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

```
import pandas as pd
df=pd.DataFrame(data.data,columns=data.feature_names)
df["target"]=data.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

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
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
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000				
max	7.900000	4.400000	6.900000	2.500000	2.000000

What can I help you build?

```
df.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 (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
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=df.drop("target",axis=1)
y=df["target"]
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=25)
x_train_scaled=sc.fit_transform(x_train)
x_test_scaled=sc.transform(x_test)
```

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

```
knn.fit(x_train,y_train)
```

```
y_pred=knn.predict(x_test)
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       1.00      0.85      0.92        13
           2       0.80      1.00      0.89         8

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

```
result=[[4.9,3.0,1.4,0.2]]
knn.predict(result)
```

```
array([0])
```