```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

```
data= pd.read_csv('iris.csv')
data.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
data.describe
```

```
<bound method NDFrame.describe of      sepal_length  sepal_width  petal_length  petal_w
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]>
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
#Define What X and y are in our Dataset
y = data['species']
X = data.drop('species', axis = 1)
print(X.head())
print(y.head())
```

```
   sepal_length  sepal_width  petal_length  petal_width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
3           4.6          3.1           1.5          0.2
4           5.0          3.6           1.4          0.2
0    setosa
1    setosa
2    setosa
3    setosa
4    setosa
Name: species, dtype: object
```

```
# Now, use the function of overall Data, For Training and Test, 20% for testing and 80% for T

X_train, X_test, y_train, y_test  = train_test_split(X, y, test_size = 0.2, random_state = 42

"""
# We will use a Standard Scaler, to scale the features for Preprocessing
# Scaling of features in data science means,
# Feature Scaling is a technique to standardize the independent features present
#in the data in a fixed range. It is performed during the data pre-processing to handle highl
#varying magnitudes or values or units.
"""
scaler = StandardScaler()
scale = scaler.fit(X_train)
X_train  = scale.transform(X_train)
X_test = scale.transform(X_test)


# Creating a model logistic regression
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)
pred = model.predict(X_test)
```

```
score = accuracy_score(y_test, pred)
score
```

```
     1.0
```

```
# Creating a Confusion Matrix

confusion_matrix(y_test, pred)
```

```
     array([[10,  0,  0],
            [ 0,  9,  0],
            [ 0,  0, 11]])
```

```
# Confusion Martix metrics, Build Classification Report
matrix = classification_report(y_test, pred)
print('classification report: \n', matrix)
```

```
     classification report: 
                   precision    recall  f1-score   support

           setosa       1.00      1.00      1.00        10
       versicolor       1.00      1.00      1.00         9
        virginica       1.00      1.00      1.00        11

         accuracy                           1.00        30
        macro avg       1.00      1.00      1.00        30
     weighted avg       1.00      1.00      1.00        30
```

✓ 0s completed at 2:20 PM ● ×