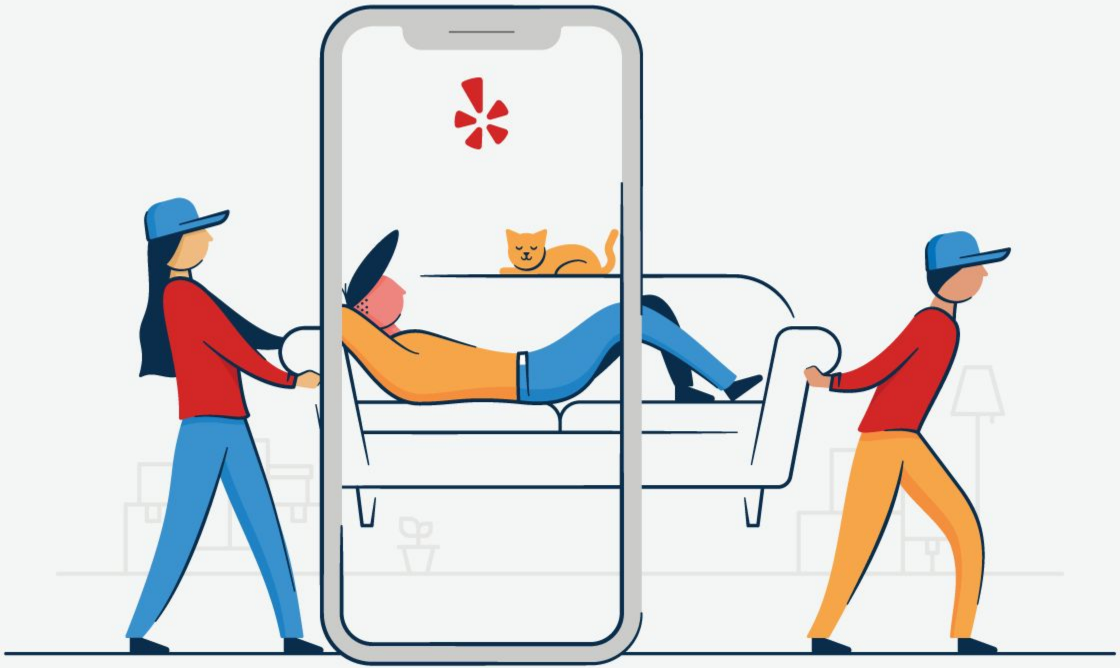
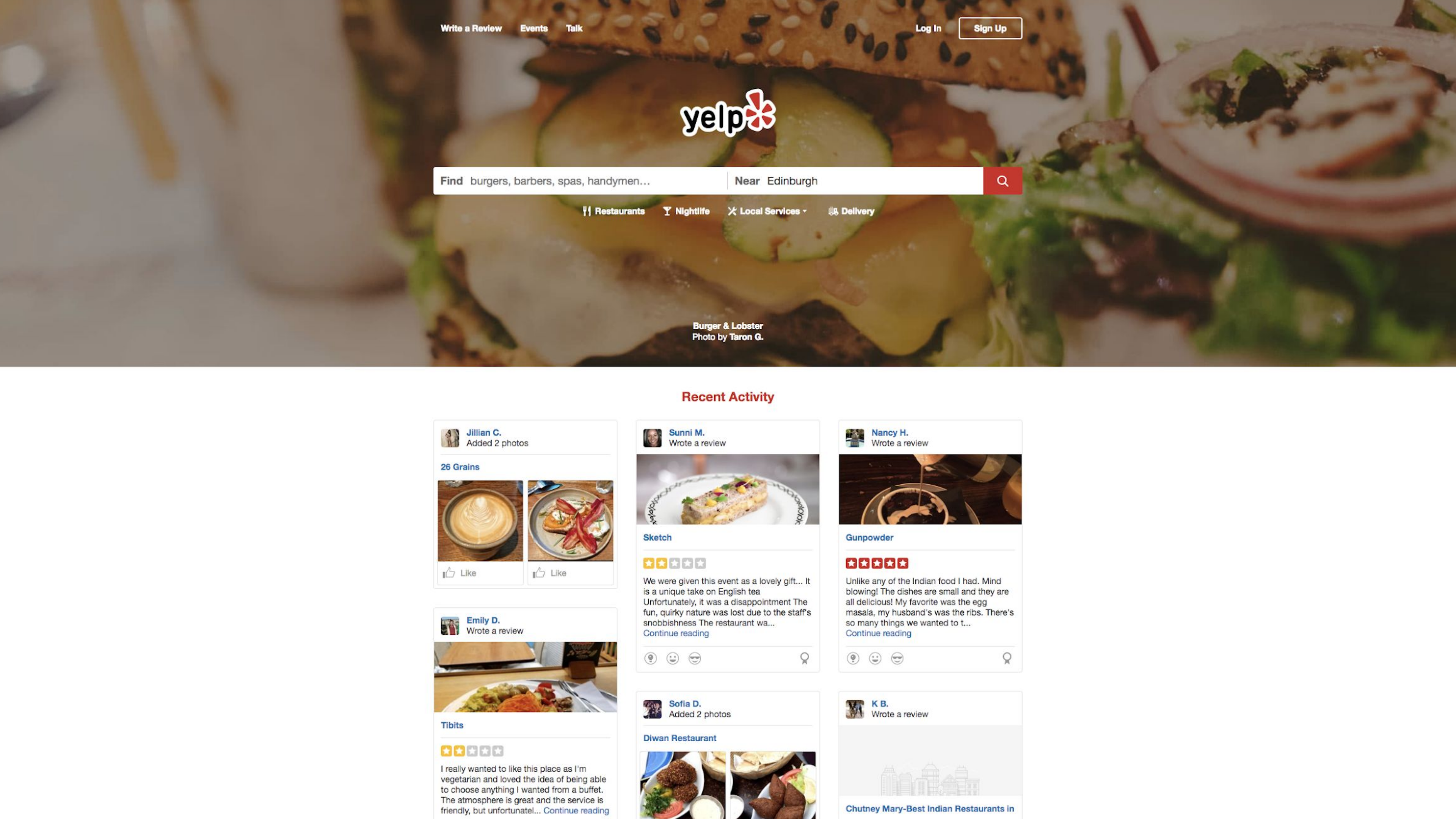
Productionizing your ML code **seamlessly**

lauris #EuroPython



europython
Edinburgh 23-29 July
2018

Connecting people with **great local businesses**


Write a Review
Events
Talk
Log In
Sign Up
yelp
Find burgers, barbers, spas, handymen...
Near Edinburgh
Restaurants
Nightlife
Local Services
Delivery
Burger & Lobster
Photo by Taron G.
Recent Activity
Jillian C.
Added 2 photos
26 Grains
Like
Like
Sunni M.
Wrote a review
Sketch
We were given this event as a lovely gift... It is a unique take on English tea Unfortunately, it was a disappointment The fun, quirky nature was lost due to the staff's snobbishness The restaurant wa...
Continue reading
Nancy H.
Wrote a review
Gunpowder
Unlike any of the Indian food I had. Mind blowing! The dishes are small and they are all delicious! My favorite was the egg masala, my husband's was the ribs. There's so many things we wanted to t...
Continue reading
Emily D.
Wrote a review
Tibits
I really wanted to like this place as I'm vegetarian and loved the idea of being able to choose anything I wanted from a buffet. The atmosphere is great and the service is friendly, but unfortunatel... Continue reading
Sofia D.
Added 2 photos
Diwan Restaurant
K B.
Wrote a review
Chutney Mary-Best Indian Restaurants in

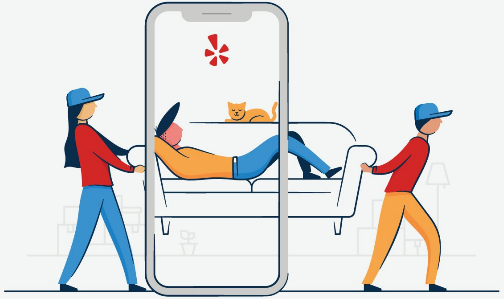
Some fun facts about Yelp

- Yelpers have written **155 million** reviews since 2004.
- We have **74 million** desktop and **30 million** mobile app monthly unique visitors.
- We have over **500 developers**.
- We have over **300 services** and our monolith yelp-main has over **3 million** lines of code!

Agenda
for today

- What does running an **ML model** in **production** involve?
- How to improve your **development workflow** to make the path to production **easier**

Starting Point: your notebook



It solves your problem

Your notebook ...

- **predicts** a desirable behavior
- **recommends** the best items to your users
- **detects** when an event is about to happen
- **forecasts** trends in stocks
- **ranks** items in search

Your notebook can be simple

```
    'n_bizsite_visit_by_yuv_has_id',
    'n_bizsite_views_to_biz',
    'n_bizsite_visit_to_biz_has_id',
    'n_bizsite_page_views',
    'bizsite_views_time_interval',
]
```

Data Preparation

XGBoost Format

In [4]:
```
from biz_data_mining.util.libsvm import read_group_local
from biz_data_mining.util.libsvm import read_libsvm_sklearn_local

# Dev set
X_xgboost, y = read_libsvm_sklearn_local('/code/dev_set/', len(feature_cols))
xgboost_group = read_group_local('/code/dev_set/')

# Prod set
# X_xgboost, y = read_libsvm_sklearn_local('/code/prod_set/feature/', len(feature_cols))
# xgboost_group = read_group_local('/code/prod_set/group/')
```

Sklearn Format

In [5]:
```
from sklearn.model_selection import GroupShuffleSplit

group_id_column = to_group_id_column(xgboost_group)
X = join_group_id_column(X_xgboost, group_id_column)

gss = GroupShuffleSplit(n_splits=1)

train_idx, test_idx = [(train, test) for (train, test) in gss.split(X,y, groups=group_id_colum
n)][0]

X_train = X[train_idx]
y_train = y[train_idx]
group_id_column_train = group_id_column[train_idx]

X_test = X[test_idx]
y_test = y[test_idx]
group_id_column_test = group_id_column[test_idx]
```

… or not so much

```
In [1]: # imports

        from yelp_redshift.client import RedshiftClient
        import pandas as pd
        import yaml
        import os
        from datetime import date, timedelta


        from scipy import stats

        import dill
        from sklearn.base import TransformerMixin
        from sklearn.preprocessing import FunctionTransformer


        from sklearn.linear_model import LogisticRegression
        from sklearn.model_selection import cross_val_predict
        from sklearn.pipeline import Pipeline
        from sklearn.metrics import roc_auc_score
        from sklearn.metrics import roc_curve
        from sklearn.metrics import confusion_matrix


        pd.options.display.max_rows = 200
        pd.options.display.max_columns = 200
```

```
In [2]: # a bunch of functions, mostly to duplicate data_science repo functionalities

        def get_from_yaml(datastore_name):
            # this will need to change for non-datascience environments
            datastore_yaml_path = "{0}/yelp_{1}.yaml".format(
                os.path.expanduser("~/.data_science_config"), datastore_name)
            with open(datastore_yaml_path, 'r') as f:
                config_opts = yaml.load(f)
            return config_opts

        def redshift_run(datastore_name, sql_text, execute=False, chunk_size=200000):
            # runs sql_text on datastore_name.
            # runs as a command if execute is true, otherwise returns results.
            config_opts = get_from_yaml(datastore_name)
            client = RedshiftClient(
                host=config_opts.get('host'),
                port=config_opts.get('port'),
                db=config_opts.get('db'),
```

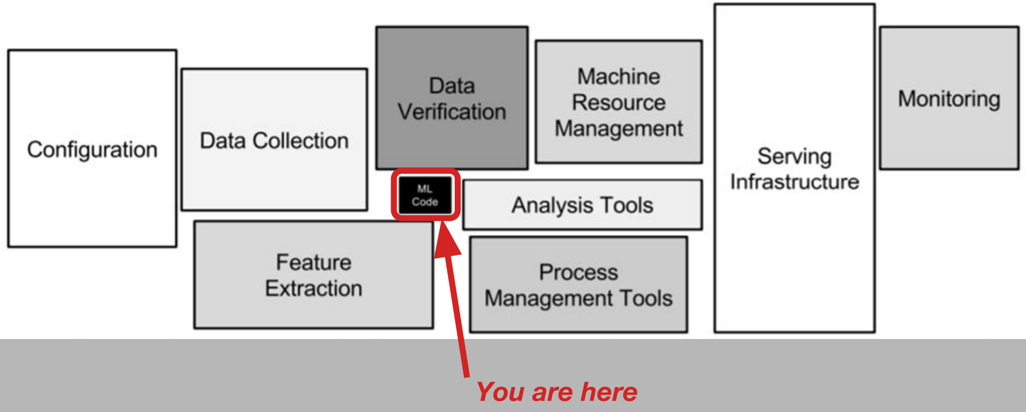
Making a model work in a notebook is only the first step to make it useful

- Getting data to predict or train on **regularly**
- **Evaluate** the model
- Use the **predictions** in your product
- Verify that your initial **objective** is accomplished, regularly

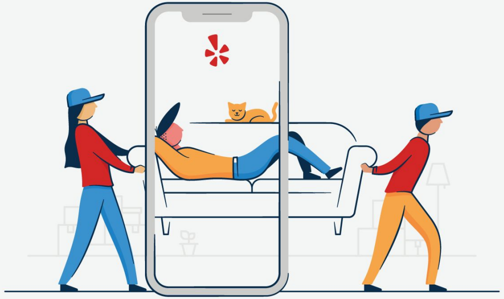
STARTING POINT: YOUR NOTEBOOK

"Only a **small fraction** of real-world ML systems is composed of the ML code [...] The required surrounding infrastructure is **vast and complex**."

Hidden Technical Debt in Machine Learning Systems - Google NIPS 2015



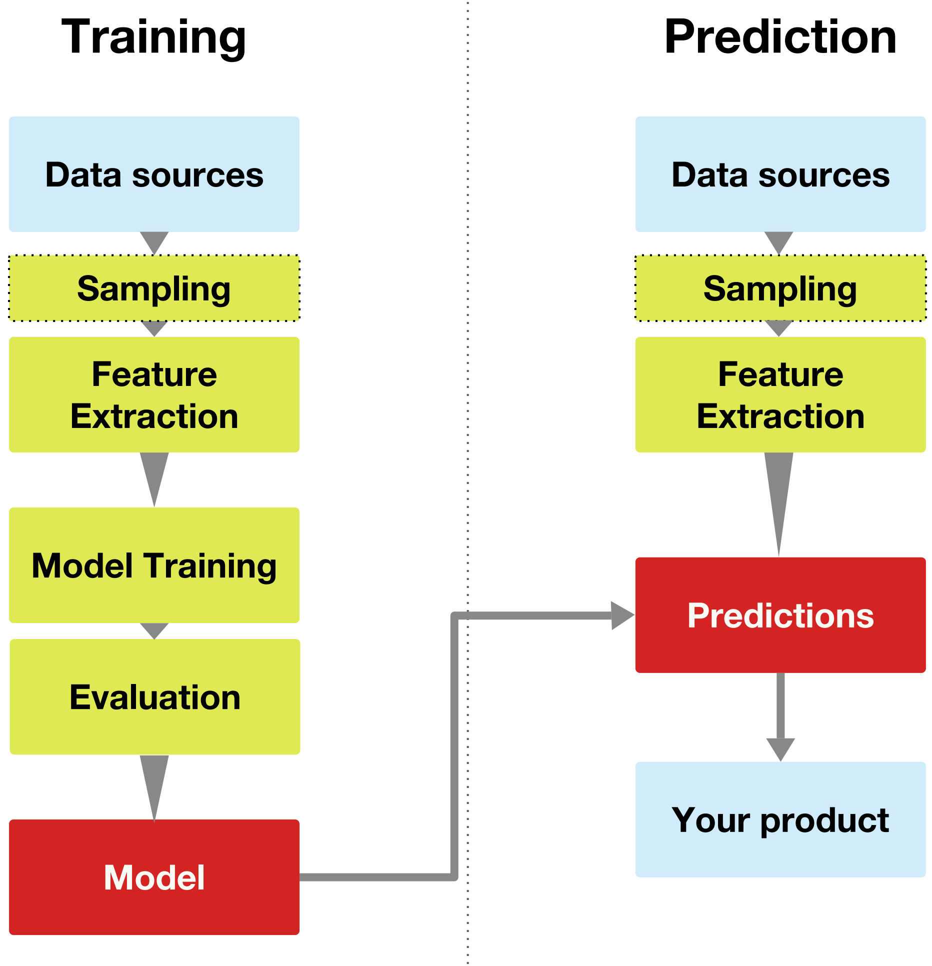
What does running an **ML model** in **production** involve?



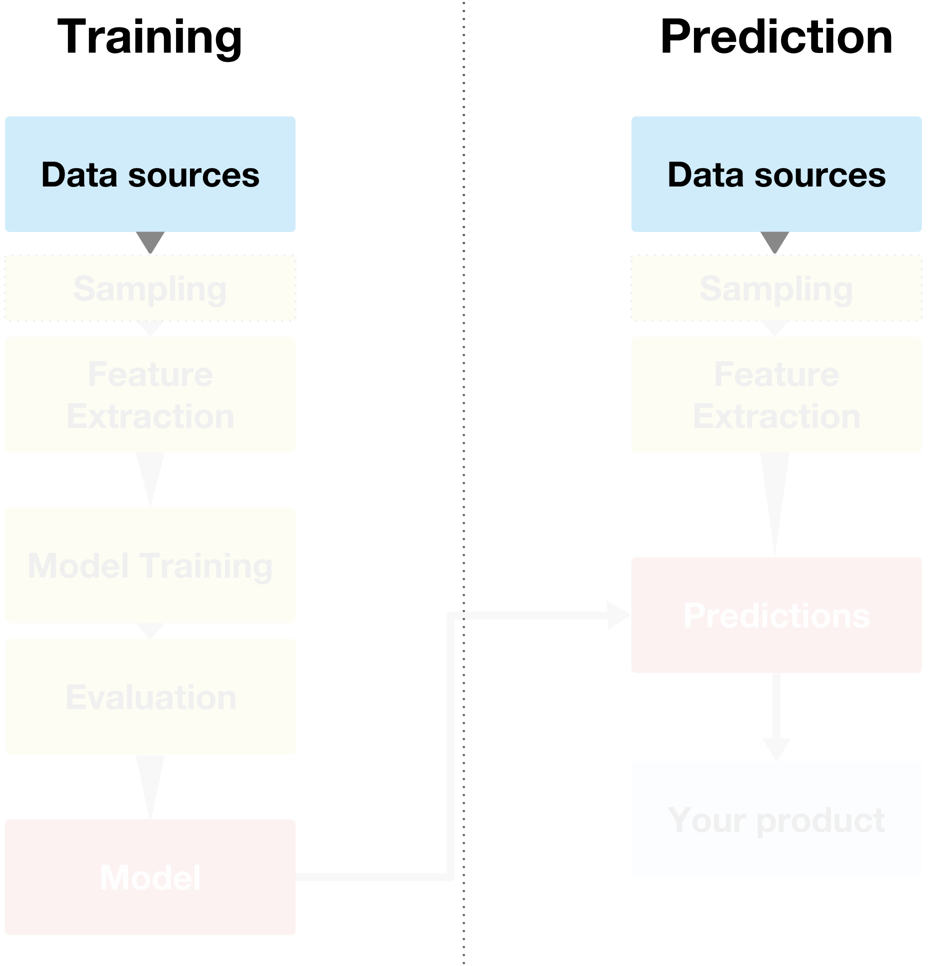
This presentation is **not** about **tooling**

This is a **framework** on how to **tackle** the **problem**

ML pipeline
A simplified view

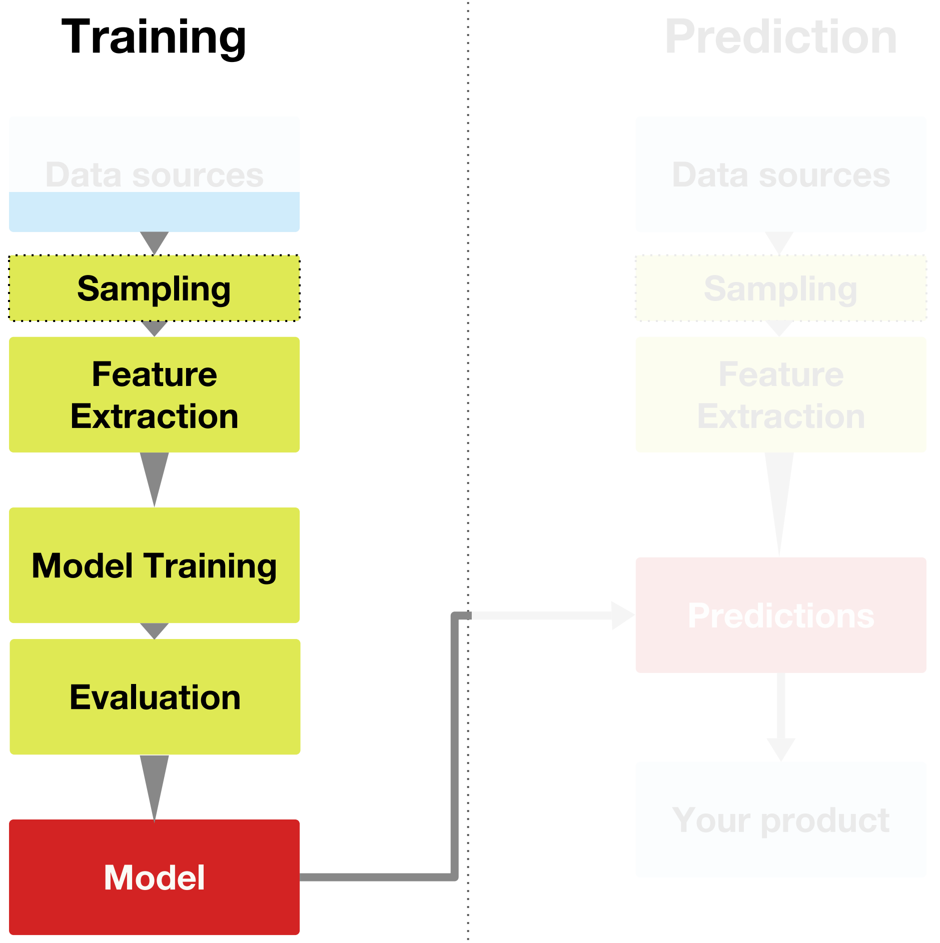


Your data source is **updating**

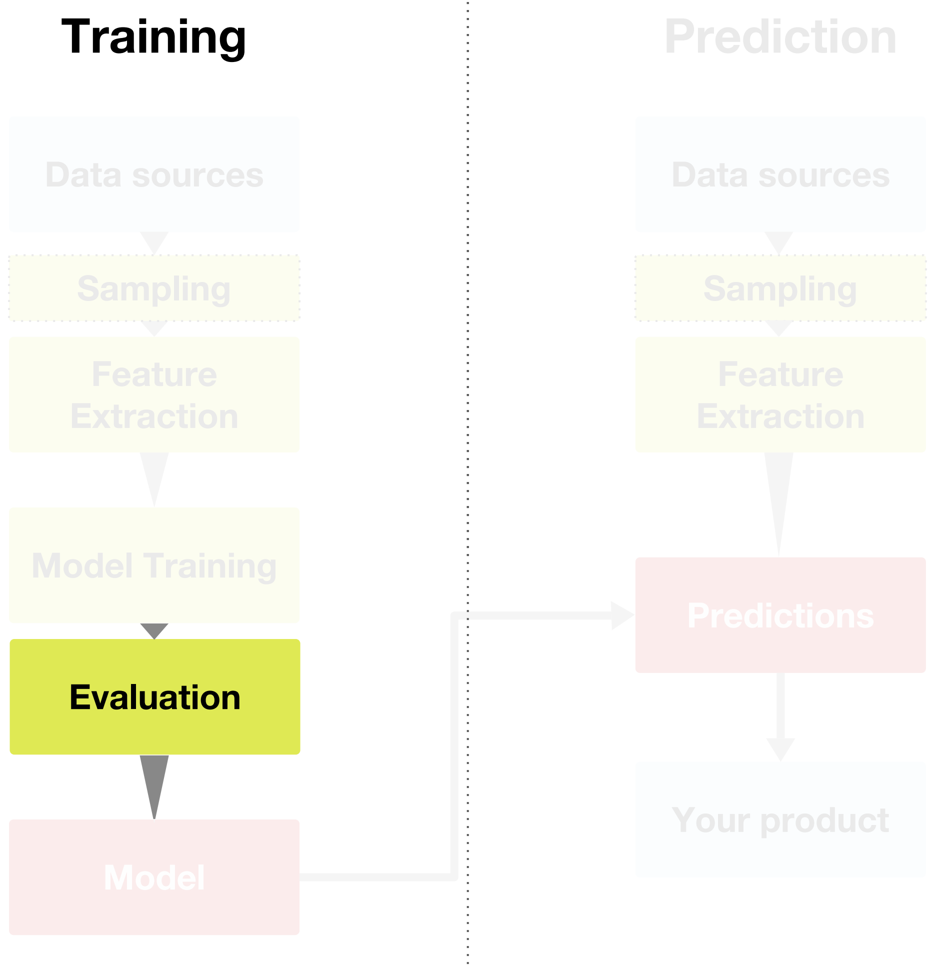


Updating the model

- Regular training
- Re-run strategies
- Scale



Evaluating Model

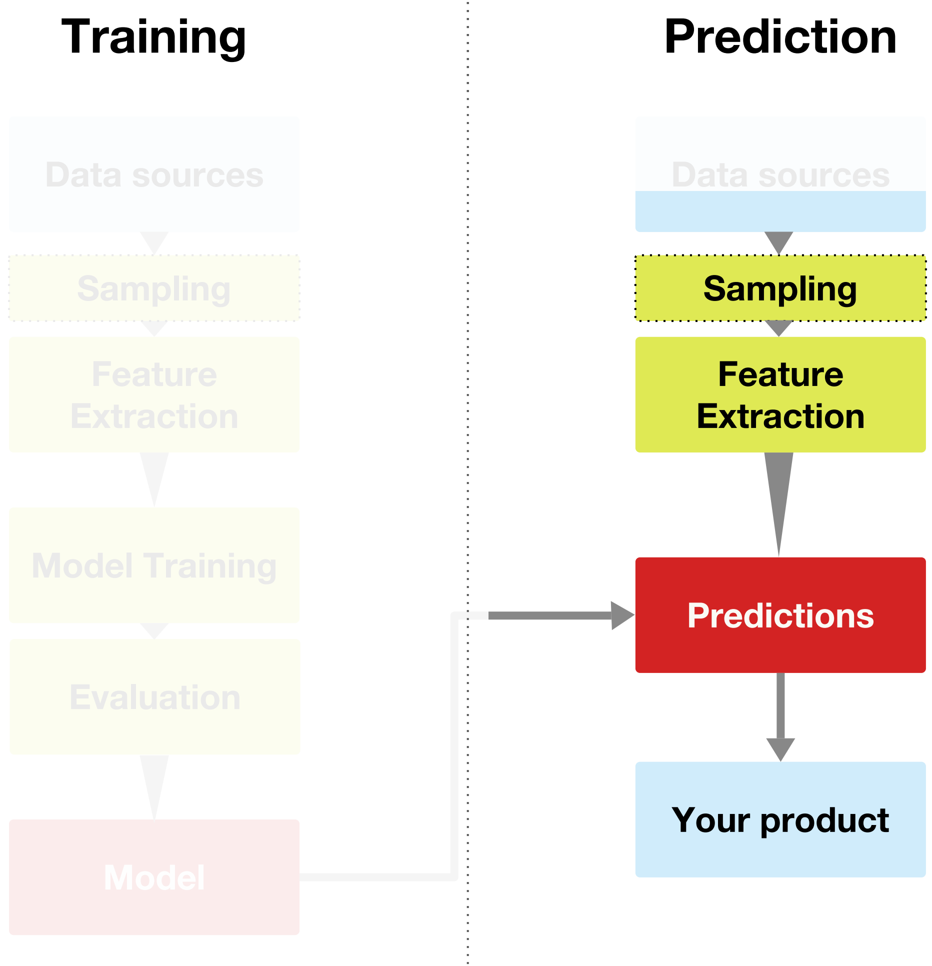


Evaluating Model

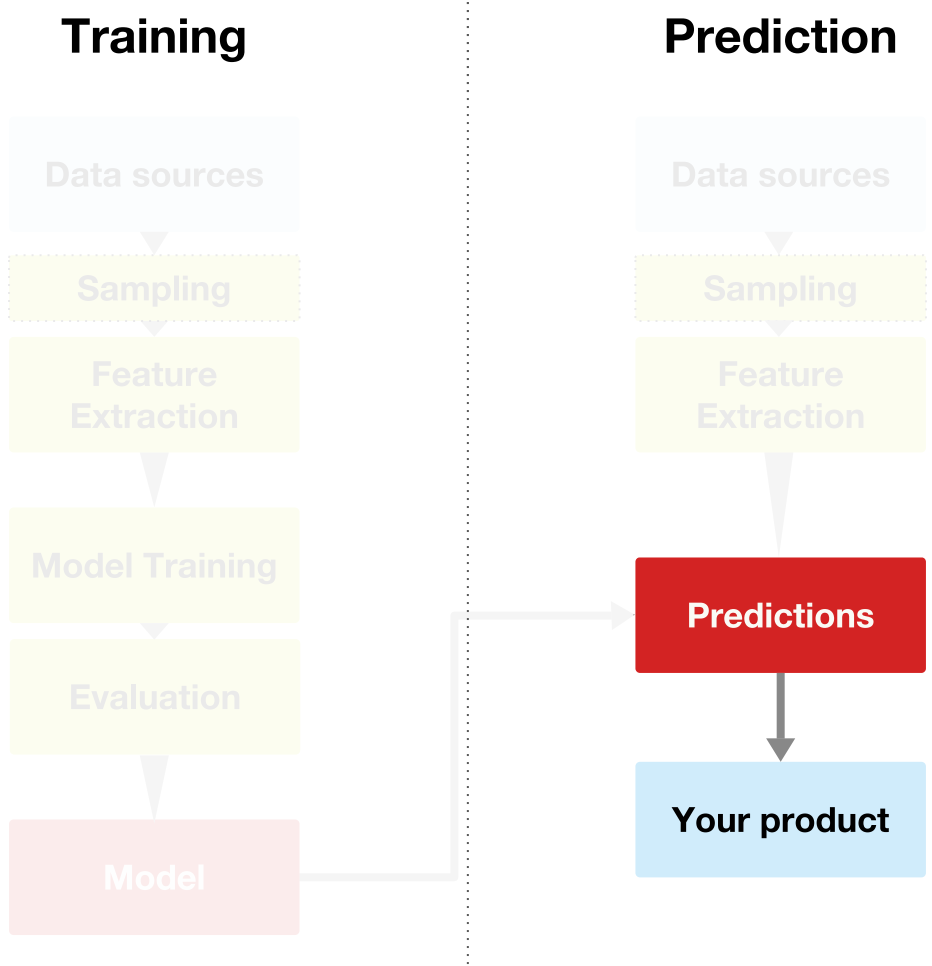
- Does my **evaluation metric** reflect how this model will be used in **production**?
- Consider both a **classic metric** and something that makes sense **business wise**.
- Think about **feedback loops**

Generating predictions

- Regular training
- Re-run strategies
- Scale



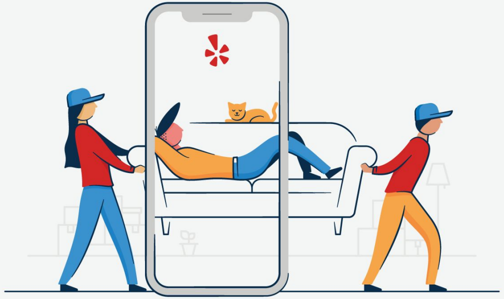
Using predictions



Measuring Success

- **Track the business metrics** you are trying to move
- **Confront** your hypothesis to reality
- **A/B test** different models
- Beware about **predicting what will happen anyway**. Think about having a **hold-out set**.

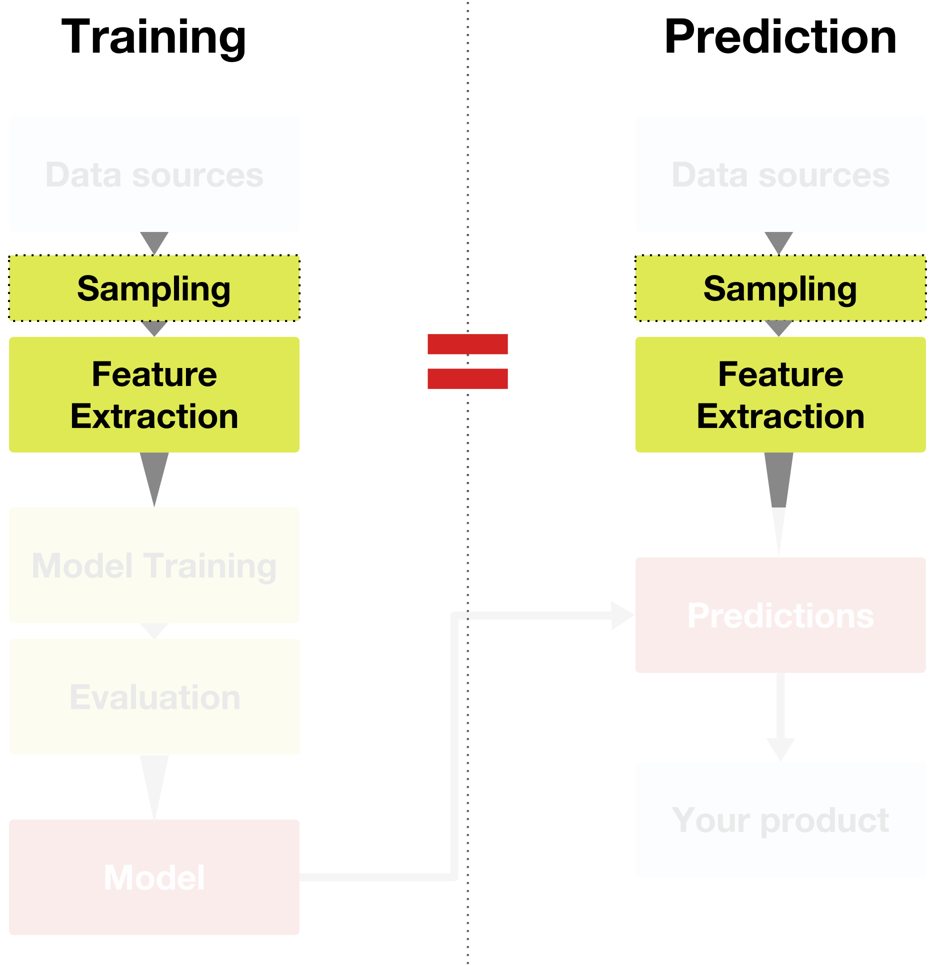
How to improve your development workflow to make the path to production easier



General Advice

- Use **containers, virtualenvs**
- **Persist** your models, logs, code ...
- Use **prod technologies** from the get go
- Rely on **already existing tools**

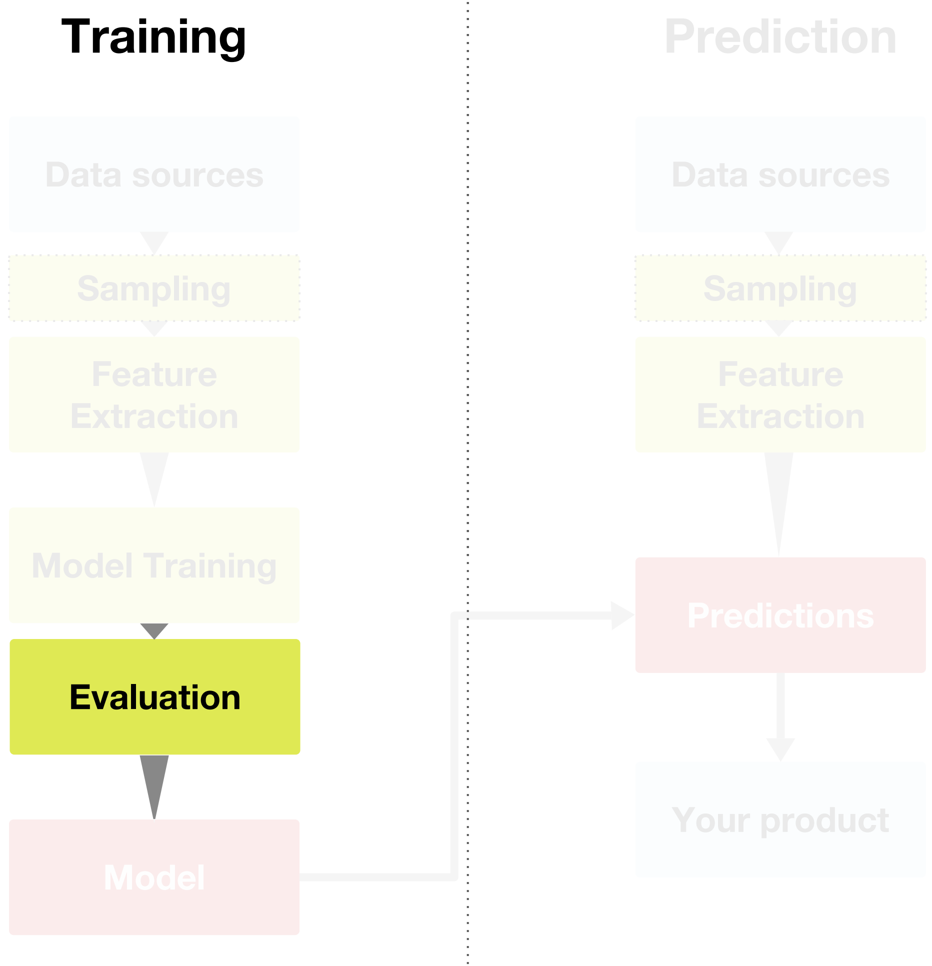
Feature Extraction



Feature Extraction

- Code should be **the same** between training and prediction
- **Unit Test** the feature extraction code
- Think about **edge cases**
- Write features as **code**
  Giant SQL queries are hard to review, test and maintain

Evaluating Model

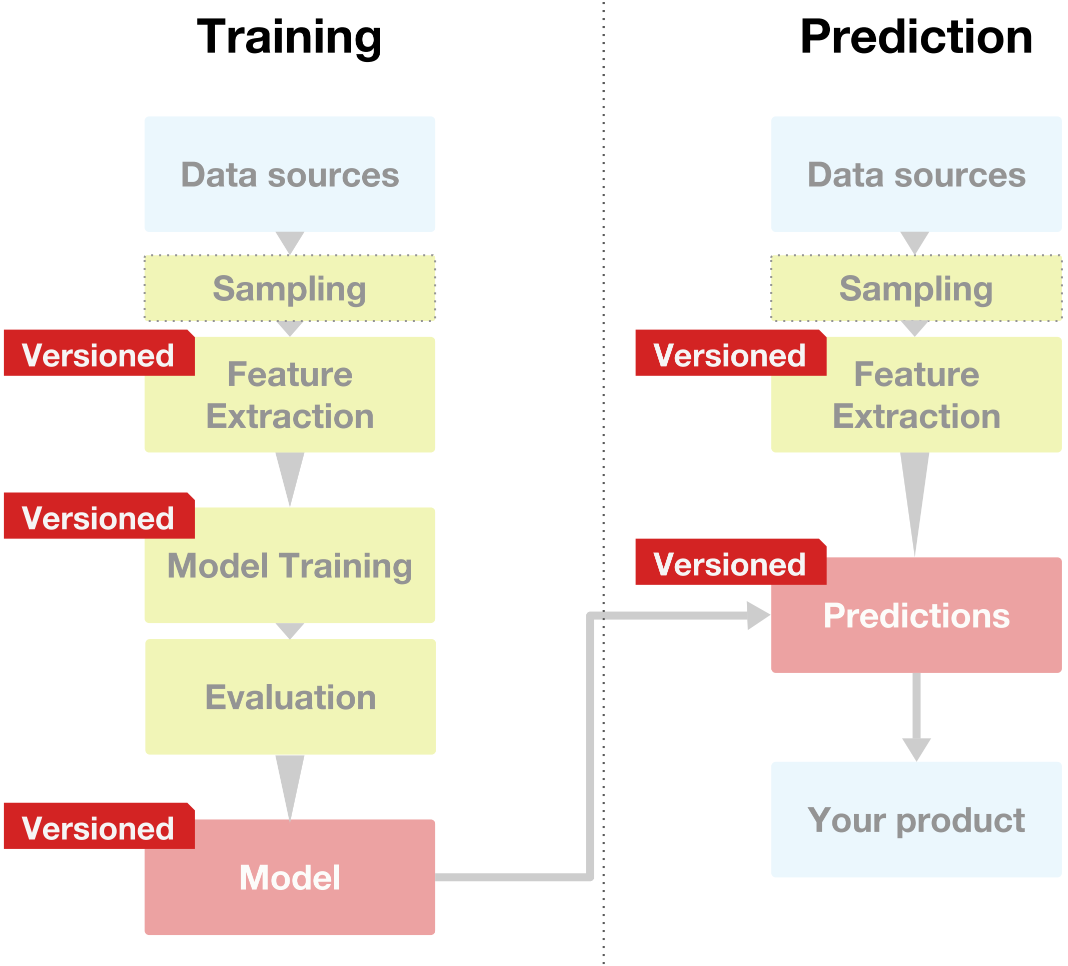


Evaluating Model

- Perform **feature importance analysis**, to be able to detect issues or big change between one training and the other.
- Set aside a **sanity check test set** for evaluation

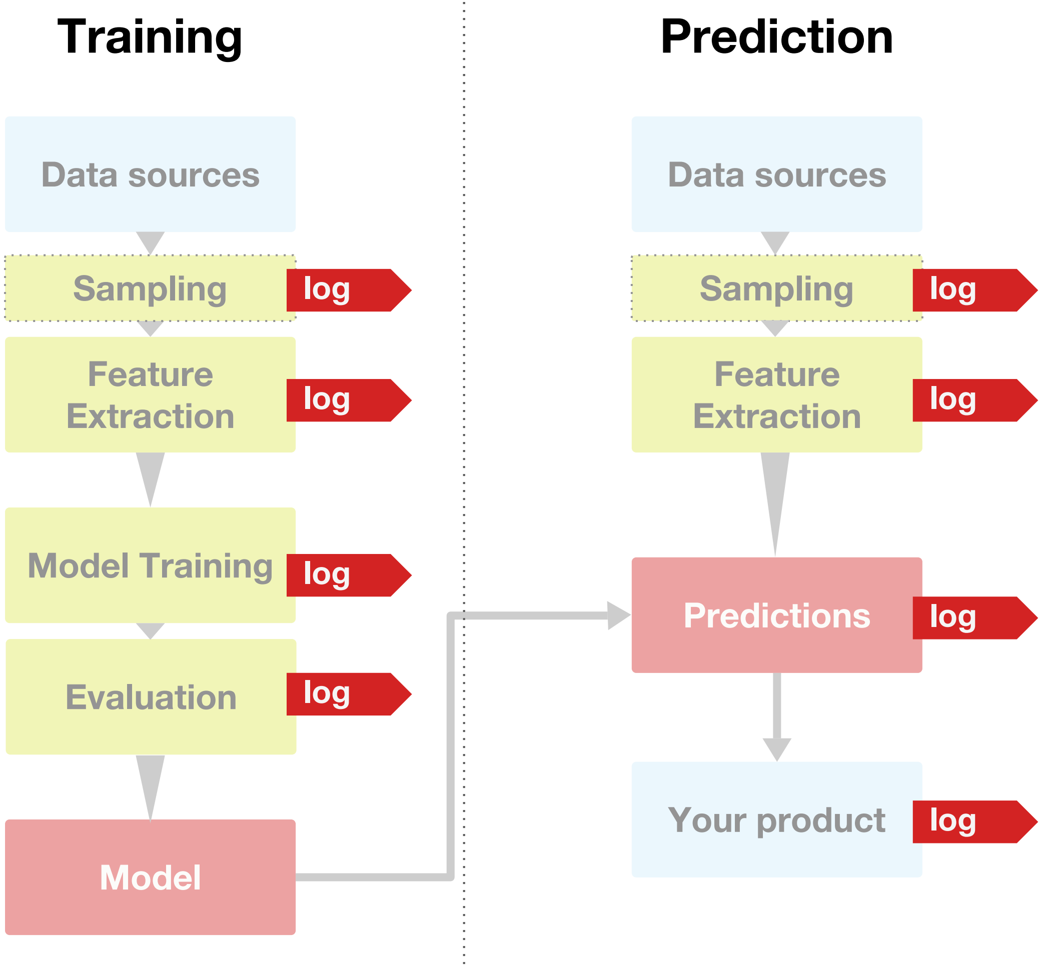
Version
All the things

Keep track of change

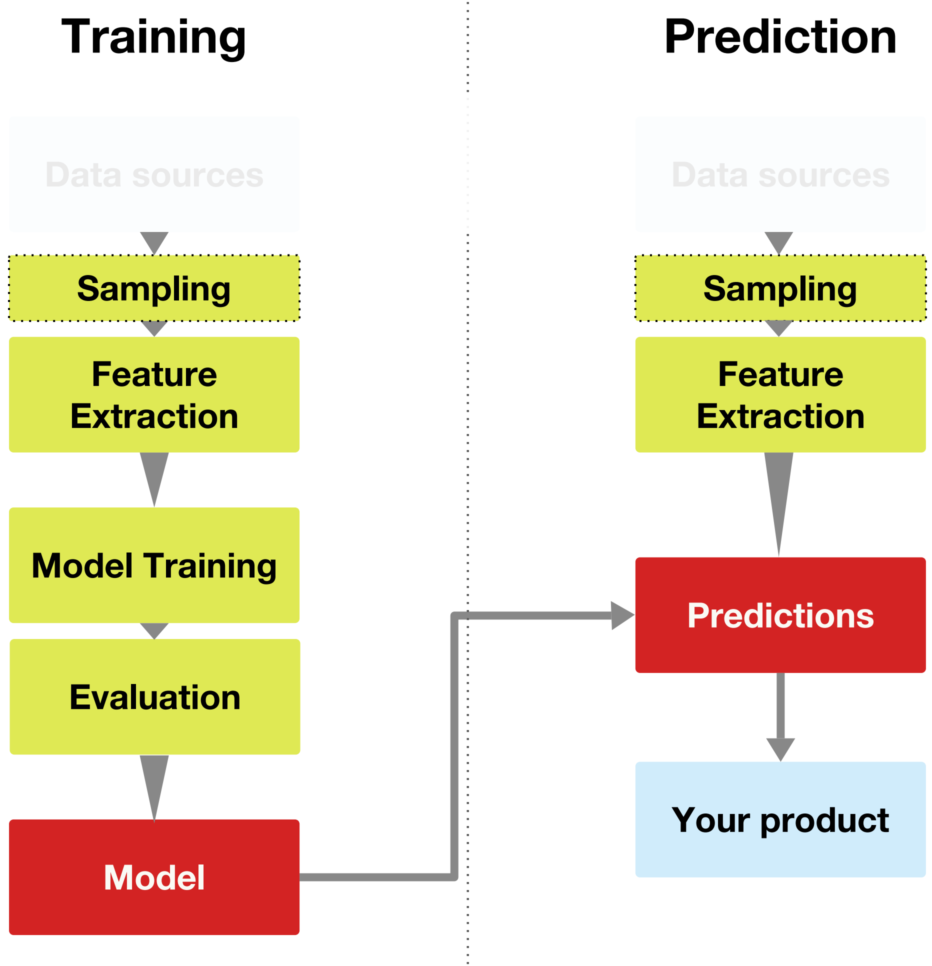


Log
All the things

And persist it



Monitoring the pipeline

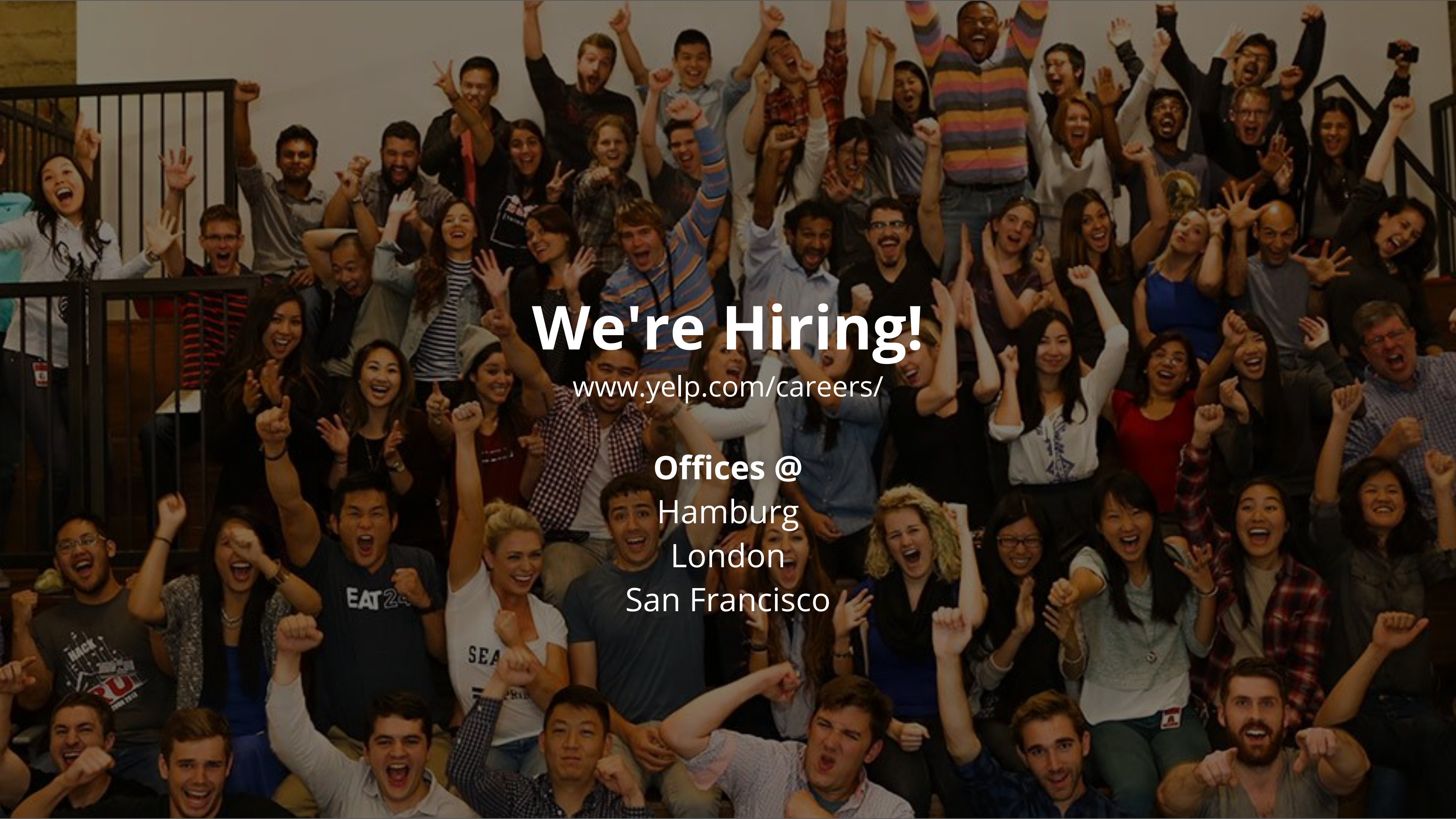


Monitoring the pipeline

- Keep track of the **number of prediction generated** (and alert when it's 0)
- Keep track of **timings**, to be able to see problems earlier
- Alert on **errors** in your pipeline's code
- Alert on the **business metrics** you are trying to move.
- Write **runbooks**

Closing word

- Design for **change**, and evolution.
- ML code is code, and all good practices **still apply**.
- **Verify your assumption** against reality. Really.



We're Hiring!

www.yelp.com/careers/

Offices @

Hamburg

London

San Francisco

Thank you!

Questions?