

LECTURE 12

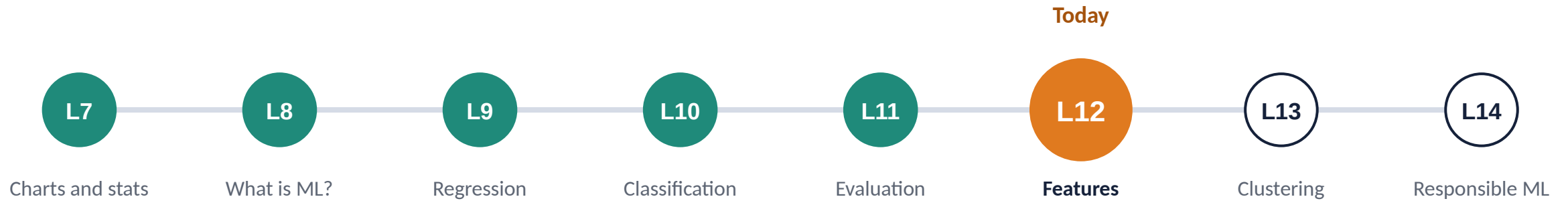
Preparing features

Missing values, categories, scaling and pipelines

Today's key line of code

```
make_pipeline(StandardScaler(),  
              KNeighborsClassifier())
```

The road ahead: machine learning



Today you will

- 1 Find and handle missing values
- 2 Turn categories into numbers with one-hot encoding
- 3 See why scaling changes k-NN, by hand
- 4 Scale without leaking test information
- 5 Build a pipeline that does it all correctly

Meet the data: invented student records

- 150 invented students, made for this course
- Target: passed (yes / no)
- `isna().sum()` counts missing values per column
- Some hours and GPAs are missing, as in real data

Code

```
import pandas as pd
st = pd.read_csv("students.csv")
print(st.head())
print(st.isna().sum())
```

Output

	hours	attendance	prior_gpa	dept	passed
0	NaN	55	3.28	CSE	yes
1	NaN	88	2.37	EEE	yes
2	9.3	81	3.52	Math	yes
3	2.7	83	3.52	EEE	no
4	3.6	78	3.44	CSE	no

hours	8
attendance	0
prior_gpa	6
dept	0
passed	0
dtype:	int64

Missing values: drop or fill?

- `dropna()` removes rows with any missing value
- Fast, but throws data away
- `fillna(median)` keeps rows, guessing a typical value
- Choose, and say which you chose in your report

Code

```
print(len(st), len(st.dropna()))  
filled = st.copy()  
for col in ["hours", "prior_gpa"]:  
    filled[col] = filled[col].fillna(filled[col].median())  
print(filled.isna().sum().sum())
```

Output

```
150 136  
0
```

One-hot encoding: categories become 0/1 columns

A model cannot multiply the word CSE. One-hot encoding gives each category its own yes/no column.

dept	dept_CSE	dept_EEE	dept_Math
CSE	1	0	0
EEE	0	1	0
Math	0	0	1
CSE	1	0	0

One-hot encoding in pandas

`pd.get_dummies` creates the 0/1 columns · `dtype=int` shows 0 and 1 instead of False/True · Every category gets its own column · In pipelines we use `OneHotEncoder` instead (later)

Code

```
dummies = pd.get_dummies(filled["dept"], prefix="dept", dtype=int)
print(dummies.head(4))
```

Output

	dept_CSE	dept_EEE	dept_Math
0	1	0	0
1	0	1	0
2	0	0	1
3	0	1	0

By hand: why scale matters for k-nearest neighbours

Distance from student A (hours 2, attendance 90) to B (hours 10, attendance 85) and C (hours 3, attendance 60).

Code

```
# Unscaled: attendance dominates.  
#   C looks far, B looks close, although  
#   B studies 8 hours more than A!  
# After scaling both columns to similar ranges,  
#   hours matter again and C is nearer.
```

pair	hours gap	attendance gap	distance
A to B	8	5	9.4
A to C	1	30	30.0
A to B, scaled	0.67	0.08	0.67
A to C, scaled	0.08	0.50	0.51

Click to reveal the next row. Predict it first!

Watch it: scaling changes the neighbours

WATCH FOR

- 14 real rows from students.csv; the star is one student
- Unscaled: neighbours chosen almost only by attendance
- The scaler stretches hours to a fair spread
- Scaled: the three nearest neighbours change

FEATURES, STEP BY STEP

Scaling changes who the nearest neighbours are

14 students from students.csv. Student Q is the star.
Which 3 students are nearest? It depends on scaling.

Click the video to play, click again to pause. Silent: talk the class through it.

StandardScaler: mean 0, standard deviation 1

Each column: subtract its mean, divide by its std · Every feature now has a similar range · Fit the scaler on training data only · Then transform both training and test data

Code

```
from sklearn.preprocessing import StandardScaler
num = filled[["hours", "attendance", "prior_gpa"]]
scaled = StandardScaler().fit_transform(num)
print(num.mean().round(1).to_dict())
print(scaled.mean(axis=0).round(2) + 0.0, scaled.std(axis=0).round(2))
```

Output

```
{'hours': 6.1, 'attendance': 70.0, 'prior_gpa': 3.0}
[0.  0.  0.] [1.  1.  1.]
```

Data leakage: the silent killer

1

What it is

Information from the test set sneaks into training, so scores look better than reality.

2

Common causes

Scaling or filling with statistics of ALL data; choosing settings on the test set; duplicate rows.

3

The fix

Split first. Fit every preparation step on training data only. Pipelines do this for you.

Watch it: leakage happening, then prevented

WATCH FOR

- Wrong: the scaler learns from all rows, test included
- Splitting afterwards is too late
- Right: split first, lock the test set away
- The pipeline fits on training rows only

FEATURES, STEP BY STEP

Leakage happening, then prevented

Two ways to scale the same data.

One lets test information sneak into training.

Click the video to play, click again to pause. Silent: talk the class through it.

A pipeline does every step, in the right order

Numbers: fill missing with the median, then scale · Categories: one-hot encode · Then the model · Inside cross-validation, each step is fitted on training folds only

Code

```
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
X = st[["hours", "attendance", "prior_gpa", "dept"]]
y = st["passed"]
prep = ColumnTransformer([
    ("num", make_pipeline(SimpleImputer(strategy="median"),
                          StandardScaler()),
     ["hours", "attendance", "prior_gpa"]),
    ("cat", OneHotEncoder(), ["dept"])]
model = make_pipeline(pre, KNeighborsClassifier())
print(round(cross_val_score(model, X, y, cv=5).mean(), 3))
```

Output

0.893

Does scaling really matter? Measure it.

- Same pipeline, with and without the scaler
- Same data, same cross-validation
- k-NN gains a lot from scaling
- A decision tree would barely change (try it)

Code

```
def build(scale):
    num_steps = [SimpleImputer(strategy="median")]
    if scale:
        num_steps.append(StandardScaler())
    prep = ColumnTransformer([
        ("num", make_pipeline(*num_steps),
         ["hours", "attendance", "prior_gpa"]),
        ("cat", OneHotEncoder(), ["dept"])]
    return make_pipeline(pre, KNeighborsClassifier())
for scale in [False, True]:
    s = cross_val_score(build(scale), X, y, cv=5).mean()
    print("scaled" if scale else "unscaled", round(s, 3))
```

Output

```
unscaled 0.687
scaled 0.893
```

Your turn

10 minutes

1

Revisit Practice 1, task B2: k-NN on wine

2

Put StandardScaler and KNeighborsClassifier in a pipeline

3

Compare 5-fold CV accuracy with and without scaling

4

How big is the difference?

Code

```
from sklearn.datasets import load_wine
Xw, yw = load_wine(return_X_y=True)
plain = KNeighborsClassifier()
scaled = make_pipeline(StandardScaler(), KNeighborsClassifier())
...
```

Starter code: replace the ... with your own code

Common feature-preparation mistakes

Read the last line of an error first. It names the problem.

What you typed	What Python says	What it means
<pre>KNeighborsClassifier().fit(filled[["hours", "dept"]], filled["passed"])</pre>	<pre>ValueError: could not convert string to float: 'CSE'</pre>	Text column: one-hot encode dept first.
<pre>KNeighborsClassifier().fit(st[["hours"]], st["passed"])</pre>	<pre>ValueError: Input X contains NaN.</pre>	Missing values: fill or drop them first (SimpleImputer).
<pre>OneHotEncoder().fit_transform(st["dept"])</pre>	<pre>ValueError: Expected a 2-dimensional container but got <class 'pandas.Series'> instead. Pass a DataFrame containing a si ...</pre>	Encoders want a table: use <code>st[["dept"]]</code> with double brackets.

Interactive lab: Feature Prep Lab

In the lab you can

1 Missing values

Drop, median, mean or 0: see the data and the CV score.

2 Categories to numbers

One-hot versus 1, 2, 3: the distances a model sees.

3 Scaling and neighbours

Click a student; the 3 nearest change after scaling.

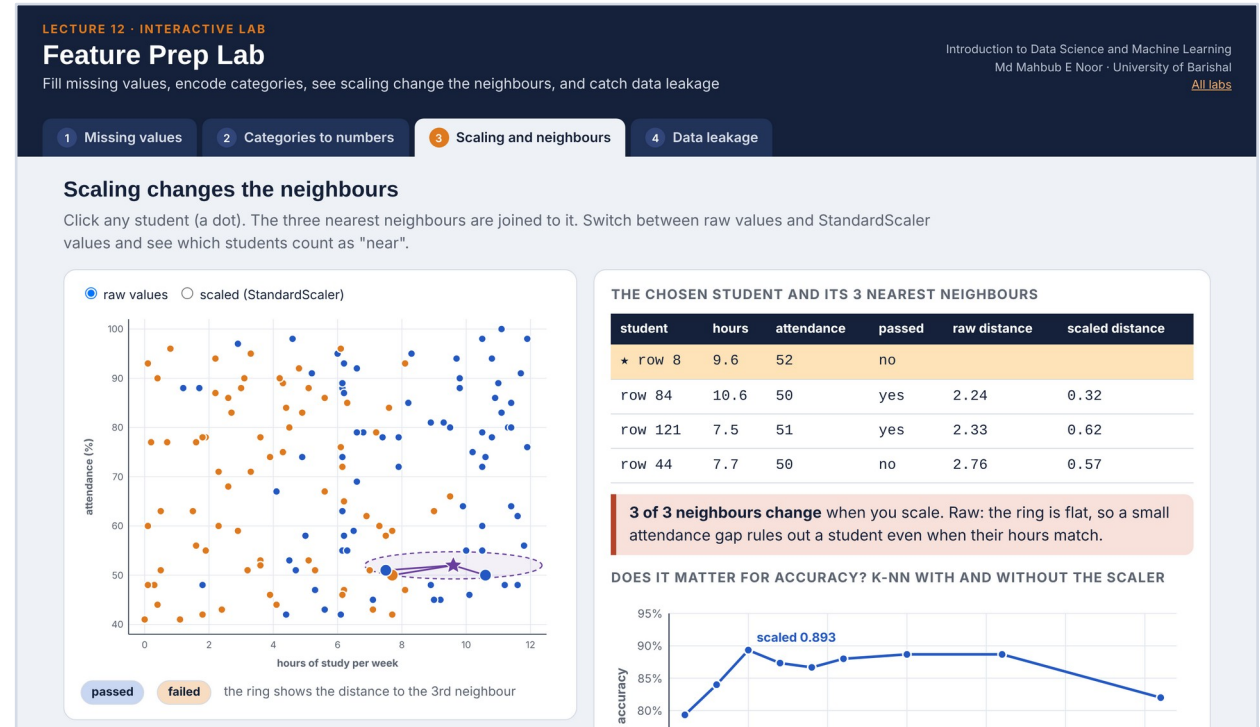
4 Data leakage

Over 90% on coin tosses in the wrong order; spot the leak.

Open the lab ►

Works offline in any browser

Labs_open_in_browser/
L12_Feature_Prep_Lab.html



Today's toolkit

<code>df.isna().sum()</code>	Count missing values
<code>fillna</code> / <code>SimpleImputer</code>	Fill missing values
<code>pd.get_dummies</code> / <code>OneHotEncoder</code>	Categories to 0/1 columns
<code>StandardScaler()</code>	Mean 0, std 1

<code>ColumnTransformer</code>	Different steps per column
<code>make_pipeline(...)</code>	Chain steps and model
<code>leakage</code>	Test info sneaking into training

Homework 12

Due before Lecture P2 • [date]

1 By hand

Scale [2, 4, 6] with StandardScaler by hand: subtract the mean, divide by the standard deviation (divide by count). Check in code.

2 Wine with scaling

Choose k for a scaled k-NN pipeline on wine with cross-validation, then report test accuracy once.

3 Tree vs k-NN

On students.csv, compare the pipeline with k-NN and with a decision tree, each scaled and unscaled. Table of 4 CV scores.

4 Leakage hunt

Describe one way leakage could happen in a project of your choice, and how a pipeline prevents it.

5 Explain

Why is one-hot encoding better than numbering categories 1, 2, 3?

Submit on [LMS / Google Classroom]: your notebook file HW12_<your ID>.ipynb, or a Colab link set to “Anyone with the link can view”.

Recap: Lecture 12 in 90 seconds

RECAP

Lecture 12 in 90 seconds

A short recap of today's lecture. Pause any time.

Click to play or pause. Watch it again at home to revise.