

- Project: Iris Flower classification

Classify Iris flowers (*Setosa*, *Versicolor*, *Virginica*) using sepal and petal measurements with a machine learning model built in Scikit-learn.

```
# Import necessary libraries
import pandas as pd

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
```

```
# Load iris data
iris = load_iris()
iris
```

```
[6.4, 2.8, 5.6, 2.1],
[7.2, 3. , 5.8, 1.6],
[7.4, 2.8, 6.1, 1.9],
[7.9, 3.8, 6.4, 2. ],
[6.4, 2.8, 5.6, 2.2],
[6.3, 2.8, 5.1, 1.5],
[6.1, 2.6, 5.6, 1.4],
[7.7, 3. , 6.1, 2.3],
[6.3, 3.4, 5.6, 2.4],
[6.4, 3.1, 5.5, 1.8],
[6. , 3. , 4.8, 1.8],
[6.9, 3.1, 5.4, 2.1],
[6.7, 3.1, 5.6, 2.4],
[6.9, 3.1, 5.1, 2.3],
[5.8, 2.7, 5.1, 1.9],
[6.8, 3.2, 5.9, 2.3],
[6.7, 3.3, 5.7, 2.5],
[6.7, 3. , 5.2, 2.3],
[6.3, 2.5, 5. , 1.9],
[6.5, 3. , 5.2, 2. ],
[6.2, 3.4, 5.4, 2.3],
[5.9, 3. , 5.1, 1.8]]),
'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
'frame': None,
'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
'DESCR': '.. _iris-dataset:\n\nIris plants dataset\n-----\n\n**Data Set Characteristics:**\n\nNumber of Instances:
150 (50 in each of three classes)\nNumber of Attributes: 4 numeric, predictive attributes and the class\nAttribute Information:\n
- sepal length in cm\n      - sepal width in cm\n      - petal length in cm\n      - petal width in cm\n      - class:\n      - Iris-Setosa\n      - Iris-Versicolour\n      - Iris-Virginica\n\nSummary Statistics:\n\n=====
=====\n\nMin Max Mean SD Class Correlation\n\n=====
=====\nsepal length:  4.3 7.9 5.84 0.83 0.7826\nsepal width:   2.0 4.4 3.05 0.43 -0.4194\npetal
length:   1.0 6.9 3.76 1.76 0.9490 (high!)\npetal width:    0.1 2.5 1.20 0.76 0.9565 (high!)\n\n=====
=====\n\nMissing Attribute Values: None\nClass Distribution: 33.3% for each of 3
classes.\nCreator: R.A. Fisher\nDonor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\nDate: July, 1988\n\nThe famous Iris
database, first used by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s paper. Note that it\'s the same as in R, but not as in
the UCI\nMachine Learning Repository, which has two wrong data points.\nThis is perhaps the best known database to be found in
the\npattern recognition literature. Fisher\'s paper is a classic in the field and\nis referenced frequently to this day. (See Duda
& Hart, for example.) The\ndata set contains 3 classes of 50 instances each, where each class refers to a\ntype of iris plant. One
class is linearly separable from the other 2; the\nlatter are NOT linearly separable from each other.\n\n.. dropdown:: References\n
- Fisher, R.A. "The use of multiple measurements in taxonomic problems"\n  Annual Eugenics, 7, Part II, 179-188 (1936); also in
"Contributions to\n  Mathematical Statistics" (John Wiley, NY, 1950).\n
- Duda, R.O., & Hart, P.E. (1973) Pattern Classification
and Scene Analysis.\n  (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.\n
- Dasarathy, B.V. (1980) "Nosing Around
the Neighborhood: A New System\n  Structure and Classification Rule for Recognition in Partially Exposed\n  Environments". IEEE
Transactions on Pattern Analysis and Machine\n  Intelligence, Vol. PAMI-2, No. 1, 67-71.\n
- Gates, G.W. (1972) "The Reduced
Nearest Neighbor Rule". IEEE Transactions\n  on Information Theory, May 1972, 431-433.\n
- See also: 1988 MLC Proceedings, 54-64.
Cheeseman et al\'s AUTOCLASS II\n  conceptual clustering system finds 3 classes in the data.\n
- Many, many more ...'\n',
'feature_names': ['sepal length (cm)',
'sepal width (cm)',
'petal length (cm)',
'petal width (cm)'],
'filename': 'iris.csv',
'data_module': 'sklearn.datasets.data}
```

```
# create A dataframe from this data
df = pd.DataFrame(data=iris.data,columns=iris.feature_names)
df['species'] = iris.target # Add the species column (target)
```

```
# Display the first few rows of the dataset
print("Iris dataset: ")
print(df.head())
```

↩

Iris dataset:




	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	

	species
0	0
1	0
2	0
3	0
4	0

```
# Slipt the dataset features into X and target (Y)
x = df.drop('species', axis = 1) # Features (sepal_length,sepal_width,prtal_length,petal_width)
y = df['species'] # Target (species: 0 = Setosa, 1 = versicolor, 2 = Virginica)
```

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


Next steps:

Generate code with x

 View recommended plots

New interactive sheet

y




species	
0	0
1	0
2	0
3	0
4	0
...	...
145	2
146	2
147	2
148	2
149	2

150 rows × 1 columns

dtype: int64

```
# Split the data into test and training sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

```
# Train a DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(x_train,y_train)
```




▼ DecisionTreeClassifier ⓘ ?

DecisionTreeClassifier()

```
# Make predicitons on the test set
y_pred = classifier.predict(x_test)
```

```
# Evaluate the model using accuracy, confusion matrix, and classification report
accuracy = metrics.accuracy_score(y_test, y_pred)
conf_matrix = metrics.confusion_matrix(y_test, y_pred)
class_report = metrics.classification_report(y_test, y_pred)
```

```
print(f'\nAccuracy: {accuracy * 100:.2f}%')
print('\nConfusion matrix: ')
print(conf_matrix)
print('\nClassification report: ')
print(class_report)
```



Accuracy: 100.00%

Confusion matrix:
[[19 0 0]
[0 13 0]
[0 0 13]]

Classification report:

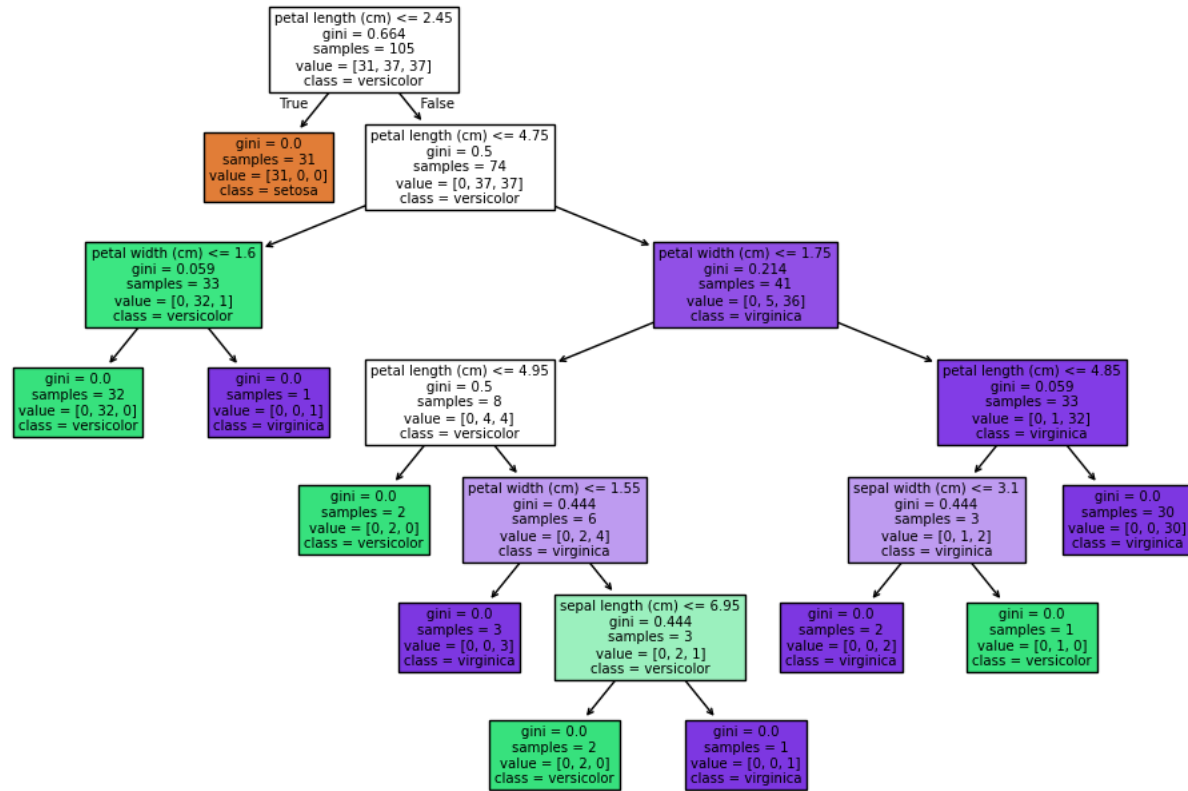
	precision	recall	f1-score	support
0	1.00	1.00	1.00	19
1	1.00	1.00	1.00	13
2	1.00	1.00	1.00	13
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

```
# Visualize the decisin tree
plt.figure(figsize=(12,8))
plot_tree(classifier, feature_names = iris.feature_names, class_names = iris.target_names, filled = True )
plt.title("Decision Tree For Iris Flower Classification")
```

plt.show()



Decision Tree For Iris Flower Classification



Results Summary

- Decision Tree Classification
- Test Accuracy: 100.0%
- Key observations: e.g., Setosa is perfectly separable; Versicolor/Virginica have some overlap.

Double-click (or enter) to edit