Letters: Optical Character Recognition Model

Support Vector Machine Model (Linear kernel)

In [38]:
```
import pandas as pd
```

In [39]:
```
pwd
```

Out[39]:
```
'/Users/pratik'
```

In [40]:
```
input_file = "/Users/pratik/Desktop/Harrisburg University programs/Courses/Late Fall Cou
```

In [41]:
```
letters = pd.read_csv(input_file)
```

In [42]:
```
letters.head(5)
```

Out[42]:

	letter	xbox	ybox	width	height	onpix	xbar	ybar	x2bar	y2bar	xybar	x2ybar	xy2bar	xedge	xe
0	T	2	8	3	5	1	8	13	0	6	6	10	8	0	
1	I	5	12	3	7	2	10	5	5	4	13	3	9	2	
2	D	4	11	6	8	6	10	6	2	6	10	3	7	3	
3	N	7	11	6	6	3	5	9	4	6	4	4	10	6	
4	G	2	1	3	1	1	8	6	6	6	6	5	9	1	

In [43]:
```
target = letters["letter"]
```

Q3:

The Target variable is letter

Train and Test Split, (convert y to numeric), import preprocessing from sklearn and LabelEncoder from preprocessing

In [44]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
x = letters.drop(['letter'], axis = 1) ## Converting letters to digits
numbers = LabelEncoder()
y = numbers.fit_transform(letters['letter'].astype('str'))
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.10, random_state
```

In [45]:
```
y_train
```

Out[45]:
```
array([ 3, 17, 18, ..., 10,  6,  6])
```

In [46]:
```
pd.Series(y_train).value_counts()
```

Out[46]:
```
20    740
3     729
15    722
16    717
0     714
```

```
19    714
13    713
23    705
24    703
12    701
6     693
11    691
1     690
4     690
5     689
17    686
21    686
8     682
18    676
22    673
14    672
9     671
2     664
7     663
25    660
10    656
dtype: int64
```

Train the model

Use Linear separator to design the model

In [49]:
```
from sklearn import svm
```

Designing the model

In [51]:
```
clf = svm.SVC(kernel = "linear")
```

In [52]:
```
clf
```

Out[52]:
```
SVC(kernel='linear')
```

In [53]:
```
clf
```

Out[53]:
```
SVC(kernel='linear')
```

In [54]:
```
clf.fit(x_train, y_train)
```

Out[54]:
```
SVC(kernel='linear')
```

In [55]:
```
y_predict = clf.predict(x_test)
```

Designing the Confusion Matrix

In [58]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
confusion_matrix(y_test, y_predict)
```

Out[58]:
```
array([[71,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,
         0,  1,  0,  0,  0,  0,  0,  0,  2,  0],
       [ 1, 67,  0,  1,  0,  0,  2,  1,  0,  0,  1,  0,  0,  0,  0,  0,
         0,  2,  1,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 67,  0,  2,  0,  1,  1,  0,  0,  0,  0,  0,  0,  1,  0,
         0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  5,  0, 67,  0,  0,  0,  1,  0,  1,  0,  0,  0,  0,  1,  0,
```

```
  0,  0,  0,  0,  1,  0,  0,  0,  0,  0],
[ 0,  1,  1,  0, 58,  1,  3,  0,  1,  0,  1,  4,  0,  0,  0,  0,
  2,  0,  1,  3,  0,  0,  0,  0,  0,  2],
[ 0,  1,  1,  0,  0, 74,  0,  0,  0,  1,  0,  0,  0,  2,  0,  1,
  0,  0,  3,  2,  0,  0,  0,  0,  1,  0],
[ 0,  0,  4,  3,  1,  0, 61,  0,  0,  0,  1,  2,  0,  0,  0,  1,
  2,  0,  4,  0,  0,  0,  1,  0,  0,  0],
[ 0,  1,  0,  0,  0,  2,  0, 52,  0,  1,  1,  1,  0,  0,  7,  0,
  0,  3,  0,  0,  2,  0,  0,  1,  0,  0],
[ 0,  1,  0,  1,  0,  2,  0,  0, 67,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  1,  0,  1],
[ 2,  0,  0,  0,  0,  1,  0,  0,  5, 67,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0,  0,  0,  1],
[ 0,  1,  2,  1,  0,  0,  3,  2,  0,  0, 68,  0,  0,  0,  0,  0,
  0,  3,  0,  1,  0,  0,  0,  2,  0,  0],
[ 0,  0,  0,  0,  1,  0,  1,  1,  0,  0,  1, 63,  0,  0,  0,  0,
  2,  0,  1,  0,  0,  0,  0,  0,  0,  0],
[ 0,  1,  0,  0,  0,  0,  0,  3,  0,  0,  0,  0, 83,  0,  1,  0,
  0,  2,  0,  0,  1,  0,  0,  0,  0,  0],
[ 0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  1,  0,  0, 65,  1,  0,
  0,  1,  0,  0,  0,  1,  0,  0,  0,  0],
[ 1,  0,  1,  1,  0,  0,  0,  7,  0,  0,  0,  0,  1,  0, 66,  0,
  1,  1,  0,  0,  1,  0,  1,  0,  0,  0],
[ 0,  0,  0,  0,  0,  9,  0,  0,  0,  0,  0,  0,  0,  0,  0, 69,
  0,  0,  0,  0,  0,  0,  1,  0,  2,  0],
[ 3,  0,  0,  1,  0,  0,  1,  2,  0,  0,  1,  0,  0,  0,  3,  1,
 49,  0,  4,  0,  0,  1,  0,  0,  0,  0],
[ 1,  3,  1,  1,  0,  0,  1,  1,  0,  0,  1,  0,  0,  0,  1,  0,
  0, 62,  0,  0,  0,  0,  0,  0,  0,  0],
[ 0,  3,  0,  0,  2,  1,  2,  0,  1,  0,  0,  1,  0,  0,  0,  0,
  4,  0, 51,  0,  0,  0,  0,  1,  0,  6],
[ 0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,
  0,  0,  1, 76,  0,  0,  0,  0,  3,  0],
[ 0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  1,  0,  1,  0,
  0,  0,  0,  0, 70,  0,  0,  0,  0,  0],
[ 1,  1,  0,  0,  0,  0,  1,  1,  0,  0,  0,  0,  1,  0,  0,  0,
  0,  0,  0,  0,  0, 69,  0,  0,  4,  0],
[ 0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  4,  1,  1,  0,
  0,  0,  0,  0,  3,  1, 68,  0,  0,  0],
[ 0,  0,  0,  0,  2,  0,  0,  0,  1,  1,  2,  2,  0,  0,  0,  0,
  0,  0,  0,  0,  0,  0,  0, 73,  1,  0],
[ 1,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  1,  0,  1,  0,  3, 76,  0],
[ 0,  0,  0,  0,  2,  0,  0,  0,  0,  3,  0,  0,  0,  0,  0,  0,
  0,  0,  7,  0,  0,  0,  0,  0,  0, 62]])
```

In [59]:
```
accuracy_score(y_test, y_predict)*100
```

Out[59]:
```
86.05000000000001
```

Improving the model by changing the kernels

Improving model 1

In [81]:
```
clf = svm.SVC(kernel='rbf', gamma =0.3)
#clf = svm.LinearSVC(C=1)
clf.fit(x_train, y_train)
y_predict = clf.predict(x_test)
accuracy_score(y_test, y_predict)*100
```

Out[81]:
```
92.9
```

Improving model 2

In [83]:
```
clf = svm.SVC(kernel='poly', degree=8)
#clf = svm.LinearSVC(C=1)
clf.fit(x_train, y_train)
y_predict = clf.predict(x_test)
accuracy_score(y_test, y_predict)*100
```

Out[83]:
```
94.39999999999999
```

Conclusion:

The Support Vector Machines using Linear Kernel identified the Letters with an accuracy of 86%, which increased to 92% on using the rbf kernel and further to 94% on using the polynomial kernel.

In [ ]: