In [51]:
```
%cd C:\Users\manoj\Downloads
```

```
C:\Users\manoj\Downloads
```

In [52]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [53]:
```
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
```

In [54]:
```
df=pd.read_csv('iris.data',names=names)
```

In [55]:
```
df.head()
```

Out[55]:

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [56]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, 4].values
```

In [57]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

In [58]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [59]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
#classifer is name you can change
```

Out[59]:
```
KNeighborsClassifier()
```

In [60]:
```
y_pred = classifier.predict(X_test)
```

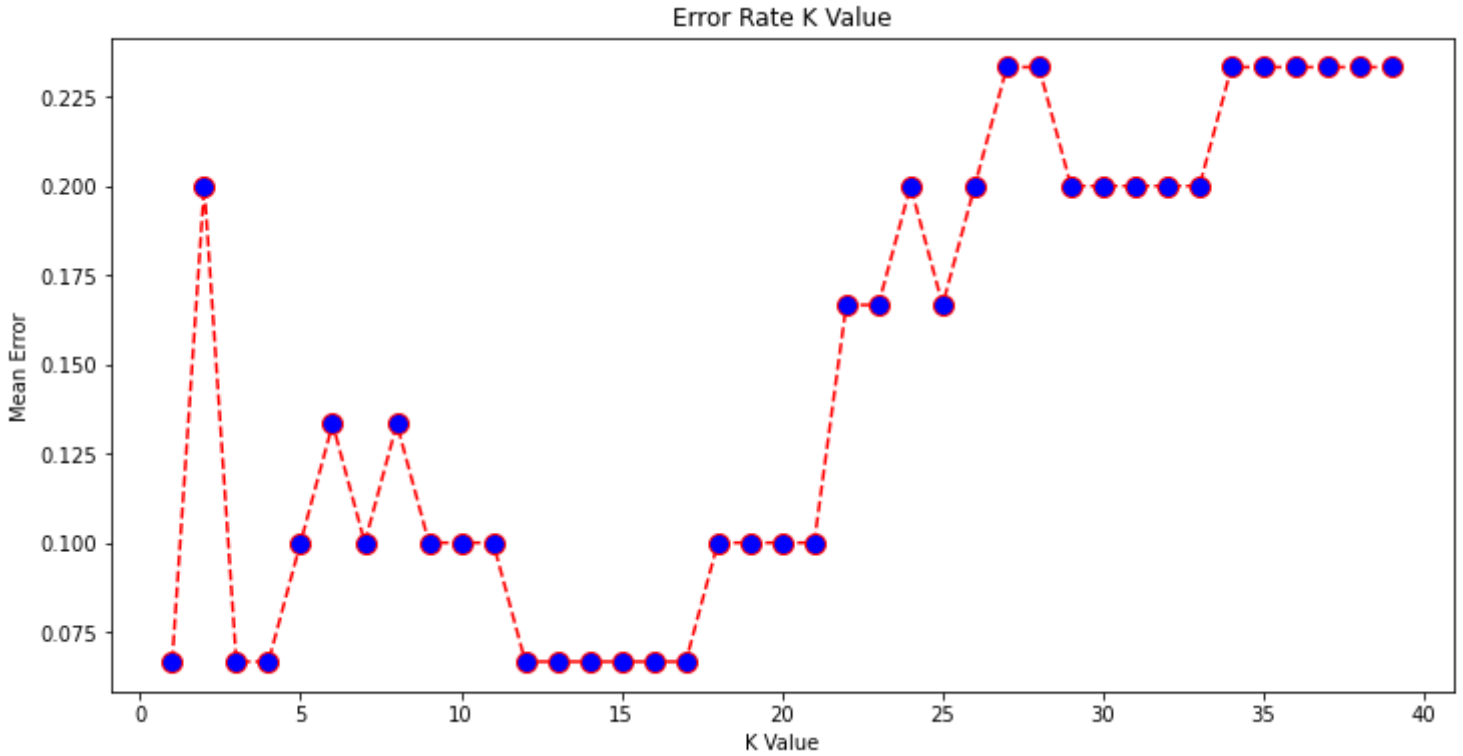
In [61]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[11  0  0]
 [ 0  6  1]
 [ 0  2 10]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       0.75      0.86      0.80         7
 Iris-virginica       0.91      0.83      0.87        12

       accuracy                           0.90        30
      macro avg       0.89      0.90      0.89        30
   weighted avg       0.91      0.90      0.90        30
```

In [62]:
```
error = []

# Calculating error for K values between 1 and 40
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error.append(np.mean(pred_i != y_test))
```

In [63]:
```
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), 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[63]: Text(0, 0.5, 'Mean Error')



In [ ]: