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

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

Out[2]:

	6.0000	-1.0000	-1.0000.1	-1.0000.2	-1.0000.3	-1.0000.4	-1.0000.5	-1.0000.6	-0.6310	0.8620	..
0	5.0	-1.0	-1.0	-1.0	-0.813	-0.671	-0.809	-0.887	-0.671	-0.853	..
1	4.0	-1.0	-1.0	-1.0	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	..
2	7.0	-1.0	-1.0	-1.0	-1.000	-1.000	-0.273	0.684	0.960	0.450	..
3	3.0	-1.0	-1.0	-1.0	-1.000	-1.000	-0.928	-0.204	0.751	0.466	..
4	6.0	-1.0	-1.0	-1.0	-1.000	-1.000	-0.397	0.983	-0.535	-1.000	..

5 rows × 258 columns

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

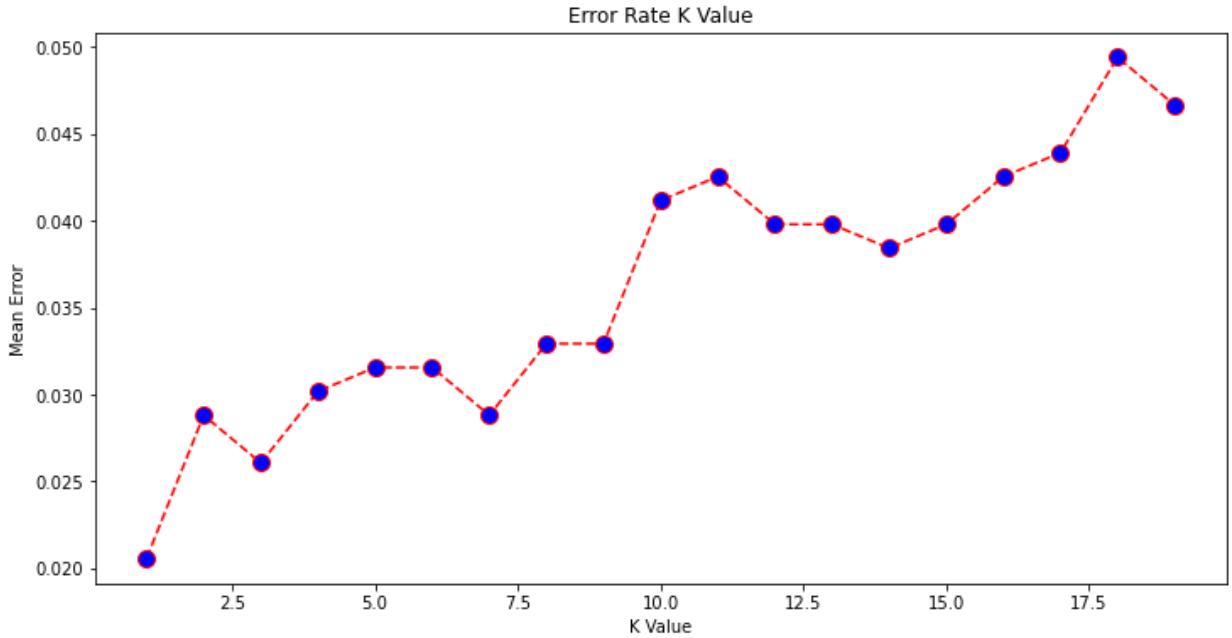
In [4]:
```
# In the section below we will use 10% of the training data as validation d
from sklearn.model_selection import train_test_split
# split dataset into training and validation data
X_train, X_validation, Y_train, Y_validation = train_test_split(
    X_train, Y_train, test_size=0.1, random_state=1, stratify=Y_train)
```

In [17]:
```
# In the section below we are training and validating KNN classifier ...
# ... for different values of K (1 to 20).
from sklearn.neighbors import KNeighborsClassifier
error = []
min_error = 1
best_knn = KNeighborsClassifier(n_neighbors=0)
# find the best classifier for k = {1:20} based on minimum mean error.
for i in range(1, 20):
    knn = KNeighborsClassifier(n_neighbors=i)
    # training the model
    knn.fit(X_train, Y_train)
    # predicting the label using test data
    y_pred = knn.predict(X_validation)
    error_i = np.mean(y_pred != Y_validation)
    error.append(error_i)
    if(error_i < min_error):
        print(i, error_i)
        min_error = error_i
        best_knn = knn
```

```
1 0.0205761316872428
```

In [18]:
```
# In the section below we are plotting to visualize mean error agaist diffe
plt.figure(figsize=(12, 6))
plt.plot(range(1, 20), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

Out[18]:
```
Text(0, 0.5, 'Mean Error')
```



In [19]:
```
# In the section below we are loading test data csv file into dataframe nam
df_test = pd.read_csv('zip.test.p.csv')
print(df_test.head())
```

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

   -0.908   0.43  0.622  -0.973  -1.137  -1.138  -1.139  -1.140  -1.141
0  -1.000 -1.000 -1.000  -1.000  -1.000  -1.000    -1.0    -1.0    -1.0
1   0.717  0.333  0.162  -0.393  -1.000  -1.000    -1.0    -1.0    -1.0
2  -1.000 -1.000 -1.000  -1.000  -1.000  -1.000    -1.0    -1.0    -1.0
3   0.901  0.901  0.290  -0.369  -0.867  -1.000    -1.0    -1.0    -1.0
4   1.000  0.988  0.187   0.139  -0.641  -0.812    -1.0    -1.0    -1.0

[5 rows x 257 columns]
```

In [20]:
```
# In the section below we are separating attributes and labels for test dat
X_test = df_test.iloc[:, 1:256].values
Y_test = df_test.iloc[:, 0].values
```

In [21]:
```
# let's predict the labels using the best model
Y_pred = best_knn.predict(X_test)
```

In [22]:
```
# 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

           5       0.93      0.91      0.92       180
           6       0.96      0.96      0.96       170
           7       0.92      0.94      0.93       147
           8       0.95      0.89      0.92       166
           9       0.89      0.95      0.92       176

    accuracy                           0.94      2006
   macro avg       0.94      0.94      0.94      2006
weighted avg       0.94      0.94      0.94      2006

[[355   0   2   0   0   0   0   1   0   1]
 [  0 255   0   0   6   0   2   1   0   0]
 [  6   1 183   2   1   0   0   2   3   0]
 [  3   0   2 153   0   6   0   0   0   2]
 [  0   3   1   0 179   1   2   3   1  10]
 [  2   0   2   4   0 145   2   0   3   2]
 [  0   0   1   0   2   3 164   0   0   0]
 [  0   1   1   1   4   0   0 138   0   2]
 [  5   0   2   5   1   1   0   1 148   3]
 [  0   0   1   0   2   0   0   4   1 168]]
```