

A2: Implement a k-NN classifier using sci-kit-learn to classify the Iris dataset into different species based on sepal and petal dimensions.

```
In [1]: import numpy as np  
import pandas as pd
```

```
In [3]: # Load dataset  
from sklearn.datasets import load_iris  
iris = load_iris() # creating an instance of iris data  
  
iris
```

```
Out[3]: {'data': array([[5.1, 3.5, 1.4, 0.2],  
[4.9, 3. , 1.4, 0.2],  
[4.7, 3.2, 1.3, 0.2],  
[4.6, 3.1, 1.5, 0.2],  
[5. , 3.6, 1.4, 0.2],  
[5.4, 3.9, 1.7, 0.4],  
[4.6, 3.4, 1.4, 0.3],  
[5. , 3.4, 1.5, 0.2],  
[4.4, 2.9, 1.4, 0.2],  
[4.9, 3.1, 1.5, 0.1],  
[5.4, 3.7, 1.5, 0.2],  
[4.8, 3.4, 1.6, 0.2],  
[4.8, 3. , 1.4, 0.1],  
[4.3, 3. , 1.1, 0.1],  
[5.8, 4. , 1.2, 0.2],  
[5.7, 4.4, 1.5, 0.4],  
[5.4, 3.9, 1.3, 0.4],  
[5.1, 3.5, 1.4, 0.3],  
[5.7, 3.8, 1.7, 0.3],  
[5.1, 3.8, 1.5, 0.3],  
[5.4, 3.4, 1.7, 0.2],  
[5.1, 3.7, 1.5, 0.4],  
[4.6, 3.6, 1. , 0.2],  
[5.1, 3.3, 1.7, 0.5],  
[4.8, 3.4, 1.9, 0.2],  
[5. , 3. , 1.6, 0.2],  
[5. , 3.4, 1.6, 0.4],  
[5.2, 3.5, 1.5, 0.2],  
[5.2, 3.4, 1.4, 0.2],  
[4.7, 3.2, 1.6, 0.2],  
[4.8, 3.1, 1.6, 0.2],  
[5.4, 3.4, 1.5, 0.4],  
[5.2, 4.1, 1.5, 0.1],  
[5.5, 4.2, 1.4, 0.2],  
[4.9, 3.1, 1.5, 0.2],  
[5. , 3.2, 1.2, 0.2],  
[5.5, 3.5, 1.3, 0.2],  
[4.9, 3.6, 1.4, 0.1],  
[4.4, 3. , 1.3, 0.2],  
[5.1, 3.4, 1.5, 0.2],  
[5. , 3.5, 1.3, 0.3],  
[4.5, 2.3, 1.3, 0.3],
```

```
[4.4, 3.2, 1.3, 0.2],  
[5. , 3.5, 1.6, 0.6],  
[5.1, 3.8, 1.9, 0.4],  
[4.8, 3. , 1.4, 0.3],  
[5.1, 3.8, 1.6, 0.2],  
[4.6, 3.2, 1.4, 0.2],  
[5.3, 3.7, 1.5, 0.2],  
[5. , 3.3, 1.4, 0.2],  
[7. , 3.2, 4.7, 1.4],  
[6.4, 3.2, 4.5, 1.5],  
[6.9, 3.1, 4.9, 1.5],  
[5.5, 2.3, 4. , 1.3],  
[6.5, 2.8, 4.6, 1.5],  
[5.7, 2.8, 4.5, 1.3],  
[6.3, 3.3, 4.7, 1.6],  
[4.9, 2.4, 3.3, 1. ],  
[6.6, 2.9, 4.6, 1.3],  
[5.2, 2.7, 3.9, 1.4],  
[5. , 2. , 3.5, 1. ],  
[5.9, 3. , 4.2, 1.5],  
[6. , 2.2, 4. , 1. ],  
[6.1, 2.9, 4.7, 1.4],  
[5.6, 2.9, 3.6, 1.3],  
[6.7, 3.1, 4.4, 1.4],  
[5.6, 3. , 4.5, 1.5],  
[5.8, 2.7, 4.1, 1. ],  
[6.2, 2.2, 4.5, 1.5],  
[5.6, 2.5, 3.9, 1.1],  
[5.9, 3.2, 4.8, 1.8],  
[6.1, 2.8, 4. , 1.3],  
[6.3, 2.5, 4.9, 1.5],  
[6.1, 2.8, 4.7, 1.2],  
[6.4, 2.9, 4.3, 1.3],  
[6.6, 3. , 4.4, 1.4],  
[6.8, 2.8, 4.8, 1.4],  
[6.7, 3. , 5. , 1.7],  
[6. , 2.9, 4.5, 1.5],  
[5.7, 2.6, 3.5, 1. ],  
[5.5, 2.4, 3.8, 1.1],  
[5.5, 2.4, 3.7, 1. ],  
[5.8, 2.7, 3.9, 1.2],  
[6. , 2.7, 5.1, 1.6],
```

[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1.],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2.],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2.],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2.],
[7.7, 2.8, 6.7, 2.],
[6.3, 2.7, 4.9, 1.8],
[6.7, 3.3, 5.7, 2.1],
[7.2, 3.2, 6. , 1.8],

R.A. Fisher
Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
Date: July, 1988
The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken from Fisher's paper. Note that it's the same as in R, but not as in the UCI Machine Learning Repository, which has two wrong data points.
This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referred frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

.. dropdown:: References
 - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
 Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to Mathematical Statistics" (John Wiley, NY, 1950).
 - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
 (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.
 - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
 Structure and Classification Rule for Recognition in Partially Exposed Environments". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 1, 67-71.
 - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions on Information Theory, May 1972, 43 1-433.
 - See also: 1988 MLC Proceedings, 54-64. Cheeseman et al's AUTOCLASS II
 conceptual clustering system finds 3 classes in the data.
 - Many, many more ...

```
'feature_names': ['sepal length (cm)',  
 'sepal width (cm)',  
 'petal length (cm)',  
 'petal width (cm)'],  
 'filename': 'iris.csv',  
 'data_module': 'sklearn.datasets.data'}
```

```
In [4]: # Identify dependent and independent variables  
X = pd.DataFrame(iris.data, columns=iris.feature_names) # Independent variables  
y = iris.target # Dependent variable
```

```
In [5]: X
```

Out[5]:

	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

In [6]: y

```
In [7]: # Split data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
In [8]: X_train
```

Out[8]:

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
5.5	2.4	3.7	1.0
6.3	2.8	5.1	1.5
6.4	3.1	5.5	1.8
6.6	3.0	4.4	1.4
7.2	3.6	6.1	2.5
...
6.1	2.8	4.0	1.3
4.9	2.5	4.5	1.7
5.8	4.0	1.2	0.2
5.8	2.6	4.0	1.2
7.1	3.0	5.9	2.1

105 rows × 4 columns

```
In [9]: y_train
```

In [10]: x test

Out[10]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
73	6.1	2.8	4.7	1.2
18	5.7	3.8	1.7	0.3
118	7.7	2.6	6.9	2.3
78	6.0	2.9	4.5	1.5
76	6.8	2.8	4.8	1.4
31	5.4	3.4	1.5	0.4
64	5.6	2.9	3.6	1.3
141	6.9	3.1	5.1	2.3
68	6.2	2.2	4.5	1.5
82	5.8	2.7	3.9	1.2
110	6.5	3.2	5.1	2.0
12	4.8	3.0	1.4	0.1
36	5.5	3.5	1.3	0.2
9	4.9	3.1	1.5	0.1
19	5.1	3.8	1.5	0.3
56	6.3	3.3	4.7	1.6
104	6.5	3.0	5.8	2.2
69	5.6	2.5	3.9	1.1
55	5.7	2.8	4.5	1.3
132	6.4	2.8	5.6	2.2
29	4.7	3.2	1.6	0.2
127	6.1	3.0	4.9	1.8

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
26	5.0	3.4	1.6	0.4
128	6.4	2.8	5.6	2.1
131	7.9	3.8	6.4	2.0
145	6.7	3.0	5.2	2.3
108	6.7	2.5	5.8	1.8
143	6.8	3.2	5.9	2.3
45	4.8	3.0	1.4	0.3
30	4.8	3.1	1.6	0.2
22	4.6	3.6	1.0	0.2
15	5.7	4.4	1.5	0.4
65	6.7	3.1	4.4	1.4
11	4.8	3.4	1.6	0.2
42	4.4	3.2	1.3	0.2
146	6.3	2.5	5.0	1.9
51	6.4	3.2	4.5	1.5
27	5.2	3.5	1.5	0.2
4	5.0	3.6	1.4	0.2
32	5.2	4.1	1.5	0.1
142	5.8	2.7	5.1	1.9
85	6.0	3.4	4.5	1.6
86	6.7	3.1	4.7	1.5
16	5.4	3.9	1.3	0.4

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
-------------------	------------------	-------------------	------------------

10	5.4	3.7	1.5	0.2
----	-----	-----	-----	-----

In [11]: `y_test`Out[11]: `array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2, 0, 2, 2, 2, 2, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1, 0, 0])`In []: `# Model Building
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=9) # creating an instance of KNN
knn.fit(X_train, y_train)`Out[]: `▼ KNeighborsClassifier ⓘ ⓘ ?``► Parameters`

<code>n_neighbors</code>	9
<code>weights</code>	'uniform'
<code>algorithm</code>	'auto'
<code>leaf_size</code>	30
<code>p</code>	2
<code>metric</code>	'minkowski'
<code>metric_params</code>	None
<code>n_jobs</code>	None

In [14]: `#Model usage``y_pred1=knn.predict(X_test)
y_pred1`

```
Out[14]: array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2,  
    0, 2, 2, 2, 2, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1, 0,  
    0])
```

```
In [15]: #Evaluate the model  
from sklearn.metrics import classification_report, confusion_matrix  
cm=confusion_matrix(y_test,y_pred1)  
print("Confusion Matrix:\n", cm)  
print("classification report", classification_report(y_test, y_pred1))
```

Confusion Matrix:

```
[[19  0  0]  
 [ 0 13  0]  
 [ 0  0 13]]
```

		precision	recall	f1-score	support
	0	1.00	1.00	1.00	19
	1	1.00	1.00	1.00	13
	2	1.00	1.00	1.00	13
accuracy			1.00	1.00	45
macro avg	1.00	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	1.00	45

```
In [ ]:
```