In [ ]:

```
# Name: Meet hiteshkuamr Trivedi
# Student Id: N01520331
```

Lab8- K Nearest Neighbour

Import Libraries

In [1]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

Get the Data(Iris dataset)

In [3]:

```
df = pd.read_csv('E:\Programming\Humber college\Humber Sem 2\Data Analytics\Week-11/iris-KN
```

In [4]:

```
df.head()
```

Out[4]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Standardize the Variables

In [7]:

```
from sklearn.preprocessing import StandardScaler
```

In [11]:

```
scaler = StandardScaler()
scaler.fit(df.drop('variety',axis=1))
scaled_features = scaler.transform(df.drop('variety',axis=1))
df_feat = pd.DataFrame(scaled_features,columns=df.columns[:-1])
df_feat.head()
```

Out[11]:

	sepal.length	sepal.width	petal.length	petal.width
0	-0.581066	0.841837	-1.012978	-1.042111
1	-0.894309	-0.207835	-1.012978	-1.042111
2	-1.207552	0.212034	-1.082312	-1.042111
3	-1.364174	0.002099	-0.943643	-1.042111
4	-0.737687	1.051772	-1.012978	-1.042111

Train Test Split

In [13]:

```
from sklearn.model_selection import train_test_split
```

In [14]:

```
X_train, X_test, y_train, y_test = train_test_split(scaled_features,df['variety'],
                                                    test_size=0.30)
```

Using KNN

In [16]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [17]:

```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train,y_train)
```

Out[17]:

```
KNeighborsClassifier(n_neighbors=1)
```

In [18]:

```
pred = knn.predict(X_test)
```

Predictions and Evaluations

In [20]:

```
from sklearn.metrics import classification_report,confusion_matrix
```

In [21]:

```
print(confusion_matrix(y_test,pred))
```

```
[[12  4]
 [ 4 10]]
```

In [22]:

```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.75      0.75        16
           1       0.71      0.71      0.71        14

    accuracy                           0.73        30
   macro avg       0.73      0.73      0.73        30
weighted avg       0.73      0.73      0.73        30
```

Choosing a K Value

In [26]:

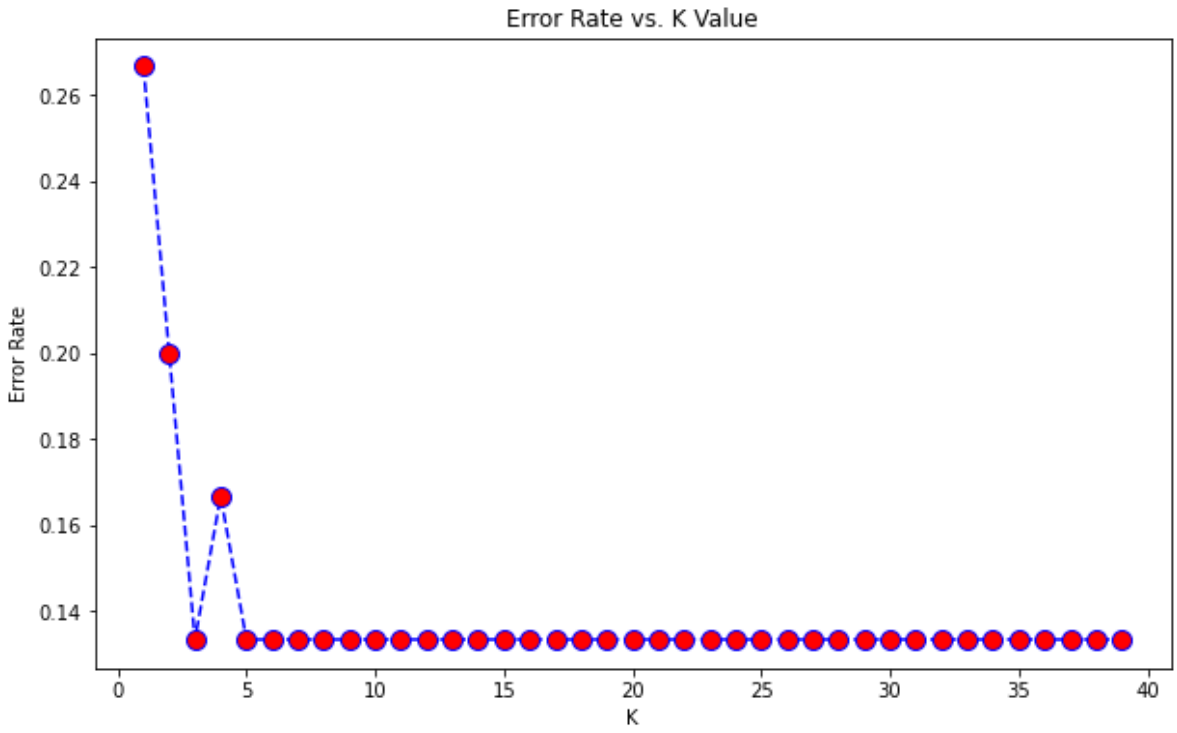
```
error_rate = []

# Will take some time
for i in range(1,40):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

In [24]:

```
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

Out[24]:

```
Text(0, 0.5, 'Error Rate')
```



In [25]:

```
# FIRST A QUICK COMPARISON TO OUR ORIGINAL K=1
knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(X_train,y_train)
pred = knn.predict(X_test)

print('WITH K=1')
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print(classification_report(y_test,pred))
```

```
WITH K=1


[[12  4]
 [ 4 10]]


              precision    recall  f1-score   support

           0       0.75      0.75      0.75        16
           1       0.71      0.71      0.71        14

    accuracy                           0.73        30
   macro avg       0.73      0.73      0.73        30
weighted avg       0.73      0.73      0.73        30
```

In [ ]:

```

```