28x28
dtypes: int64(785)
memory usage: 59.9 MB
None
```

```
## Separating the X and Y variable
y_train = train_data['label']
y_test = test_data['label']
## Dropping the variable 'label' from X variable
```

```
X_train = train_data.drop(columns = 'label')
X_test = test_data.drop(columns = 'label')


X_train = X_train/255.0
X_test=X_test/255.0
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
```

```
X_train: (60000, 784)
X_test: (10000, 784)
```

```
# linear model

model_linear = SVC(kernel='linear')
model_linear.fit(X_train, y_train)

# predict
y_pred = model_linear.predict(X_test)


# confusion matrix and accuracy

from sklearn import metrics
from sklearn.metrics import confusion_matrix
# accuracy
print("accuracy:", metrics.accuracy_score(y_test,y_pred), "\n")
print("Precision:",metrics.precision_score(y_test, y_pred,average='macro'), "\n")
print("Recall:",metrics.recall_score(y_test, y_pred,average='macro'), "\n")

# cm
print(metrics.confusion_matrix(y_true=y_test, y_pred=y_pred))
```

```
accuracy: 0.9404

Precision: 0.9399179178125057

Recall: 0.9394223313176161

[[ 957    0    4    1    1    6    9    1    0    1]
 [   0 1122    3    2    0    1    2    1    4    0]
 [   8    6  967   11    3    3    7    8   17    2]
 [   4    3   16  947    1   16    0    9   12    2]
 [   1    1   10    1  942    2    4    2    3   16]
 [  10    4    3   36    6  803   13    1   14    2]
 [   9    2   13    1    5   16  910    1    1    0]
 [   1    8   21   10    8    1    0  957    3   19]
 [   8    4    6   25    7   26    6    7  877    8]
 [   7    7    2   11   33    4    0   18    5  922]]
```

```
# polynomial model
```

```
model_poly = SVC(kernel='poly')
model_poly.fit(X_train, y_train)

# predict
y_pred_poly = model_poly.predict(X_test)


# confusion matrix and accuracy

# accuracy
print("accuracy:", metrics.accuracy_score(y_test,y_pred_poly), "\n")
print("Precision:",metrics.precision_score(y_test, y_pred_poly,average='macro'), "\n")
print("Recall:",metrics.recall_score(y_test, y_pred_poly,average='macro'), "\n")

# cm
print(metrics.confusion_matrix(y_true=y_test, y_pred=y_pred_poly))
```

```
accuracy: 0.9771

Precision: 0.9771145861721802

Recall: 0.9769380249358539

[[ 969    0    2    0    0    4    2    1    2    0]
 [   0 1127    2    1    0    0    3    0    2    0]
 [   6    2 1006    0    2    1    3    9    3    0]
 [   0    2    3  984    0    6    0    6    5    4]
 [   1    0    3    0  966    0    4    0    0    8]
 [   2    0    0    8    1  869    4    1    5    2]
 [   4    4    2    0    3    6  937    0    2    0]
 [   0   15    8    1    1    0    0  995    0    8]
 [   1    1    2    5    5    5    0    3  949    3]
 [   3    6    1    4   14    5    0    6    1  969]]
```

```
# rbf model

model_rbf = SVC(kernel='rbf')
model_rbf.fit(X_train, y_train)

# predict
y_pred_rbf = model_rbf.predict(X_test)


# confusion matrix and accuracy

# accuracy
print("accuracy:", metrics.accuracy_score(y_test,y_pred_rbf), "\n")
print("Precision:",metrics.precision_score(y_test, y_pred_rbf,average='macro'), "\n")
print("Recall:",metrics.recall_score(y_test, y_pred_rbf,average='macro'), "\n")
# cm
print(metrics.confusion_matrix(y_true=y_test, y_pred=y_pred_rbf))
```

```
accuracy: 0.9792

Precision: 0.9791973966593345

Recall: 0.9790919842945065

[[ 973    0    1    0    0    2    1    1    2    0]
 [   0 1126    3    1    0    1    1    1    2    0]
 [   6    1 1006    2    1    0    2    7    6    1]
 [   0    0    2  995    0    2    0    5    5    1]
 [   0    0    5    0  961    0    3    0    2   11]
 [   2    0    0    9    0  871    4    1    4    1]
 [   6    2    0    0    2    3  944    0    1    0]
 [   0    6   11    1    1    0    0  996    2   11]
 [   3    0    2    6    3    2    2    3  950    3]
 [   3    4    1    7   10    2    1    7    4  970]]
```