In [1]:
```
# In the section below, we are importing libraries which will be used in the pro
# <numpy> to compute mean error for the predicted values
# <pandas> to load and parse the csv file into meaningful dataframes
import numpy as np
import pandas as pd
```

In [2]:
```
# In the section below, we are loading training data csv file into dataframe nam
df_train = pd.read_csv('zip.train.p.csv')
print(df_train.head())
```

```
   6.0000  -1.0000  -1.0000.1  -1.0000.2  -1.0000.3  -1.0000.4  -1.0000.5  \
0     5.0     -1.0       -1.0       -1.0     -0.813     -0.671     -0.809
1     4.0     -1.0       -1.0       -1.0     -1.000     -1.000     -1.000
2     7.0     -1.0       -1.0       -1.0     -1.000     -1.000     -0.273
3     3.0     -1.0       -1.0       -1.0     -1.000     -1.000     -0.928
4     6.0     -1.0       -1.0       -1.0     -1.000     -1.000     -0.397

   -1.0000.6  -0.6310  0.8620  ...  0.8230  1.0000.39  0.4820  -0.4740  \
0     -0.887   -0.671  -0.853  ...  -0.671     -0.033   0.761    0.762
1     -1.000   -1.000  -1.000  ...  -1.000     -1.000  -0.109    1.000
2      0.684    0.960   0.450  ...   1.000      0.536  -0.987   -1.000
3     -0.204    0.751   0.466  ...   0.639      1.000   1.000    0.791
4      0.983   -0.535  -1.000  ...   0.015     -0.862  -0.871   -0.437

   -0.9910  -1.0000.121  -1.0000.122  -1.0000.123  -1.0000.124  Unnamed: 257
0    0.126       -0.095       -0.671       -0.828         -1.0           NaN
1   -0.179       -1.000       -1.000       -1.000         -1.0           NaN
2   -1.000       -1.000       -1.000       -1.000         -1.0           NaN
3    0.439       -0.199       -0.883       -1.000         -1.0           NaN
4   -1.000       -1.000       -1.000       -1.000         -1.0           NaN

[5 rows x 258 columns]
```

In [3]:
```
# In the section below, we are separating attributes and labels for training dat
X_train = df_train.iloc[:, 1:256].values
Y_train = df_train.iloc[:, 0].values
```

In [4]:
```
# In the section below, we are loading test data csv file into dataframe named d
df_test = pd.read_csv('zip.test.p.csv')
df_test.head()
```

Out[4]:

	9	-1	-1.1	-1.2	-1.3	-1.4	-0.948	-0.561	0.148	0.384	...	-1.136	-0.908	0.43	0.
0	6	-1.0	-1.0	-1.0	-1.000	-1.0	-1.0	-1.000	-1.000	-1.000	...	-1.000	-1.000	-1.000	-1.
1	3	-1.0	-1.0	-1.0	-0.593	0.7	1.0	1.000	1.000	1.000	...	1.000	0.717	0.333	0
2	6	-1.0	-1.0	-1.0	-1.000	-1.0	-1.0	-1.000	-1.000	-1.000	...	-1.000	-1.000	-1.000	-1.
3	6	-1.0	-1.0	-1.0	-1.000	-1.0	-1.0	-1.000	-0.858	-0.106	...	0.901	0.901	0.901	0.
4	0	-1.0	-1.0	-1.0	-1.000	-1.0	-1.0	0.195	1.000	0.054	...	0.224	1.000	0.988	0

5 rows × 257 columns

In [5]:
```
# In the section below, we are separating attributes and labels for test data.
X_test = df_test.iloc[:, 1:256].values
Y_test = df_test.iloc[:, 0].values
```

In [6]:
```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

#Create a Gaussian Classifier
model = GaussianNB()

#Train the model using the training sets
model.fit(X_train,Y_train)

#Predict Output
Y_pred= model.predict(X_test)
```

In [7]:
```
# Print out classification report
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(Y_test, Y_pred))
# Print out confusion matrix
print(confusion_matrix(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.82      0.87       359
           1       0.91      0.96      0.94       264
           2       0.77      0.72      0.74       198
           3       0.90      0.48      0.62       166
           4       0.75      0.28      0.41       200
           5       0.81      0.48      0.60       160
           6       0.66      0.89      0.76       170
           7       0.85      0.90      0.87       147
           8       0.44      0.66      0.53       166
           9       0.45      0.86      0.59       176

    accuracy                           0.72      2006
   macro avg       0.75      0.70      0.69      2006
weighted avg       0.77      0.72      0.72      2006

[[294   1  13   0   4   2  12   1  31   1]
 [  0 254   1   0   1   0   4   0   1   3]
 [  2   0 142   3   3   4  12   1  30   1]
 [  6   0  11  79   1   3   4   6  47   9]
 [  0   4   4   0  57   1   7   5   5 117]
 [  9   1   2   5   2  77  36   2  19   7]
 [  3   6   5   0   0   2 151   0   3   0]
 [  0   1   1   0   3   0   0 132   1   9]
 [  0   1   5   1   3   6   2   0 109  39]
 [  0  10   0   0   2   0   0   9   3 152]]
```