

LECTURE 9

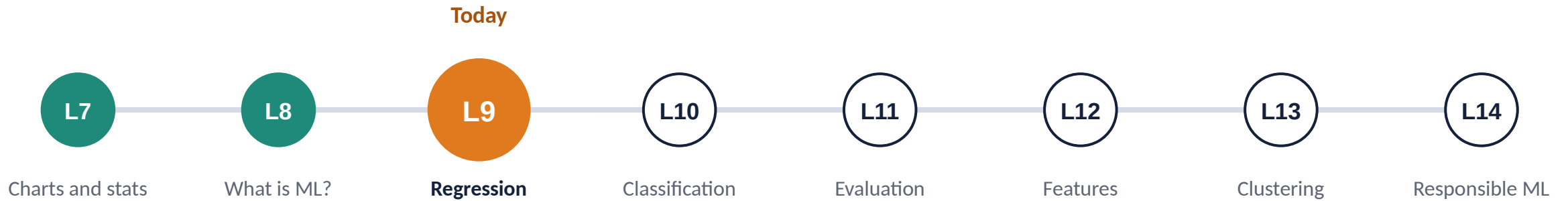
Regression

Predicting numbers with a straight line

Today's key line of code

```
model = LinearRegression()  
model.fit(X, y)  
model.predict([[6]])
```

The road ahead: machine learning



Today you will

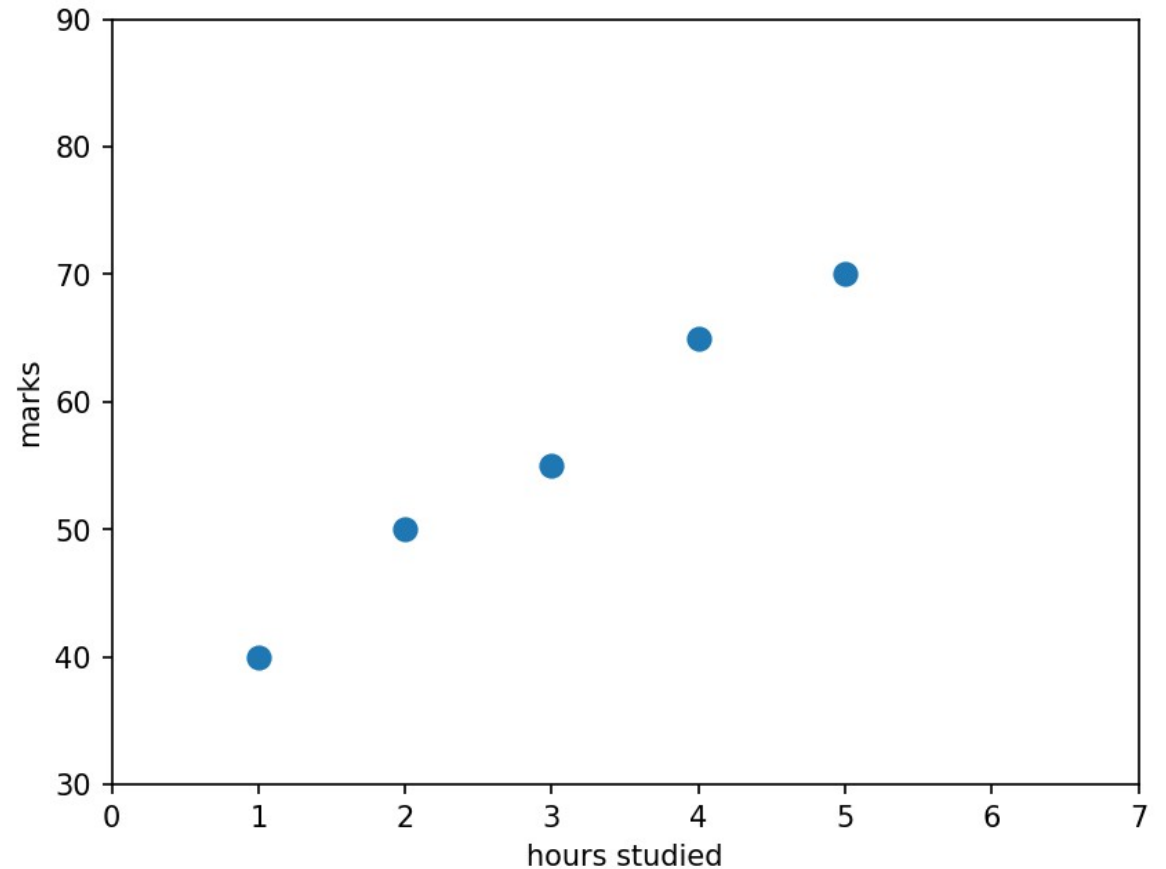
- 1 Recognise regression problems
- 2 Read a line: slope and intercept
- 3 Measure error with MAE by hand
- 4 Find the best line by hand, then with LinearRegression
- 5 Evaluate on real data with MAE and R^2 , and know when a line misleads

Five students, one question

Invented data: hours studied per week and exam marks. What mark would we expect for 6 hours?

Code

```
import matplotlib.pyplot as plt
hours = [1, 2, 3, 4, 5]
marks = [40, 50, 55, 65, 70]
plt.scatter(hours, marks, s=60)
plt.xlabel("hours studied")
plt.ylabel("marks")
plt.xlim(0, 7)
plt.ylim(30, 90)
plt.show()
```



Anatomy of a straight line

Code

```
slope = 7.5  
intercept = 33.5  
h = 6  
predicted = slope * h + intercept  
print(predicted)
```

Slope: marks gained per extra hour

Intercept: predicted marks at 0 hours

Slope times the input (6 hours)

plus the starting point

Output

78.5

By hand: how wrong is a guessed line?

Try the line marks = $8 \times \text{hours} + 30$. Error = actual - predicted. MAE = average of the error sizes.

Code

```
# guessed line:  
# 8 * hours + 30  
# MAE = (2+4+1+3+0) / 5  
# = 2.0 marks
```

hours	actual	predicted	error	error
1	40	38	2	2
2	50	46	4	4
3	55	54	1	1
4	65	62	3	3
5	70	70	0	0
			MAE	2.0

Click to reveal the next row. Predict it first!

By hand: the best line (least squares)

slope = sum of $(h - \text{mean } h)(m - \text{mean } m)$ / sum of $(h - \text{mean } h)^2$. Means: hours 3, marks 56.

Code

```
# slope = 75 / 10 = 7.5
# intercept = 56 - 7.5 * 3
#           = 33.5
# best line:
# marks = 7.5 * hours + 33.5
```

h	m	h - 3	m - 56	product	$(h - 3)^2$
1	40	-2	-16	32	4
2	50	-1	-6	6	1
3	55	0	-1	0	0
4	65	1	9	9	1
5	70	2	14	28	4
			sum	75	10

Click to reveal the next row. Predict it first!

The same line with scikit-learn

- X must be a table: one row per student, so `[[1], [2], ...]`
- `fit` finds the slope (`coef_`) and intercept
- Same numbers as our hand calculation
- `predict` answers new questions: 6 hours gives 78.5

Code

```
from sklearn.linear_model import LinearRegression
X = [[1], [2], [3], [4], [5]]
y = [40, 50, 55, 65, 70]
model = LinearRegression()
model.fit(X, y)
print(model.coef_, model.intercept_)
print(model.predict([[6]]))
```

Output

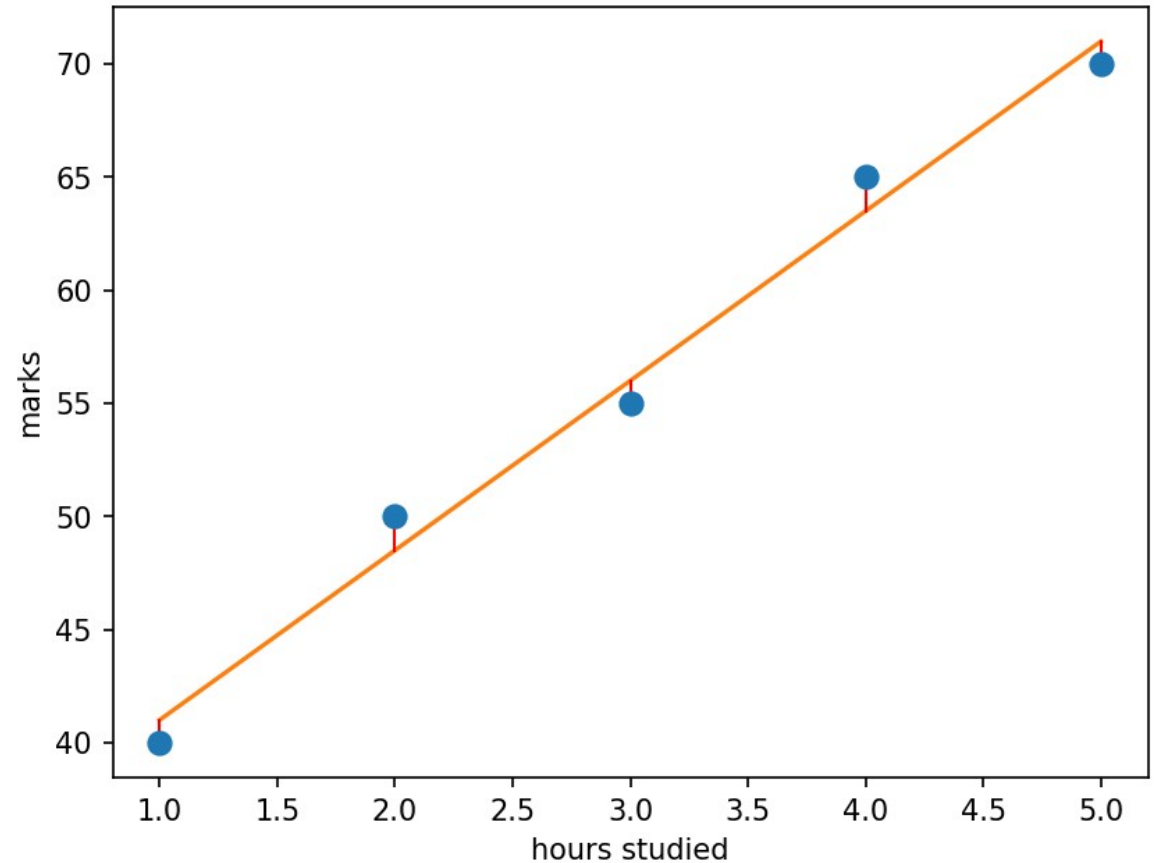
```
[7.5] 33.5
[78.5]
```

Draw the fitted line

The line passes as close as possible to all points at once. The red segments are the errors it tries to keep small.

Code

```
pred = model.predict(X)
plt.scatter(hours, marks, s=60, zorder=3)
plt.plot(hours, pred, color="tab:orange")
for h, m, p in zip(hours, marks, pred):
    plt.plot([h, h], [m, p], color="red", lw=1)
plt.xlabel("hours studied")
plt.ylabel("marks")
plt.show()
```



Watch it: searching for the best line

WATCH FOR

- Each frame tries a different line
- Red segments: the errors
- MAE goes down as the line improves
- The search ends at slope 7.5, intercept 33.5

REGRESSION, STEP BY STEP

Searching for the best line

The computer tries a line, measures the error, and adjusts.
Watch the red error segments shrink.

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

Real data: diabetes progression

- 442 patients, built into scikit-learn (offline)
- Feature: body mass index (bmi), already scaled
- Target: disease progression one year later
- Real data is noisy: errors are much larger

Code

```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
data = load_diabetes(as_frame=True)
X = data.data[["bmi"]]
y = data.target
X_tr, X_te, y_tr, y_te = train_test_split(
    X, y, test_size=0.25, random_state=0)
model = LinearRegression().fit(X_tr, y_tr)
pred = model.predict(X_te)
print(round(mean_absolute_error(y_te, pred), 1))
```

Output

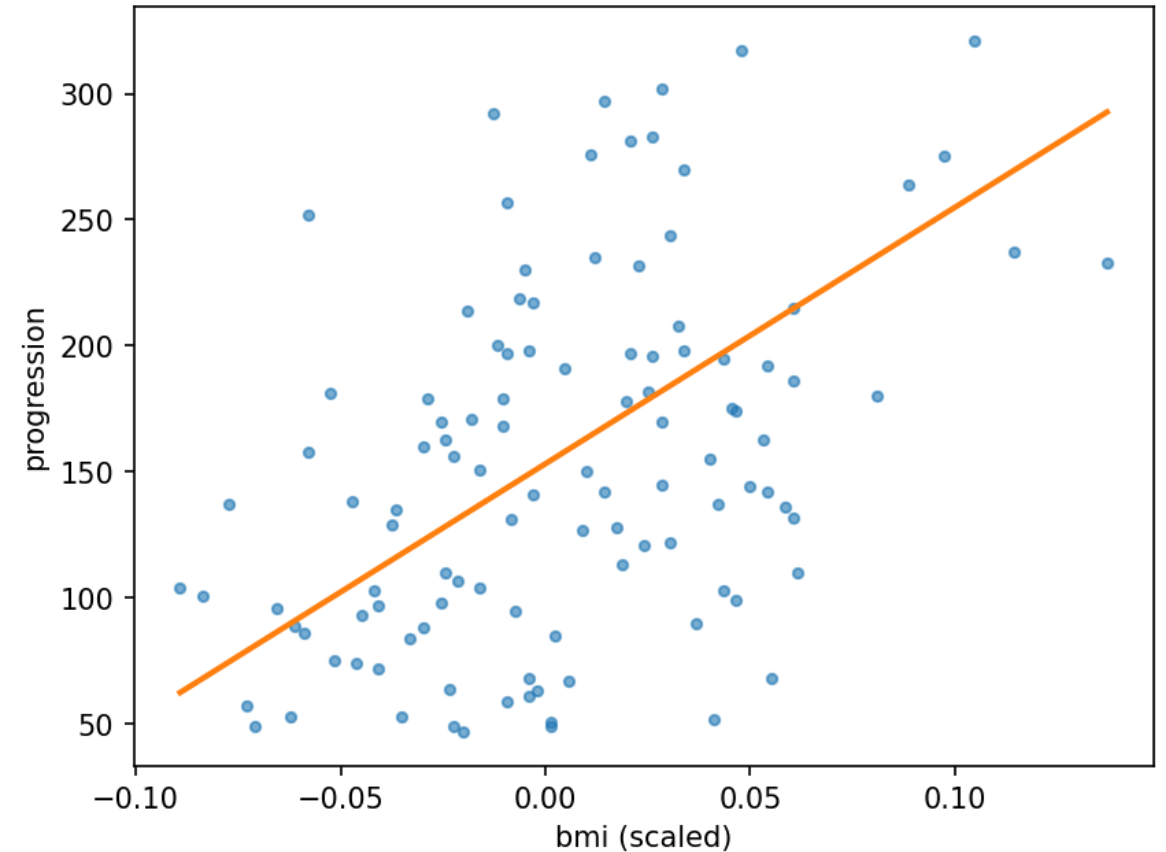
53.3

See why: one feature, lots of noise

The trend is real, but patients with the same bmi differ a lot. A line can only capture the average trend.

Code

```
plt.scatter(X_te["bmi"], y_te, s=12, alpha=0.6)
xs = X_te.sort_values("bmi")
plt.plot(xs["bmi"], model.predict(xs),
         color="tab:orange", lw=2)
plt.xlabel("bmi (scaled)")
plt.ylabel("progression")
plt.show()
```



All 10 features, and two more ways to score

- MAE: average error size, in target units
- R^2 : share of variation explained (1 = perfect, 0 = no better than the mean)
- More features: lower MAE, higher R^2
- Always compare on the test set

Code

```
from sklearn.metrics import r2_score
for cols in ["bmi", list(data.data.columns)]:
    X = data.data[cols]
    X_tr, X_te, y_tr, y_te = train_test_split(
        X, y, test_size=0.25, random_state=0)
    m = LinearRegression().fit(X_tr, y_tr)
    p = m.predict(X_te)
    print(len(cols), "features: MAE",
          round(mean_absolute_error(y_te, p), 1),
          "R2", round(r2_score(y_te, p), 2))
```

Output

```
1 features: MAE 53.3 R2 0.16
10 features: MAE 45.1 R2 0.36
```

What if someone studies 20 hours?

Predict first

Code

```
X = [[1], [2], [3], [4], [5]]  
y = [40, 50, 55, 65, 70]  
small = LinearRegression().fit(X, y)  
print(small.predict([[20]]))
```



Use our 5-student model to predict marks for 20 hours of study. Is the answer sensible?

Answer: What if someone studies 20 hours?

Code

```
X = [[1], [2], [3], [4], [5]]  
y = [40, 50, 55, 65, 70]  
small = LinearRegression().fit(X, y)  
print(small.predict([[20]]))
```

Output

```
[183.5]
```

- The line predicts 183.5 marks out of 100
- Lines continue forever; real limits do not
- We only had data from 1 to 5 hours
- Never trust predictions far outside the training data

Your turn

10 minutes

- 1 Use the diabetes data with the feature **bp** (blood pressure) instead of bmi
- 2 Fit LinearRegression on the training part
- 3 Print MAE and R^2 on the test part
- 4 Is bp a better single feature than bmi?

Code

```
X = data.data[["bp"]]
X_tr, X_te, y_tr, y_te = train_test_split(
    X, y, test_size=0.25, random_state=0)
...
```

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

Common regression mistakes

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

What you typed	What Python says	What it means
<pre>LinearRegression().fit([1, 2, 3], [4, 5, 6])</pre>	<pre>ValueError: Expected 2D array, got 1D array instead:</pre>	X must be 2-D: [[1], [2], [3]]. Python says so in the message.
<pre>LinearRegression().fit([[1], [2]], [4, 5, 6])</pre>	<pre>ValueError: Found input variables with inconsistent numbers of samples: [2, 3]</pre>	X and y have different numbers of rows.
<pre>LinearRegression().predict([[6]])</pre>	<pre>NotFittedError: This LinearRegression instance is not fitted yet. Call 'fit' with appropriate arguments before using this estimator.</pre>	The model was never fitted. Call fit first.

Interactive lab: Regression Lab

In the lab you can

1 Fit by hand

Move slope and intercept; watch MAE, MSE and the errors.

2 Outliers and extrapolation

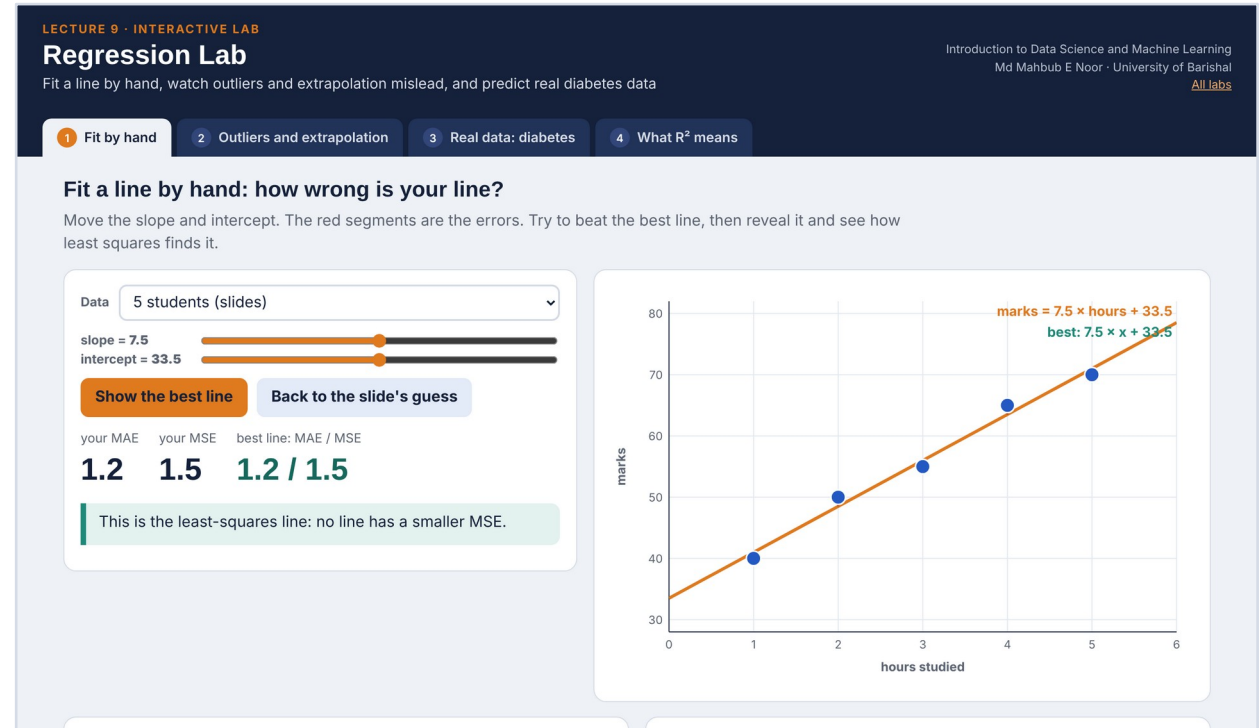
One odd student swings the line; 20 hours gives 183.5.

3 Real data: diabetes

Every single feature ranked by test MAE and R^2 .

4 What R^2 means

Errors drawn as squares: your line against the average.



Open the lab ►

Works offline in any browser

Labs_open_in_browser/
L9_Regression_Lab.html

Today's toolkit

$y = \text{slope} \cdot x + \text{intercept}$	A straight line
MAE	Average error size
R^2	Share of variation explained
<code>LinearRegression()</code>	Fits the best line

<code>model.coef_</code>	Slope(s)
<code>model.intercept_</code>	Intercept
<code>mean_absolute_error(y, p)</code>	MAE in code
<code>r2_score(y, p)</code>	R^2 in code

Homework 9

Due before Lecture 10 • [date]

1 By hand

For hours 2, 4, 6 and marks 50, 60, 76: compute the best slope and intercept with the table method. Check with `LinearRegression`.

2 MAE by hand

For your line, compute the three errors and the MAE by hand.

3 Real data

On diabetes, try three single features of your choice. Report MAE and R^2 for each in a small table.

4 Extrapolation

Give one real-life example where a line would make an impossible prediction far outside the data.

5 Plot

Plot your best single feature against the target with the fitted line, as in the lecture.

Submit on [LMS / Google Classroom]: your notebook file `HW9_<your ID>.ipynb`, or a Colab link set to “Anyone with the link can view”.

Recap: Lecture 9 in 90 seconds

RECAP

Lecture 9 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.