```
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
print(iris.data.shape)
print(iris.target.shape)
```

```
(150, 4)
(150,)
```

```
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df['target'] = iris.target
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
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

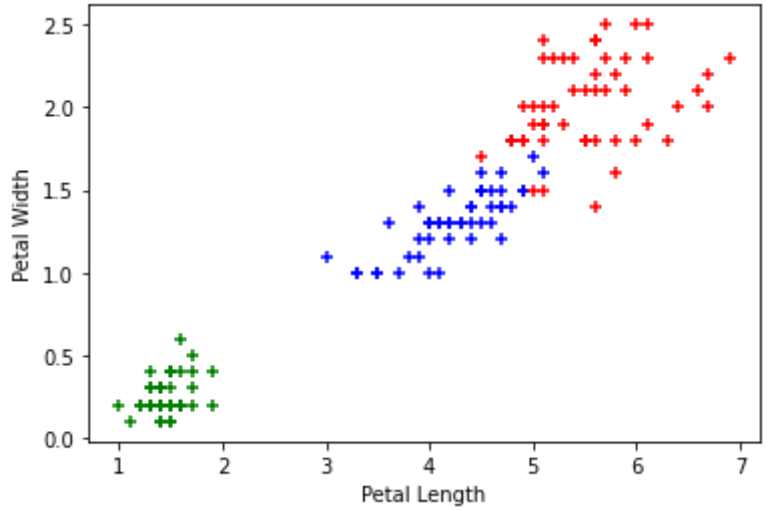
```
df['flower_name'] =df.target.apply(lambda x: iris.target_names[x])
df[45:55]
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
45	4.8	3.0	1.4	0.3	0	setosa
46	5.1	3.8	1.6	0.2	0	setosa
47	4.6	3.2	1.4	0.2	0	setosa
48	5.3	3.7	1.5	0.2	0	setosa
49	5.0	3.3	1.4	0.2	0	setosa
50	7.0	3.2	4.7	1.4	1	versicolor
51	6.4	3.2	4.5	1.5	1	versicolor
52	6.9	3.1	4.9	1.5	1	versicolor
53	5.5	2.3	4.0	1.3	1	versicolor
54	6.5	2.8	4.6	1.5	1	versicolor

```
df0 = df[:50]
df1 = df[50:100]
df2 = df[100:]

import matplotlib.pyplot as plt
```

```
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",marker='+')
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",marker='+')
plt.scatter(df2['petal length (cm)'], df2['petal width (cm)'],color="red",marker='+')
```

```
<matplotlib.collections.PathCollection at 0x7f5aed8b5c50>
```




```
X=iris.data
y=iris.target
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X=scaler.fit_transform(X)
```

```
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,random_state = 1)
```

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=33)
model.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=33)
```

```
y_pred = model.predict(X_test)
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

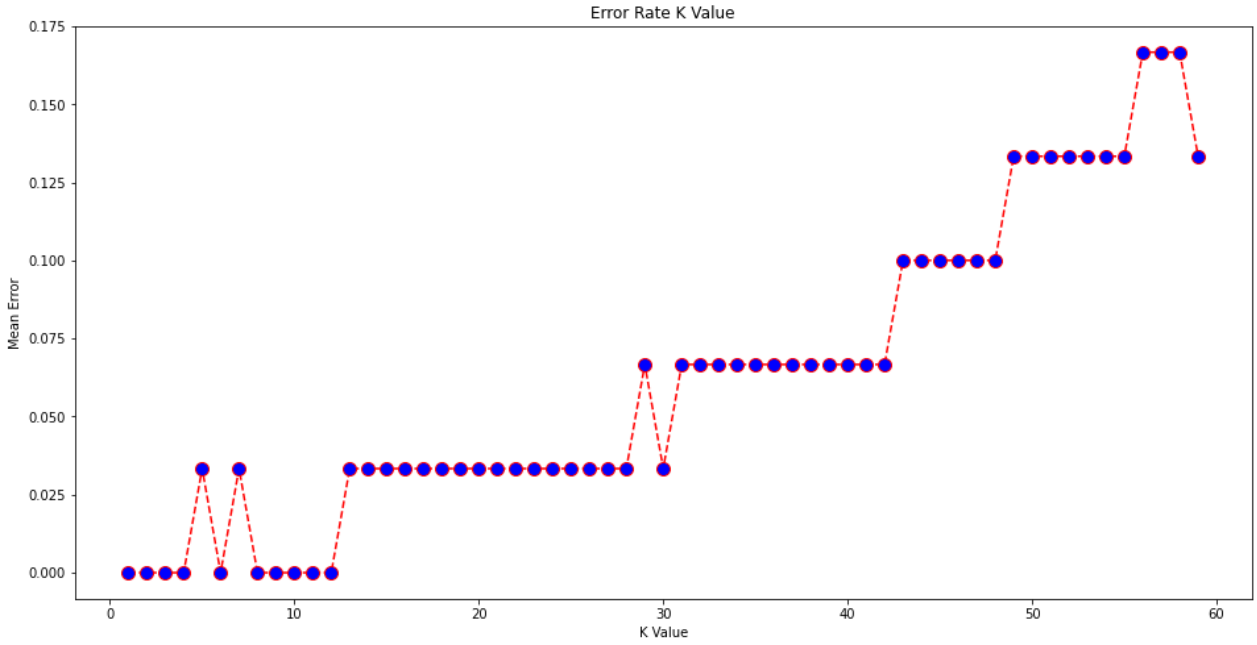
```
[[11  0  0]
 [ 0 11  2]
 [ 0  0  6]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.85      0.92        13
```

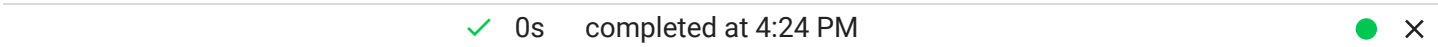
```
                2       0.75      1.00      0.86         6

         accuracy                           0.93        30
        macro avg       0.92      0.95      0.92        30
     weighted avg       0.95      0.93      0.94        30
```

```
import numpy as np
error = []

# Calculating error for K values between 1 and 60
for i in range(1, 60):
    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))
```

```
plt.figure(figsize=(16, 8))
plt.plot(range(1, 60), error, color='red', linestyle='dashed', marker='o', markerfacecolor
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

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




0s
completed at 4:24 PM