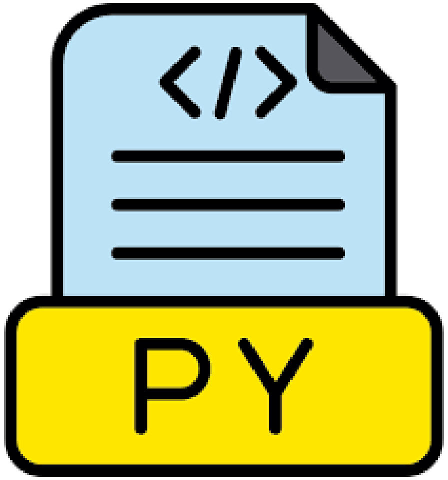
Fouille de Données

Data Mining

Classification - Partie 1

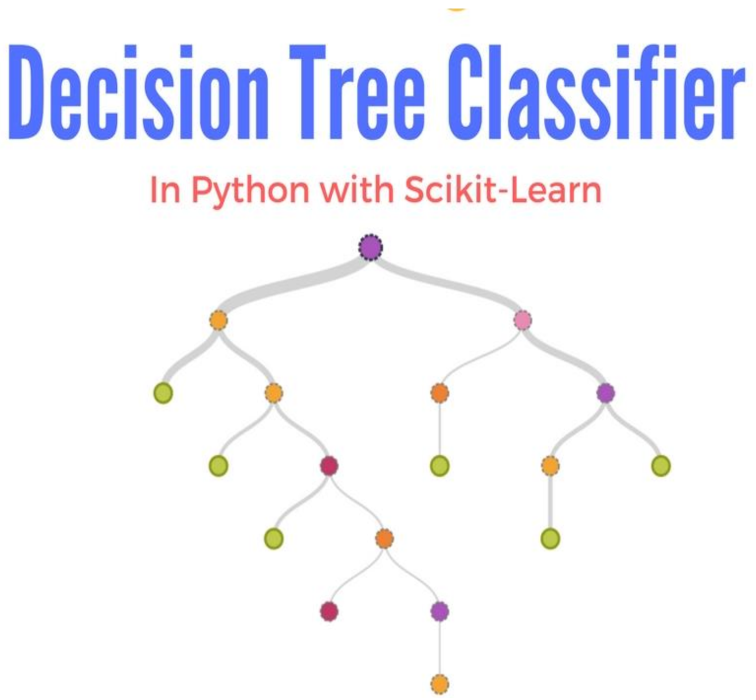
La classification avec les arbres de décision

- **Série TP 3 – Decision Trees with Scikit Learn**



La classification avec les arbres de décision

- **Série TP 3 – Decision Trees with Scikit Learn**

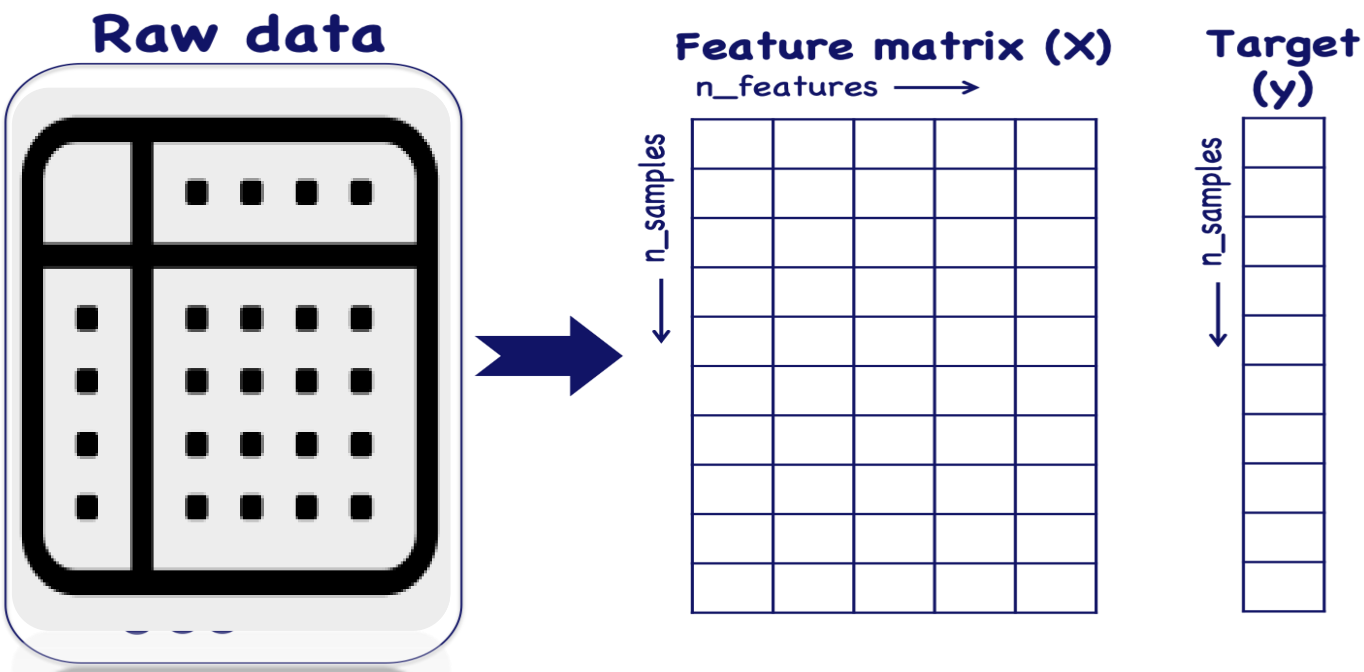


https://scikit-learn.org/stable/modules/tree.html

La classification avec les arbres de décision

- **sklearn.tree.DecisionTreeClassifier** is a class capable of performing multi-class classification on a dataset.

- Takes as **input** two arrays: an **array X** of shape (n_samples, n_features) holding the **training samples**, and an **array Y** of integer values, shape (n_samples,), holding the **class labels** for the training samples.



La classification avec les arbres de décision

- scikit-learn (sklearn) uses an optimised version of the **CART algorithm:** The Classification and Regression Tree;

- **Gini index (default)** or **Information gain** are metrics to measure the quality of a split for classification tasks in CART.

- scikit-learn implementation **does not support categorical variables** for now. => Transform categorical to numerical (**Encoding**).

- CART constructs **binary trees**, meaning each internal node has exactly two child nodes (left and right).

- Scikit-learn relies on **pre-pruning** (early stopping) through hyperparameters like: max_depth, min_samples_split, min_samples_leaf, and max_features.

La classification avec les arbres de décision

1. Import necessary modules
2. Load & explore the dataset
3. Split the DataFrame into features (X) and target/class (y)
4. Create training and test sets
5. Encode categorical data as numbers : OneHotEncoding
6. Train the decision tree
7. Plot the decision tree
8. Predict and Evaluate : Accuracy & Confusion matrix
9. Tree Pruning

La classification avec les arbres de décision

Import necessary modules : scikit-learn package

In [3]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder
from sklearn import tree
```

Load & explore the dataset : playing tennis

In [43]:

```
df = pd.read_csv('tennis.csv')
```

La classification avec les arbres de décision

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes
10	sunny	mild	normal	True	yes

La classification avec les arbres de décision

```
df.describe()
```

Out[20]:

	outlook	temp	humidity	windy	play
count	14	14	14	14	14
unique	3	3	2	2	2
top	sunny	mild	high	False	yes
freq	5	6	7	8	9

In [84]:

```
df['temp'].value_counts()
```

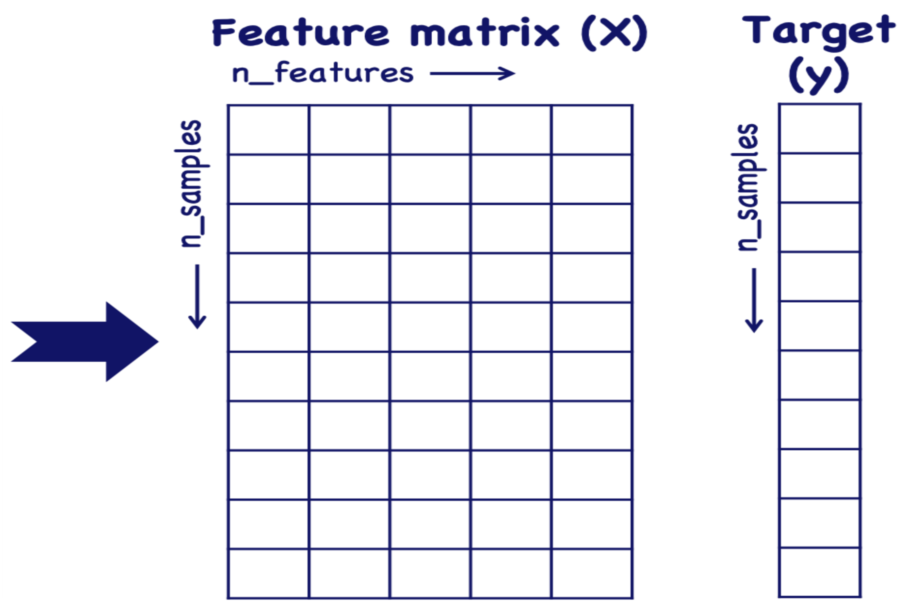
Out[84]:

```
mild     6
cool     4
hot      4
```

La classification avec les arbres de décision

Split the DataFrame into features (X) and target/class (y)

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes
10	sunny	mild	normal	True	yes



La classification avec les arbres de décision

Split the DataFrame into features (X) and target/class (y)

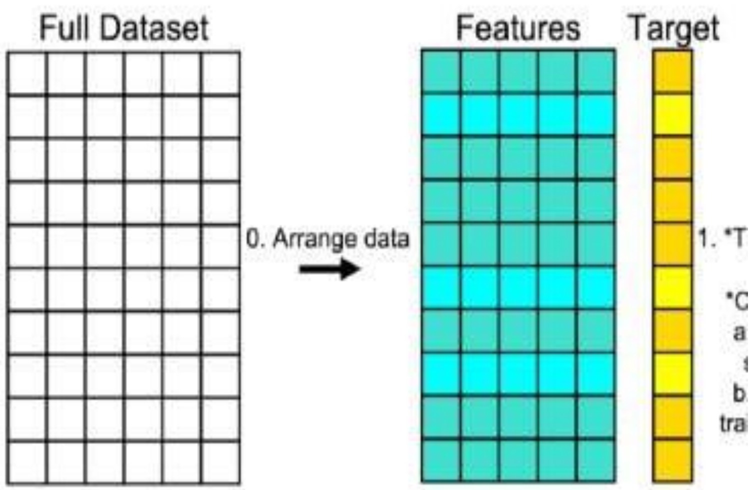
In [7]:

```
X = df[['outlook', 'temp', 'humidity', 'windy']]
y = df[['play']]
```

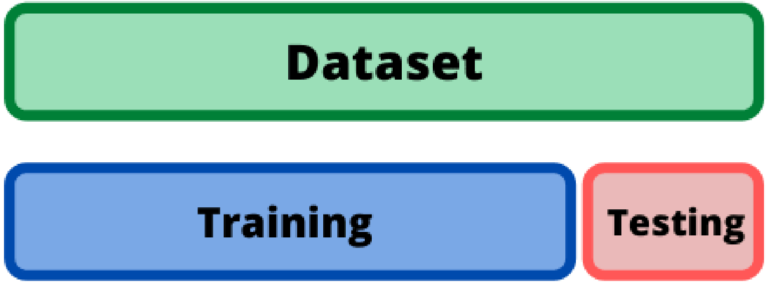
outlook	temp	humidity	windy	play

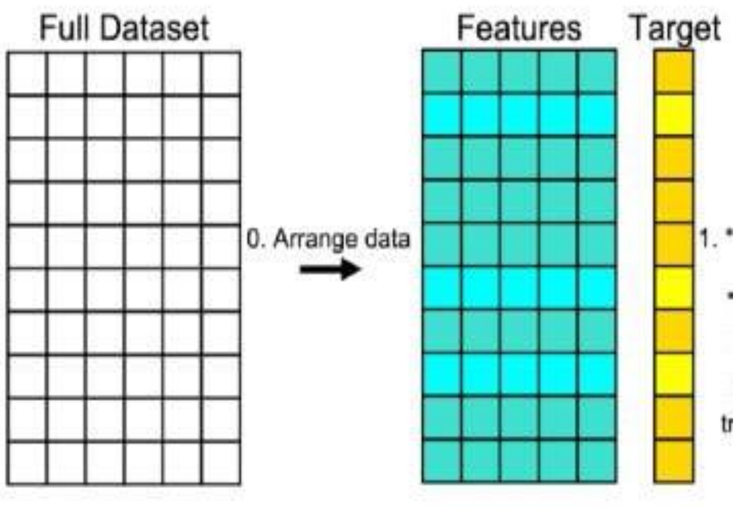
```
X = df[['outlook', 'temp', 'humidity', 'windy']]

y = df[['play']]
```

La classification avec les arbres de décision

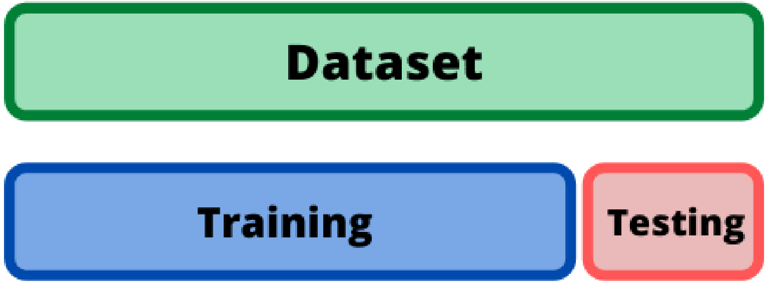


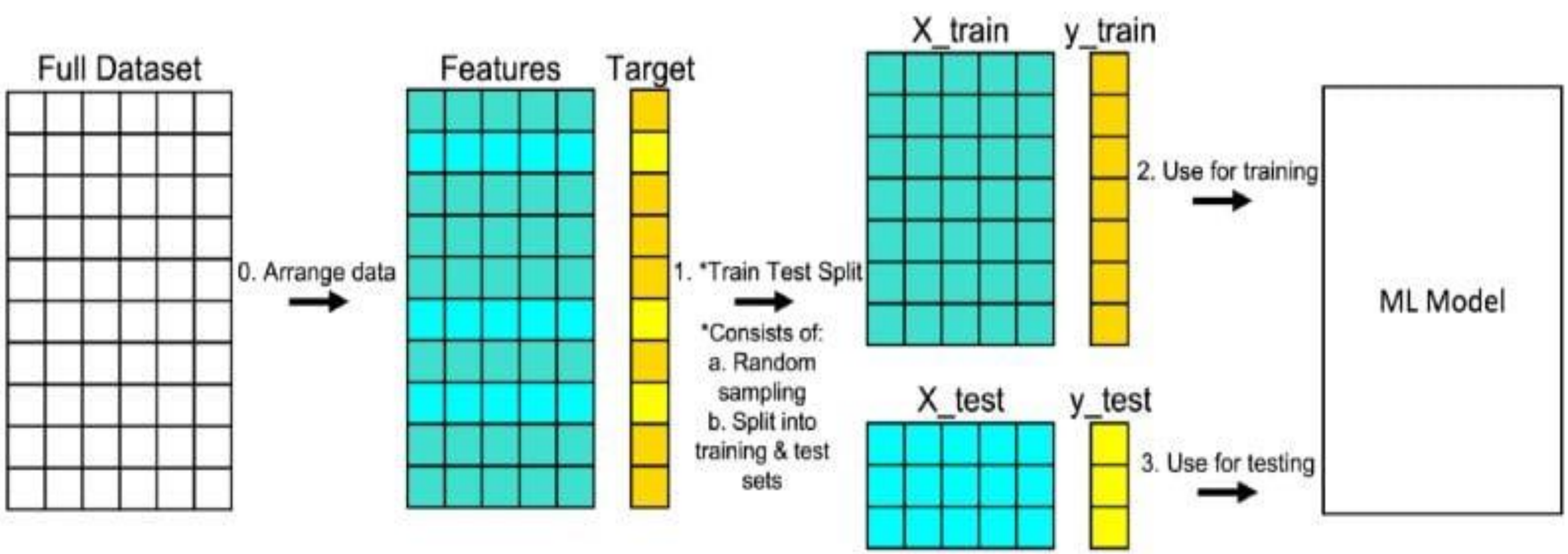
La classification avec les arbres de décision



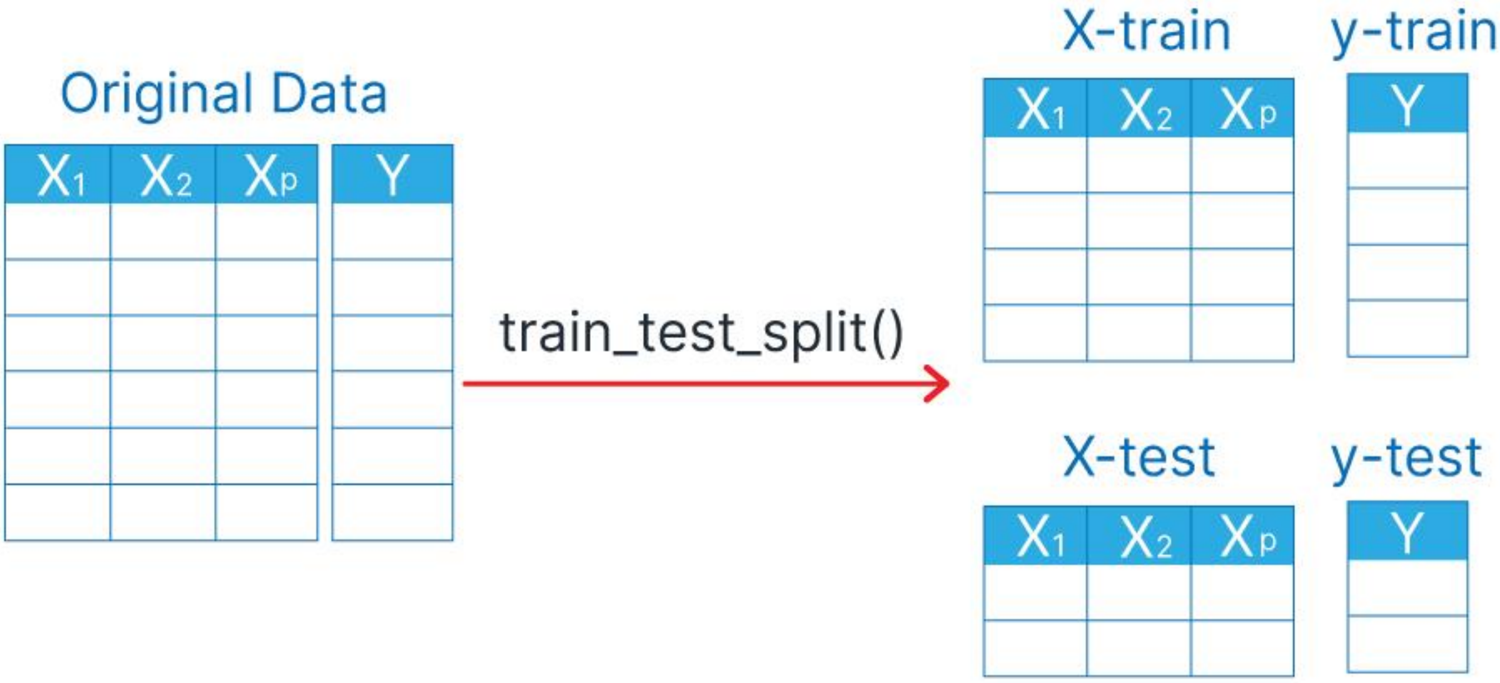


La classification avec les arbres de décision

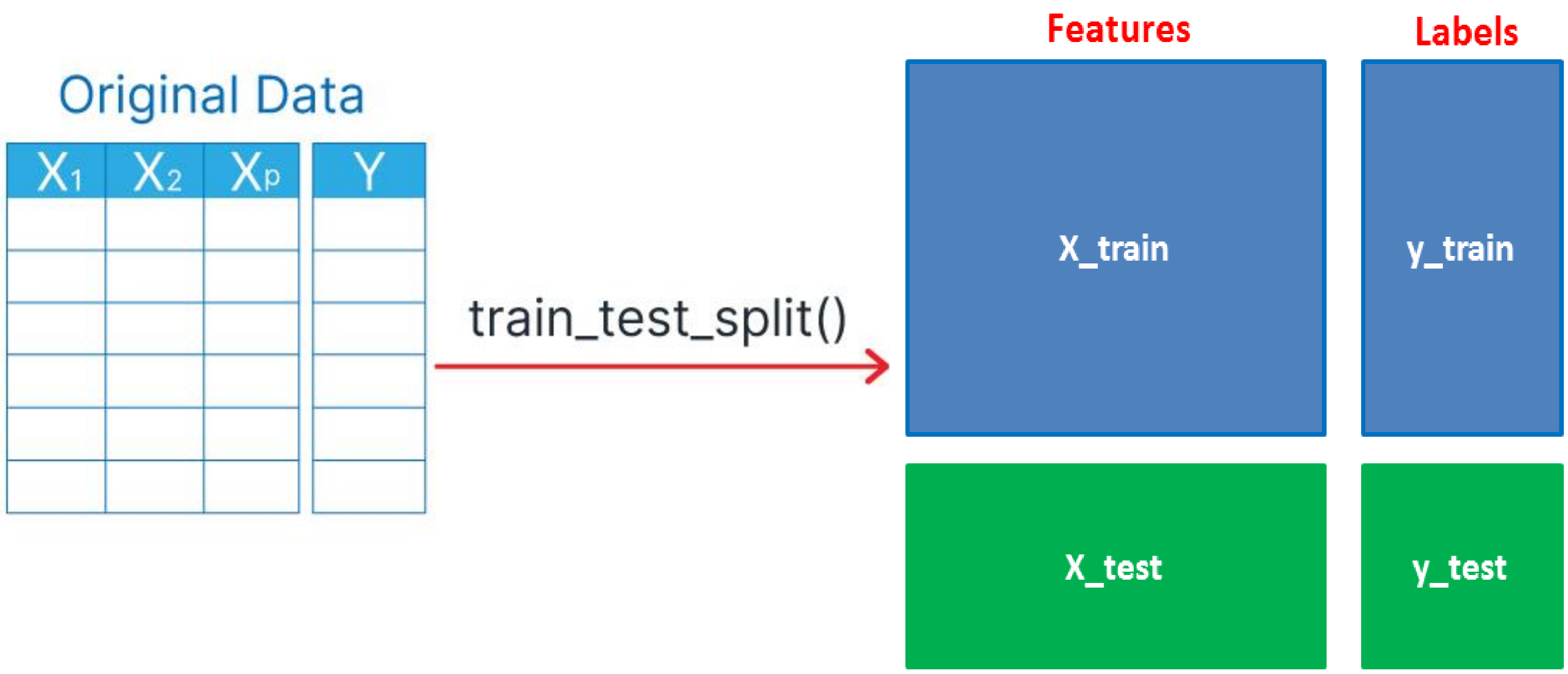




La classification avec les arbres de décision



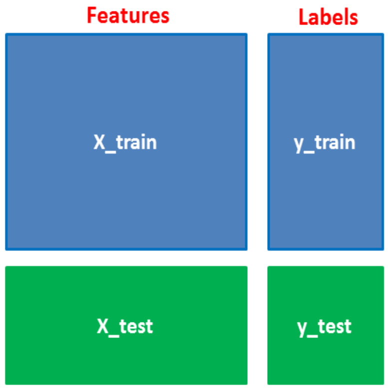
La classification avec les arbres de décision



La classification avec les arbres de décision

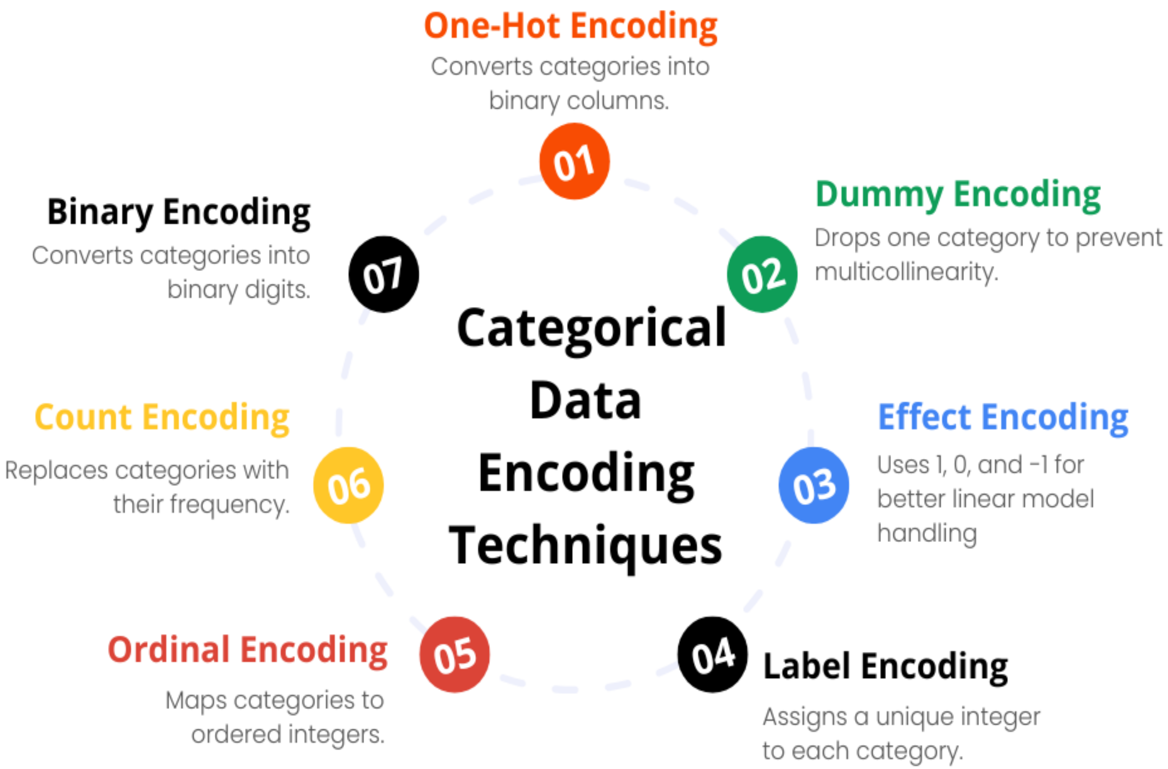
- **Create training and test sets : 70% of it is in the training set, and 30% of it is in the testing set.**

```
X_train, X_test, y_train, y_test   =

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



La classification avec les arbres de décision

- **Encoding : Encode categorical data as numbers**



https://blog.dailydoseofds.com/p/7-categorical-data-encoding-techniques

La classification avec les arbres de décision

- **Encoding : One-Hot Encoding**

Human-Readable

Pet
Cat
Dog
Turtle
Fish

Machine-Readable

Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

La classification avec les arbres de décision

- **Encoding : Label Encoding**

Human-Readable Machine-Readable

La classification avec les arbres de décision

- **Encoding : One-Hot Encoding**

```
ohe = OneHotEncoder()

ohe.fit(X_train)

X_train_ohe = ohe.transform(X_train).toarray()
```

La classification avec les arbres de décision

- **Encoding : One-Hot Encoding**

```
ohe = OneHotEncoder()

ohe.fit(X_train)

X_train_ohe = ohe.transform(X_train).toarray()


X_train_ohe  = ohe.fit_transform(X_train).toarray()
```

La classification avec les arbres de décision

- **Encoding : One-Hot Encoding**

```
ohe.get_feature_names_out()
```

```
array(['outlook_overcast', 'outlook_rainy', 'outlook_sunny', 'temp_cool',
       'temp_hot', 'temp_mild', 'humidity_high', 'humidity_normal',
       'windy_False', 'windy_True'], dtype=object)
```

La classification avec les arbres de décision

- **Encoding : One-Hot Encoding**

ohe = OneHotEncoder()

ohe.fit(X_train)

X_train_ohe = ohe.transform(X_train).toarray()

outlook temp humidity windy

X_train_ohe

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

La classification avec les arbres de décision

- **Encoding : One-Hot Encoding**

```
ohe_df = pd.DataFrame(X_train_ohe, columns=ohe.get_feature_names(X_train.columns))

ohe_df.head()
```

Out[13]:

	outlook_overcast	outlook_rainy	outlook_sunny	temp_cool	temp_hot	temp_mild	humi
0	0.0	0.0	1.0	1.0	0.0	0.0	...
1	1.0	0.0	0.0	0.0	1.0	0.0	...
2	0.0	0.0	1.0	0.0	1.0	0.0	...
3	0.0	1.0	0.0	0.0	0.0	1.0	...
4	0.0	1.0	0.0	1.0	0.0	0.0	...

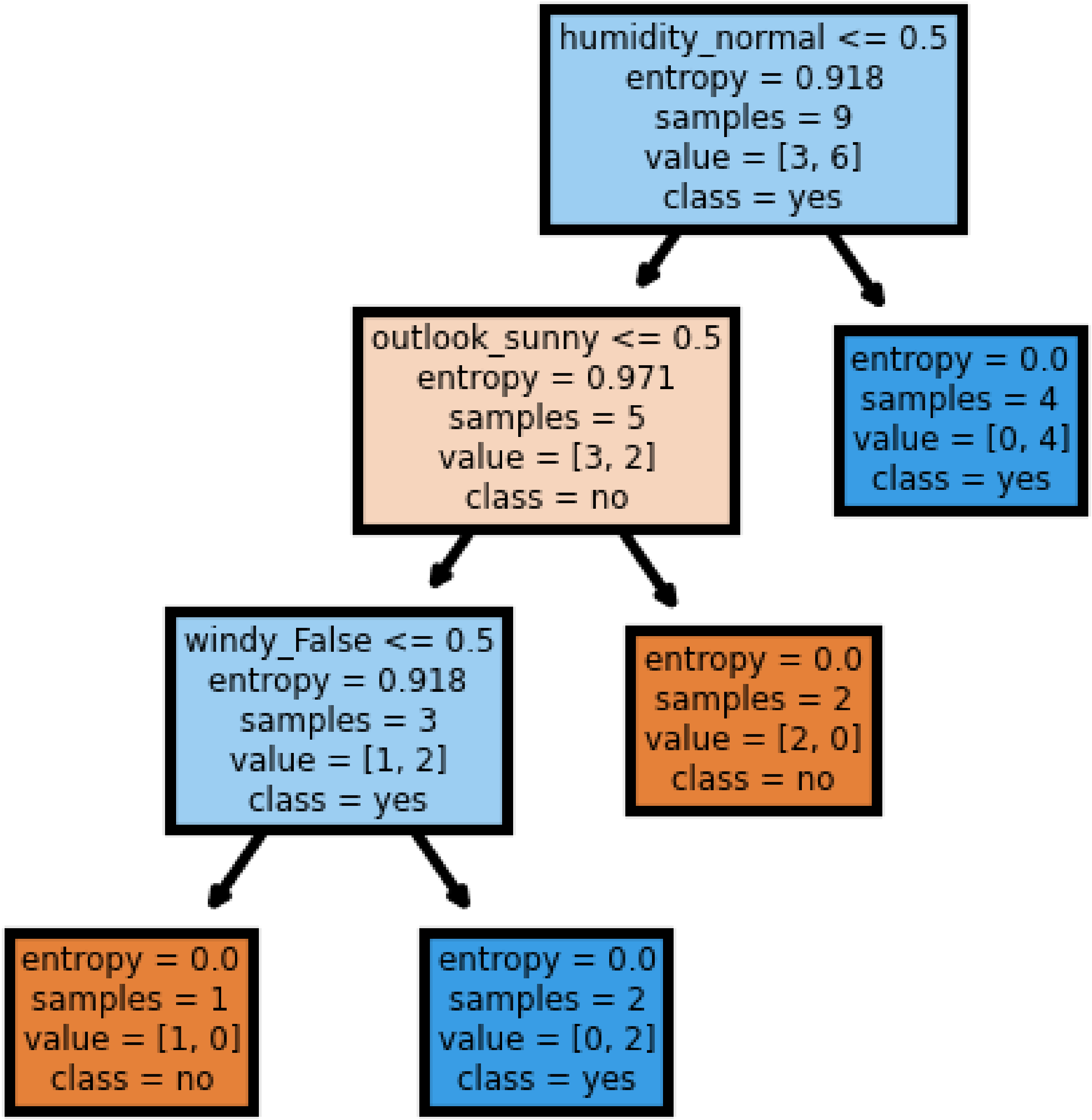
La classification avec les arbres de décision

- **Train the decision tree & Plot**

  https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

```
clf = DecisionTreeClassifier(criterion='entropy')

clf.fit(X_train_ohe, y_train)
```

criterion{“gini”, “entropy”, “log_loss”}, default=”gini”


humidity_normal <= 0.5
entropy = 0.918
samples = 9
value = [3, 6]
class = yes
outlook_sunny <= 0.5
entropy = 0.971
samples = 5
value = [3, 2]
class = no
entropy = 0.0
samples = 4
value = [0, 4]
class = yes
windy_False <= 0.5
entropy = 0.918
samples = 3
value = [1, 2]
class = yes
entropy = 0.0
samples = 2
value = [2, 0]
class = no
entropy = 0.0
samples = 1
value = [1, 0]
class = no
entropy = 0.0
samples = 2
value = [0, 2]
class = yes

La classification avec les arbres de décision

humidity_normal \	play
1.0	yes
0.0	yes
0.0	no
0.0	no
1.0	yes
0.0	no
1.0	yes
0.0	yes
1.0	yes

La classification avec les arbres de décision

humidity_normal	play
1.0	yes
0.0	yes
0.0	no
0.0	no
1.0	yes
0.0	no
1.0	yes
0.0	yes
1.0	yes

La classification avec les arbres de décision

humidity_normal	outlook_sunny	play
1.0	1.0	yes
0.0	0.0	yes
0.0	1.0	no
0.0	0.0	no
1.0	0.0	yes
0.0	1.0	no
1.0	1.0	yes
0.0	0.0	yes
1.0	0.0	yes

La classification avec les arbres de décision

humidity_normal	outlook_sunny	play
1.0	1.0	yes
0.0	0.0	yes
0.0	1.0	no
0.0	0.0	no
1.0	0.0	yes
0.0	1.0	no
1.0	1.0	yes
0.0	0.0	yes
1.0	0.0	yes

La classification avec les arbres de décision

1. Import necessary modules
2. Load & explore the dataset
3. Split the DataFrame into features (X) and target/class (y)
4. Create training and test sets
5. Encode categorical data as numbers : OneHotEncoding
6. Train the decision tree
7. Plot the decision tree
8. **Predict and Evaluate : Accuracy & Confusion matrix**
9. **Tree Pruning**

La classification avec les arbres de décision

- **Predict and Evaluate : Accuracy**

```
X_test_ohe = ohe.transform(X_test)

y_preds = clf.predict(X_test_ohe)
```

```
print('Accuracy: ', accuracy_score(y_test, y_preds))
```

```
Accuracy:  0.6
```

La classification avec les arbres de décision

- **Predict and Evaluate : Accuracy**

```
y_test
```

	play
9	yes
11	yes
0	no
12	yes
5	no

```
list(y_preds)
```

```
['yes', 'no', 'no', 'yes', 'yes']
```

La classification avec les arbres de décision

- **Predict and Evaluate : Confusion matrix**

```
cf_matrix = confusion_matrix(y_test, y_preds)

print(cf_matrix)
```

```
[[1 1]
 [1 2]]
```

The default order of labels in the confusion matrix is the **lexicographical order** of the unique classes in *y*.

La classification avec les arbres de décision

- **Predict and Evaluate : Confusion matrix**

```
cf_matrix = confusion_matrix(y_test, y_preds,
                             labels=["yes", "no"])
print(cf_matrix)
```

```
[[2 1]
 [1 1]]
```

La classification avec les arbres de décision

- **Predict and Evaluate : Confusion matrix**

```
print(precision_score(y_test, y_preds, pos_label='yes'))

print(recall_score(y_test, y_preds, pos_label='yes'))
```

```
0.6666666666666666
0.6666666666666666
```

La classification avec les arbres de décision

- **Predict and Evaluate : Confusion matrix**

```
print(precision_score(y_test, y_preds, pos_label='no'))

print(recall_score(y_test, y_preds, pos_label='no'))
```

```
0.5
0.5
```

La classification avec les arbres de décision

- **Tree Pruning : Elagage - Pre-Pruning Strategies**

DecisionTreeClassifier

```
class sklearn.tree.DecisionTreeClassifier(*, criterion='gini',
splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_features=None, random_state=None,
max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None,
ccp_alpha=0.0, monotonic_cst=None)
```

[source]

La classification avec les arbres de décision

- **Tree Pruning : Elagage - Pre-Pruning Strategies**

max_depth : *int, default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*

The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

La classification avec les arbres de décision

- **Tree Pruning : Elagage - Pre-Pruning Strategies**

min_samples_leaf : ***int or float, default=1***

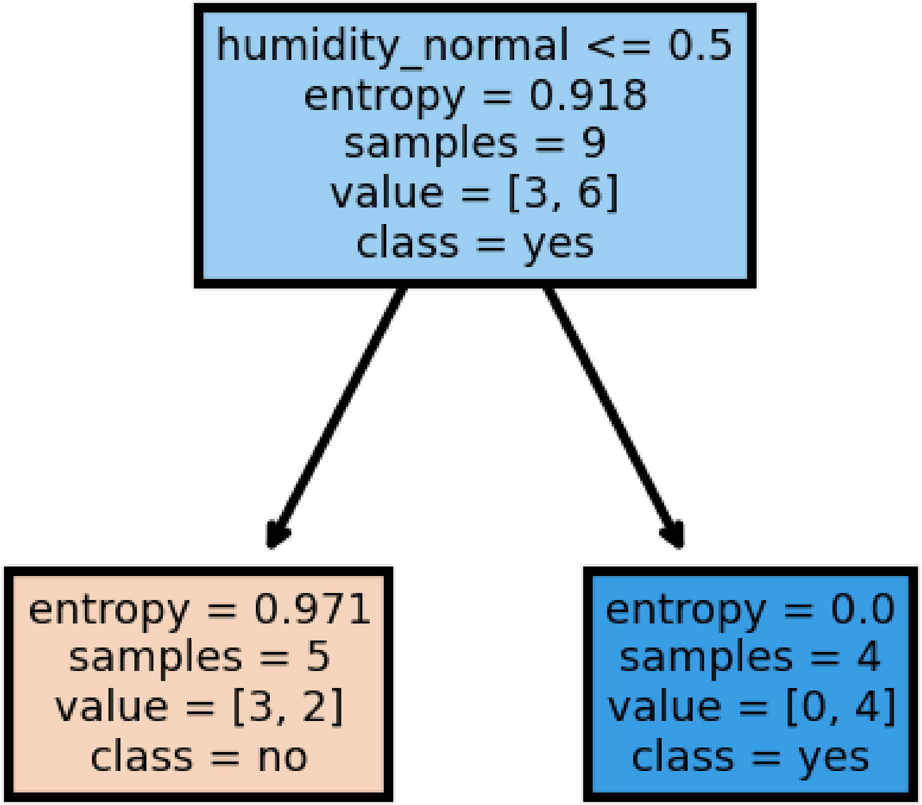
The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider `min_samples_leaf` as the minimum number.
- If float, then `min_samples_leaf` is a fraction and `ceil(min_samples_leaf * n_samples)` are the minimum number of samples for each node.

La classification avec les arbres de décision

- **Tree Pruning : Elagage - Pre-Pruning Strategies**

max_depth=1

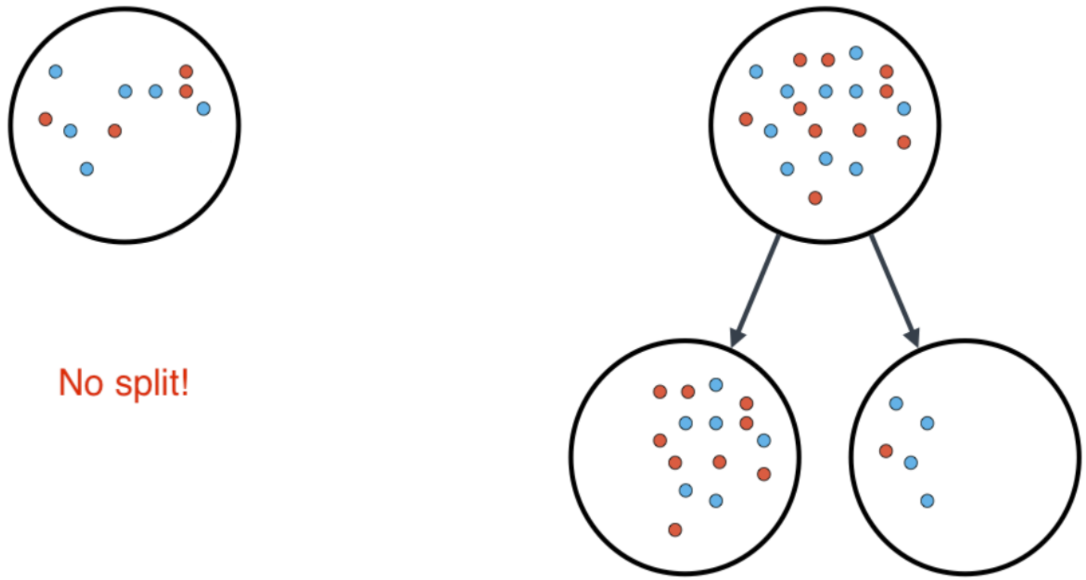


La classification avec les arbres de décision

- **Tree Pruning : Elagage - Pre-Pruning Strategies**

min_samples_split = 11

If a node has fewer samples than min_samples_split, it will not be split further.



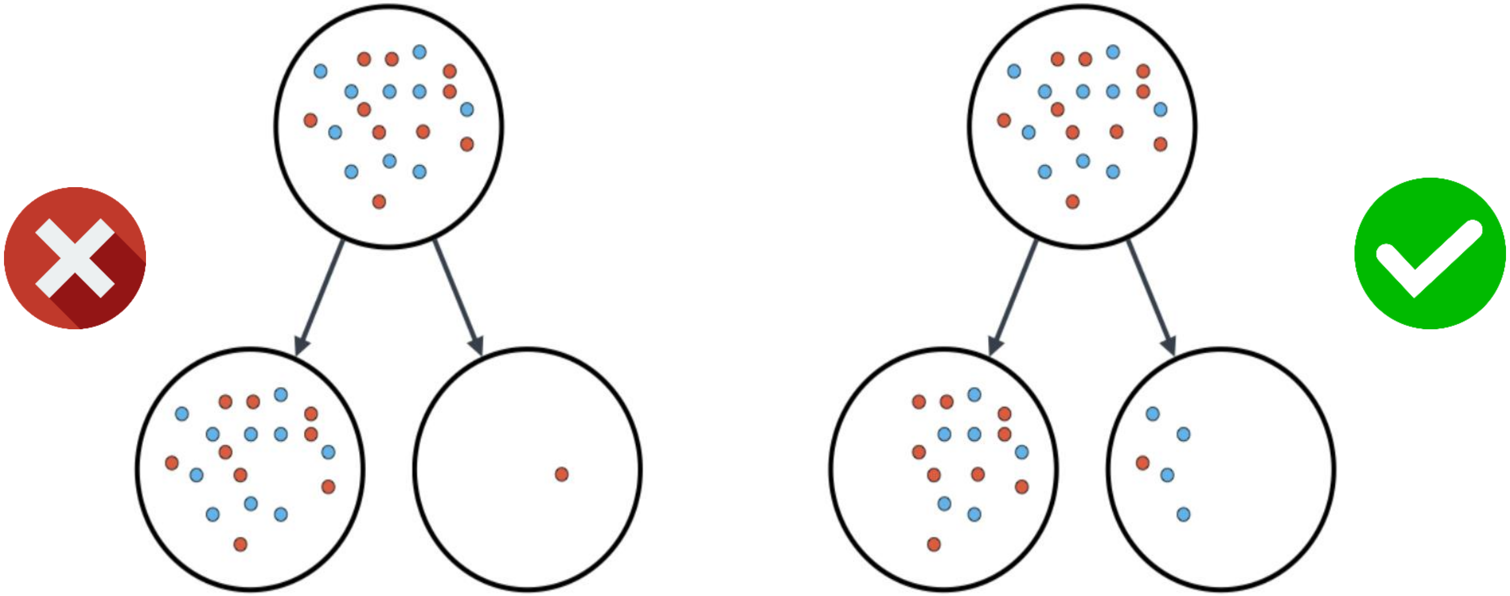
Minimum number of samples to split = 11   Minimum number of samples to split = 11

La classification avec les arbres de décision

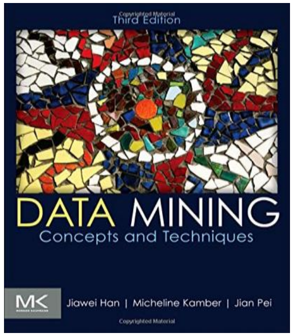
- **Tree Pruning : Elagage - Pre-Pruning Strategies**

min_samples_leaf = 2

If a split results in a leaf node with fewer samples than min_samples_leaf, the split is not allowed. The node is not split further and becomes a leaf node.

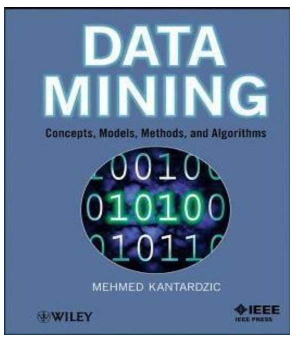


Ressources



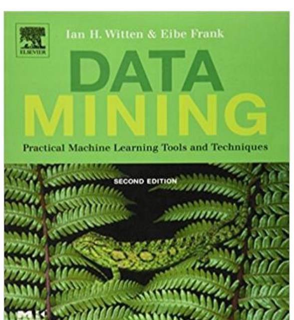
Data Mining : concepts and techniques, 3rd Edition

- ✓ Auteur : Jiawei Han, Micheline Kamber, Jian Pei
- ✓ Éditeur : Morgan Kaufmann Publishers
- ✓ Edition : Juin 2011 - 744 pages - ISBN 9780123814807



Data Mining : concepts, models, methods, and algorithms

- ✓ Auteur : Mehmed Kantardzi
- ✓ Éditeur : John Wiley & Sons
- ✓ Edition : Aout 2011 – 552 pages - ISBN : 9781118029121



Data Mining: Practical Machine Learning Tools and Techniques

- ✓ Auteur : Ian H. Witten & Eibe Frank
- ✓ Éditeur : Morgan Kaufmann Publishers
- ✓ Edition : Juin 2005 - 664 pages - ISBN : 0-12-088407-0