PARCOURS : DATA SCIENTIST

Model Interpretability with SHAP

Open the black box — understand why your model predicts what it predicts

PARCOURS : DATA SCIENTIST

DURATION: ~90 MIN

Liora

ABOUT THIS MASTERCLASS

What you will learn

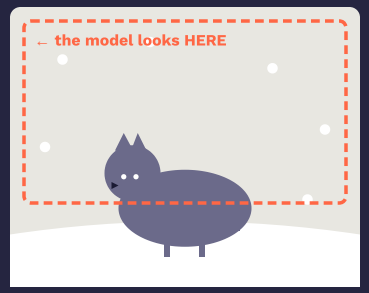
LEARNING OBJECTIVES

- Explain **why interpretability matters** — beyond accuracy alone
- Use **classic techniques** (trees, coefficients, PCA) and know their limits
- Apply **SHAP** to tree models for global **and** local explanations
- Read **summary, dependence, and force plots** with confidence
- Extend SHAP to a **multiclass text classification** setting
- Run an **independent SHAP analysis** on your own dataset

Liora

THE PROBLEM

A 98% accurate model can still be wrong



THE CLASSIC STORY

A wolf vs. husky classifier reaches **98% accuracy**. Great model? It actually looks at **the snow in the background**, not the animal.

THE LESSON

High performance is **not enough**. Interpretability is the bridge between performance and **trust**.

WHY IT MATTERS

Four reasons to open the black box

1 · TRUST

Stakeholders won't deploy what they can't understand.

2 · DEBUGGING

Spot shortcuts, leakage, spurious correlations.

3 · FAIRNESS

Audit sensitive features like gender, age, ethnicity.

4 · REGULATION

GDPR, EU AI Act, credit scoring: a right to explanation.

> Two questions, always: **Is the model effective?** And **is its reasoning acceptable?**

Liora

ROADMAP

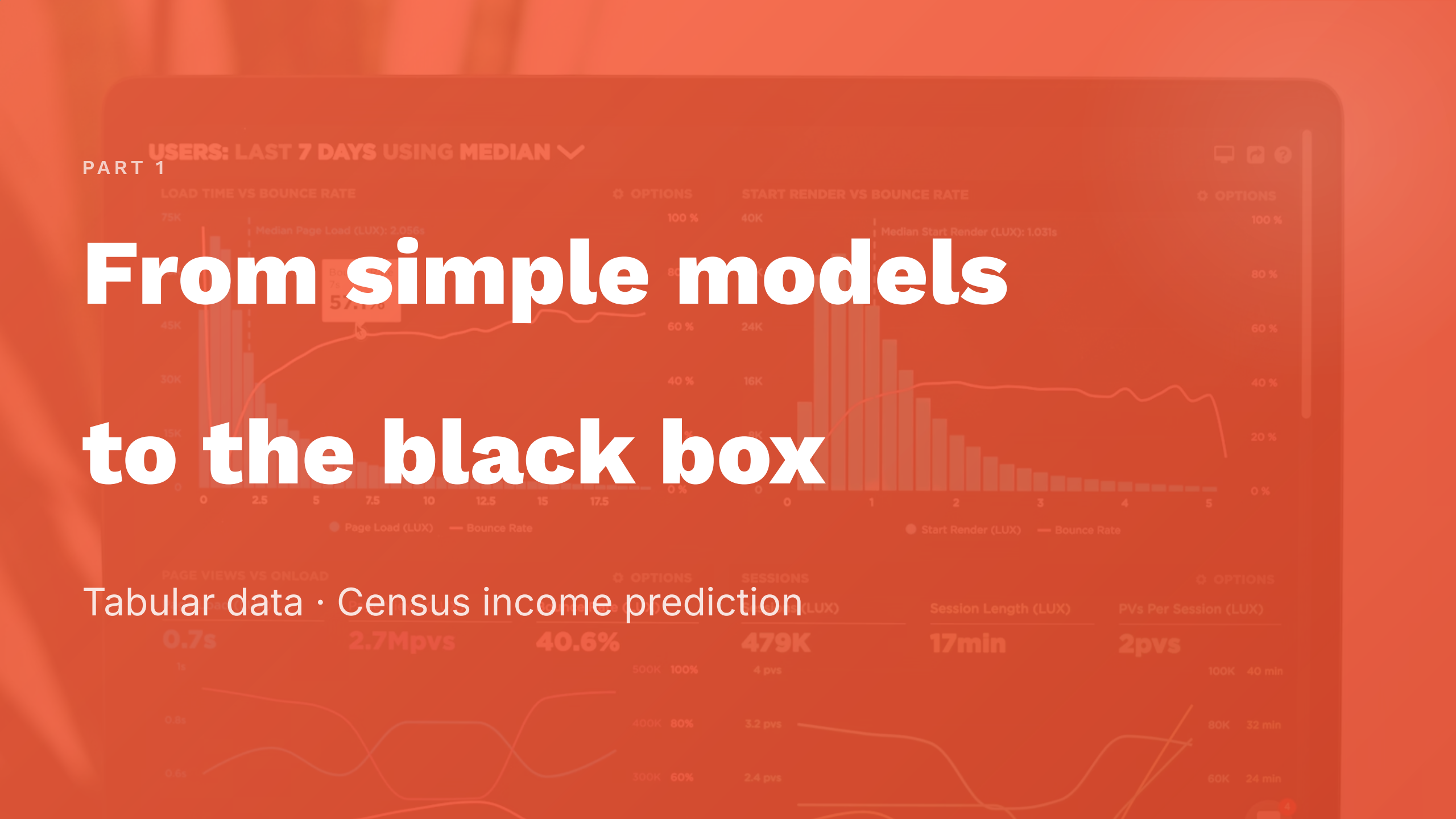
Our three-part journey

PART	QUESTION WE'LL ANSWER	DATA	MAIN TOOL
Part 1	How far can classic interpretability take us?	Census (tabular)	Tree, coefficients, PCA, SHAP
Part 2	Does the same logic work on text and multiclass?	Emotions (text)	Multiclass SHAP
Part 3	Can you run the workflow alone?	Your dataset	Independent SHAP analysis

PART 1

From simple models to the black box

Tabular data · Census income prediction



THE DATASET

Predicting US Census income

Target

Does this person earn **> $50K** per year? → binary classification

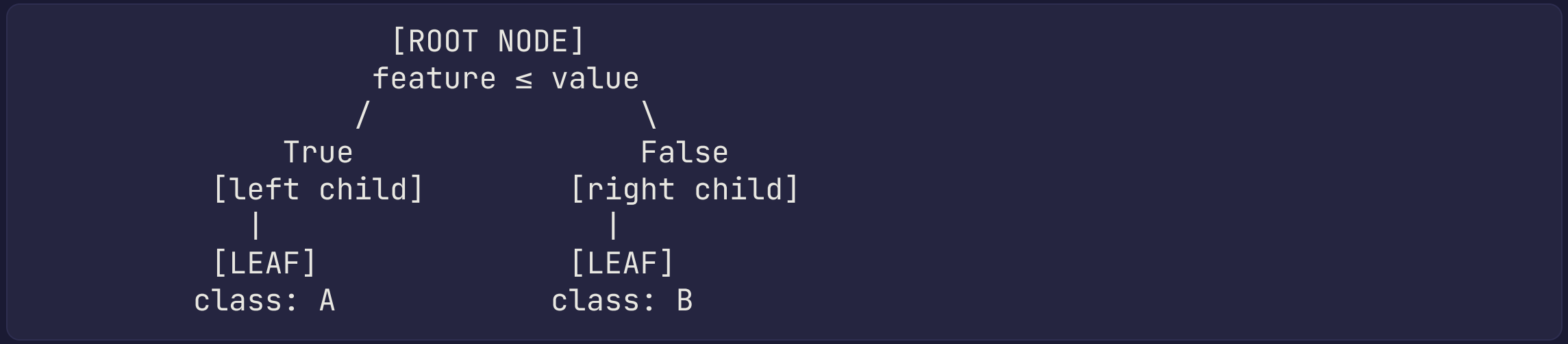
Features

age · education · marital status · occupation · gender · capital gain/loss · hours worked · native country

WHY THIS DATASET?

Features are **human-readable**. When the model highlights **marital-status** or **hours-per-week**, we can immediately judge whether the pattern is reasonable — or worth auditing.

Reading a decision tree



gini — impurity · 0 = pure leaf, 0.5 = mixed

samples — how many training rows reached this node

value — class counts → predicted **majority class**

In the notebook → `DecisionTreeClassifier(max_depth=3, random_state=42)` — depth 3 to stay readable.

The feature at the root is the strongest first separator in the data.

INTERPRETABLE BY DESIGN · 2

What the tree tells us (and hides)

WHAT IT GIVES US

- **Global view** — main decision rules
- **Local view** — one person's path
- **Full transparency** — read line by line

WHAT IT HIDES

- Rich **interactions** between many variables
- Strong **predictive performance**
- A method that **transfers** to non-tree models

> Readable only because it's shallow. Grow the tree, lose the interpretability.

INTERPRETABLE BY DESIGN · 3

Feature importance & coefficients

TREE IMPORTANCE

Split frequency × impurity reduction.

Limit — tied to the fitted tree, unstable.

LINEAR COEFFICIENTS

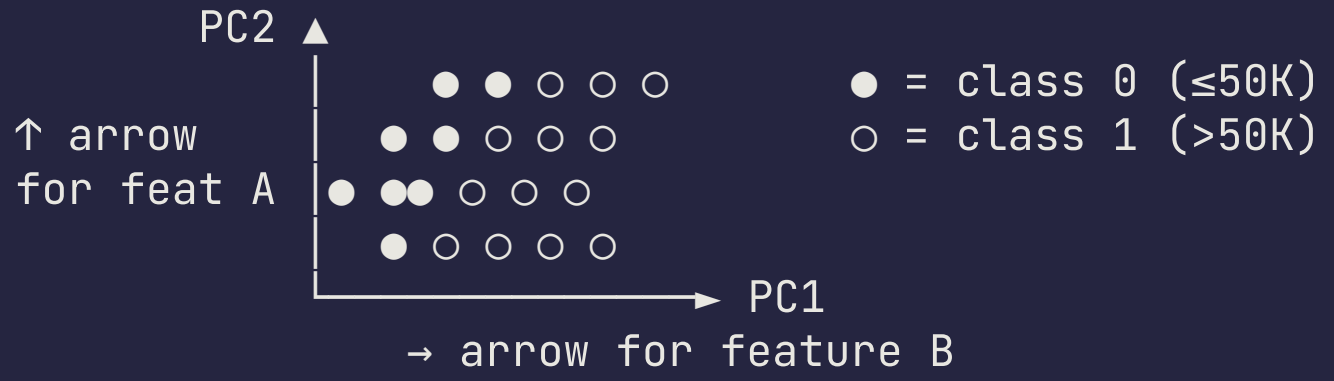
Positive → class 1 · negative → class 0.

Limit — assumes linearity, scale-sensitive.

⚠ A sensitive feature like gender_Female near the top = signal to audit the model.

In the notebook → `tree_clf.feature_importances_` · `pipeline['logistic_regression'].coef_[0]` (with `StandardScaler`).

PCA biplot — a global map



Points — one per person

Arrow direction — where the feature is high

Arrow length — influence on the 2D projection

PCA is excellent for exploration. But projection loses information — never a final explanation.

Performance vs. interpretability

MODEL	F1 SCORE	INTERPRETABILITY
Decision tree (depth 3)	~0.75	Readable rule set
Logistic regression	~0.78	Coefficients directly inspectable
XGBoost	~0.81	Not directly inspectable

THE PROBLEM

Better performance → less readable. No single tree to look at. No coefficient table. Reasoning is **distributed across hundreds of trees**.

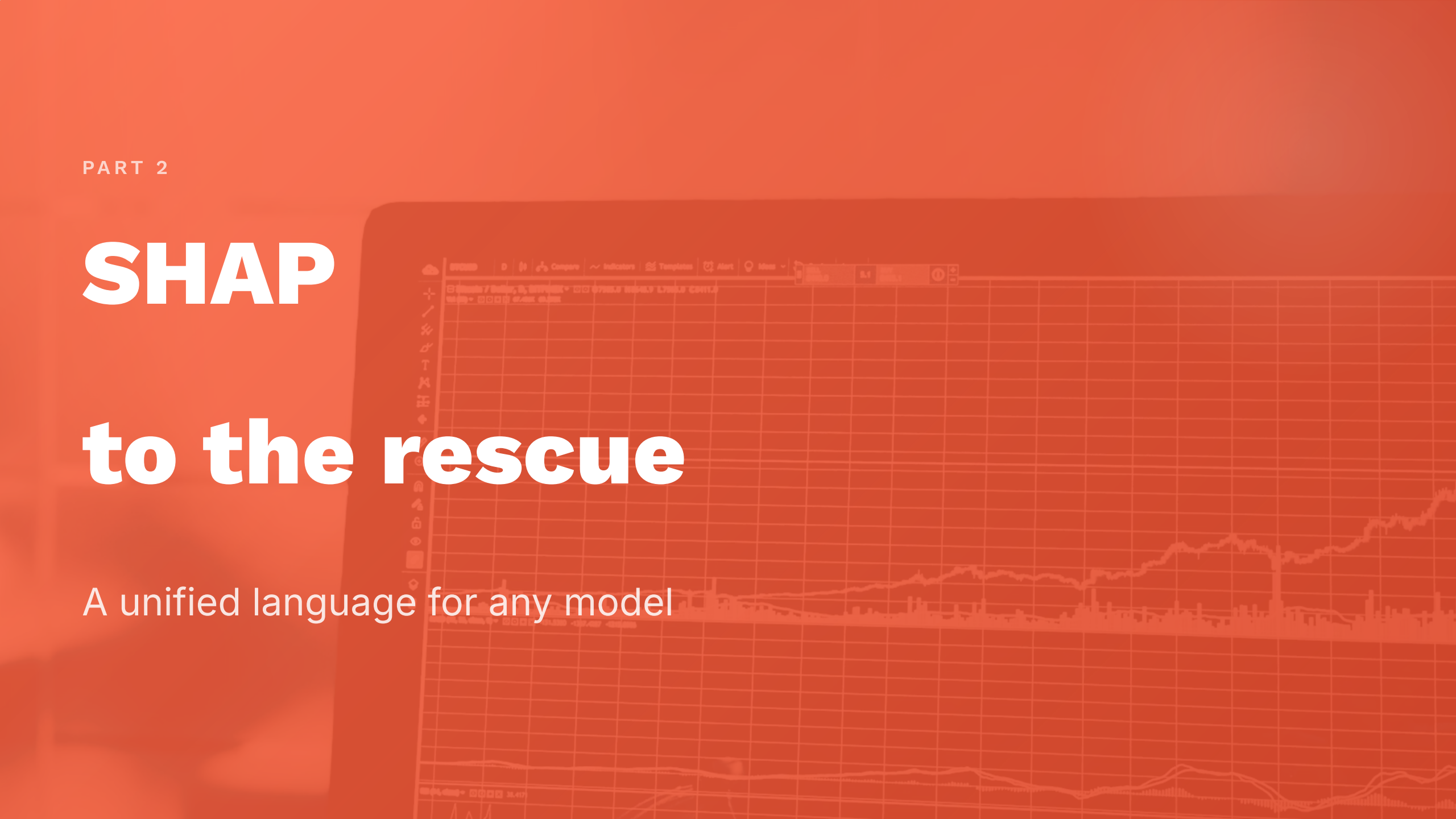
In the notebook → `xgb.XGBClassifier(eval_metric='logloss', random_state=42)` — 100 trees by default, `max_depth=6`. This is the model we'll explain with SHAP.

PART 2

SHAP

to the rescue

A unified language for any model



SHAP in one mental picture

THE QUESTION SHAP ANSWERS

For this prediction, how much did each feature contribute, relative to a baseline?

```
baseline prediction (expected value)
      + contribution from feature 1
      + contribution from feature 2
      + contribution from feature 3
      + ...
      = model output for this person
```

Think of SHAP as a prediction-decomposition tool. It splits one prediction into additive pieces you can read.

THE FOUNDATION

Why SHAP is trustworthy

GAME THEORY

Built on **Shapley values** — a Nobel-prize idea for fairly splitting a team's reward among its players.

FAIR PROPERTIES

Efficiency · Symmetry · Dummy · Additivity — these **four axioms** uniquely define SHAP.

MODEL-AGNOSTIC

Works for trees, linear models, neural nets — with an optimized `TreeExplainer` for XGBoost.

THE WORKFLOW

Four steps to apply SHAP

1. **Train your model**

   Any model — but tree boosters (XGBoost, LightGBM) pair with a fast `TreeExplainer`.

2. **Create the explainer**

   `explainer = shap.TreeExplainer(xgb_model, data=X_train, model_output="probability")` · `data=X_train` is the **background set** — the mean prediction over it becomes the expected value.

3. **Compute SHAP values**

   `shap_values = explainer.shap_values(X_test)` → a matrix **(n_samples, n_features)**

4. **Visualize & interpret**

   Bar plot · beeswarm · dependence · force — we'll read all four next.

Liora

OUTPUT STRUCTURE

The SHAP values matrix

	feature 1	feature 2	feature 3	...
person 1	+0.12	-0.03	+0.01	
person 2	-0.08	+0.10	0.00	
person 3	+0.02	-0.01	-0.05	

ROWS

One per person in the test set

COLUMNS

One per feature

SIGN

+ pushes up · − pushes down

Near zero → the feature barely influenced this particular prediction.

THE BASELINE

Expected value = average prediction

The **expected value** is simply the **model's average prediction** over the background data — `E[f(X)] = np.mean(model.predict(X_background))`.

CENSUS EXAMPLE · BINARY

```
model.predict_proba(X_train)[:,1].mean()
≈ 0.24   ← expected_value
(~24% earn >50K in X_train)
```

How to read it — before seeing **one** person's features, the model would start from **0.24**. SHAP values measure the distance between this average and the **actual** prediction for that person.

```
expected_value + Σ shap_values[person] = model.predict_proba(person)
```

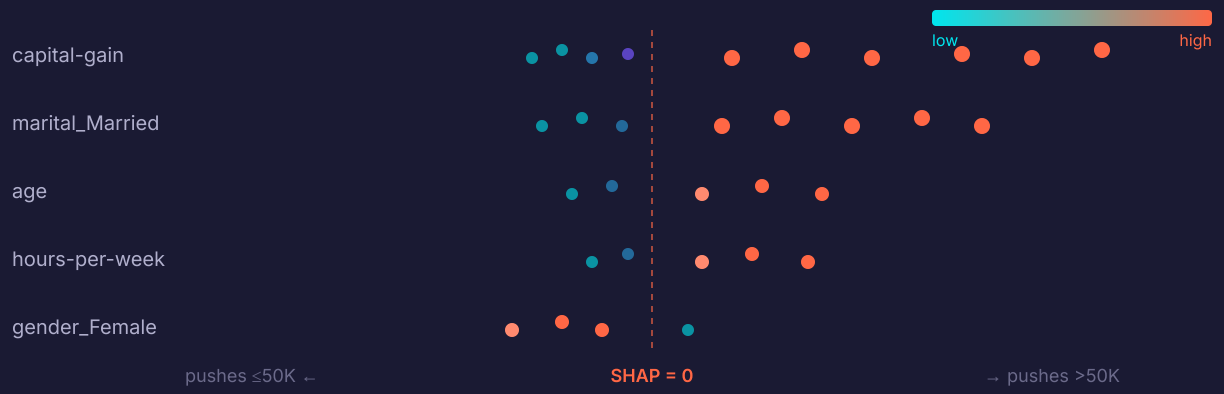
GLOBAL VIEW · 1

The bar plot — what matters on average

Each bar = **mean |SHAP|** — influence with no direction

⚠ Ranks features — **does not** say whether they push toward ≤50K or >50K

The beeswarm — matters how and for whom



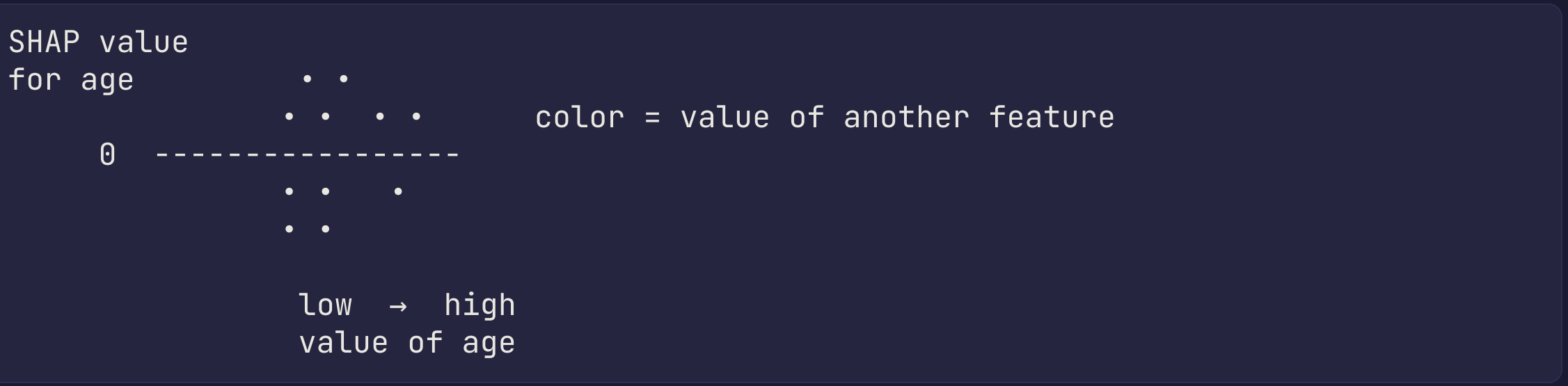
Each dot = one person

X position = SHAP value

Color = low (cyan) / high (orange)

High capital-gain → >50K. gender_Female slightly pushes toward ≤50K — worth an audit.

The dependence plot — how one feature behaves



Upward trend — higher feature value raises the prediction

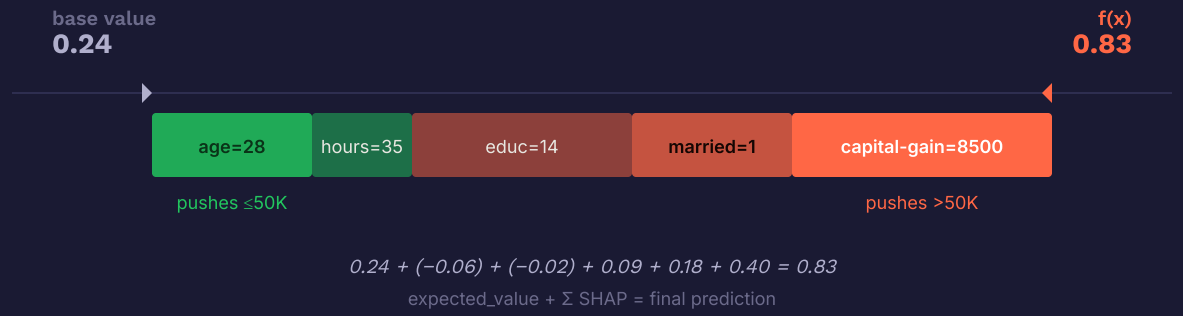
Flat shape — feature barely matters over this range

Color pattern — reveals **interaction** with another feature

Liora

LOCAL VIEW · 2

The force plot — one specific decision



Red blocks → push up

Green blocks → push down

Block width = strength of contribution

The force plot tells you why this specific person got this prediction.

KNOW THE LIMITS

SHAP is powerful — not magic

CAVEAT	WHAT IT MEANS
Correlated features	Credit may split between twins in a messy way
Compute cost	`TreeExplainer` is fast — other explainers can be slow
Not causality	A strong SHAP value is association, not cause
Local instability	Similar people can get visibly different explanations
Explainer choice	Different explainers behave differently across models

> Use SHAP as a **structured way to understand** the model — not as absolute truth.

PART 3

SHAP on text and multiclass

Emotion classification with 6 classes

From tabular to text, from 2 classes to 6

ASPECT	PART 1 (BINARY)	PART 3 (MULTICLASS TEXT)
Features	Tabular columns	Words / tokens
Task	Binary	6-class (sadness · joy · fear · anger · surprise · disgust)
SHAP output space	Probabilities	Logits (raw class scores)
Explanation unit	One per prediction	One per class, per prediction

✓ Same workflow — `TreeExplainer`, `shap_values`, force plots. We slow down on 2 new concepts.

In the notebook → `xgb.XGBClassifier(objective='multi:softprob', num_class=6)`

Logits vs. probabilities

```
raw model scores (logits)   ——softmax——▶   probabilities that sum to 1

sadness     3.1                           sadness    0.72
joy         0.7                           joy        0.08
fear        1.0                           fear       0.10
anger       0.2                           anger      0.05
surprise   -0.3                           surprise   0.03
disgust    -0.6                           disgust    0.02
```

$$\text{probability}_k = \frac{e^{\text{logit}_k}}{\sum_j e^{\text{logit}_j}}$$

In multiclass, `TreeExplainer` explains the raw class scores — probabilities come after softmax.

NEW CONCEPT · 2

One explanation per class

```
shap_values
│
├── class 0: sadness   → array (n_samples, n_features)
├── class 1: joy       → array (n_samples, n_features)
├── class 2: fear      → array (n_samples, n_features)
├── class 3: anger     → array (n_samples, n_features)
├── class 4: surprise  → array (n_samples, n_features)
└── class 5: disgust   → array (n_samples, n_features)
```

List format → `shap_values[3][5]` for person 5, anger class

3-D array → `shap_values[5, :, 3]` same thing

READING MULTICLASS FORCE PLOTS

Which words fire each emotion?

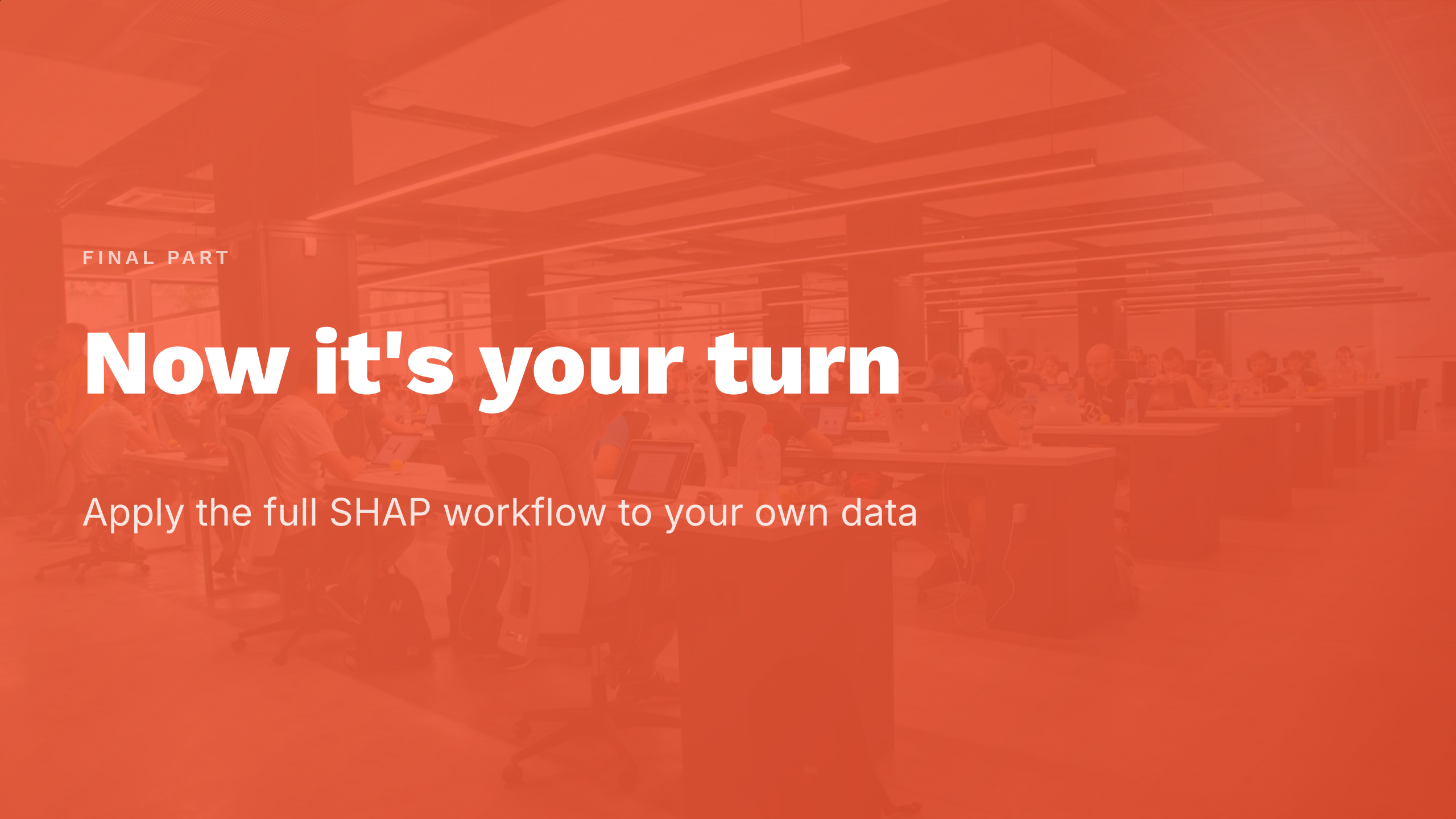
SADNESS · OBS #4

Words like **horrible**, **ungrateful** push the sadness score up. Other words pull gently the other way.

ANGER · OBS #5

A single strong term (**profane**) dominates. Calming words like **feeling** appear on the opposite side.

The point isn't just which words appear — it's **which class they support** in this explanation.



FINAL PART

Now it's your turn

Apply the full SHAP workflow to your own data

Liora

OPEN CHALLENGE

Your 5-step SHAP workflow

1. **Load & explore**
   Missing values, target balance, sensitive columns.
2. **Prepare & train**
   Encode, split, fit an `XGBClassifier`, print `classification_report`.
3. **Global SHAP — bar + beeswarm**
   Top 3 features · direction · domain check.
4. **Local SHAP — force plots**
   One clearly positive · one negative · one uncertain.
5. **Fairness / risk check**
   Sensitive attributes · proxies · extra validation needed?

Liora

SUMMARY

Key takeaways

REMEMBER THIS

- High accuracy ≠ trustworthy. Always ask: **why did it predict that?**
- Classic tools (tree, coefficients, PCA) are **intuitive** but **limited**
- SHAP gives a **unified language**: decomposition = baseline + contributions
- Two views: **global** (bar, beeswarm) and **local** (force)
- Multiclass? One explanation **per class** — in logit space
- SHAP ≠ causality. It reveals **learned patterns**, not real-world truth

What does a positive SHAP value mean?

A — The feature is globally important across the whole dataset

B — This feature pushed this prediction **above** the baseline ✓

C — The feature caused the outcome in the real world

D — The prediction is definitely correct

SHAP values are local and associative — one prediction, one person, one contribution.

FURTHER READING

Go deeper

BOOKS

- Interpretable ML — C. Molnar
- Hands-On ML — A. Géron (Ch. 6 + bonus)

PAPERS

- Lundberg & Lee (2017) — A Unified Approach to Interpreting Model Predictions
- Ribeiro et al. (2016) — LIME: Why Should I Trust You?

LIBRARIES

- `shap` — shap.readthedocs.io
- `lime` · `eli5` · `captum` (PyTorch)

MASTERCLASS COMPLETE

Thank you!

Questions? Let's discuss.