```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
from sklearn.datasets import load_iris
irisdataset = load_iris()
```

```
irisdataset.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
irisdataset.feature_names
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

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

```
df
```

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   target             150 non-null    int64  
 5   flower_name        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
df.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000

```
df['flower_name'].unique()
```

```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

75%	6.400000	3.300000	5.100000	1.800000	2.000000

```
df[df['flower_name']=="setosa"].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

```
df[df['flower_name']=="virginica"].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
100	6.3	3.3	6.0	2.5	2	virginica
101	5.8	2.7	5.1	1.9	2	virginica
102	7.1	3.0	5.9	2.1	2	virginica
103	6.3	2.9	5.6	1.8	2	virginica
104	6.5	3.0	5.8	2.2	2	virginica

```
df[df['flower_name']=="versicolor"].head()
```

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

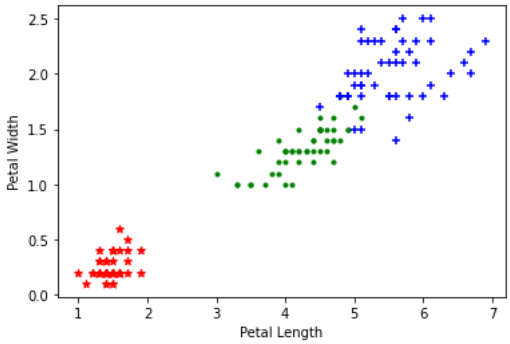
```
df0 = df[:50] #setosa
df1 = df[50:100] #versicolor
df2 = df[100:] #virginica

#SETOSA VS versicolor VS virginica IN SEPAL LENGTH AND SEPAL WIDTH
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="red",marker='*')
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="green",marker='.')
plt.scatter(df2['sepal length (cm)'], df2['sepal width (cm)'],color="blue",marker='+')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
```

```
Text(0, 0.5, 'Sepal Width')
```

```
#SETOSA VS versicolor VS virginica IN PETAL LENGTH AND PETAL WIDTH
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="red",marker='*')
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="green",marker='.')
plt.scatter(df2['petal length (cm)'], df2['petal width (cm)'],color="blue",marker='+')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
```

```
Text(0, 0.5, 'Petal Width')
```




```
x = df.drop(['target','flower_name'], axis='columns')
y = df.target
```

```
x
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
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
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

```
y
```

```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: target, Length: 150, dtype: int64
```

```
#train test split
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.3)
```

```
#KNN classifier
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=6)
knn.fit(xtrain,ytrain)
```

```
KNeighborsClassifier(n_neighbors=6)
```

```
#Accuracy
knn.score(xtest,ytest)
```

```
1.0
```

```
ypred = knn.predict(xtest)
ypred
```

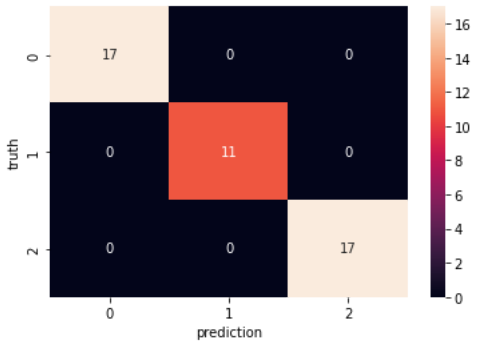
```
array([2, 0, 1, 0, 0, 2, 0, 0, 1, 2, 2, 2, 2, 0, 1, 1, 1, 2, 0, 0, 0, 2,
       0, 2, 2, 1, 0, 1, 1, 2, 2, 0, 0, 0, 2, 0, 0, 0, 1, 2, 1, 2, 1, 2,
       2])
```

```
from sklearn.metrics import confusion_matrix
ypred = knn.predict(xtest)
cm = confusion_matrix(ytest,ypred)
cm
```

```
array([[17,  0,  0],
       [ 0, 11,  0],
       [ 0,  0, 17]])
```

```
sns.heatmap(cm,annot=True)
plt.xlabel("prediction")
plt.ylabel("truth")
```

```
Text(33.0, 0.5, 'truth')
```




```
from sklearn.metrics import classification_report
cr = classification_report(ytest,ypred)
print(cr)
```

```
                  precision    recall  f1-score   support

               0       1.00      1.00      1.00        17
               1       1.00      1.00      1.00        11
               2       1.00      1.00      1.00        17

        accuracy                           1.00        45
       macro avg       1.00      1.00      1.00        45
    weighted avg       1.00      1.00      1.00        45
```