Validation

Validating models in the real world

Luis Moneda, Data Scientist at Nubank

About me

Academic

- MSc in Computer Science student (IME-USP)
- Bachelor in Computer Engineering (Poli-USP)
- Bachelor in Economics (FEA-USP)

Work & activities

- Data Scientist at Nubank (2017 - Current)
- Teaching Machine Learning for MBA courses at FIA
- Udacity mentor and project reviewer for data related courses
- Organizer of the Nubank's Machine Learning meetup
- Kaggler (competitions and datasets)
- Twitter and Blog: @lgmoneda and lgmoneda.github.io

Outline

1. Supervised Learning summarized
2. ML 101 validation
3. Real world supervised learning

There are some code examples at:

https://github.com/lgmoneda/presentations

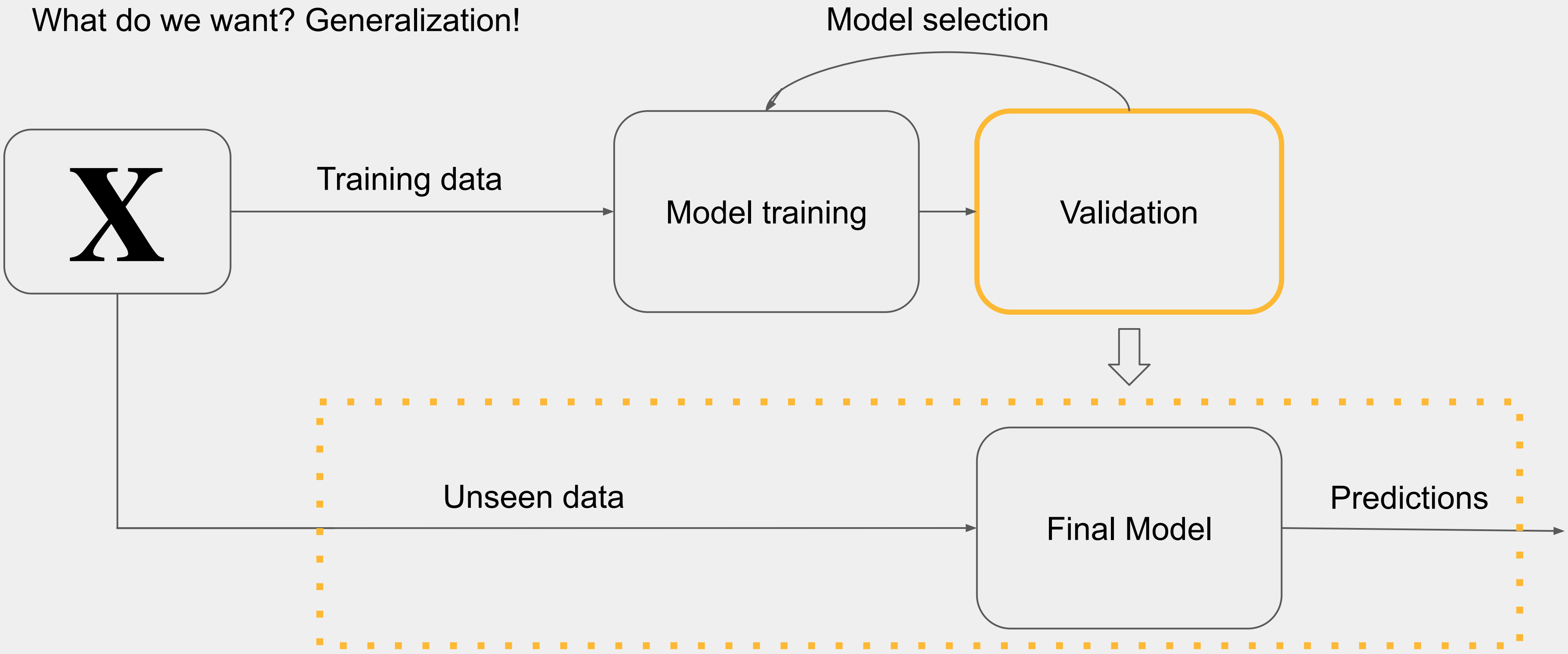
Supervised Learning summarized

$$\mathbf{X} \xrightarrow{f} \mathrm{y}$$

- Statistical Learning theory
- Empirical Risk Minimization
- Independently identically distributed (iid)
- We want to predict things nicely, we don't care about what is the f

ML101 Validation

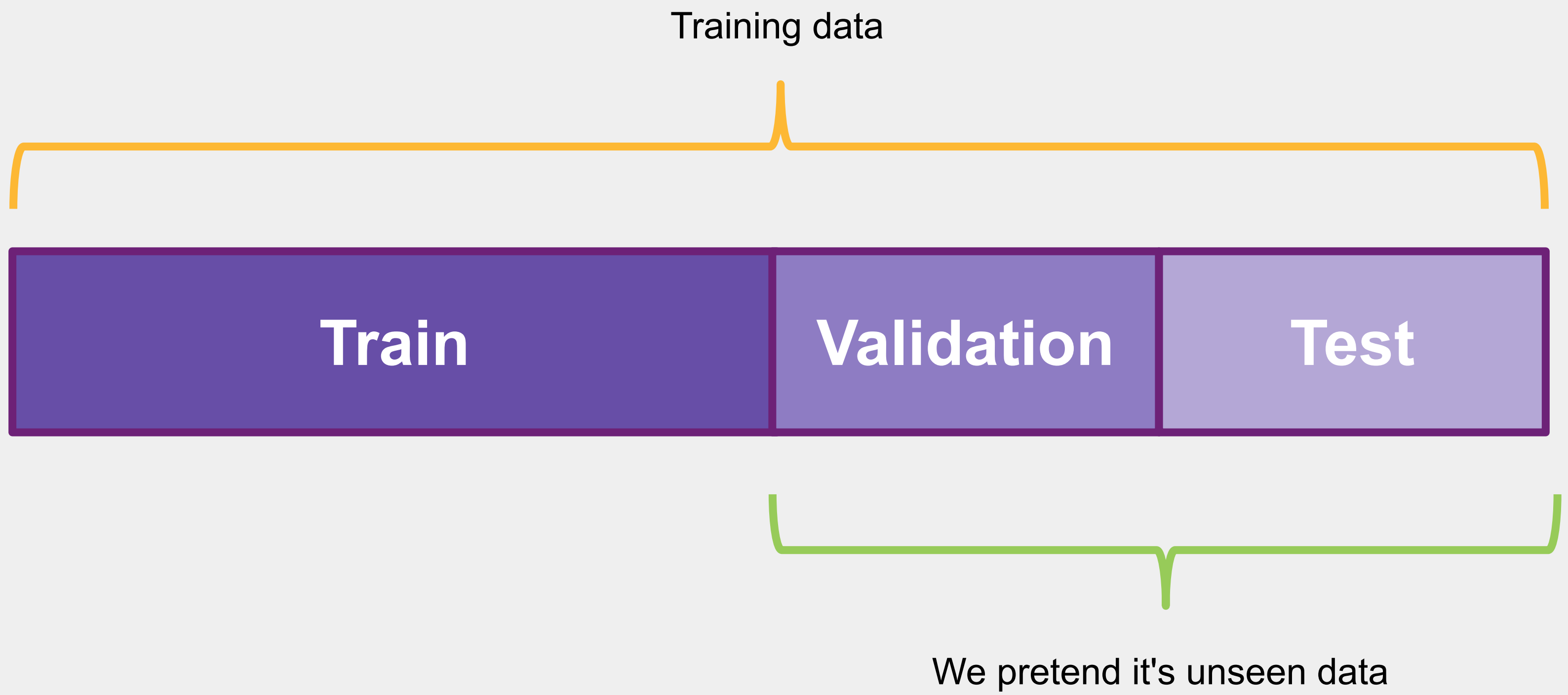
What do we want? Generalization!



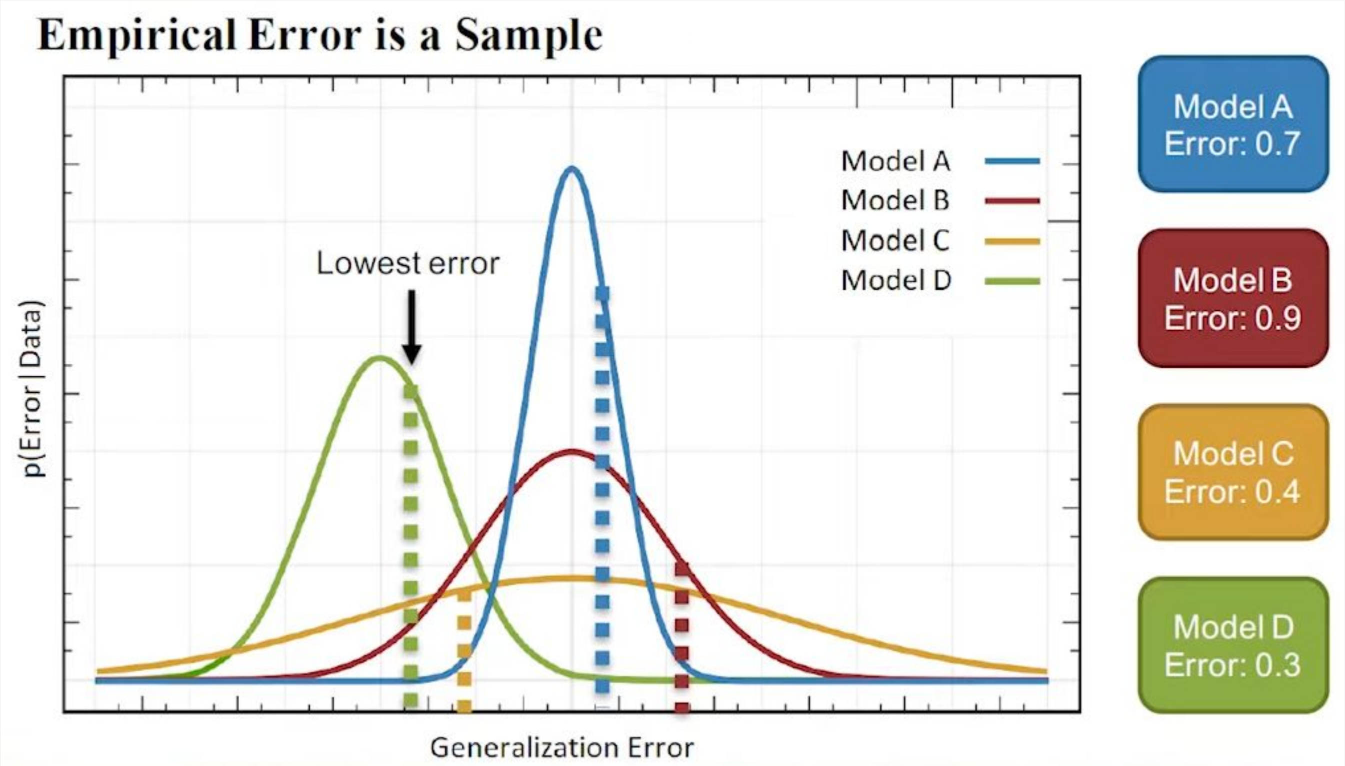
ML101 Validation: Asses performance

Model selection	Generalization power estimation
Which model am I going to **select**?	How the selected model is going to perform when **deployed**?
We want to **order** models from worst to best	We want to **estimate it assertively**
Validation set	Test set
Hyper parameters optimization	Solution selection, impact estimation

ML101 Validation: Simple split

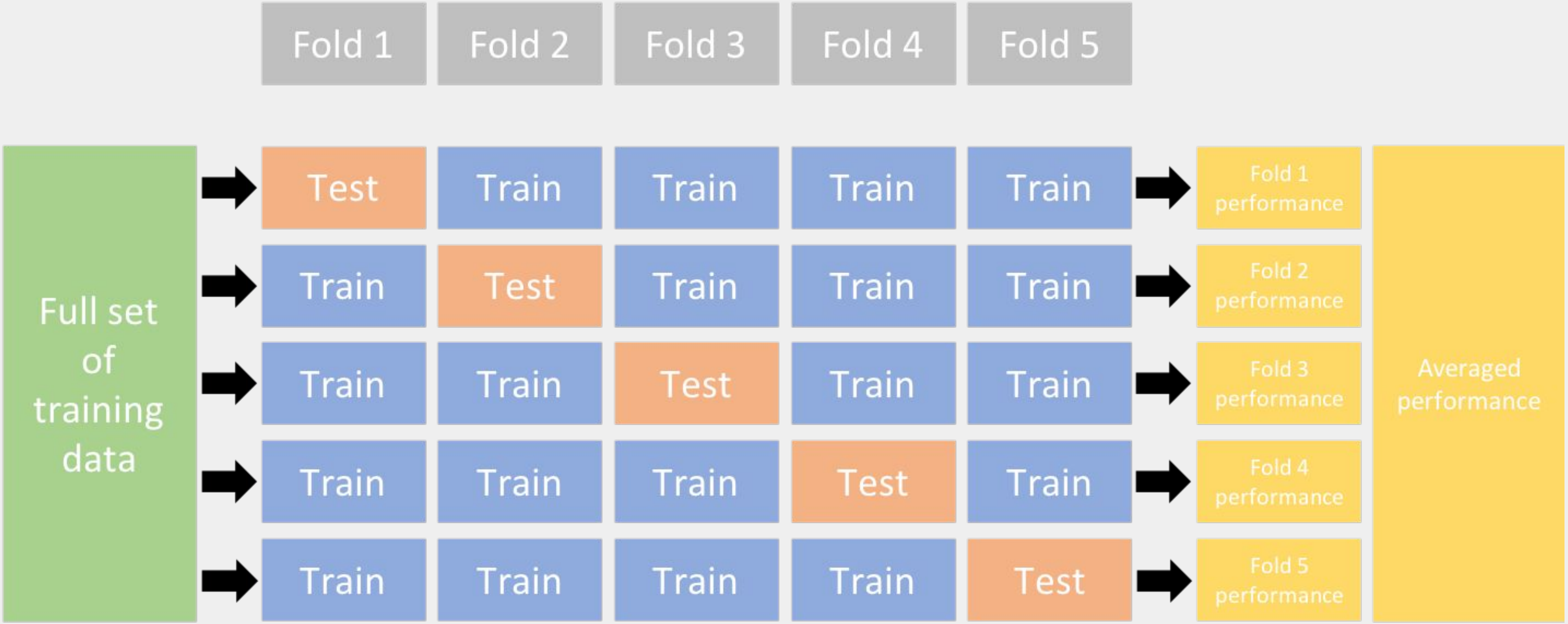


ML101 Validation



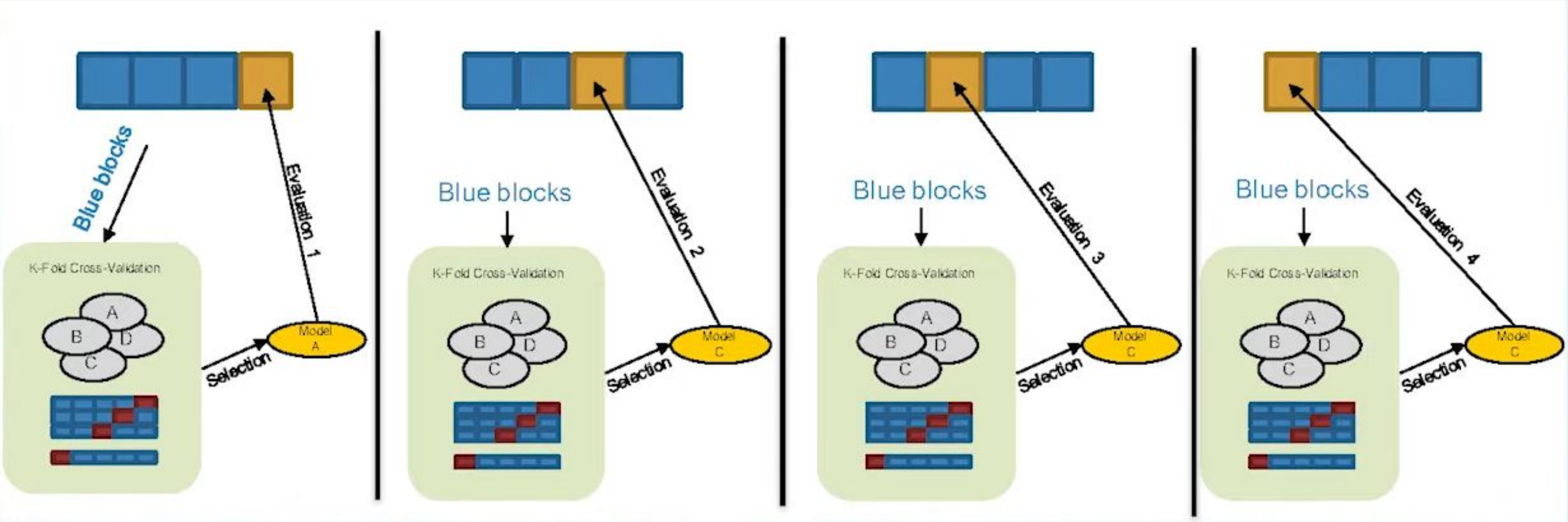
Source: Sergey Feldman, You Should Probably Be Doing Nested Cross-Validation | PyData Miami 2019

ML101 Validation: K-Fold



Source: **UC Business Analytics R Programming Guide**

ML101 Validation: Nested K-Fold



Source: Sergey Feldman, You Should Probably Be Doing Nested Cross-Validation | PyData Miami 2019

ML101 Validation

So after your ML101 classes **it should look very clear**:

We want generalization, i.e. performing well on unseen data, so:

1) Leave some data out of the training process and pretend it's unseen;
2) Check if the learned model performs well on this unseen data;
3) If it performs reasonably, pick it!
4) Put in production!

What could possibly go wrong?

Then you go to the real world and...

Real World Supervised Learning

$$\mathbf{X} \xrightarrow{f} \mathrm{y}$$

Real World Supervised Learning

$$\xrightarrow{f} \mathrm{y}$$

Well, it turns out that in **most of the cases** the $\mathbf{X}$ is **mutant**!

Real World Supervised Learning

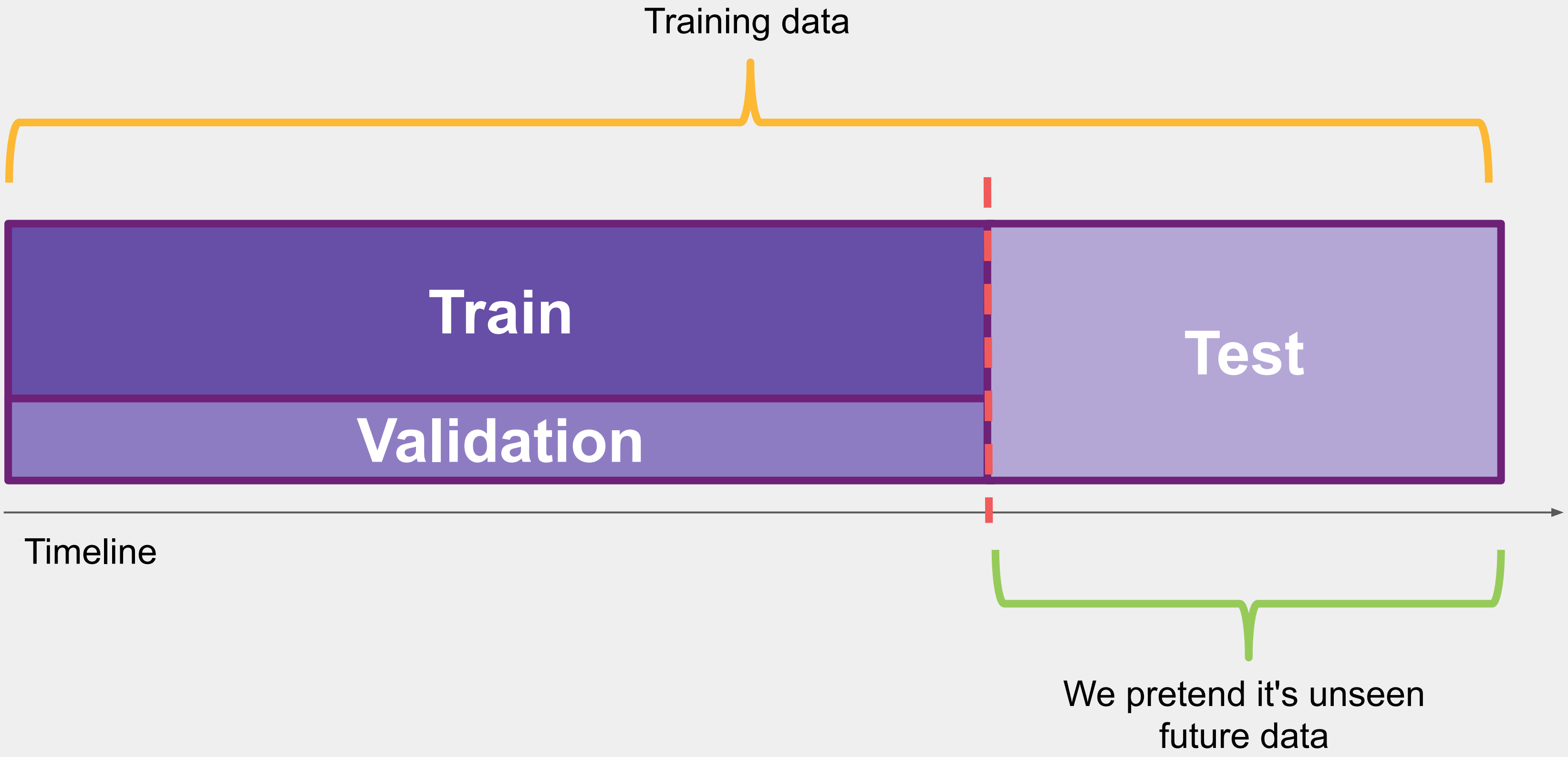
$$\xrightarrow{f} \mathrm{y}$$

Well, it turns out that in **most of the cases** the $\mathbf{X}$ is **mutant**!

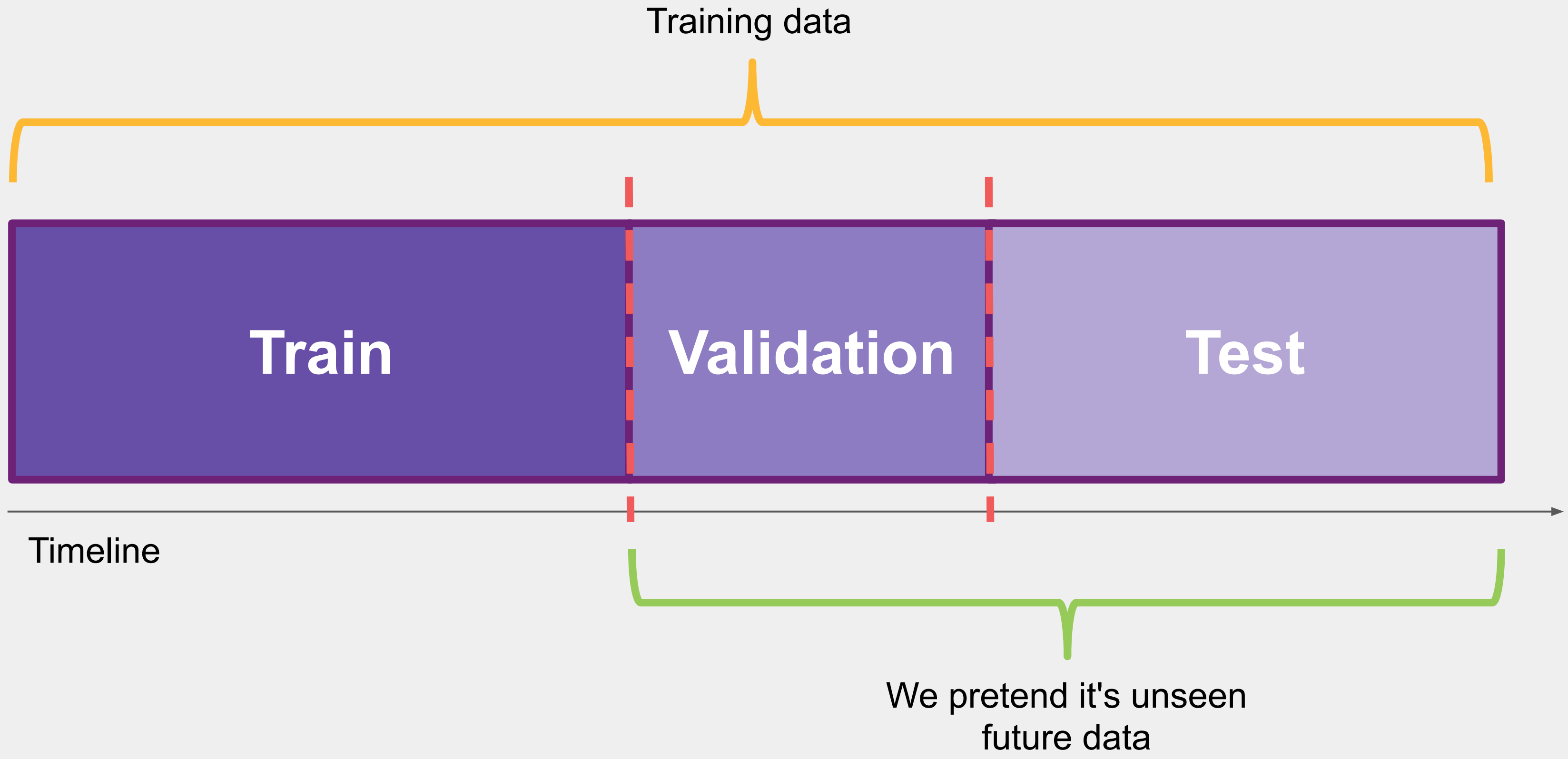
Temporally, spatially… **bye bye, i.i.d!**

Random splits imply future data being used to predict past data

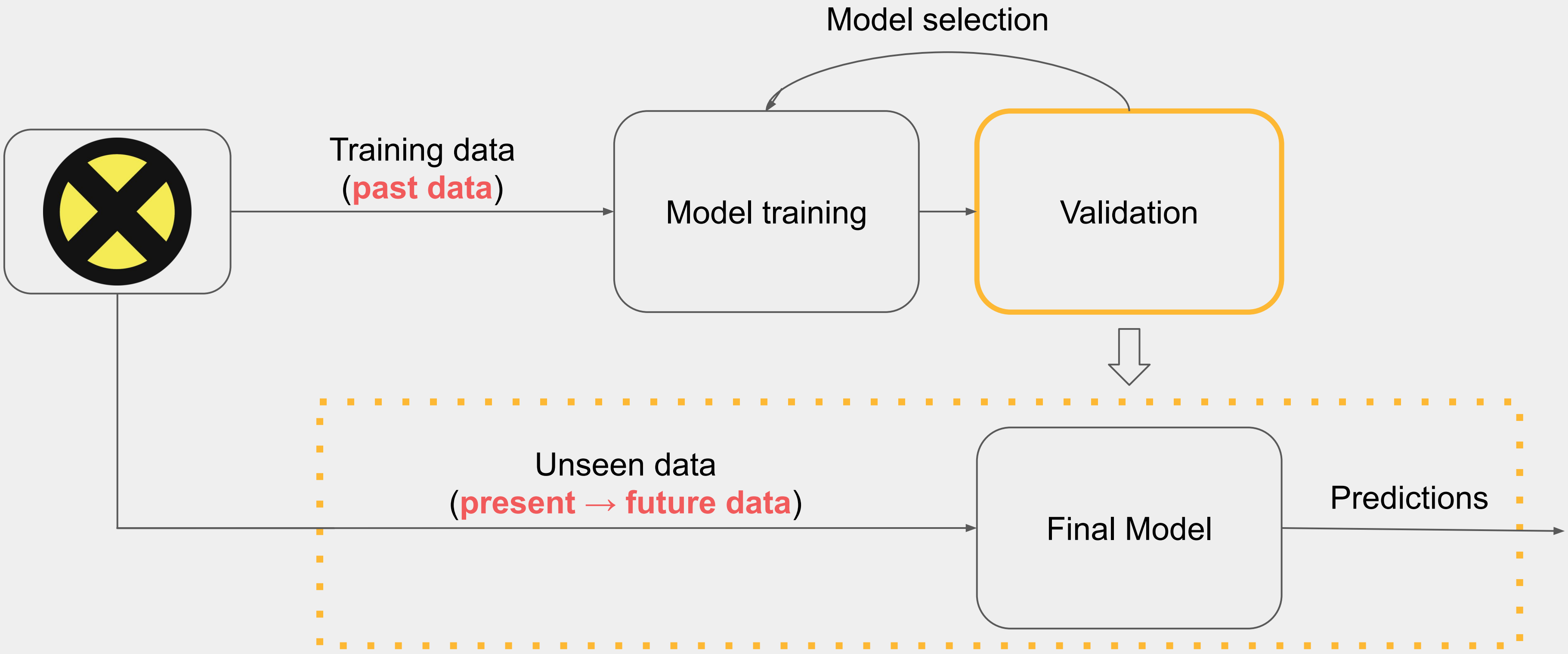
Real World Validation: Temporal split



Real World Validation: Temporal split



Real World Validation



When temporal validation can help us?

Basically, **always**!

All datasets have a temporal aspect because they are generated as the time passes by, but time effect depends on the problem.

Weak*

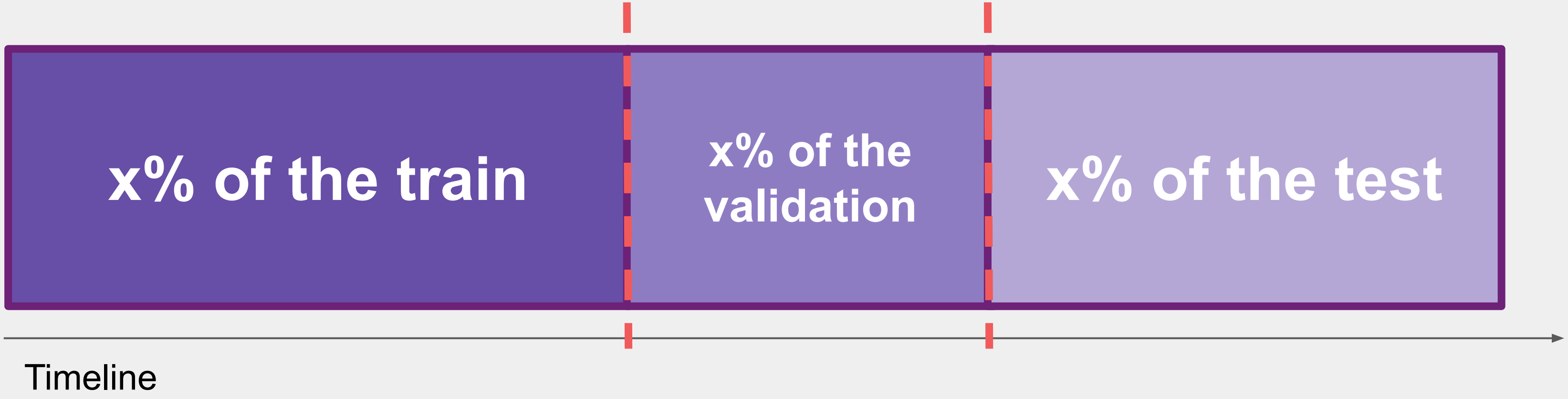
- Images
- Text

Strong

- Time series
- Tabular data

What about the point estimate problem?

Real World Validation: Temporal split



Small dataset: bootstrap

Enough data: you're fine!

Is the generalization estimation right now?

Real World Supervised Learning

$$X \xrightarrow{f} y$$

- Default
- Churn
- Fraud

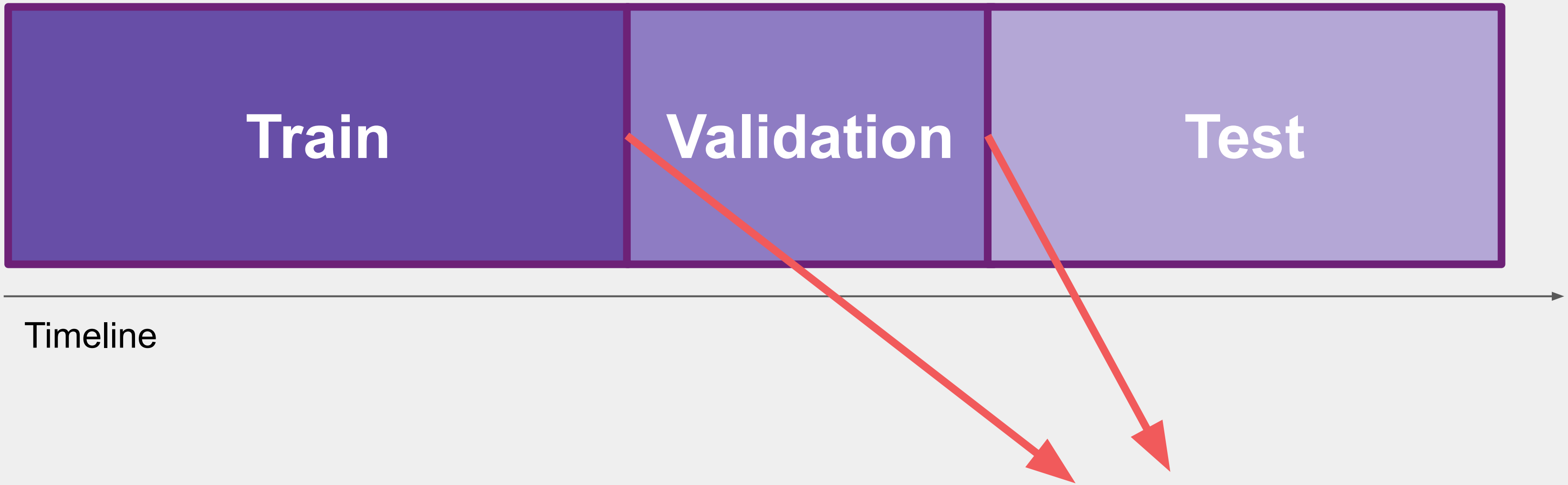
What is default?
Not paying after n days.

What is churn?
Not using the product for n days.

What is fraud?
Being reported as fraudulent after n days of the transaction / operation.

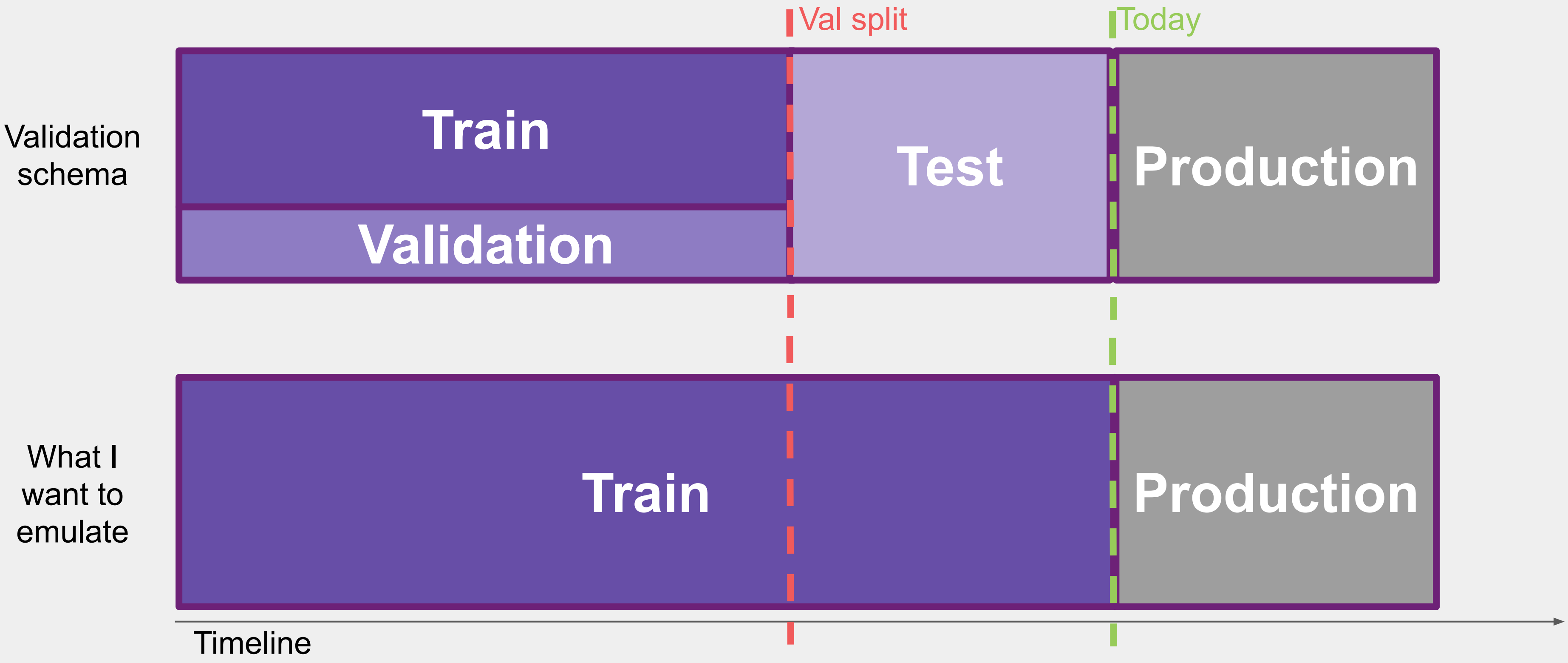
All of them involve **observing the phenomena in a time frame** to finally annotate the example!

Real World Problem: target observation

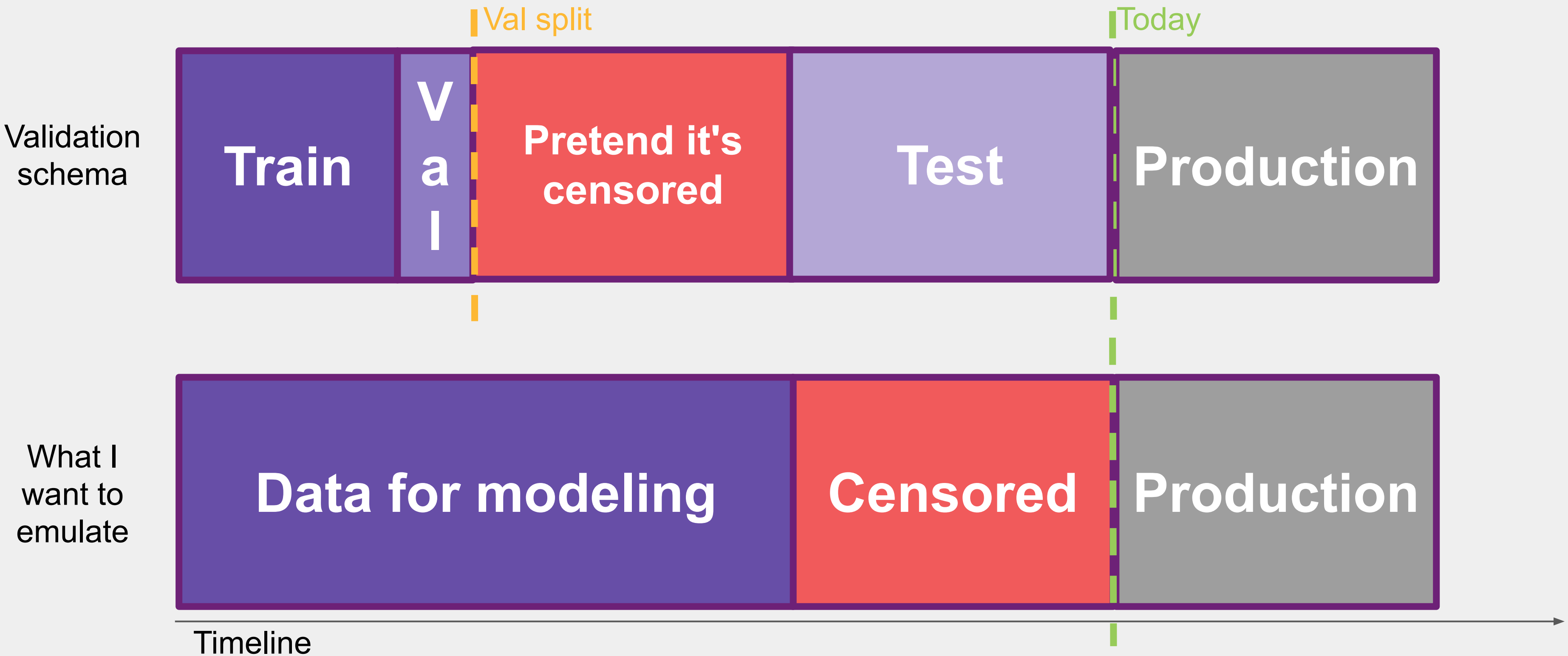


Having annotated data to train your model just before the period you're going to evaluate it won't be possible in practice!

Real World Problem: target observation



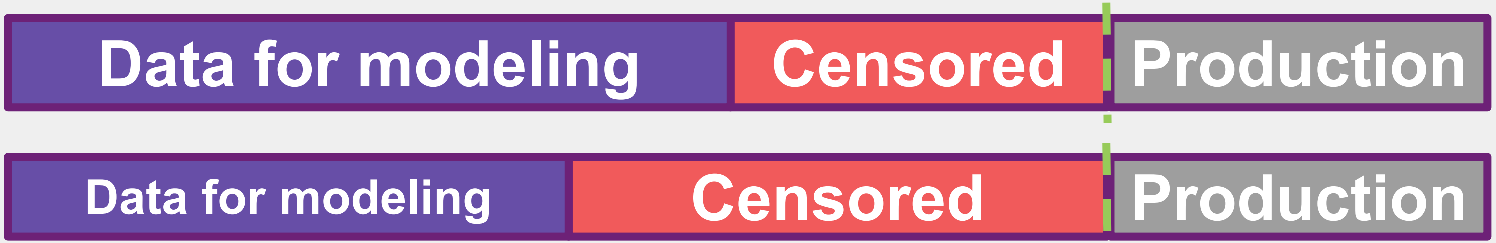
Real World Problem: target observation



Prediction gap

When is it relevant for model selection?

Testing different target definition: churn as an inactive user for 5, 10… 60 days.
It will change the censored length.



So the validation can mimic the production environment and address the trade-off between **target stability** and **fewer and older training data**.

Examples: churn, default

Model degradation

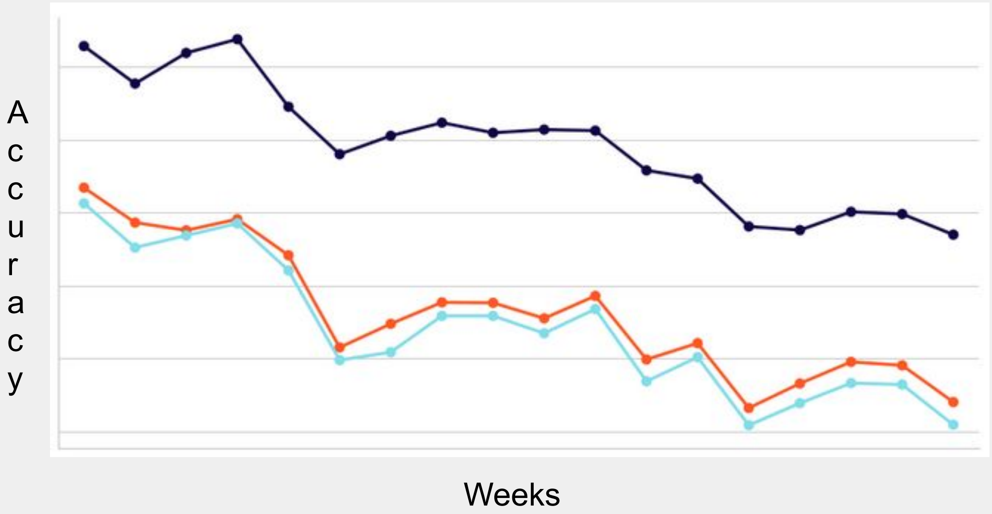
Model degradation

When is it relevant?

More complex models degrade faster!

So it impacts model selection.

A real model performance by week



Ready to rock!

Ok, let's summarize it:

- Now you know the **inherent role of time** in every dataset;
- You can design a validation schema that considers the **prediction gap;**
- When reporting the generalization power you consider **model degradation** and the time frame your model will operate.

Now you pick a company's problem and ask how they solve it currently.

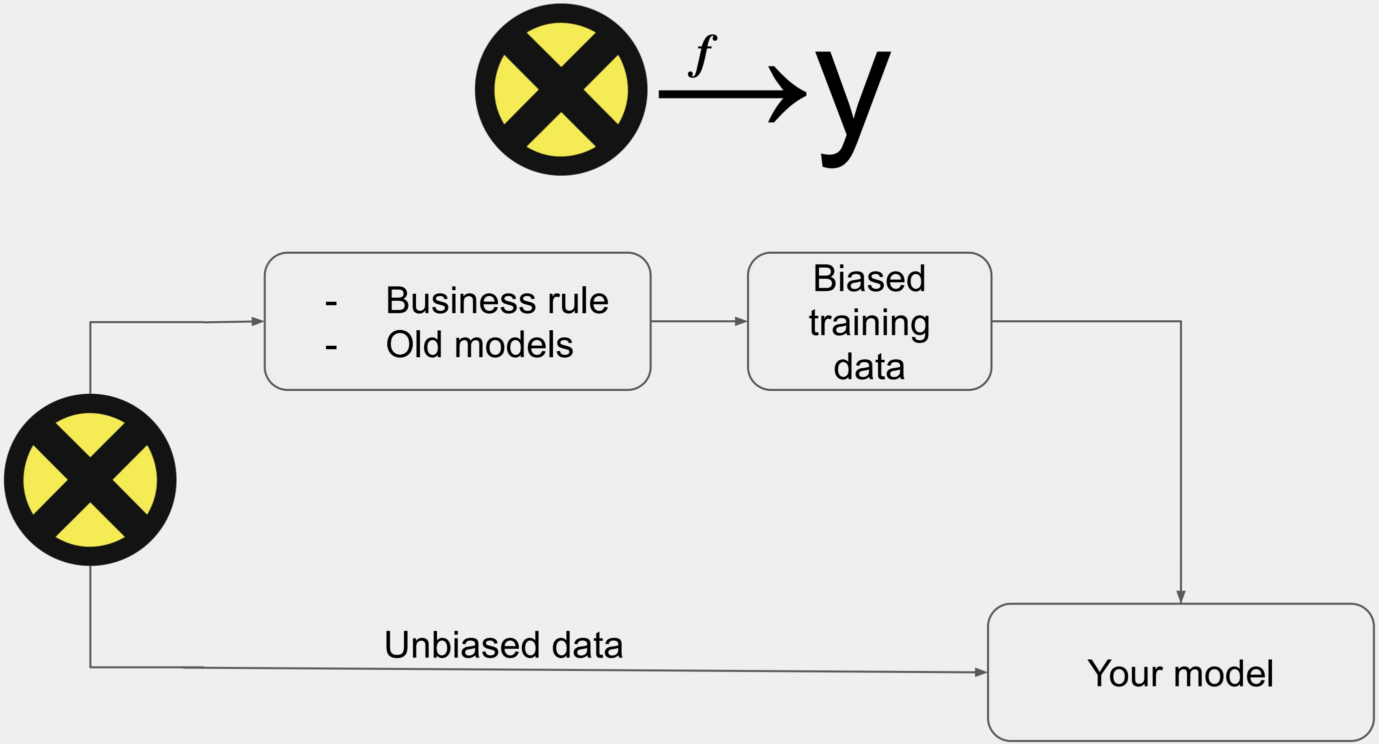
"We have some business rules to decide what to do: we apply some IFs, ELSE... and..."

"Oh, do you think you can improve it?"

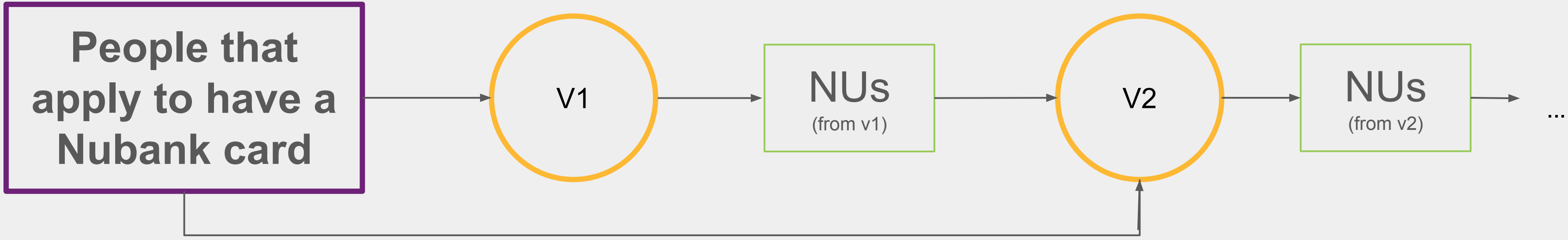
1) Get the historical data
2) Train a model on it
3) Validate using out of fold data
4) Get rid of all the crap business rules
5) Deploy your awesome ML model

But then...

Old policies and models bias



Old policies and models bias example



How can I evaluate V2 if **I can't observe the outcome for people rejected** by V1?

Counterfactual evaluation and rejected inference

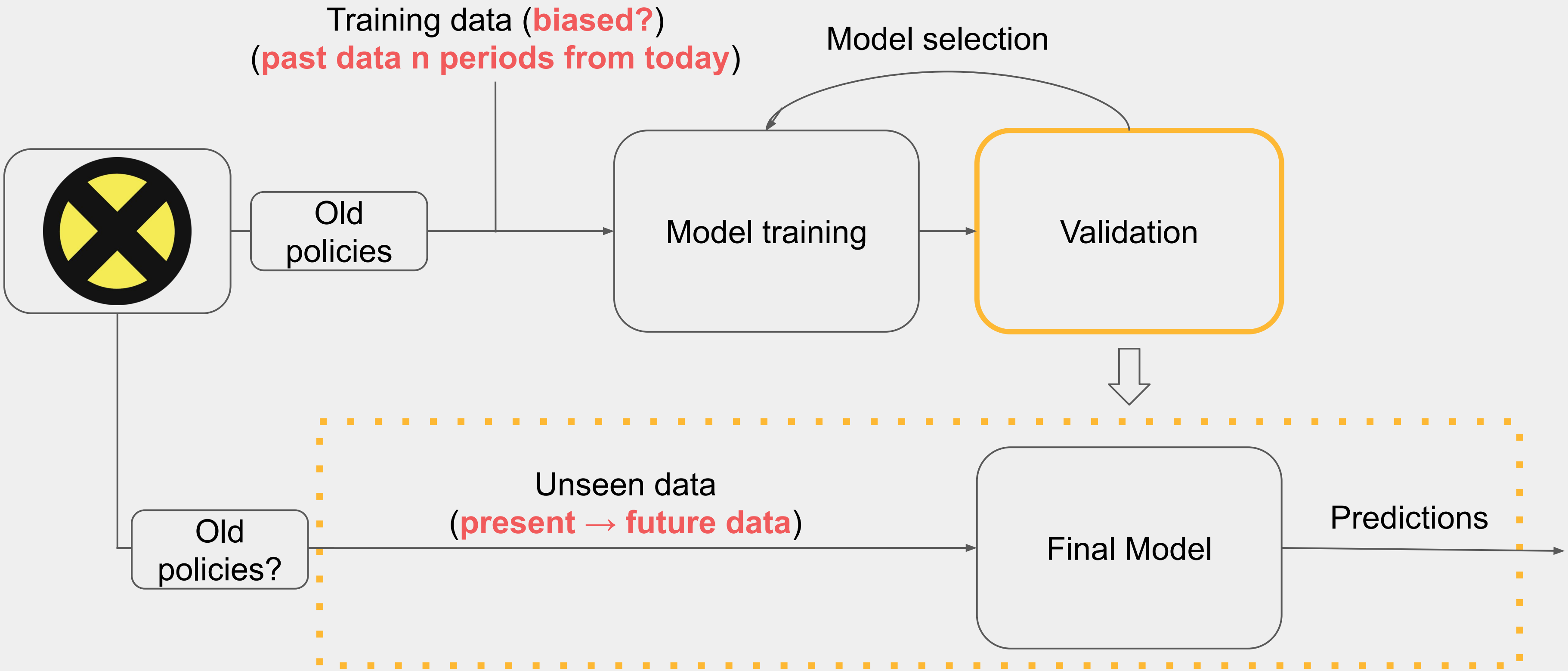
Counterfactual evaluation

In production, disobey your model decision with a probability p, then you can oversample them to evaluate the next model version.

Rejected Inference

Find a way to make an inference about the outcome from the examples you can't observe the ground truth.

Real World Validation



In a company, data science joins business and engineering to deliver value.

Engineering

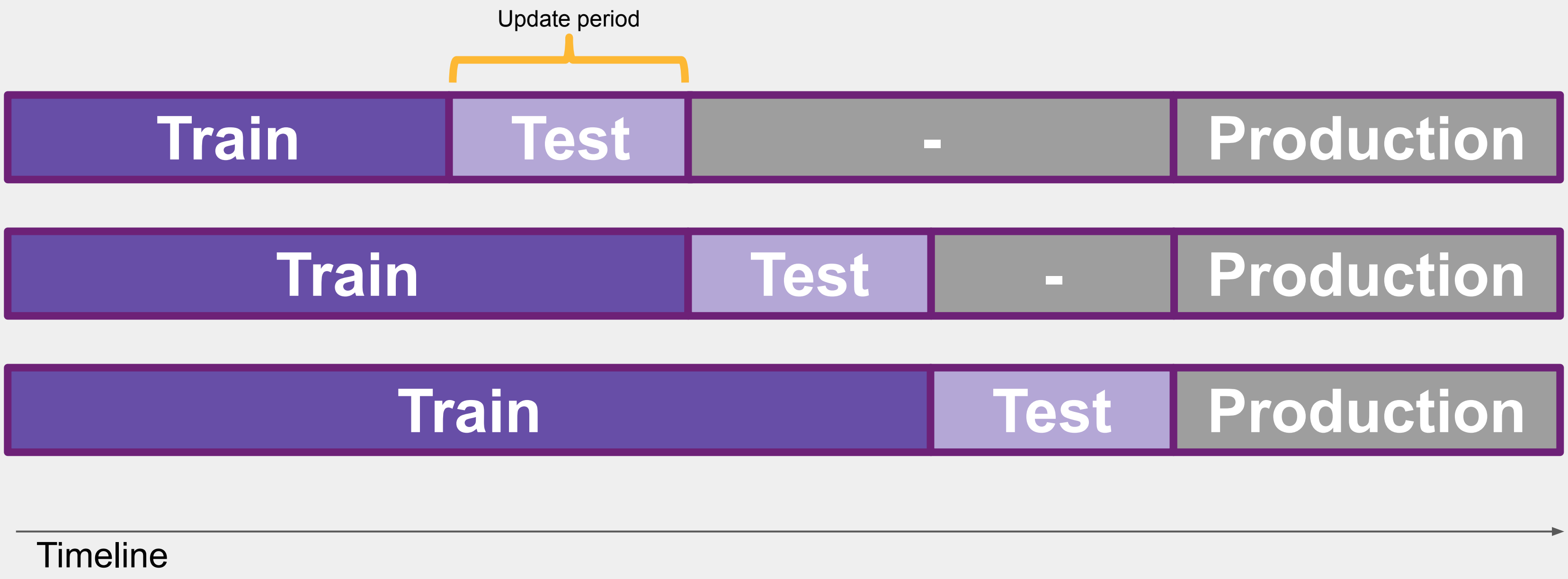
Engineering

- How often can I update my model?
- Is there any time constraint?

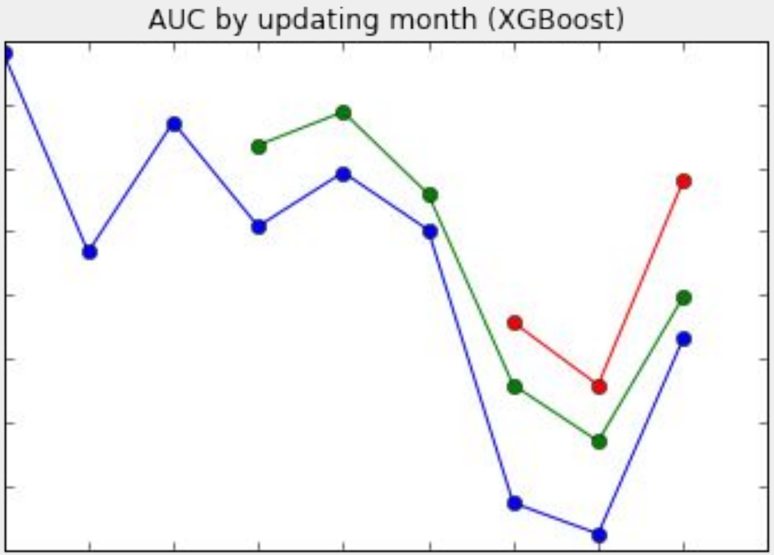
So the production environment we want to validate may become something like "what is the best model considering it can be updated every N periods?"

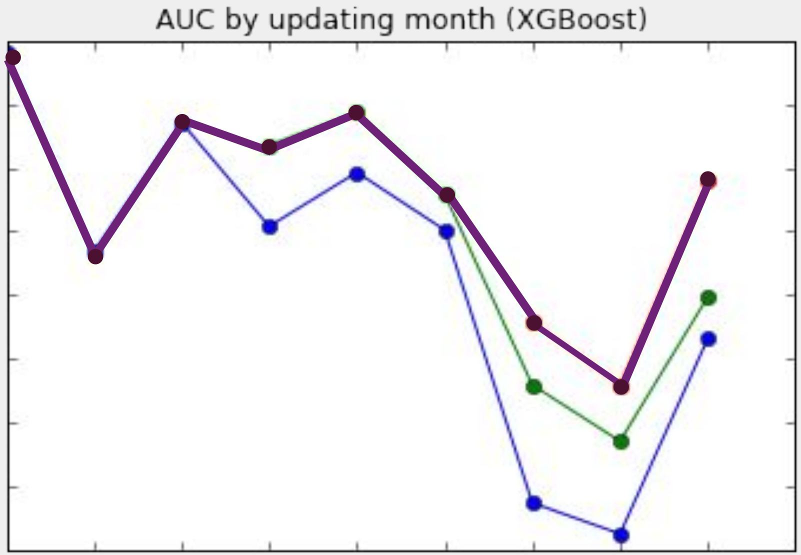
Real World Validation - Engineering

Validate considering update



RW Validation - Engineering example





Business

Business

- A lot of things can change the $\mathbf{X}$ distribution:
    - Marketing
    - New products
    - Communication
    - Growth/maturity
- You want to produce meaningful/profitable/useful predictions
- Update and running time constraints also
- Model objective

Business: how does it impact validation?

Business

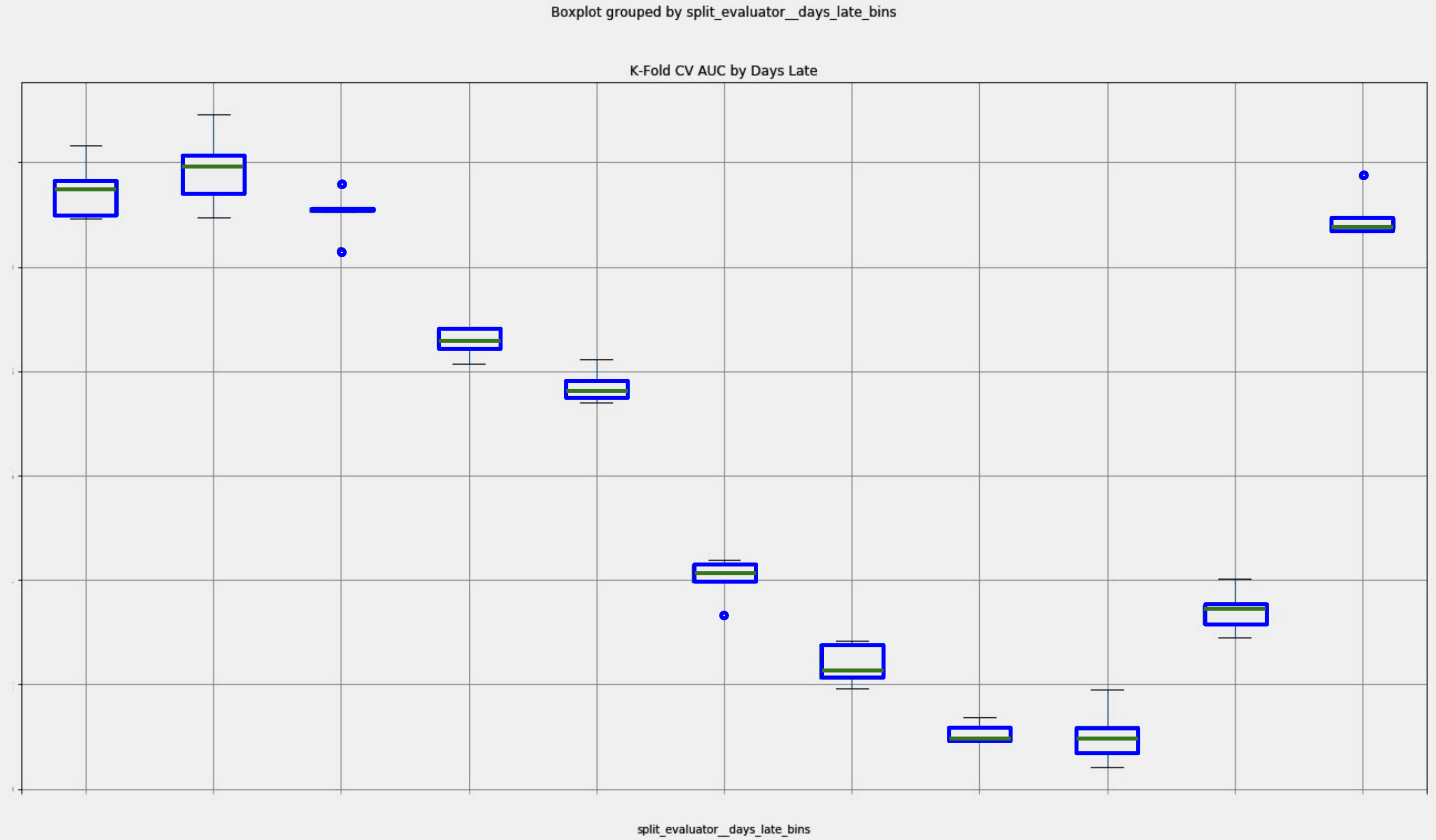
- A lot of things can change the $\mathbf{X}$ distribution
  You can't do anything at validation time for future changes, but **monitor**! You shipped something to score over X, but people won't care about, while you should.

- You want to produce meaningful/profitable/useful predictions
  Validate **considering business value**. Split by important features/groups, analyze past events that changed the $\mathbf{X}$ distribution.

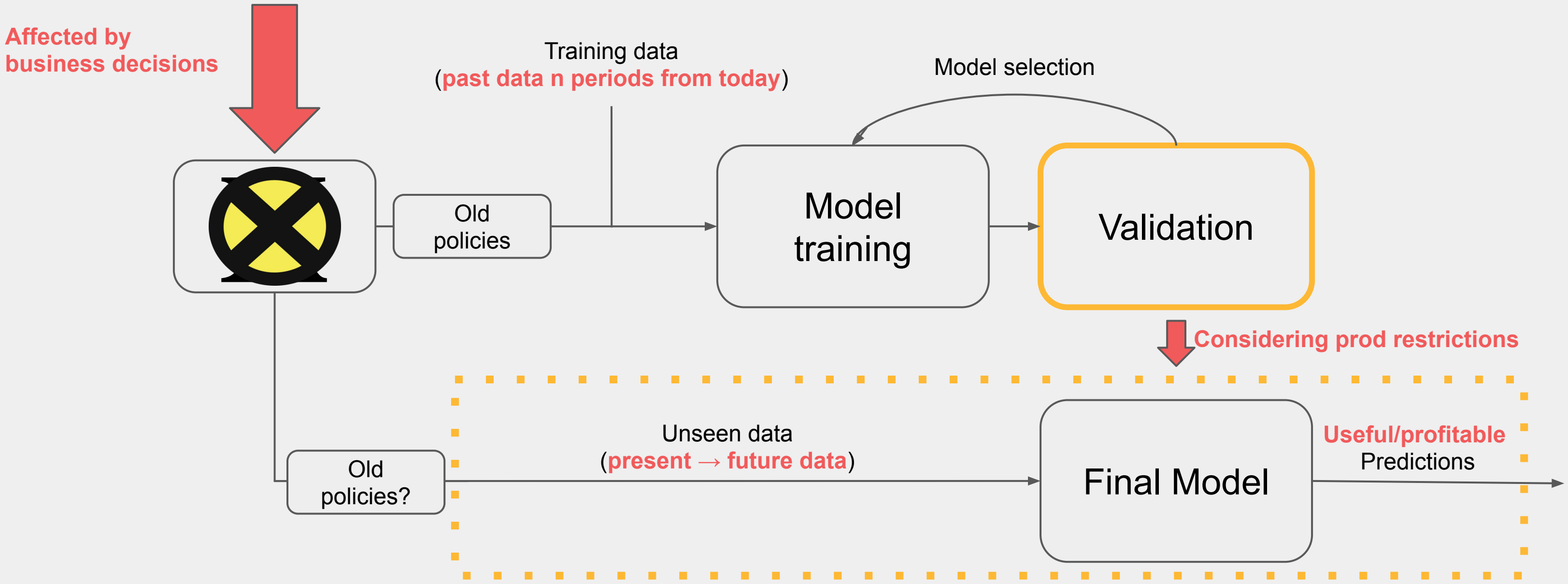
- Update and time constraints also:
  Consider model performance x delay to take decisions! **Calculate the monetary trade-off** between them.

- Model objective
  If you know how your data was collected and how your model is going to be applied, it can be a **leverage** instead of a trap.

Real World Validation - Deeper look



Real World Validation



"Wait a minute! If I'm not doing any of this, how am I not blowing my company?"

Well...

Validation strategy	Model Selection	Generalization power estimation	Impact
Mimics application environment	You choose the best model in terms of predictive power	Provides the best estimation about the model performance when in production	You’re doing great!
It doesn't mimic, but it's fair	A wrong but fair comparison has a good chance to keep model ordering for the selection (include current model!)	Bad estimation, probably overestimating model performance.	Replacing the current solution/model by a worse one. Adopting a not profitable solution.
It doesn't mimic, unfair comparison	Picking a sub optimal model	Bad estimation	Same as above, but probably with a worse model

Is it possible at all to replicate prod environment for validation?

So at the end...

X	
Train: A nice and invariant distribution I have a reasonable random sample. Apply: In an unseen random sample.	Train: Old, far from prediction time, biased by old policies and models, unequally distributed in the features you care about. Apply: In an unseen future data I'm not sure about how it's going to change accordingly to time and other business decisions.

Takeaways

It's hard to define a recipe for validation, but keep in mind the general idea of "**mimic the production environment / application case**":

- Use a **temporal split**
- Observe the **model degradation** in time
- Consider the **censored period** to observe the target
- Do a internal research about **how the data was collected** to be aware of all the old policies and models and **its bias**
- Know **how/when** your model is going to be applied
- Consider all the **engineering restrictions and possibilities**
- Think about the **important business aspects** to do a deeper validation

Takeaways

It's hard to define a recipe for validation, but keep in mind the general idea of "**mimic the production environment / application case**":

- Use a **temporal split**
- Observe the **model degradation** in time
- Consider the **censored period** to observe the target
- Do a internal research about **how the data was collected** to be aware of all the old policies and models and **its bias**
- Know **how/when** your model is going to be applied
- Consider all the **engineering restrictions and possibilities**
- Think about the **important business aspects** to do a deeper validation
- Be aware of **population shifts** caused by business decisions

Questions?

@lgmoneda

lg.moneda@gmail.com

http://lgmoneda.github.io/