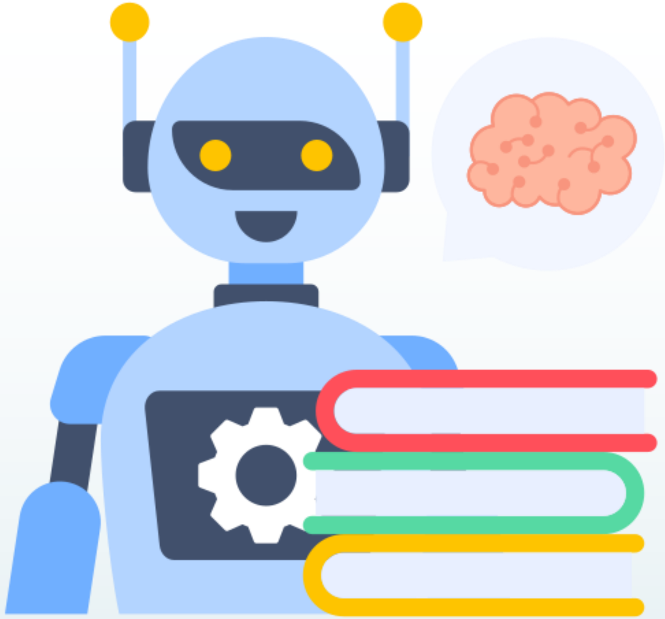
Train-test split in ML



Presentation by:

Dr. Ashima Tyagi

Outline

- Train-test split
- Working example

Two Splitting: Train-Test Split

- A train test split is when you split your data into a training set and a testing set.
- The training set is used for training the model, and the testing set is used to test your model.
- This allows you to train your models on the training set, and then test their accuracy on the unseen testing set.
- For example 80% for training and 20% for testing. This ensures that both sets are representative of the entire dataset, and gives you a good way to measure the accuracy of your models.

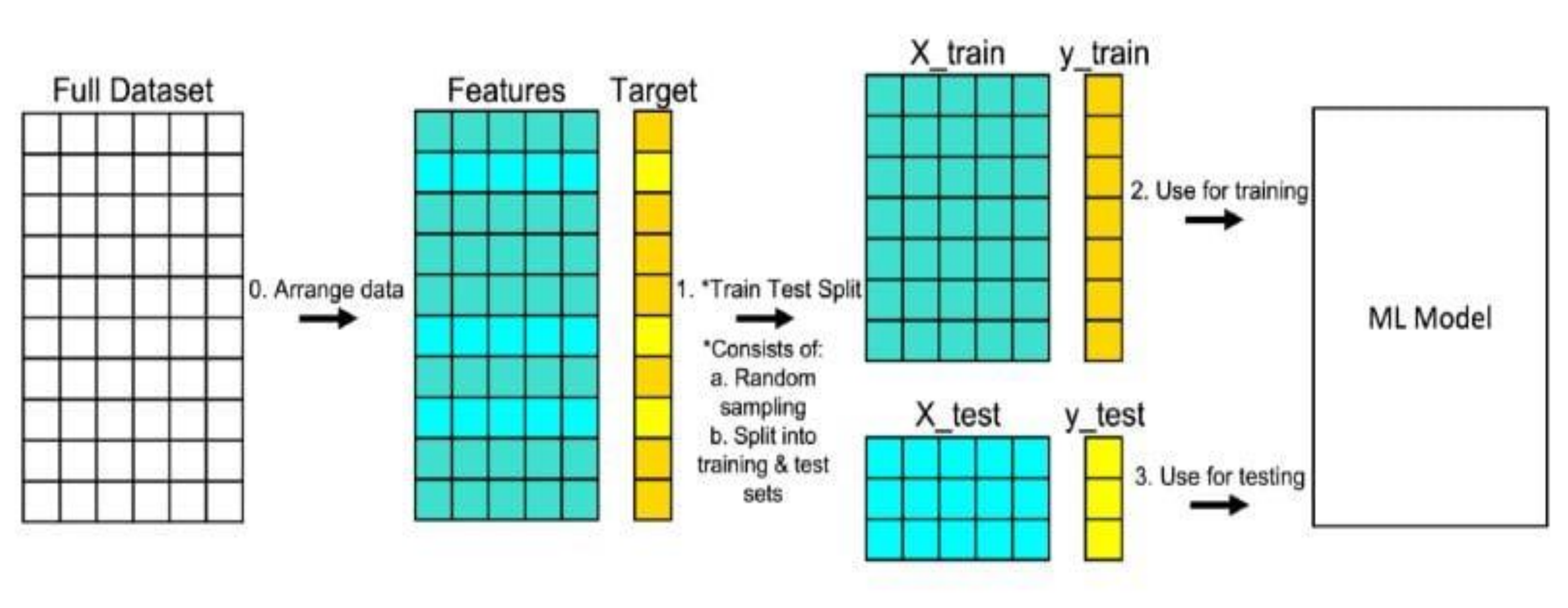
Here's how the train-test split works:

1. **Splitting the Data:** The dataset is divided into two subsets: the training set and the test set. The training set is used to train the model, while the test set is used to evaluate its performance.

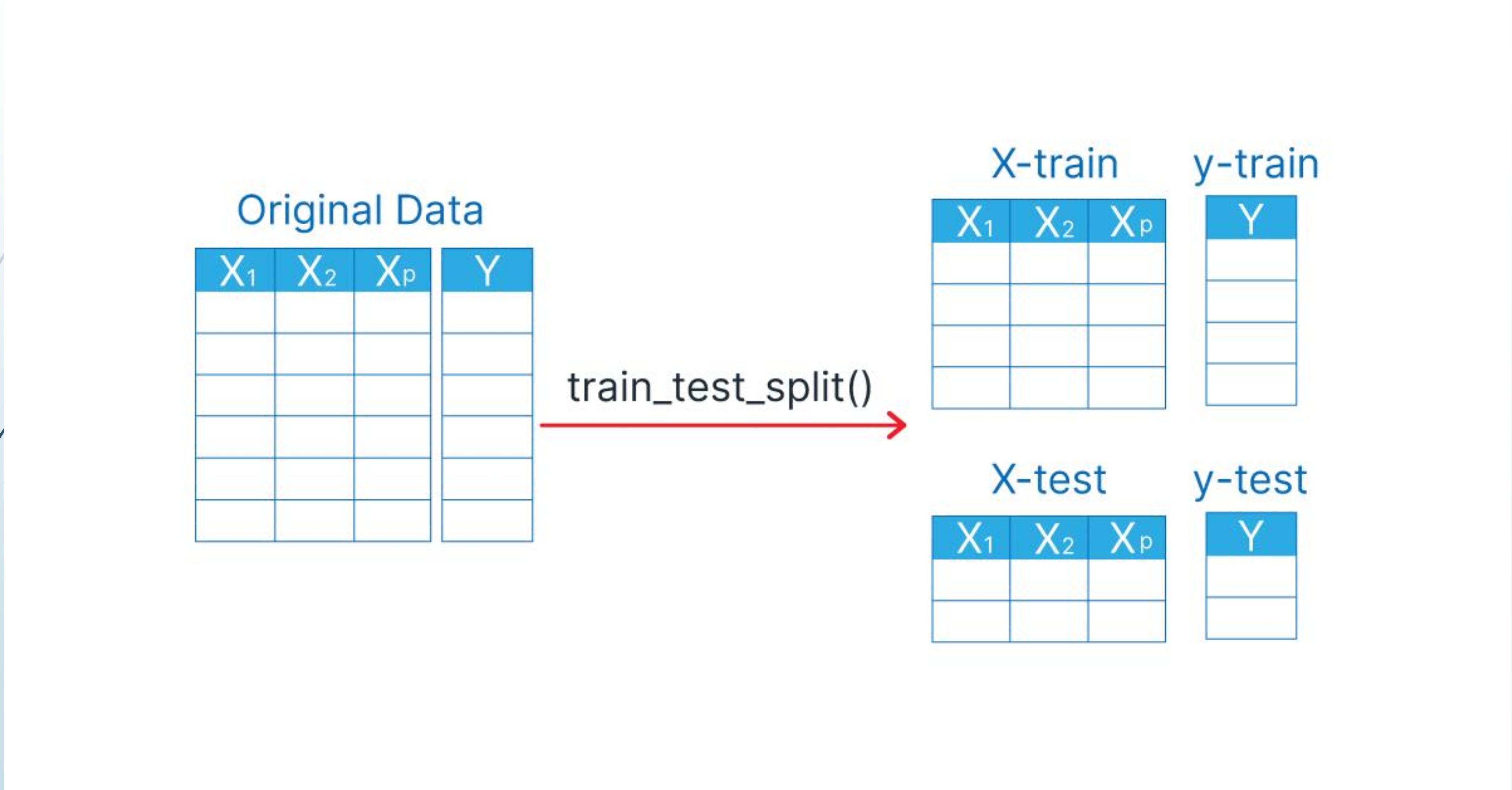
2. **Training the Model:** The model is trained on the training set using a machine learning algorithm. The model learns patterns and relationships in the data to make predictions.

3. **Evaluating the Model:** Once the model is trained, it is evaluated on the test set. This provides an estimate of how well the model will perform on new, unseen data.

Train-test split



Train-test split



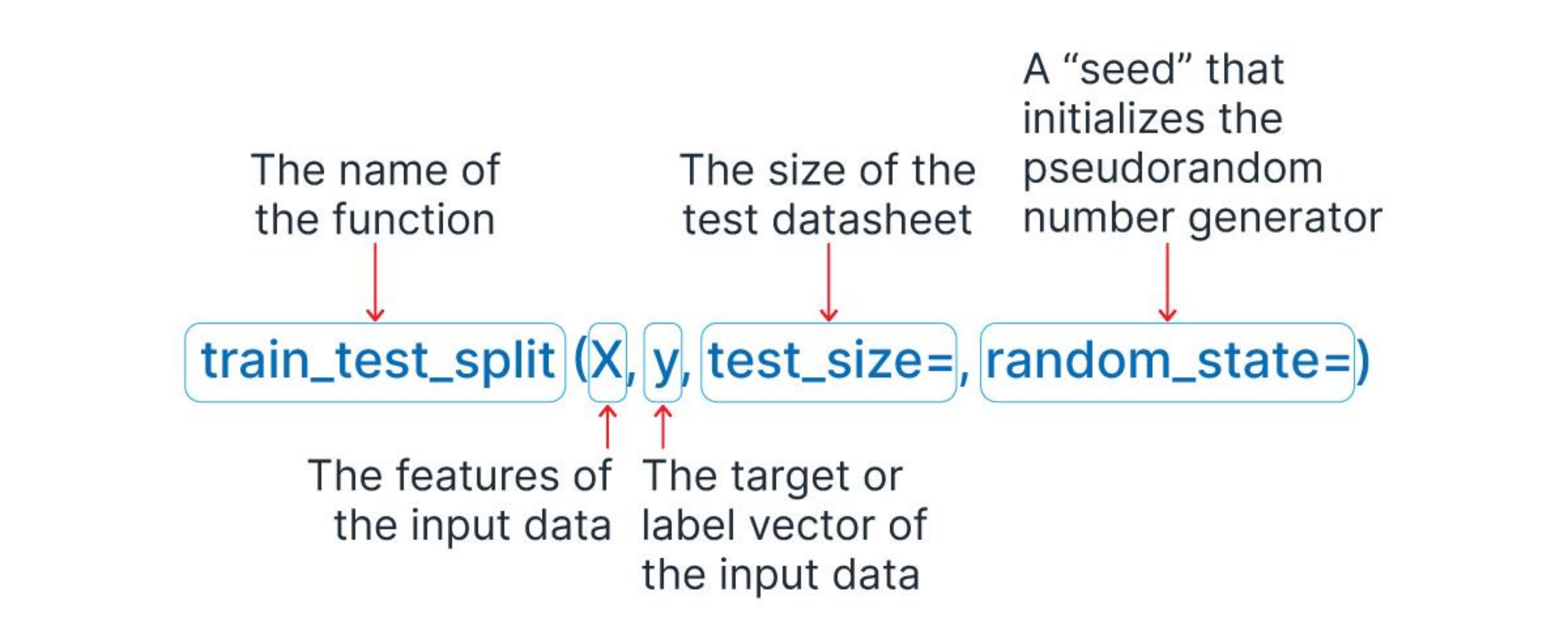
Syntax of Train Test Split

Before continuing, please note that in order to use this feature, you must first import it.

```
from sklearn.model_selection import train_test_split
```

After importing the function as above, call it as train_test_split() .

Train-test split



The name of the function

The size of the test datasheet

A “seed” that initializes the pseudorandom number generator

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

The features of the input data

The target or label vector of the input data

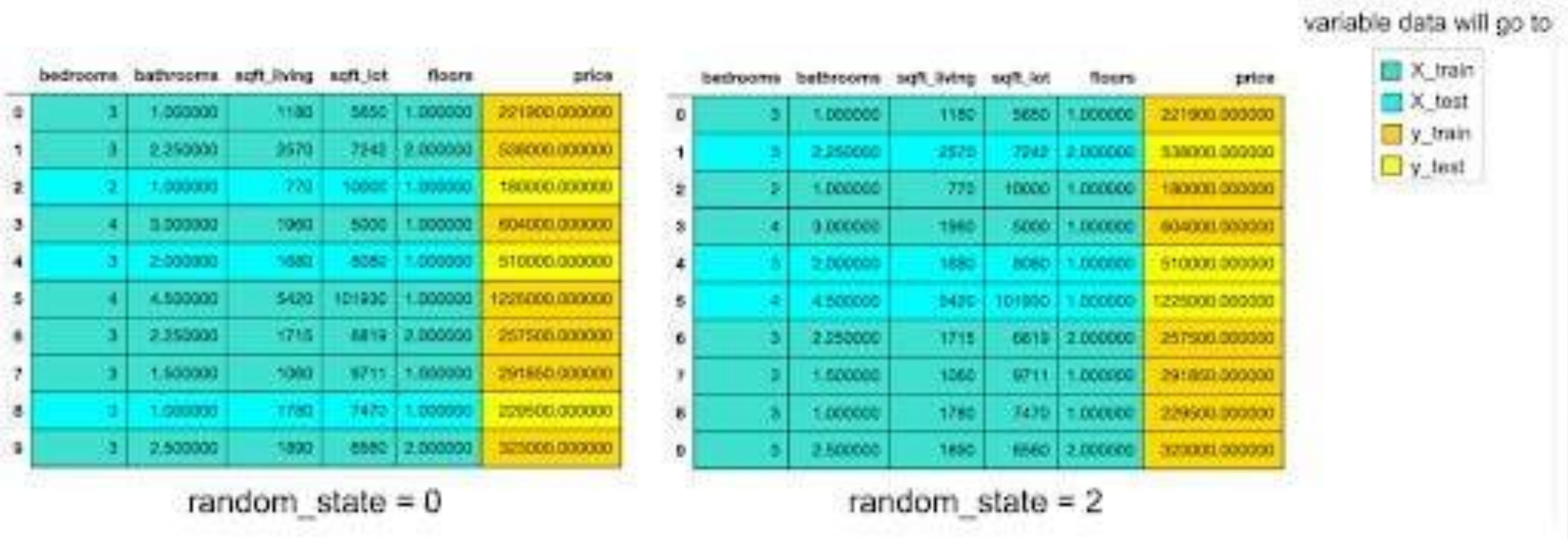
Train-test split

If the train-test split/ **test_size** is: **0.2** then,

Split the data set into two pieces — a training set and a testing set. This consists of random sampling without replacement about 80 percent of the rows (you can vary this) and putting them into your training set. The remaining 20 percent is put into your test set. Note that the colors in “Features” and “Target” indicate where their data will go (“X_train,” “X_test,” “y_train,” “y_test”) for a particular train test split.

Train-test split

Random State: The random_state is a pseudo-random number parameter that allows you to reproduce the same train test split each time you run the code.


variable data will go to
X_train
X_test
y_train
y_test

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	price
0	3	1.000000	1180	5650	1.000000	221900.000000
1	3	2.250000	2570	7242	2.000000	538000.000000
2	2	1.000000	770	10000	1.000000	180000.000000
3	4	3.000000	1960	5000	1.000000	604000.000000
4	3	2.000000	1680	8080	1.000000	510000.000000
5	4	4.500000	5420	101930	1.000000	1225000.000000
6	3	2.250000	1715	6819	2.000000	257500.000000
7	3	1.500000	1060	9711	1.000000	291850.000000
8	3	1.000000	1780	7470	1.000000	229500.000000
9	3	2.500000	1890	6560	2.000000	323000.000000

random_state = 0

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	price
0	3	1.000000	1180	5650	1.000000	221900.000000
1	3	2.250000	2570	7242	2.000000	538000.000000
2	2	1.000000	770	10000	1.000000	180000.000000
3	4	3.000000	1960	5000	1.000000	604000.000000
4	3	2.000000	1680	8080	1.000000	510000.000000
5	4	4.500000	5420	101930	1.000000	1225000.000000
6	3	2.250000	1715	6819	2.000000	257500.000000
7	3	1.500000	1060	9711	1.000000	291850.000000
8	3	1.000000	1780	7470	1.000000	229500.000000
9	3	2.500000	1890	6560	2.000000	323000.000000

random_state = 2

The image above shows that if you select a different value for random_state, different information would go to “X_train,” “X_test,” “y_train” and “y_test”.

Which random number to choose?

In machine learning, the choice of the random number to use for the random_state parameter is arbitrary. You can use any non-negative integer value, and the specific value you choose does not matter as long as you use the same value consistently if you want to reproduce the same random splits.

For example, you could use *random_state=0, random_state=42,* or any other integer value. The important thing is to use the same value consistently if you want to ensure that your results are reproducible.

- **What does 42 mean?**

Nothing special 😄

Just a fixed seed number.

You can use:

```
0, 1, 21, 42, 100
```

As long as it's constant.

Without random_state

Every run gives:

Run	Accuracy
1	82%
2	87%
3	79%

You can't compare models properly 😵

Train-test split

- **With random_state**

Every run gives:

Run	Accuracy
1	84%
2	84%
3	84%

Stable and fair.

Example

Let's consider a dataset of iris flowers with features such as sepal length, sepal width, petal length, and petal width. We want to predict the species of the iris flower based on these features.

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Load the iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 'X_train' and 'y_train' are used to train the model
# 'X_test' and 'y_test' are used to evaluate the model's performance
```

Thank You