Lab9- K Nearest Neighbour

In [ ]:
```
Harini Rajarathinam
N01419700
```

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 [2]:
```
df = pd.read_csv(r'C:\Users\rcher\Documents\Humber work\Semester 2\Intro to Data
df.head()
```

Out[2]:

	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

In [56]:

Out[56]:

	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 [4]:
```
from sklearn.preprocessing import StandardScaler
```

In [5]:
```
scaler = StandardScaler()
```

In [6]:
```
scaler.fit(df.drop('variety',axis=1))
```

Out[6]:
```
StandardScaler()
```

In [7]:
```
scaled_features = scaler.transform(df.drop('variety',axis=1))
scaled_features
```

Out[7]:
```
array([[-5.81065904e-01,  8.41837140e-01, -1.01297765e+00,
        -1.04211089e+00],
       [-8.94308978e-01, -2.07835104e-01, -1.01297765e+00,
        -1.04211089e+00],
       [-1.20755205e+00,  2.12033793e-01, -1.08231219e+00,
        -1.04211089e+00],
       [-1.36417359e+00,  2.09934449e-03, -9.43643106e-01,
        -1.04211089e+00],
       [-7.37687441e-01,  1.05177159e+00, -1.01297765e+00,
        -1.04211089e+00],
       [-1.11201292e-01,  1.68157493e+00, -8.04974023e-01,
        -6.86441647e-01],
       [-1.36417359e+00,  6.31902691e-01, -1.01297765e+00,
        -8.64276271e-01],
       [-7.37687441e-01,  6.31902691e-01, -9.43643106e-01,
        -1.04211089e+00],
       [-1.67741667e+00, -4.17769553e-01, -1.01297765e+00,
        -1.04211089e+00],
       [-8.94308978e-01,  2.09934449e-03, -9.43643106e-01,
```

In [8]:
```
df_feat = pd.DataFrame(scaled_features,columns=df.columns[:-1])
df_feat.head()
```

Out[8]:

	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 [25]:
```
from sklearn.model_selection import train_test_split
```

In [29]:
```
X_train, X_test, y_train, y_test = train_test_split(scaled_features, df['variety'
 random_state=50)
```

Using KNN

In [31]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [32]:
```
knn = KNeighborsClassifier(n_neighbors=1)
```

In [33]:
```
knn.fit(X_train,y_train)
```

Out[33]:
```
KNeighborsClassifier(n_neighbors=1)
```

In [34]:
```
pred = knn.predict(X_test)
```

Predictions and Evaluations

In [35]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [36]:
```
print(confusion_matrix(y_test,pred))
```

```
[[18  6]
 [ 5 21]]
```

In [37]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.75      0.77        24
           1       0.78      0.81      0.79        26

    accuracy                           0.78        50
   macro avg       0.78      0.78      0.78        50
weighted avg       0.78      0.78      0.78        50
```

Choosing a K Value

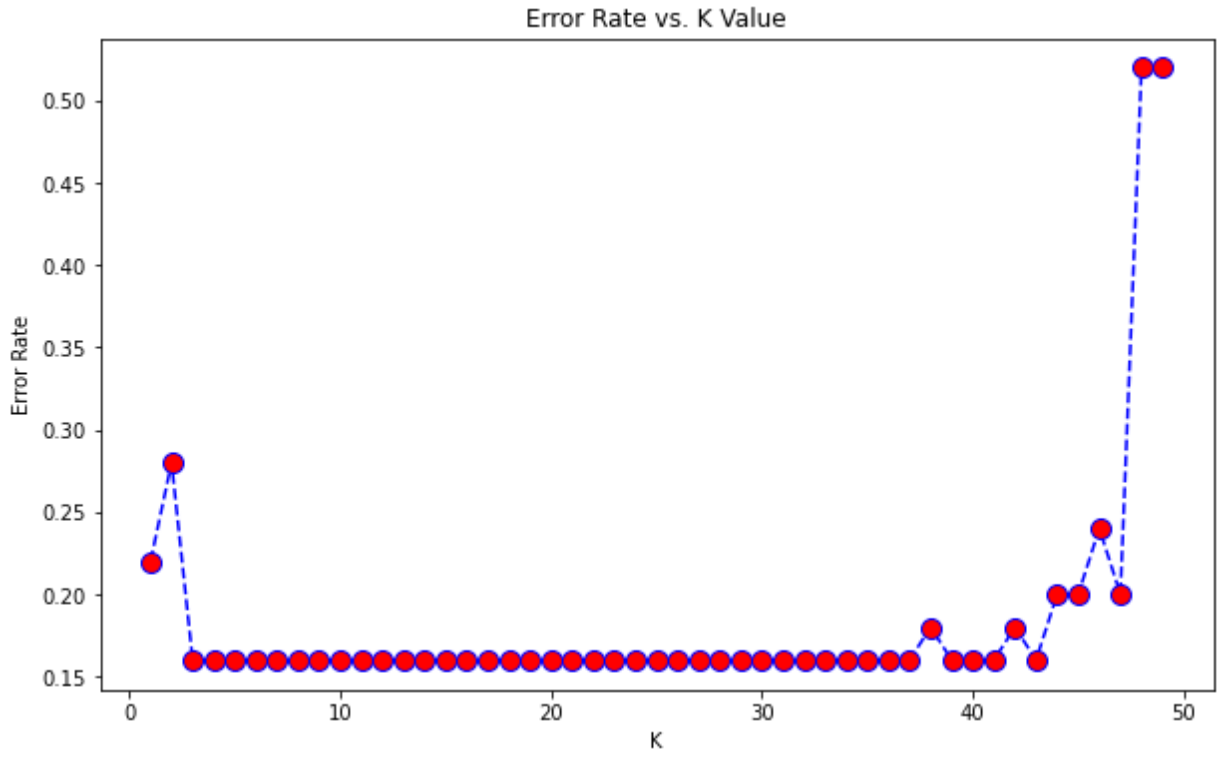
In [38]:
```
error_rate = []
for i in range (1,50):
 knn = KNeighborsClassifier(n_neighbors = i)
 knn.fit(X_train,y_train)
 pred_i = knn.predict(X_test)
 error_rate.append(np.mean(pred_i != y_test))
```

In [21]:
```
error_rate
```

Out[21]:
```
[]
```

In [39]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1,50),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[39]:
```
Text(0, 0.5, 'Error Rate')
```



In [40]:
```
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


[[18  6]
 [ 5 21]]


              precision    recall  f1-score   support

           0       0.78      0.75      0.77        24
           1       0.78      0.81      0.79        26

    accuracy                           0.78        50
   macro avg       0.78      0.78      0.78        50
weighted avg       0.78      0.78      0.78        50
```

In [41]:
```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,y_train)
pred = knn.predict(X_test)
print('WITH K=3')
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print(classification_report(y_test,pred))
```

```
WITH K=3


[[20  4]
 [ 4 22]]


              precision    recall  f1-score   support

           0       0.83      0.83      0.83        24
           1       0.85      0.85      0.85        26

    accuracy                           0.84        50
   macro avg       0.84      0.84      0.84        50
weighted avg       0.84      0.84      0.84        50
```

In [ ]:
```

```