

```
import seaborn as sns
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
import pickle
```

```
print(sns.get_dataset_names())
```

```
['anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'diamonds', 'dots', 'dowjones']
```

```
df = sns.load_dataset('iris').head()
df
```

	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

Next steps:

[Generate code with df](#)
[View recommended plots](#)
[New interactive sheet](#)

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   sepal_length    5 non-null     float64
1   sepal_width     5 non-null     float64
2   petal_length    5 non-null     float64
3   petal_width     5 non-null     float64
4   species         5 non-null     object
dtypes: float64(4), object(1)
memory usage: 328.0+ bytes
```

```
df = df.dropna()
```

```
x = df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y = df['species']
```

```
y = LabelEncoder().fit_transform(y)
```

```
train_test_split(x, y, test_size=0.3, random_state=42)
```

```

↗ [   sepal_length  sepal_width  petal_length  petal_width
    2           4.7           3.2           1.3           0.2
    0           5.1           3.5           1.4           0.2
    3           4.6           3.1           1.5           0.2,
      sepal_length  sepal_width  petal_length  petal_width
    1           4.9           3.0           1.4           0.2
    4           5.0           3.6           1.4           0.2,
  array([0, 0, 0]),
  array([0, 0])]

```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

```

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

```

```

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train, y_train)

```

```

↗
  ▾ KNeighborsClassifier ⓘ ?
  KNeighborsClassifier(n_neighbors=3)

```

```
y_pred = knn.predict(x_test)
```

```
print(y_pred)
```

```
↗ [0]
```

```

accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)

```

```

↗ /usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:409: UserWarning: A single
  warnings.warn(

```

```

print("Accuracy:", accuracy)
print("Classification Report:\n", classification_rep)
print("Confusion Matrix:\n", confusion_mat)

```

```

↗ Accuracy: 1.0
  Classification Report:
              precision    recall  f1-score   support

         0           1.00      1.00      1.00         1

   accuracy                1.00         1
  macro avg           1.00      1.00      1.00         1
 weighted avg           1.00      1.00      1.00         1

```

```
Confusion Matrix:
```

[[1]]

```
with open('knn_model_iris.pkl', 'wb') as model_file:
    pickle.dump(knn, model_file)
```

```
with open('knn_model_iris.pkl', 'rb') as file:
    loaded_knn = pickle.load(file)
```

```
df.head()
```



1 to 5 of 5 entries

Filter



index	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

Show 25 per page



Like what you see? Visit the [data table notebook](#) to learn more about interactive tables.

Next steps:

[Generate code with df](#)

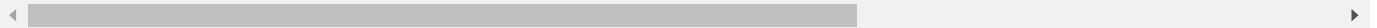
[View recommended plots](#)
[New interactive sheet](#)

```
new_instance = [[5.1, 3.5, 1.4, 0.2]]
```

```
scaled_new_instance = scaler.transform(new_instance)
```



```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names. You should pass feature names to the estimator if you want to use feature names in the model.
```



```
new_data_result = loaded_knn.predict(scaled_new_instance)
```

```
print(new_data_result)
```



[0]

Start coding or [generate](#) with AI.

