In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
```

In [ ]:
```
from sklearn import svm
from sklearn.svm import SVC
```

In [ ]:
```
train_data = pd.read_csv("archive\MNIST\mnist_train.csv") #reading the csv files
test_data = pd.read_csv("archive\MNIST\mnist_test.csv")
```

In [ ]:
```
df = train_data

df.describe()
```

Out[ ]:

	label	1x1	1x2	1x3	1x4	1x5	1x6	1x7	
count	60000.000000	60000.0	60000.0	60000.0	60000.0	60000.0	60000.0	60000.0	600(
mean	4.453933	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
std	2.889270	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
min	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
25%	2.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
50%	4.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
75%	7.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
max	9.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

8 rows × 785 columns

In [ ]:
```
df.shape
```

Out[ ]:
```
(60000, 785)
```

In [ ]:
```
df.head()
```

Out[ ]:

	label	1x1	1x2	1x3	1x4	1x5	1x6	1x7	1x8	1x9	...	28x19	28x20	28x21	28
0	5	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	
2	4	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	
4	9	0	0	0	0	0	0	0	0	0	...	0	0	0	

5 rows × 785 columns

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
label    0
1x1      0
1x2      0
1x3      0
1x4      0
        ..
28x24    0
28x25    0
28x26    0
28x27    0
28x28    0
Length: 785, dtype: int64
```

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['label', '1x1', '1x2', '1x3', '1x4', '1x5', '1x6', '1x7', '1x8', '1x9',
       ...
       '28x19', '28x20', '28x21', '28x22', '28x23', '28x24', '28x25', '28x26',
       '28x27', '28x28'],
      dtype='object', length=785)
```

In [ ]:
```
order = list(np.sort(df['label'].unique()))
print(order)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [ ]:
```
y = train_data['label']

X = train_data.drop(columns = 'label')

print(train_data.shape)
```

```
(60000, 785)
```

In [ ]:
```
## Normalization

X = X/255.0
test_data = test_data/255.0

print("X:", X.shape)
print("test_data:", test_data.shape)
```

```
X: (60000, 784)
test_data: (10000, 785)
```

In [ ]:
```
from sklearn.preprocessing import scale
X_scaled = scale(X)

# train test split
x_train, x_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.3
```

In [ ]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

In [ ]:
```
linearSVM = svm.SVC(kernel='linear')
polynomialSVM = svm.SVC(kernel='poly', degree=3)
rbfSVM = svm.SVC(kernel='rbf')
sigmoidSVM = svm.SVC(kernel='sigmoid')
```

In [ ]:
```
linearSVM.fit(x_train,y_train)
linear_predictions = linearSVM.predict(x_test)
```

In [ ]:
```
polynomialSVM.fit(x_train,y_train)
poly_predictions = polynomialSVM.predict(x_test)
```

In [ ]:
```
rbfSVM.fit(x_train,y_train)
rbf_predictions = rbfSVM.predict(x_test)
```

In [ ]:
```
sigmoidSVM.fit(x_train,y_train)
sigmoid_predictions = sigmoidSVM.predict(x_test)
```

In [ ]:
```
linear_accuracy = accuracy_score(y_test, linear_predictions)
poly_accuracy = accuracy_score(y_test, poly_predictions)
rbf_accuracy = accuracy_score(y_test, rbf_predictions)
sigmoid_accuracy = accuracy_score(y_test, sigmoid_predictions)

print("Linear SVM Accuracy:", linear_accuracy)
print("Polynomial SVM Accuracy:", poly_accuracy)
print("RBF SVM Accuracy:", rbf_accuracy)
print("Sigmoid SVM Accuracy:", sigmoid_accuracy)
```

```
Linear SVM Accuracy: 0.9103333333333333
Polynomial SVM Accuracy: 0.9132222222222223
RBF SVM Accuracy: 0.943
Sigmoid SVM Accuracy: 0.9010555555555556
```

In [ ]:
```
from sklearn import metrics

print("\nConfusion matrix for linear kernel\n" )
print(metrics.confusion_matrix(y_true=y_test, y_pred=linear_predictions))
print("\nConfusion matrix for poly kernel\n" )
print(metrics.confusion_matrix(y_true=y_test, y_pred=poly_predictions))
print("\nConfusion matrix for rbf kernel\n" )
print(metrics.confusion_matrix(y_true=y_test, y_pred=rbf_predictions))
print("\nConfusion matrix for sigmoid kernel\n" )
print(metrics.confusion_matrix(y_true=y_test, y_pred=sigmoid_predictions))
```

```
Confusion matrix for linear kernel

[[1719    0   10    5    3   16   12    1    6    0]
 [   1 1951   11    5    5    4    0    3   11    1]
 [  11   26 1676   30   23    5   23   19   17    1]
 [  10    4   47 1627    4   66    5   19   42   10]
 [   4    8   21    1 1658    5   14    6    5   50]
 [  21    9   19   87   12 1423   30    1   39   11]
 [  20    7   23    1   14   20 1666    2    4    0]
 [   7   13   19   16   31    4    2 1774    5   93]
 [  25   44   49   54   12   58   18   11 1436   17]
 [   4   11   19   23   90    9    1   69   20 1456]]

Confusion matrix for poly kernel

[[1649    0    7    2    8    9   11    1   84    1]
 [   0 1941    8    5    6    0    2    1   28    1]
 [   4    8 1576   15   53    2    5    7  159    2]
 [   1    2   15 1644    6   24    0   11  113   18]
 [   0    5   14    0 1685    3    4    0    5   56]
 [   2    1    1   33   27 1388   18    2  149   31]
 [   3    4    3    0   26   14 1657    0   50    0]
 [   1   14    5    1   76    1    0 1692   28  146]
 [   3    6   14   12   11   12    4    1 1650   11]
 [   2    6    4   14   62    4    0   16   38 1556]]

Confusion matrix for rbf kernel

[[1722    0   15    4    1    6   13    2    8    1]
 [   1 1947   21    7    5    0    1    2    7    1]
 [   5    6 1747   11   12    3   14   16   14    3]
 [   2    3   52 1685    1   37    2   21   26    5]
 [   1    5   31    1 1664    5    9    7    4   45]
 [   3    5   28   33    3 1526   27    5   15    7]
 [   8    4   21    0    4   14 1698    1    7    0]
 [   3   11   52    7   14    0    0 1828    2   47]
 [  10   22   32   18    9   26   11    5 1585    6]
 [   3    5   25   17   23    4    0   38   15 1572]]

Confusion matrix for sigmoid kernel

[[1695    0   19    4    4   26    9    1   12    2]
 [   1 1940   15    8    3    8    0    2   11    4]
 [  23   19 1593   26   26    6   84   12   34    8]
 [   8    4   56 1603    5   86    6   26   30   10]
 [   4    7   33    2 1623    6   13   10    3   71]
 [  16   21   22   62   15 1424   29    4   36   23]
 [  19   11   41    0   20   26 1633    0    7    0]
 [   7   20   25   25   22    2    4 1749    5  105]
 [  20   49   43   38    8   57   14    8 1473   14]
 [   7    8   23   23   71    9    0   64   11 1486]]
```

TRAINING ACCURACIES

In [ ]:
```
svm_linear_model = svm.SVC(kernel='linear')
svm_linear_model.fit(x_train, y_train)

train_predictions_linear = svm_linear_model.predict(x_train)
```

```
train_accuracy_linear = accuracy_score(y_train, train_predictions_linear)
print("Training Accuracy (Linear Kernel):", train_accuracy_linear)
```

```
Training Accuracy (Linear Kernel): 1.0
```

In [ ]:
```
svm_poly_model = svm.SVC(kernel='poly')
svm_poly_model.fit(x_train, y_train)

train_predictions_poly = svm_poly_model.predict(x_train)
train_accuracy_poly = accuracy_score(y_train, train_predictions_poly)
print("Training Accuracy (poly Kernel):", train_accuracy_poly)
```

```
Training Accuracy (poly Kernel): 0.95025
```

In [ ]:
```
svm_rbf_model = svm.SVC(kernel='rbf')
svm_rbf_model.fit(x_train, y_train)

train_predictions_rbf = svm_rbf_model.predict(x_train)
train_accuracy_rbf = accuracy_score(y_train, train_predictions_rbf)
print("Training Accuracy (rbf Kernel):", train_accuracy_rbf)
```

```
Training Accuracy (rbf Kernel): 0.98075
```

In [ ]:
```
svm_sigmoid_model = svm.SVC(kernel='sigmoid')
svm_sigmoid_model.fit(x_train, y_train)

train_predictions_sigmoid = svm_sigmoid_model.predict(x_train)
train_accuracy_sigmoid = accuracy_score(y_train, train_predictions_sigmoid)
print("Training Accuracy (sigmoid Kernel):", train_accuracy_sigmoid)
```

```
Training Accuracy (sigmoid Kernel): 0.9099166666666667
```

In [ ]:
```
print("SVM model accuracies for different kernels\n")


print("Training accuracis:")
print("\n\t\tLinear kerenel: ",train_accuracy_linear)
print("\n\t\tpolynomial kerenel: ",train_accuracy_poly)
print("\n\t\trbf kerenel: ",train_accuracy_rbf)
print("\n\t\tSigmoid kerenel: ",train_accuracy_sigmoid)


print("\n\nTesting accuracis:")
print("\n\t\tLinear kerenel: ",linear_accuracy)
print("\n\t\tpolynomial kerenel: ",poly_accuracy)
print("\n\t\trbf kerenel: ",rbf_accuracy)
print("\n\t\tSigmoid kerenel: ",sigmoid_accuracy)
```

```
SVM model accuracies for different kernels

Training accuracis:

                Linear kerenel:  1.0

                polynomial kerenel:  0.95025

                rbf kerenel:  0.98075

                Sigmoid kerenel:  0.9099166666666667


Testing accuracis:

                Linear kerenel:  0.9103333333333333

                polynomial kerenel:  0.9132222222222223

                rbf kerenel:  0.943

                Sigmoid kerenel:  0.9010555555555556
```

All kernels of SVM models produce a good accuracy.