SH Dr. Soufiane Hourri

drsoufiane.com

DUT · SEMESTER 3 · COMPUTER SCIENCE

Machine Learning

Session 02 · Preparing data for a model

Dr. Soufiane Hourri, PhD, HDR

Associate Professor & AI Consultant

École Supérieure de Technologie de Safi · Cadi Ayyad University

2026-2027

Today's session

01 Split the data first

02 Missing values

03 Encoding categories

04 Scaling numerical features

05 Pipelines

06 Exercise: a preparation plan

TODAY'S FOCUS

A model only accepts complete numerical tables. Every preparation step uses statistics, and every statistic must come from the training data only.

Learning objectives

By the end of this session, you will be able to

1. Split a dataset with train_test_split, with and without stratification
2. Choose how to handle missing values and explain the effect on the statistics
3. Encode nominal and ordinal variables
4. Standardise or min-max scale numerical features
5. Build a preprocessing pipeline that learns only from the training set

PREREQUISITES

- Session 01: mean, median, standard deviation, z-score, proportions
- Your Session 01 lab notebook

Warm-up from Session 01

QUESTION

A column contains a few extreme values. Which summary is least affected by them?

A

The mean

B

The median

C

The standard deviation

D

The range

01

CHAPTER

Split the data first

The test set is put aside before any other step, and drawn so that it represents the data.

Why we split before preparing anything

- The test set plays the role of future data that the model has never seen
- Every preparation step learns something from data: a median, a mean, a list of categories
- If those values are computed on all the data, the test set influences training
  - The test score becomes optimistic: this is data leakage

KEY TAKEAWAY

Split first. Learn every preparation value on the training set only.

A proportion and a random sample

DEFINITION

A proportion is a count divided by a total: p = count / n. In a random sample, every observation has the same chance of being selected.

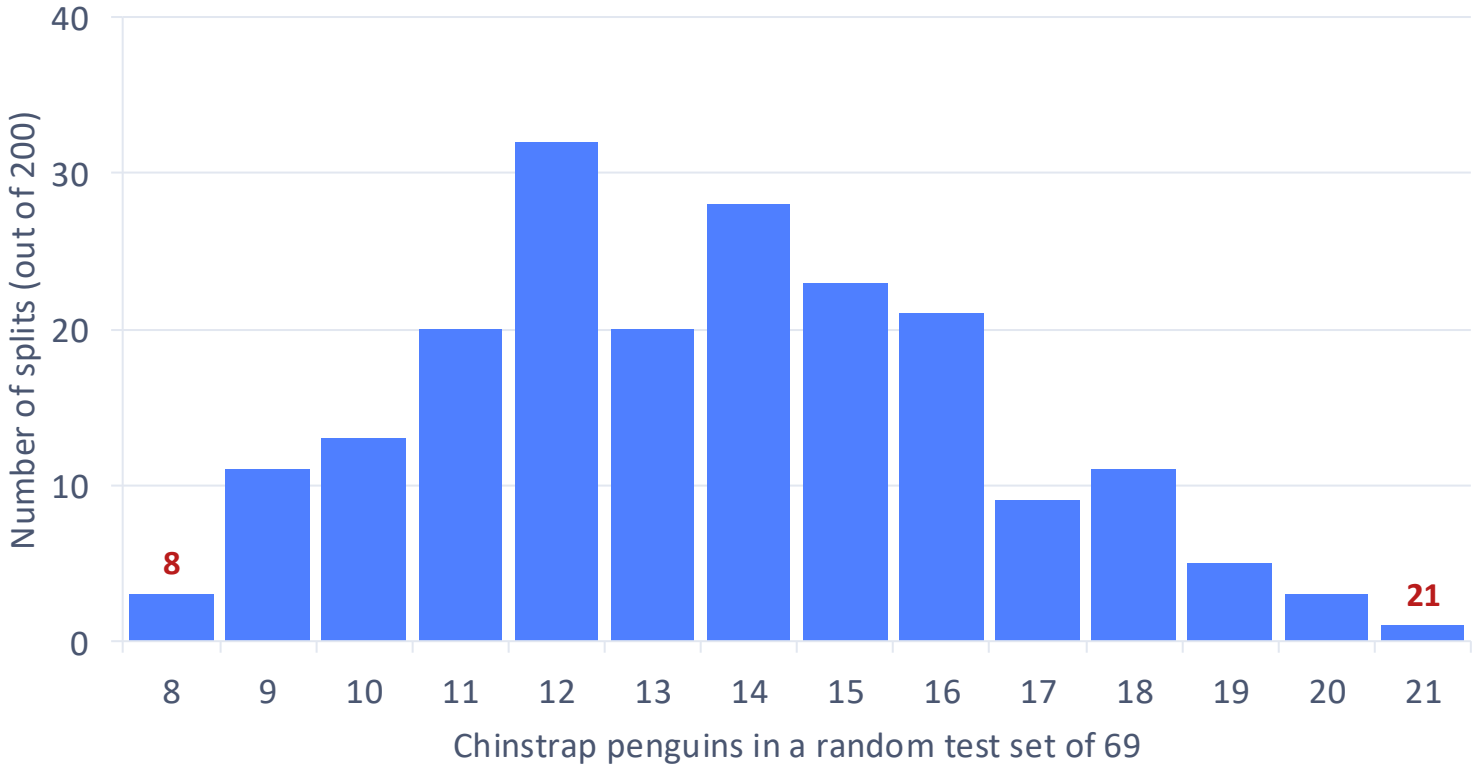
EXAMPLE

Chinstrap penguins: 68 / 344 = 0.198, about 20%. A random test set of 69 penguins should therefore contain about 0.198 × 69 ≈ 14 Chinstraps.

WATCH OUT

The penguins file is sorted by species. Taking the last 69 rows as a test set would give 69 Gentoo penguins and no other species. Always shuffle.

Random samples do not all look alike



200 random splits of the Palmer penguins (test_size = 0.2), computed with scikit-learn 1.9.1.

MAIN FINDING

The expected count is about 14, but random test sets held from 8 to 21 Chinstraps: 11.6% to 30.4%.

A proportion varies from one sample to another.

Stratified split: same proportions in both sets

CODE · PYTHON

```python
from sklearn.model_selection import (
    train_test_split)

X = df.drop(columns="species")
y = df["species"]
X_train, X_test, y_train, y_test = (
    train_test_split(X, y,
        test_size=0.2, random_state=42,
        stratify=y))
print(len(X_train), len(X_test))
print(y_test.value_counts(
    normalize=True).round(3))
```

OUTPUT

```
275 69
species
Adelie       0.435
Gentoo       0.362
Chinstrap    0.203
```

Full data: 0.442, 0.360, 0.198

Preparing the data before splitting

✗ INCORRECT APPROACH

```
X = scaler.fit_transform(X)
X_tr, X_te = train_test_split(X)
```

✓ CORRECTION

```
X_tr, X_te = train_test_split(X)
X_tr = scaler.fit_transform(X_tr)
X_te = scaler.transform(X_te)
```

WHY IT FAILS

The scaler computed its mean and standard deviation with the test rows included. The test set is no longer unseen, so the score is optimistic.

02

CHAPTER

Missing values

Every choice for a gap changes the data: we must know how.

Finding missing values, and asking why

Find them

- `df.isna().sum()` counts gaps per column
- Penguins: 2 rows with no measurements (0.6%)
- 11 penguins with no recorded sex (3.2%)
- Proportion missing: `df.isna().mean()`

Ask why

- By chance: a lost page, a sensor glitch. Simple fixes are reasonable
- Linked to the value: heavy people skip the weight question. Simple fixes introduce bias
- Ask the people who collected the data

Three ways to handle a gap

Drop the rows

WHEN

Few rows affected, missing by chance

COST

Loses data; impossible for a new case to predict

EXAMPLE

The 2 penguins with no measurements

Drop the column

WHEN

Most of the column is missing

COST

Loses a whole feature

EXAMPLE

A column 80% empty

Fill (impute)

WHEN

Keeps every row

COST

Invents values: changes the statistics

EXAMPLE

Median body mass, most frequent sex

Filling with the mean shrinks the spread

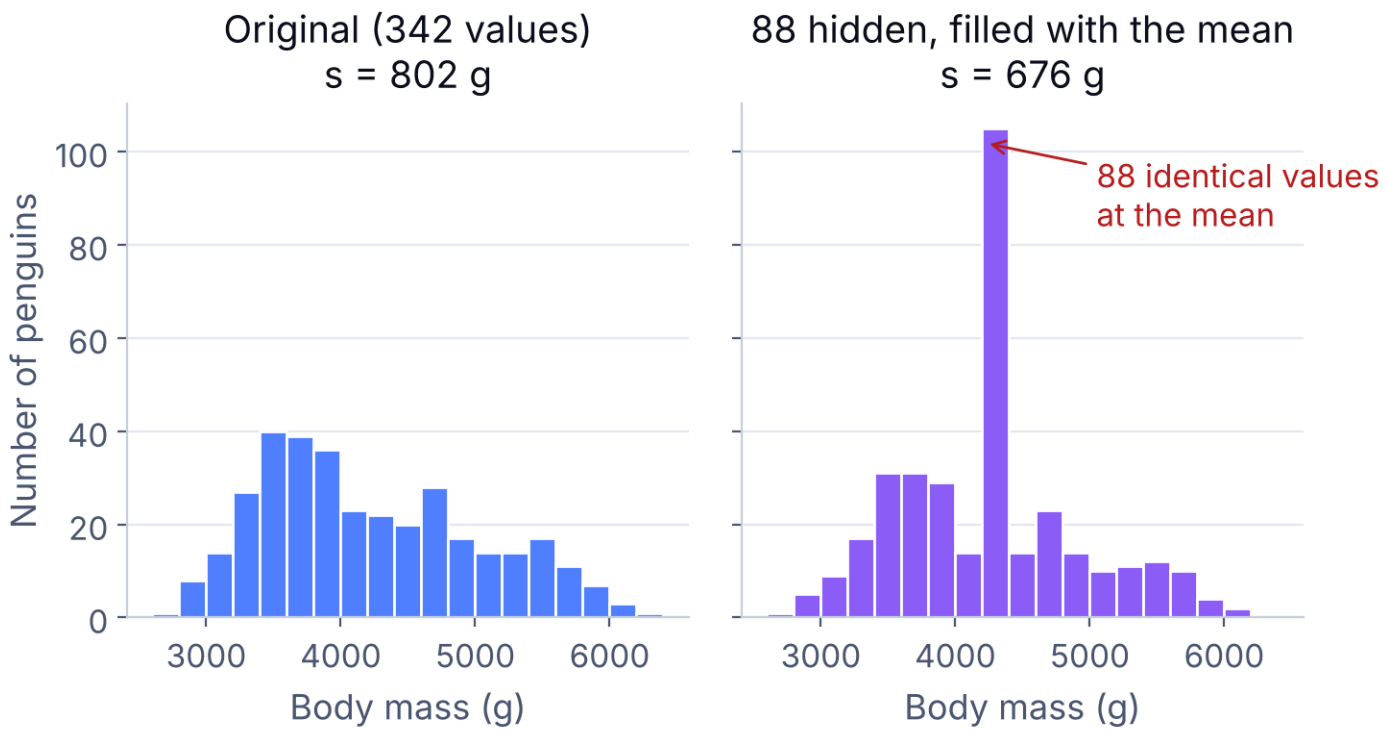


Figure 1. 88 of 342 body masses hidden at random, then filled with the mean (4,215 g). Data: Palmer penguins.

WHAT TO NOTICE

- The mean hardly changes: 4,202 → 4,215 g
- The standard deviation drops: 802 → 676 g
- 88 identical values form an artificial spike
- Fill only when few values are missing

Which value should fill the gap?

Column type	Fill with	scikit-learn	Why
Numerical, skewed or with outliers	Median	`strategy="median"`	Robust to extreme values
Numerical, roughly symmetric	Mean	`strategy="mean"`	Uses every value
Categorical	Mode (most frequent)	`strategy="most_frequent"`	The only center for categories
Categorical, missingness informative	A category "missing"	`strategy="constant"`	Keeps the information

Table 1. Every fill value is learned on the training set, then reused unchanged on the test set and on new data.

SimpleImputer learns on the training set

CODE · PYTHON

```python
from sklearn.impute import (
    SimpleImputer)

imp = SimpleImputer(strategy="median")
imp.fit(X_train[["body_mass_g"]])
print(imp.statistics_)

# same median applied to any data
mass_test = imp.transform(
    X_test[["body_mass_g"]])
```

OUTPUT

```
[4050.]
```

The median learned from the 275 training penguins. It is stored, then reused.

Knowledge check

QUESTION

A bank dataset has monthly incomes, with 4% missing and a few very high values. How should the gaps be filled?

A

With the median of the whole dataset, test set included

B

With the mean of the training set

C

With the median of the training set

D

With 0, since the income is unknown

03

CHAPTER

Encoding categories

Models compute with numbers. The type of the variable decides how to convert it.

One-hot encoding for nominal variables

island (original)	island_Biscoe	island_Dream	island_Torgersen
Torgersen	0	0	1
Biscoe	1	0	0
Dream	0	1	0

Table 2. One column per island. Each row has a single 1. No order is implied between islands.

Ordinal variables keep their order

Bac mention (ordinal)

- Passable → 0
- Assez bien → 1
- Bien → 2
- Très bien → 3
- One column; the order is kept

In scikit-learn

```
OrdinalEncoder(categories=[[
  "Passable", "Assez bien",
  "Bien", "Très bien"]])
```

Give the order explicitly: by default the categories are sorted alphabetically, which puts Passable after Bien

Numbering a nominal variable

✗ INCORRECT APPROACH

Encode the city as Safi = 1, Marrakech = 2, Agadir = 3

✓ CORRECTION

Use one-hot encoding: one 0/1 column per city

WHY IT FAILS

The model reads the numbers as quantities: Agadir becomes "three times" Safi and Marrakech lies "between" them. The order and distances are invented.

OneHotEncoder and categories never seen before

CODE · PYTHON

```python
from sklearn.preprocessing import (
    OneHotEncoder)

enc = OneHotEncoder(handle_unknown="ignore",
                    sparse_output=False)
enc.fit(X_train[["island"]])
print(enc.get_feature_names_out())

new = pd.DataFrame(
    {"island": ["Dream", "Anvers"]})
print(enc.transform(new))
```

OUTPUT

```
['island_Biscoe'
 'island_Dream'
 'island_Torgersen']

[[0. 1. 0.]
 [0. 0. 0.]]
```

Anvers was never seen: all zeros, no error.

04

CHAPTER

Scaling numerical features

Grams and millimetres should not be compared directly.

k-nearest neighbours: predicting from similar cases

DEFINITION

k-nearest neighbours (kNN) predicts the class of a new observation from the k most similar training observations: those at the smallest distance. They vote.

EXAMPLE

k = 5: among the 5 closest penguins, 4 are Gentoo and 1 is Adelie. Prediction: Gentoo.

DISTANCE BETWEEN TWO PENGUINS

$$d = \sqrt{(\Delta\text{bill_length}^2 + \Delta\text{bill_depth}^2 + \Delta\text{flipper}^2 + \Delta\text{mass}^2)}$$

Δ = difference between the two penguins

Without scaling, grams dominate the distance

Feature	Penguin 1	Penguin 2	Difference	Squared difference
bill_length_mm	39.1	39.5	0.4	0.16
bill_depth_mm	18.7	17.4	1.3	1.69
flipper_length_mm	181	186	5	25
body_mass_g	3750	3800	50	**2,500**
Distance				**√2,526.85 ≈ 50.3**

Table 3. The first two penguins of the dataset. Body mass contributes 2,500 of 2,526.85, about 99% of the squared distance.

Two ways to put features on the same scale

$$z = (x - \bar{x}) \,/\, s \qquad\qquad x' = (x - \min) \,/\, (\max - \min)$$

STANDARDISATION (LEFT)

Mean 0, standard deviation 1

`StandardScaler`: the usual default

$\bar{x}$ and s learned on the training set

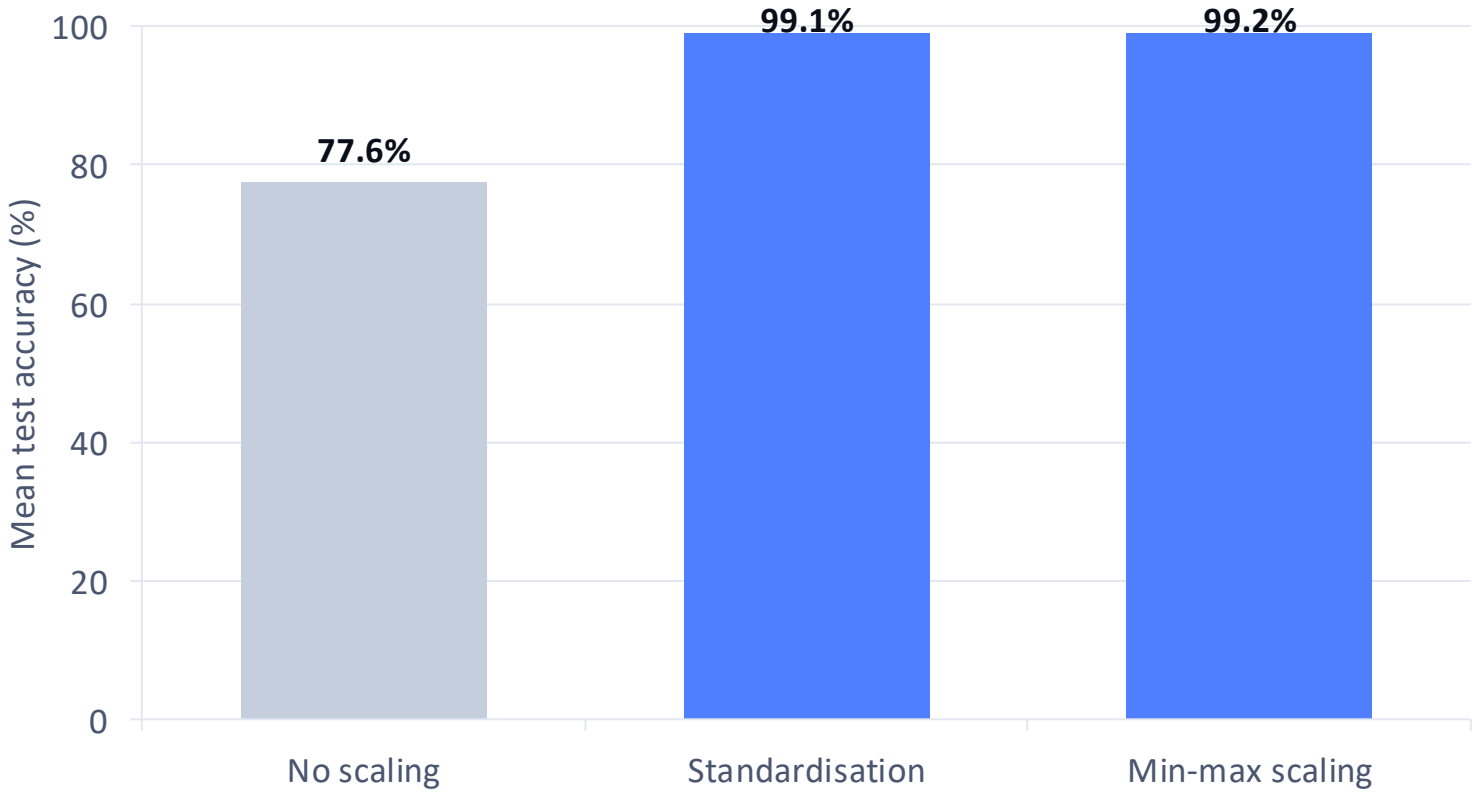
MIN-MAX SCALING (RIGHT)

Values between 0 and 1

`MinMaxScaler`: bounded ranges, images

Sensitive to outliers, like the range

Scaling changes the result



k-nearest neighbours (k = 5), four measurements, 30 stratified 80/20 splits of the penguins. scikit-learn 1.9.1.

MAIN FINDING

Same model, same data: accuracy rises from 77.6% to about 99% once the features are scaled.

Distance-based and linear models need scaling; trees do not.

05

CHAPTER

Pipelines

One object that learns every preparation step from the training set, then applies them everywhere.

fit learns, transform applies

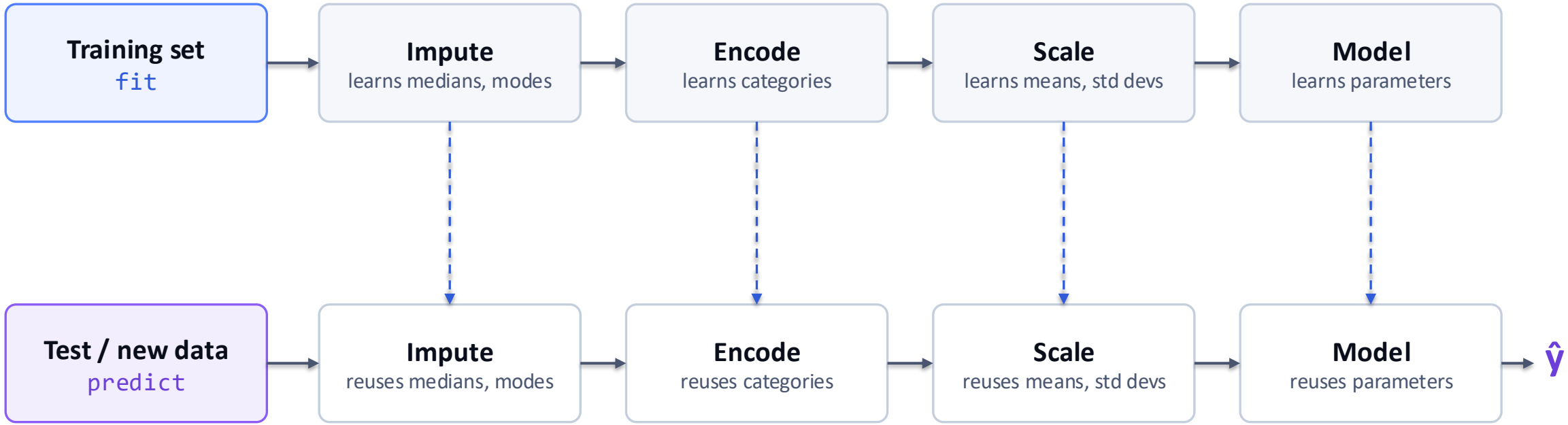


Figure 2. Dashed arrows: values learned during fit on the training set are reused, unchanged, for every later prediction.

ColumnTransformer: different steps for different columns

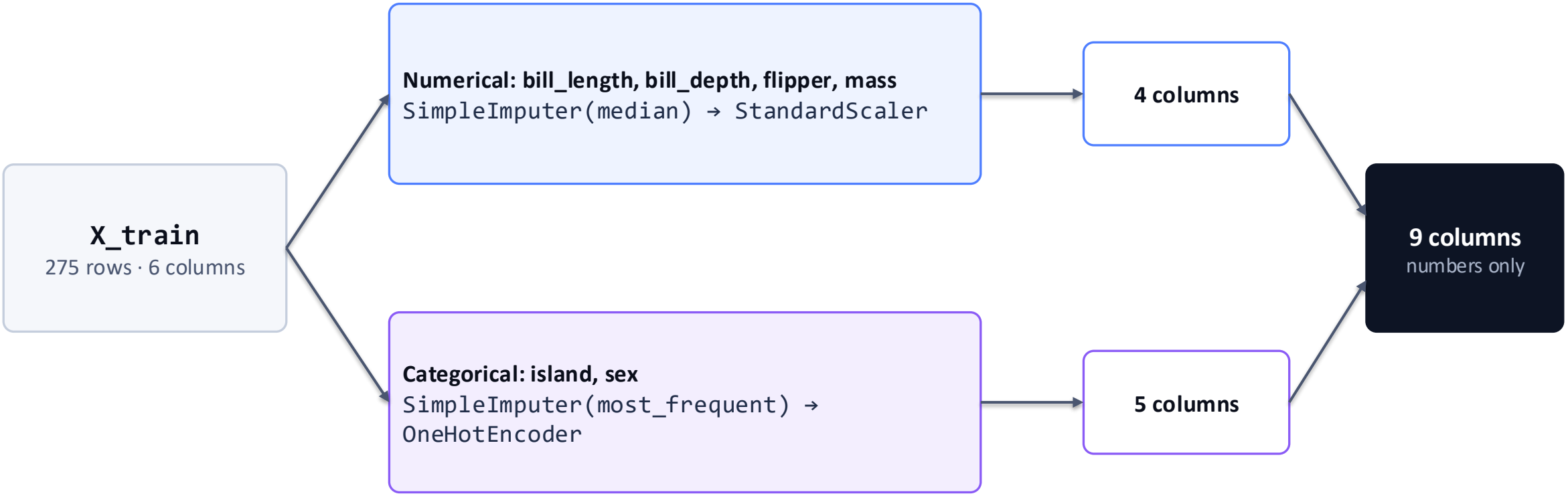


Figure 3. Penguins: 4 numerical + 2 categorical columns become 9 numerical columns ready for the model.

Building the preparation step

PYTHON · SCIKIT-LEARN

```
1 num = ["bill_length_mm", "bill_depth_mm",
2        "flipper_length_mm", "body_mass_g"]
3 cat = ["island", "sex"]
4
5 num_pipe = make_pipeline(
6     SimpleImputer(strategy="median"),
7     StandardScaler())
8 cat_pipe = make_pipeline(
9     SimpleImputer(strategy="most_frequent"),
10    OneHotEncoder(handle_unknown="ignore"))
11
12 prep = ColumnTransformer([
13     ("num", num_pipe, num),
14     ("cat", cat_pipe, cat)])
```

LINES 1-3

Which columns are numbers, which are categories

LINES 5-10

One small pipeline per column type

LINES 12-14

Route each list of columns to its pipeline

Preparation and model in one object

CODE · PYTHON

```python
knn = KNeighborsClassifier(n_neighbors=5)
model = make_pipeline(prep, knn)

model.fit(X_train, y_train)
print(model.score(X_test, y_test))

# shape after preparation
print(prep.fit_transform(X_train).shape)
```

OUTPUT

```
0.9855072463768116
(275, 9)
```

68 of 69 test penguins correctly classified: 98.6%.

Knowledge check

QUESTION

In our pipeline model, where does the StandardScaler learn its means and standard deviations?

A

On the whole dataset, before the split

B

On X_train, when model.fit(X_train, y_train) runs

C

On X_test, every time model.predict is called

D

On X_train and X_test together

EXERCISE

Exercise · A preparation plan on paper

GOAL

For each column of a student dataset, decide: fill? how? encode? how? scale?

TIME

8 min, in pairs

COLUMNS

1. city: 2% missing
2. age: 1% missing
3. bac_mention (Passable … Très bien): 6% missing
4. monthly_data_gb: skewed, a few huge values, 9% missing
5. absences: complete
6. passed (yes / no): the target

DELIVERABLE

A table: column, type, fill, encode, scale. Plus the first line of your code: how do you split?

Recap

KEY TAKEAWAYS

- Split first; stratify on a class target to keep proportions
- Filling gaps changes the statistics; the median is the safe default
- Nominal → one-hot; ordinal → ordered integers
- Scale for distance-based and linear models
- A pipeline fits on training data, then reuses what it learned

NEXT

- Session 03: regression, fitting a line through data
- Homework: Session 02 notebook
- Week 4: first practice session

References

(1) scikit-learn 1.9 User Guide: 6.3 Preprocessing data; 6.4 Imputation of missing values; Pipelines and composite estimators. scikit-learn.org/stable

(2) scikit-learn 1.9 API: train_test_split, SimpleImputer, OneHotEncoder, OrdinalEncoder, StandardScaler, MinMaxScaler, ColumnTransformer. scikit-learn.org/stable/api

(3) Kapoor, S., & Narayanan, A. (2023). Leakage and the reproducibility crisis in machine-learning-based science. Patterns. doi.org/10.1016/j.patter.2023.100804

(4) Diez, D., Çetinkaya-Rundel, M., & Barr, C. D. (2019). OpenIntro Statistics (4th ed.), sections 1.3 (sampling) and 2.1. openintro.org/book/os

(5) James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). An Introduction to Statistical Learning, with Applications in Python. Springer. statlearning.com

(6) Horst, A. M., Hill, A. P., & Gorman, K. B. (2020). palmerpenguins: Palmer Archipelago (Antarctica) penguin data. doi.org/10.5281/zenodo.3960218

SH

Thank you · Questions?

Next session: regression. Finish the Session 02 notebook before Week 4.

Dr. Soufiane Hourri, PhD, HDR

drsoufiane.com