In [3]:
```
#Author : Sayali Kudale

#loading the required libraries

import scipy.io
import numpy as np
import pandas as pd
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
import time
```

In [4]:
```
#Data Loading

mat = scipy.io.loadmat('Data/USPS_all.mat')
df_features = pd.DataFrame(mat['fea'])
df_gnd = pd.DataFrame(mat['gnd'])


#Split data into training and testing set

train_fea= df_features.iloc[:7291]
test_fea= df_features.iloc[-2007:]
train_dec= df_gnd.iloc[:7291]
test_dec= df_gnd.iloc[-2007:]
```

SVM Model - Linear Kernel

In [5]:
```
# Creating the SVM model
model = OneVsRestClassifier(SVC(kernel='linear'))

trainStart = time.time()
# Fitting the model with training data
model.fit(train_fea, train_dec)

trainEnd = time.time()

print("Training time : ", trainEnd-trainStart)
```

```
Training time :  20.14596390724182
```

In [6]:
```
# Making a prediction on the test set
prediction = model.predict(test_fea)
```

In [7]:
```
# Evaluating the model
score=accuracy_score(test_dec, prediction) * 100
print("Accuracy Score : %s" %score)
```

```
Accuracy Score : 91.6292974589
```

In [10]:
```
classsification_report=classification_report(test_dec, prediction)
```

In [11]:
```
print('Classification report : \n',classsification_report)
```

```
Classification report :
              precision    recall  f1-score   support

           1       0.95      0.97      0.96       359
           2       0.98      0.97      0.98       264
           3       0.87      0.85      0.86       198
           4       0.86      0.86      0.86       166
           5       0.86      0.89      0.87       200
           6       0.89      0.89      0.89       160
           7       0.96      0.95      0.96       170
           8       0.93      0.90      0.92       147
           9       0.88      0.87      0.87       166
          10       0.91      0.93      0.92       177

avg / total       0.92      0.92      0.92      2007
```

In [16]:
```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(test_dec, prediction)
print(cm)
```

```
[[350   0   2   1   3   0   0   0   3   0]
 [  0 255   1   3   2   0   2   0   1   0]
 [  5   0 168   6   9   1   1   2   6   0]
 [  2   0   4 142   2   8   0   3   3   2]
 [  1   2   7   0 177   1   2   2   3   5]
 [  3   0   0   7   3 143   0   0   0   4]
 [  0   0   3   0   2   2 162   0   1   0]
 [  2   0   2   2   4   0   0 133   0   4]
 [  5   0   4   4   1   6   1   0 144   1]
 [  0   2   2   0   2   0   0   3   3 165]]
```

SVM Model - Polynomial Kernel function

In [26]:
```
# Creating the SVM model
modelPoly = OneVsRestClassifier(SVC(kernel='poly'))
trainStart = time.time()
# Fitting the model with training data
modelPoly.fit(train_fea, train_dec)
trainEnd = time.time()

print("Training time : ", trainEnd-trainStart)
```

```
Training time :  18.499656915664673
```

In [18]:
```
# Making a prediction on the test set
predictionPoly = modelPoly.predict(test_fea)
```

In [19]:
```
# Evaluating the model
scorePoly=accuracy_score(test_dec, predictionPoly) * 100
print("Accuracy Score : %s" %scorePoly)
```

```
Accuracy Score : 93.2735426009
```

In [20]:
```
classsification_reportPoly=classification_report(test_dec, predictionPol
y)
print('Classification report of Polynomial kernel function: \n',classsif
ication_reportPoly)
```

```
Classification report of Polynomial kernel function:
              precision    recall  f1-score   support

          1       0.94      0.99      0.96       359
          2       0.98      0.97      0.98       264
          3       0.90      0.89      0.90       198
          4       0.92      0.90      0.91       166
          5       0.88      0.93      0.90       200
          6       0.90      0.91      0.90       160
          7       0.96      0.92      0.94       170
          8       0.96      0.91      0.94       147
          9       0.94      0.88      0.91       166
         10       0.93      0.95      0.94       177

avg / total       0.93      0.93      0.93      2007
```

In [21]:
```
cm_poly= confusion_matrix(test_dec, predictionPoly)
print("Confusion Matrix\n",cm_poly)
```

```
Confusion Matrix
 [[354   0   2   0   2   0   0   0   0   1]
 [  0 255   1   1   4   0   3   0   0   0]
 [  6   0 177   3   5   2   1   1   3   0]
 [  3   0   4 149   0   6   0   2   1   1]
 [  1   2   6   0 185   0   2   1   0   3]
 [  5   0   0   4   1 146   0   0   0   4]
 [  4   0   3   0   3   2 157   0   1   0]
 [  0   0   2   0   6   0   0 134   2   3]
 [  4   1   2   5   1   6   0   0 146   1]
 [  0   1   0   0   3   1   0   1   2 169]]
```