

# **Deploying ML in Production**

**Suhaib Mujahid**

[smujahid@mozilla.com](mailto:smujahid@mozilla.com)

# Outline

# Outline

- **Use Case:** Software Defect Prediction

# Outline

- **Use Case:** Software Defect Prediction
- **ML architecture**

# Outline

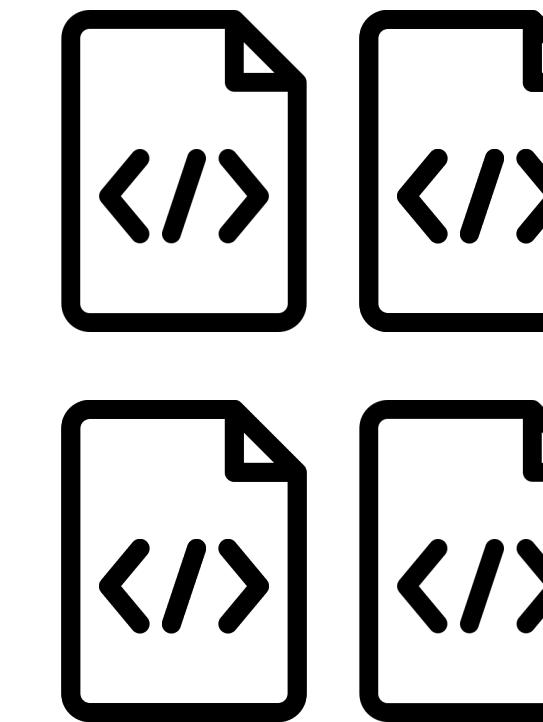
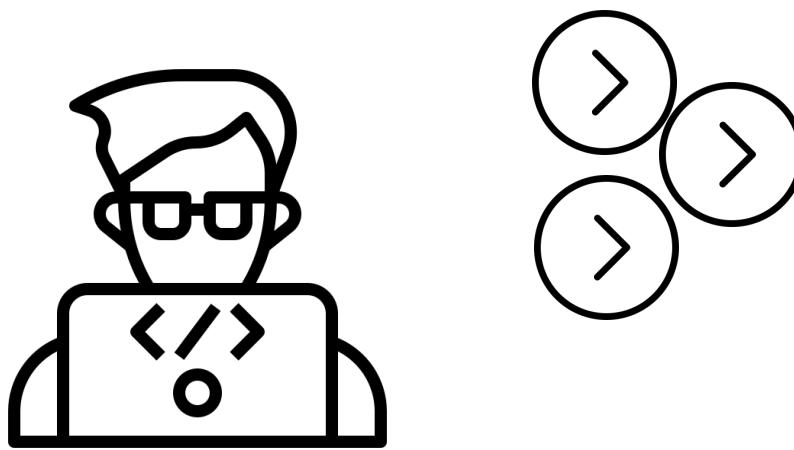
- **Use Case:** Software Defect Prediction
- **ML architecture**
- **Monitoring ML in production**

# Outline

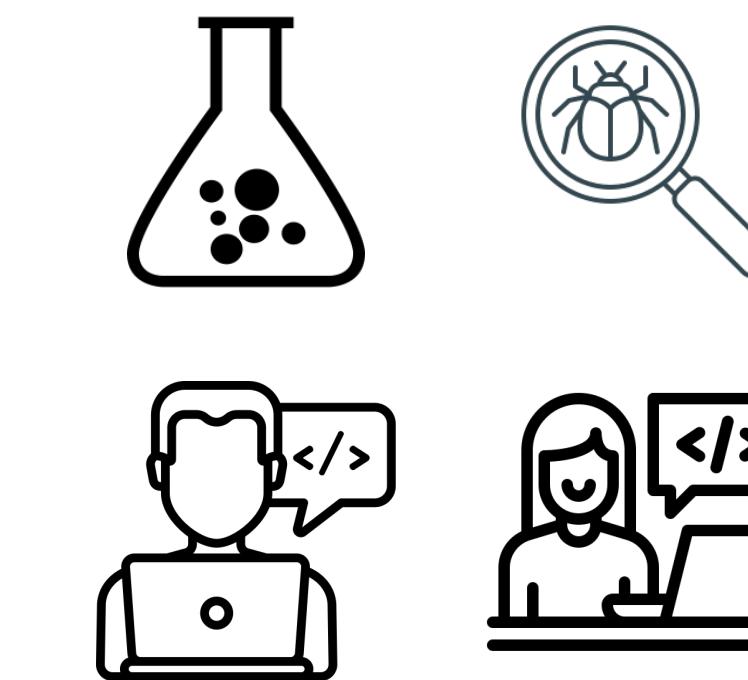
- **Use Case:** Software Defect Prediction
- **ML architecture**
- **Monitoring ML in production**
- **Retraining strategies**

# Software Development Process

## Develop

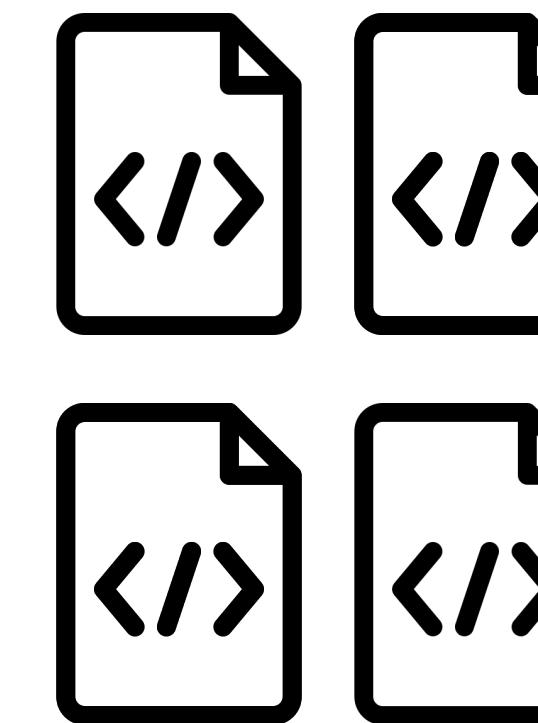
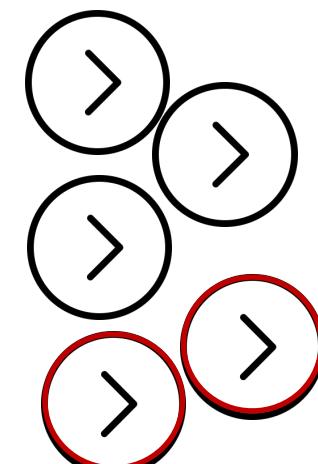
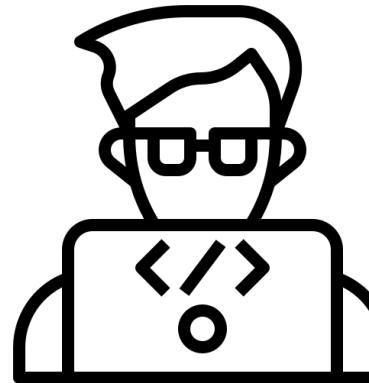


## Review & Test

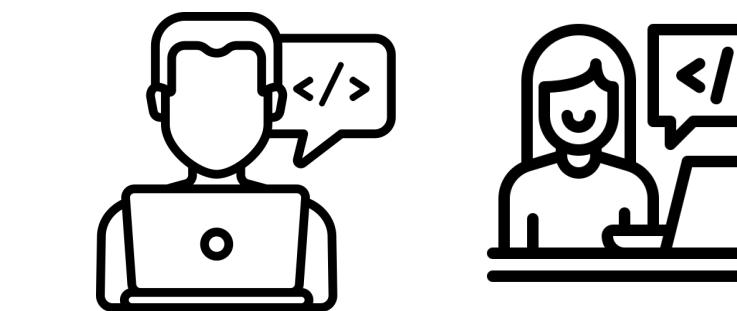
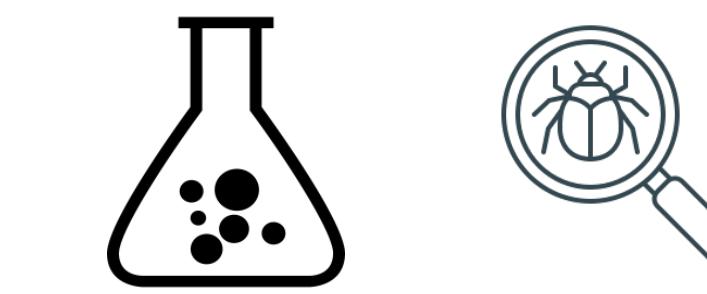


# Software Development Process

## Develop

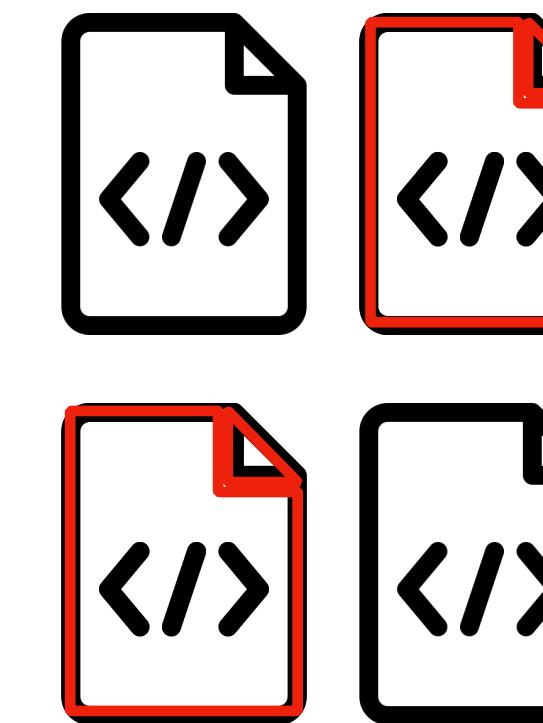
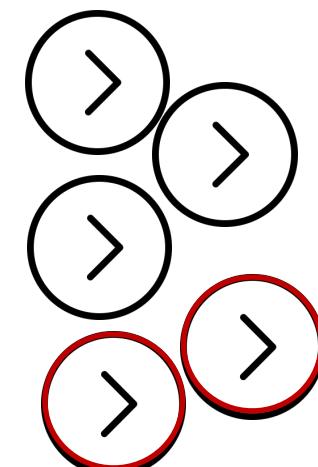
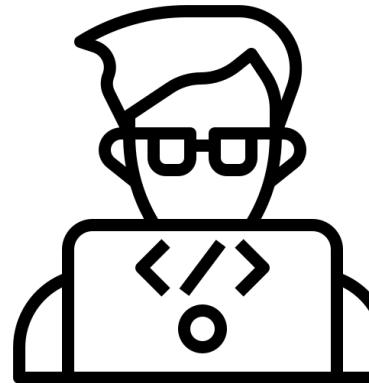


## Review & Test

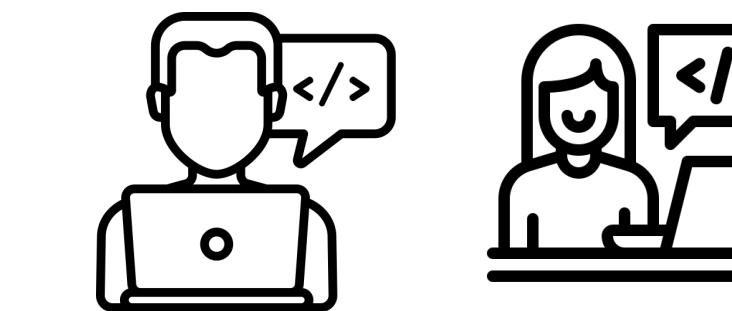
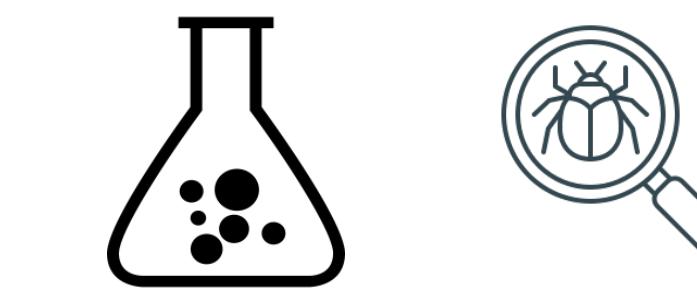


# Software Development Process

## Develop

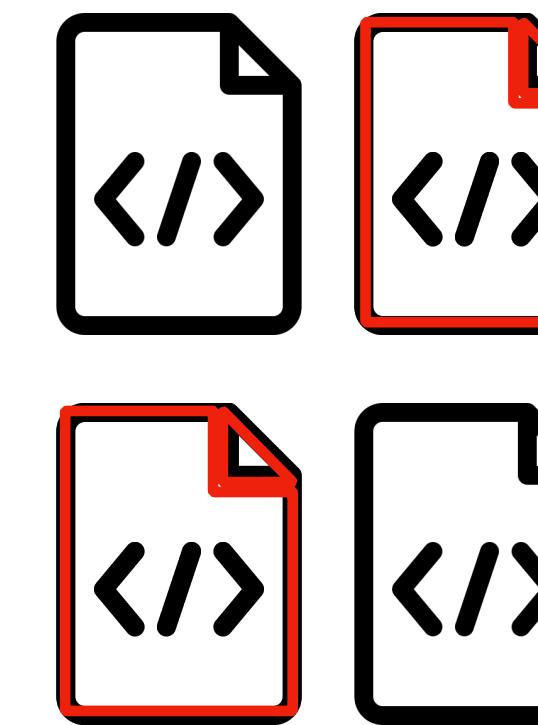
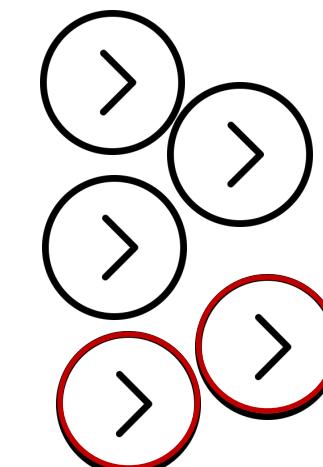
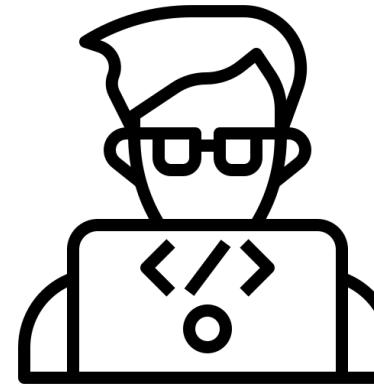


## Review & Test

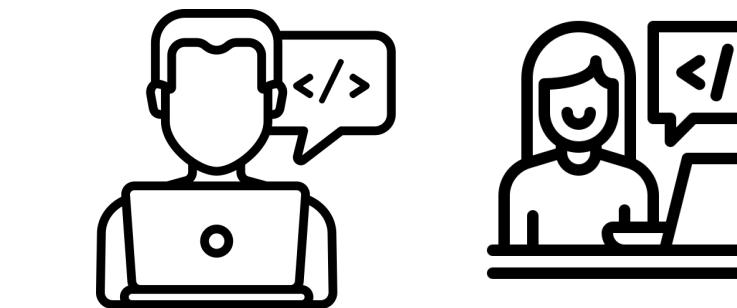
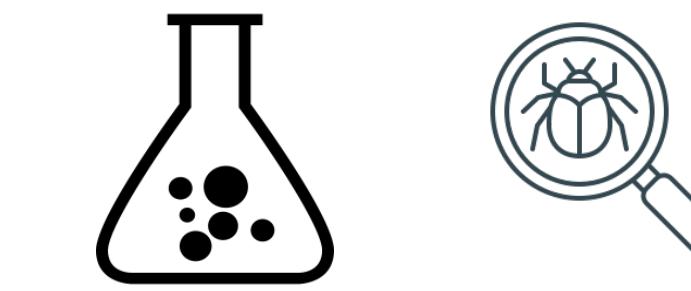


# Software Development Process

Develop



Review & Test

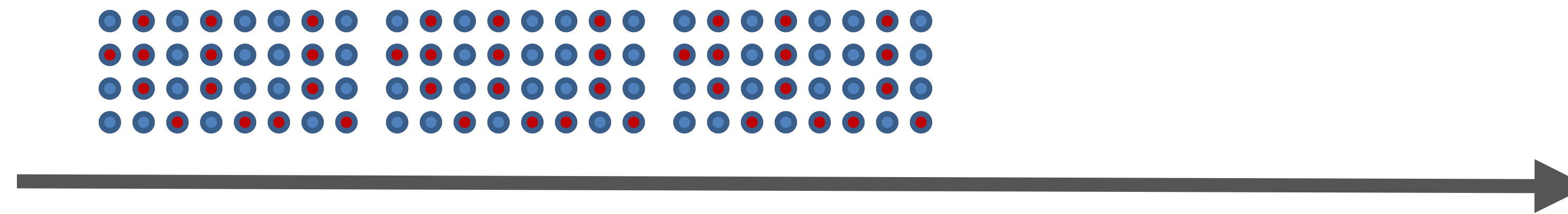


Defect  
Prediction

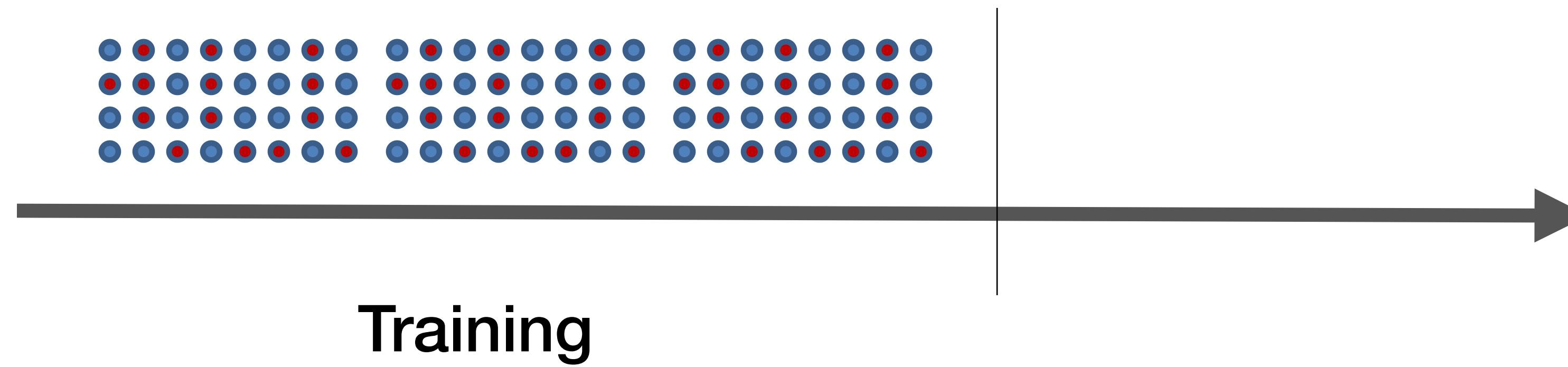
# How it works...



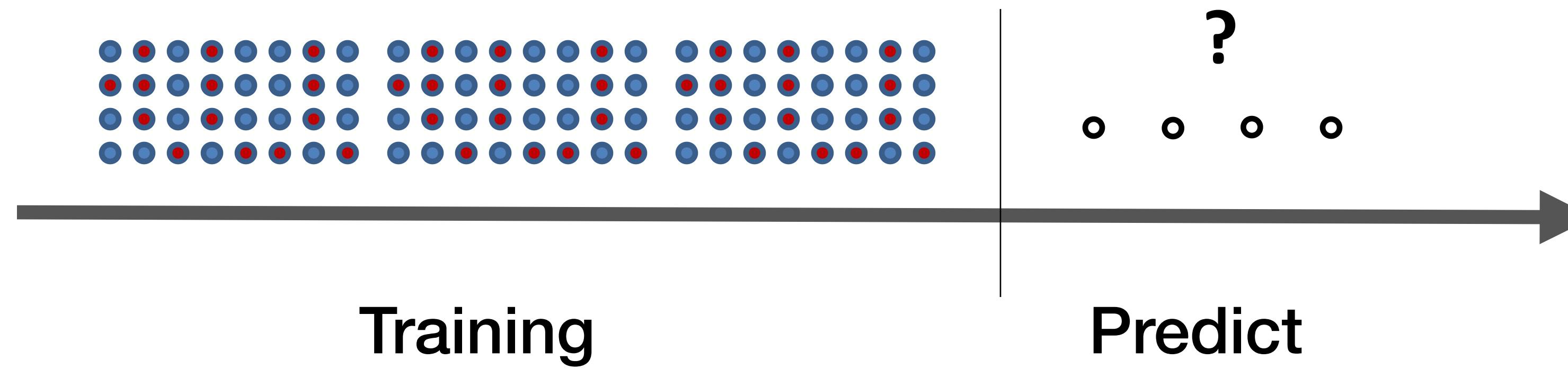
# How it works...



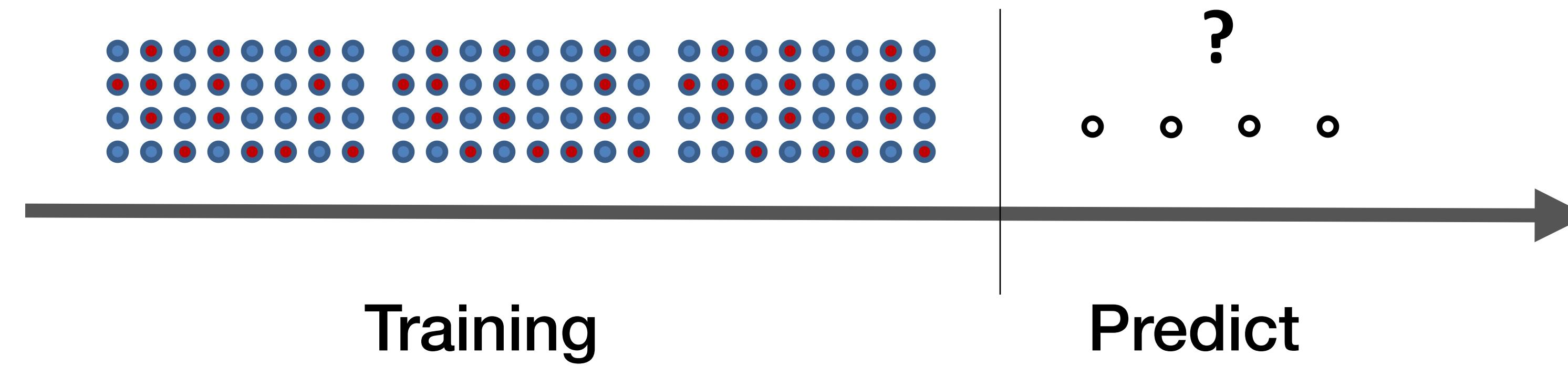
# How it works...



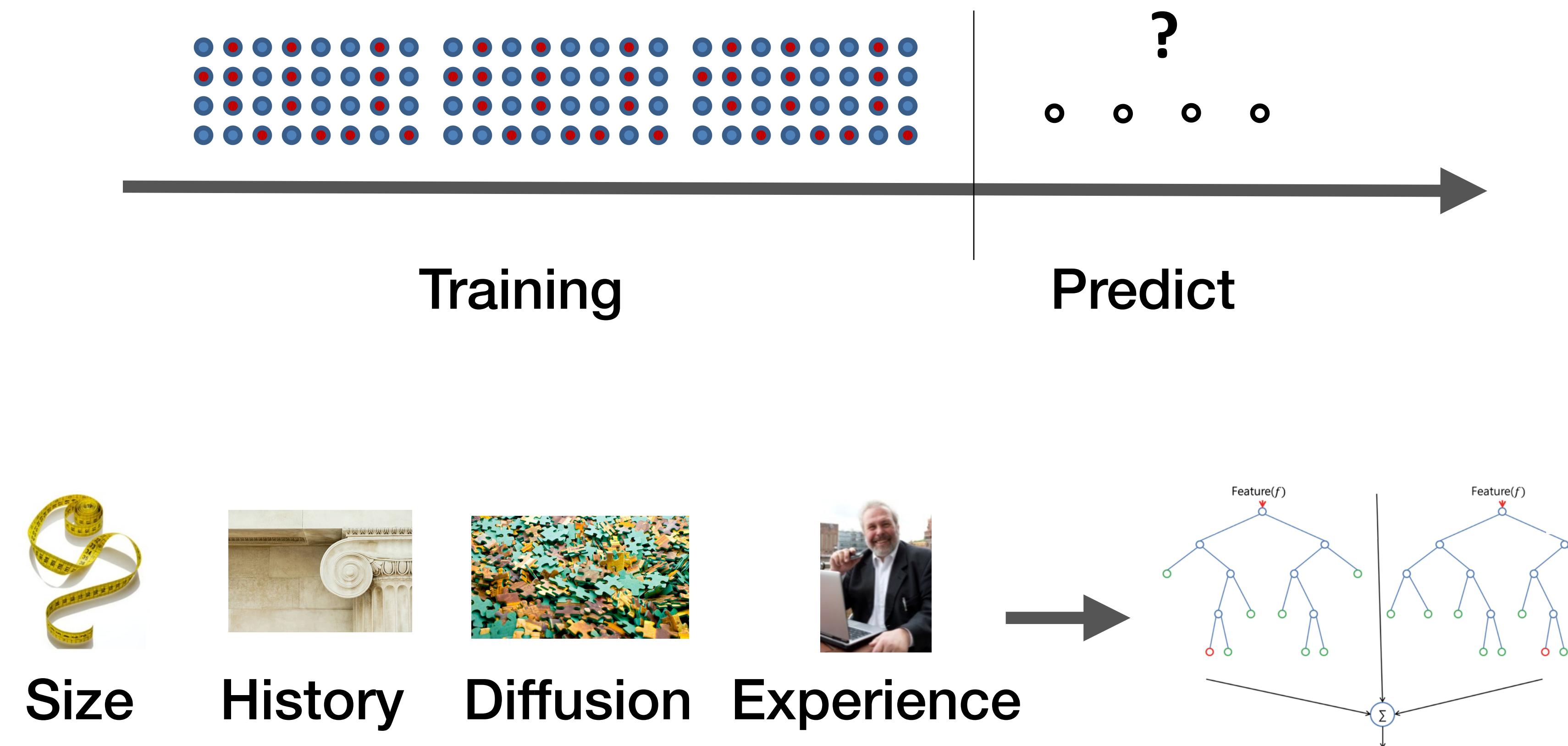
# How it works...



# How it works...



# How it works...



# Software Defect Prediction Use Case

<https://bit.ly/se4ai-repo>

The screenshot shows a GitHub repository page. The repository name is `suhaimujahid/se4ai-2022-03-21`, described as a `Public template`. The page includes navigation links for `Pull requests`, `Issues`, `Marketplace`, and `Explore`. The main content area displays a list of code commits by user `suhaimujahid`:

File	Message	Commit Hash	Time Ago
<code>data</code>	Add the data dumps	<code>4043d58</code>	2 hours ago
<code>Readme.md</code>	Add an example script		3 minutes ago
<code>example.py</code>	Add an example script		3 minutes ago
<code>requirements.txt</code>	Add an example script		3 minutes ago

Below the commits, there is a `Readme.md` file with an edit icon.

**About** section details:

- No description, website, or topics provided.
- README
- 0 stars
- 1 watching
- 0 forks

**Releases** section:

No releases published  
Create a new release

**Packages**

**SOEN 691: Engineering AI-based Software Systems (March 21th, 2022)**

# Metrics

```
type ChangeMeasures struct {
    // NS is the number of modified subsystems
    NS
    // ND is the number of modified directories
    ND
    // NF is the number of modified files
    NF
    // Entropy is the distribution of modified code across each file
    Entropy
    // LA is lines of code added in the commit
    LA
    // LD is lines of code deleted in the commit
    LD
    // HA is hunks of code added in the commit
    HA
    // HD is hunks of code deleted in the commit
    HD
    // LT is lines of code in the modified files before the commit
    LT
    // NDEV is the number of developers that changed the modified files in the past
    NDEV
    // AGE is the average time interval between the last and the current change of modified files
    AGE
    // NUC is the number of unique last changes that touched the modified files
    NUC
    // EXP is the developer experience, measured by the number of submitted commits
    EXP
    // REXP is the recent developer experience in the modified files
    REXP
    // SEXP is the developer experience on modified subsystems
    SEXP
}
```

# Serving the Model

# Serving the Model

**Main challenges:**

# Serving the Model

## Main challenges:

- Reproducibility

# Serving the Model

## Main challenges:

- Reproducibility
  - Dependency management

# Serving the Model

## Main challenges:

- **Reproducibility**
  - Dependency management
    1. Constrain the dependencies of the model

# Serving the Model

## Main challenges:

- **Reproducibility**
  - Dependency management
    - 1. Constrain the dependencies of the model
    - 2. Use containers

# Serving the Model

## Main challenges:

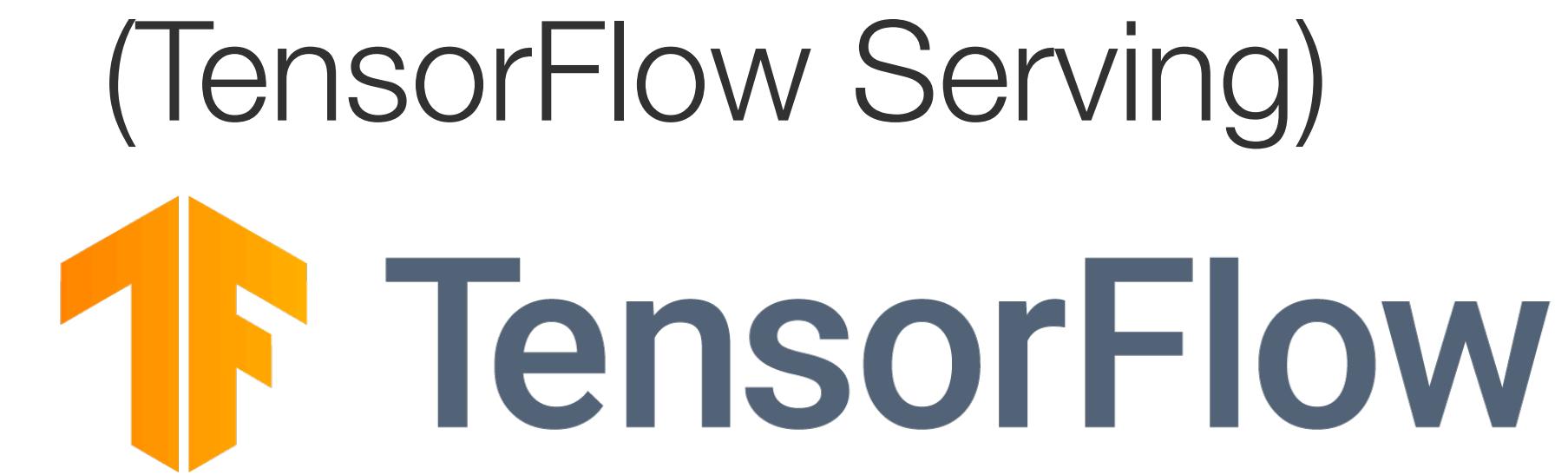
- **Reproducibility**
  - Dependency management
    1. Constrain the dependencies of the model
    2. Use containers
- **Scalability**

# Serving the Model

## Main challenges:

- **Reproducibility**
  - Dependency management
    1. Constrain the dependencies of the model
    2. Use containers
- **Scalability**
- **Performance (latency)**

# Framework Serving Tools



# Framework Interoperability

The ONNX website features a dark blue header with the ONNX logo and navigation links: GET STARTED, SUPPORTED TOOLS, NEWS, ABOUT, SLACK, and GITHUB. The main section is titled "Open Neural Network Exchange" with the subtitle "The open standard for machine learning interoperability". A "GET STARTED" button is prominent. Below this, a box contains the text: "ONNX is an open format built to represent machine learning models. ONNX defines a common set of operators - the building blocks of machine learning and deep learning models - and a common file format to enable AI developers to use models with a variety of frameworks, tools, runtimes, and compilers." A "LEARN MORE" link is provided. The "KEY BENEFITS" section includes two items: "Interoperability" (represented by a server icon) and "Hardware Access" (represented by a GPU icon). Each benefit has a brief description and a "SUPPORTED FRAMEWORKS" or "SUPPORTED ACCELERATORS" link.

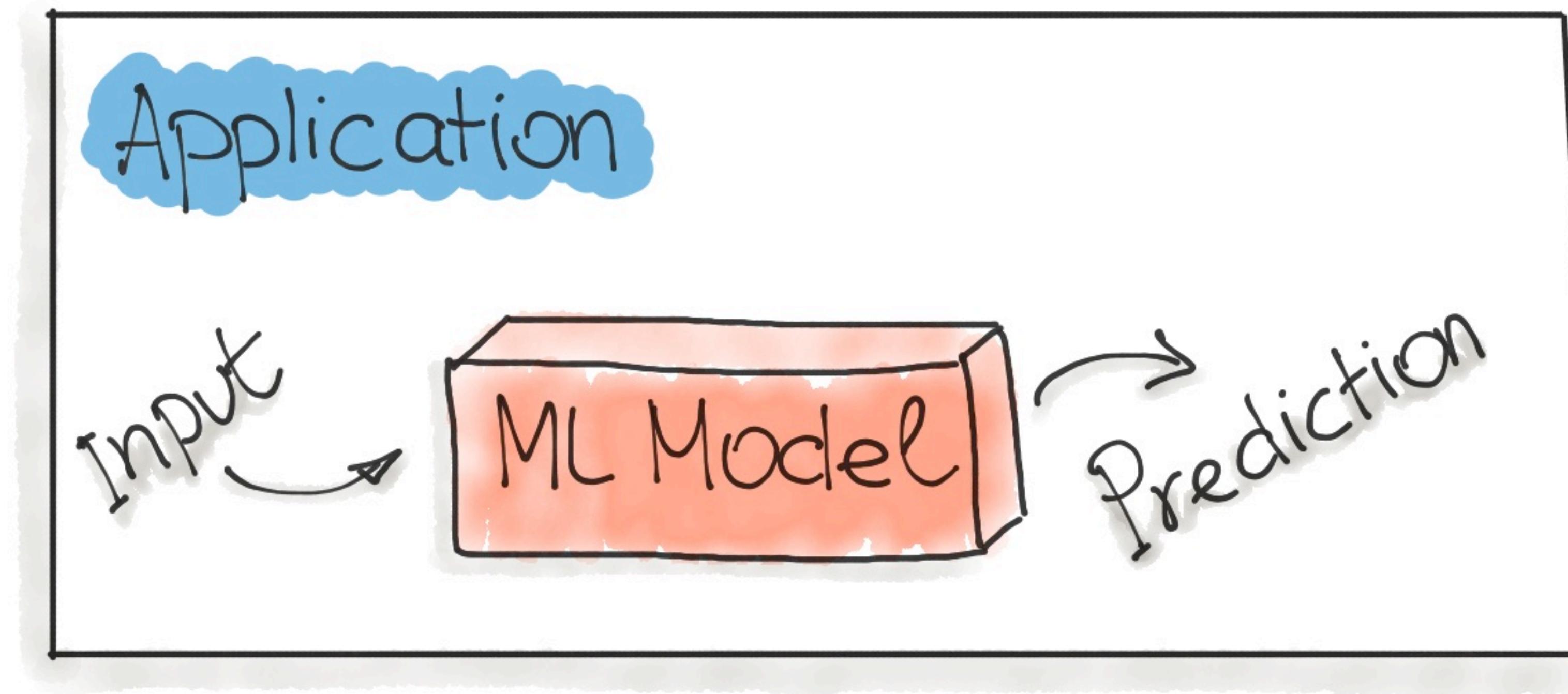
The "Build Model" section of the ONNX website lists various frameworks and converters. The heading "Frameworks & Converters" includes the sub-instruction "Use the frameworks you already know and love." The grid contains logos for Caffe2, Yandex CatBoost, Chainer, Cognitive Toolkit, CoreML, Keras, LibSVM, MATLAB, MindSpore, MyCaffe, NCNN, NeoML, Neural Network Libraries, PaddlePaddle, PyTorch, SAS, Siemens, Singa, SciKit Learn, TensorFlow, Tengine, XGBoost, and WOLFRAM.

# Framework Interoperability

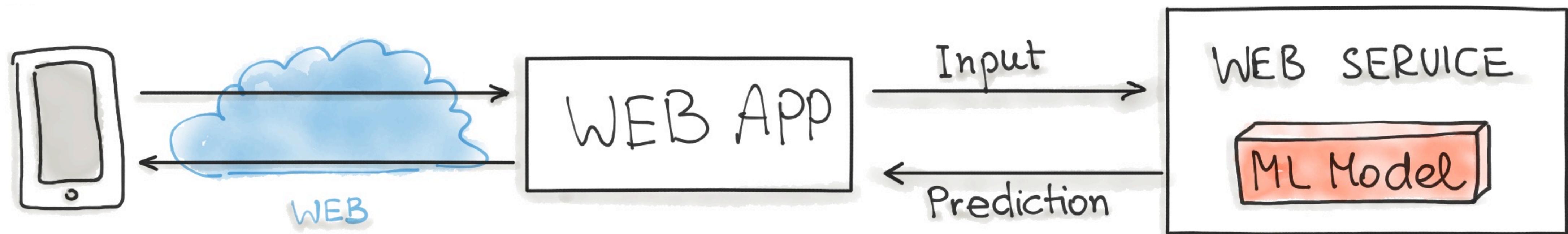
The image displays three screenshots of the ONNX website, illustrating its features and interoperability across various machine learning frameworks.

- ONNX Main Page:** Shows the "Open Neural Network Exchange" logo and tagline "The open standard for machine learning interoperability". It includes a "GET STARTED" button and a detailed description of ONNX as an open format for machine learning models.
- ONNX Build Model Page:** Features a "Build Model" section with a "Frameworks & Converters" heading. It lists supported frameworks and converters, each with a logo and name: Caffe2, Yandex CatBoost, Chainer, Cognitive Toolkit, Keras, LibSVM, MATLAB, MindSpore, MyCaffe, NCNN, NeoML, Neural Network Libraries, PaddlePaddle, and SciKit Learn.
- ONNX Runtime Page:** Focuses on optimization and acceleration. It features the "ONNX RUNTIME" logo and tagline "Optimize and Accelerate Machine Learning Inferencing and Training". It highlights three key benefits: "Speed up machine learning process", "Plug into your existing technology stack", and "Build using proven technology".

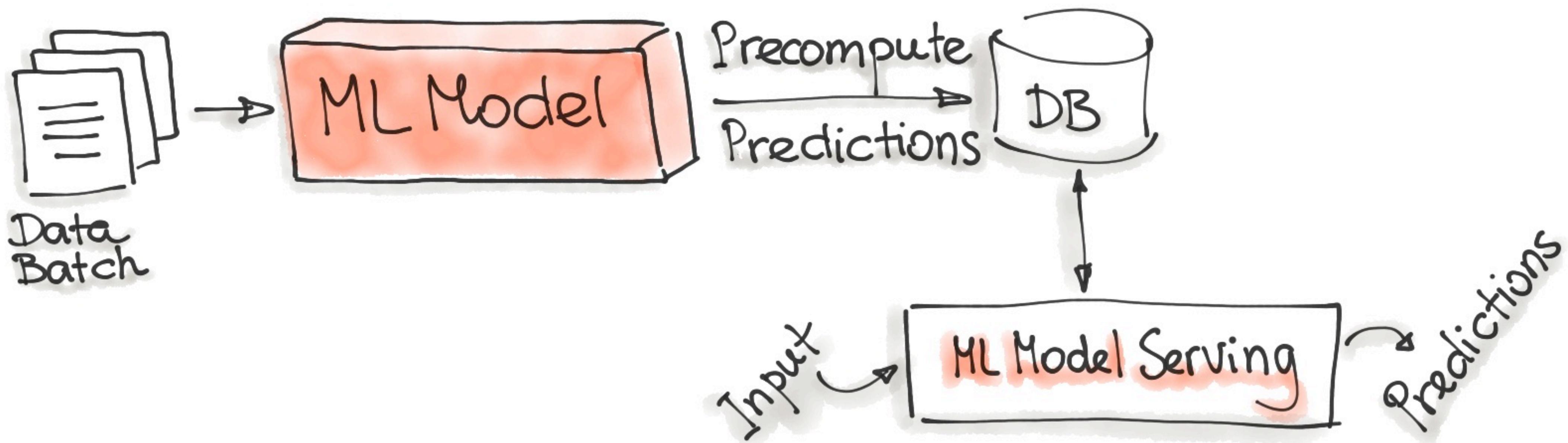
# Model-as-Dependency



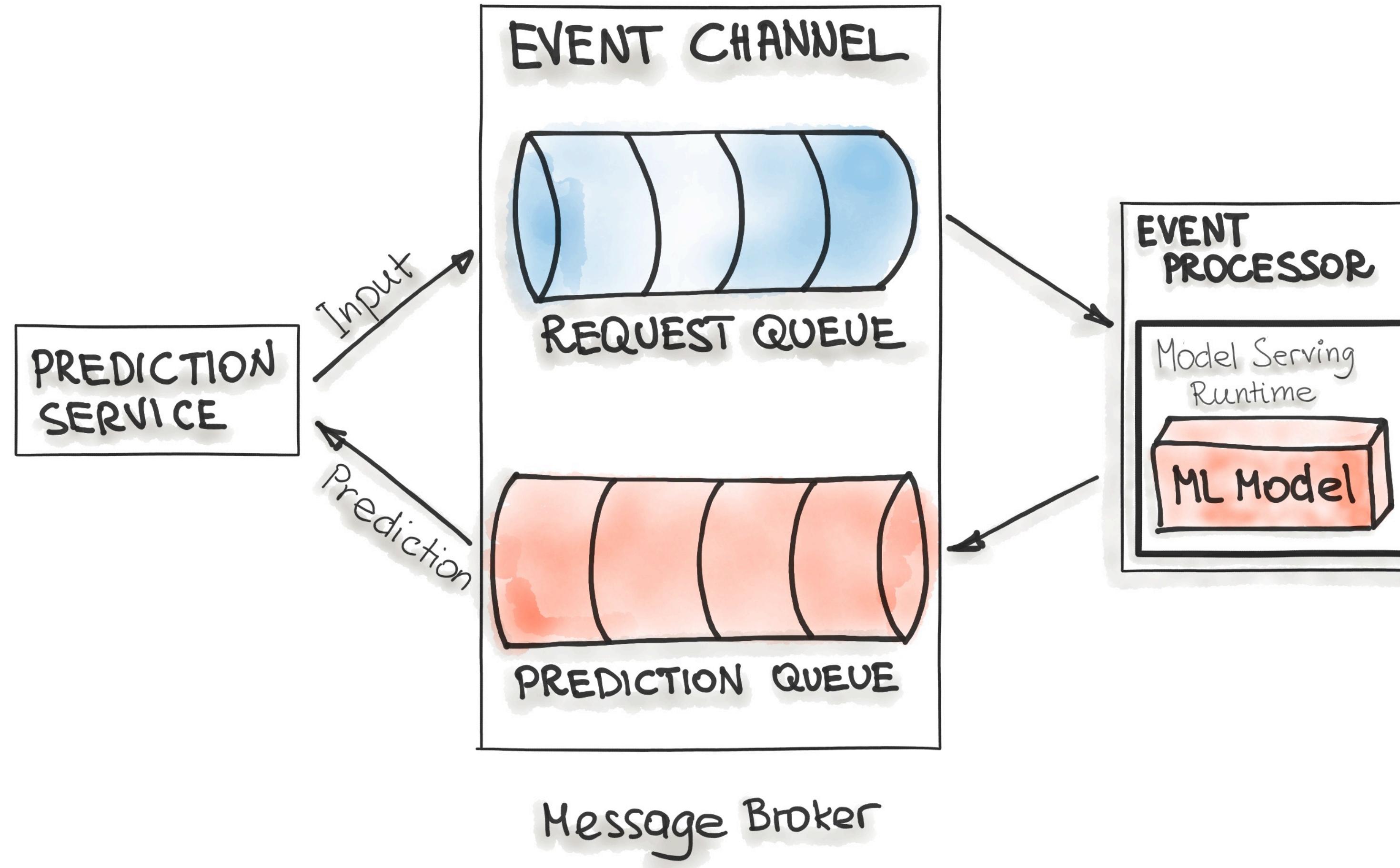
# Model-as-Service



# Precompute Serving Pattern



# Model-on-Demand



**Which architecture** will you  
adopt to deploy our  
use case?

# What next?

Models will keep **working as expected forever** after deployment

# What next?

Models will keep ~~working as expected~~  
~~forever after deployment~~

# What next?

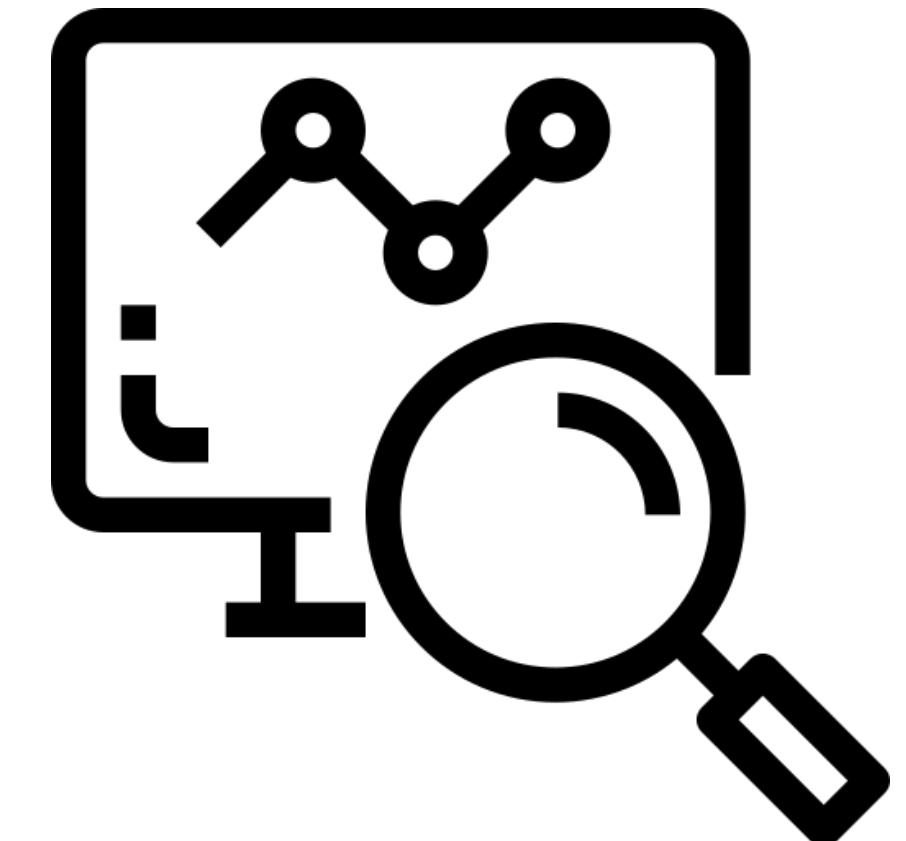
Models will keep ~~working as expected~~  
~~forever after deployment~~

Deployment is not the last  
stage in model management,  
**it's just the beginning**

# What we should monitor?

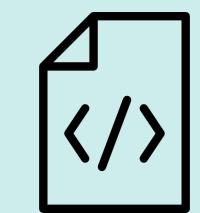
**A good model monitoring pipeline should monitor:**

- The availability of the model
- Computational performance of the ML system
- **Model prediction and performance**

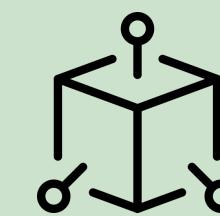


# How to continuously monitor models in production?

In machine learning, the development pipeline includes **3 levels of changes**



code



model



data

# Monitoring is more central to ML than for traditional software



The image shows the official website for MLflow. At the top left is the 'mlflow™' logo in white and blue. To the right are navigation links: 'DOCS', 'COMMUNITY', 'CODE', and social media icons for LinkedIn and Twitter. Below the navigation is a large, abstract blue background image featuring a 3D wireframe mesh of glowing blue dots, resembling a neural network or data flow. In the lower half of the page, centered text reads: 'An open source platform for the machine learning lifecycle'.

An open source platform for the  
machine learning lifecycle

# Add MLflow to the Example

```
example.py
.... @@ -5,6 +5,9 @@ from sklearn.model_selection import RandomizedSearchCV
5   5     from sklearn.model_selection import train_test_split
6   6     from sklearn.metrics import classification_report
7   7
8  8     +import mlflow
9  9     +import mlflow.sklearn
10 10    +
11 11    max_training_rows = 10000
12 12    target = 'buggy'
13 13    date_column = 'author_date'
.... @@
14 14    @@ -66,6 +69,13 @@ def main():
15 15
16 16        report = classification_report(y_test, y_pred)
17 17        print(report)
18 18
19 19        +    mlflow.sklearn.autolog()
20 20        +    mlflow.log_param("ignored_days", ignored_days)
21 21        +    mlflow.log_metric("training_rows", len(X_train))
22 22        +    mlflow.log_metric("testing_rows", len(X_test))
23 23        +    mlflow.sklearn.log_model(model, "model")
24 24        +    mlflow.sklearn.eval_and_log_metrics(model, X_test, y_test, prefix="val_")
25 25        +
26 26
27 27    if __name__ == '__main__':
28 28        main()
```

# Data validation

**Your Model Is Only as Good as Your Data Quality**  
**(Garbage in, Garbage out)**

# Data validation

**Your Model Is Only as Good as Your Data Quality  
(Garbage in, Garbage out)**

**Major data quality issues to monitor:**

1. Data Schema
2. Data Drift
3. Concept Drift

# What is data drift?

Data-drift is defined as **a variation in the production data** from the data that was used to test and validate the model before deploying it in production.

# How to monitor data drift in production?

# How to monitor data drift in production?

- **Rule-based methods**
  - Calculating the data range
  - Comparing data attributes

# How to monitor data drift in production?

- **Rule-based methods**
  - Calculating the data range
  - Comparing data attributes
- **Statistical methods**
  - Compare the data distributions

# How to monitor data drift in production?

- **Rule-based methods**
  - Calculating the data range
  - Comparing data attributes
- **Statistical methods**
  - Compare the data distributions
- **Continuously monitor your data with advanced MLOps tools**
  - Fiddler AI platform to monitor data drift in production

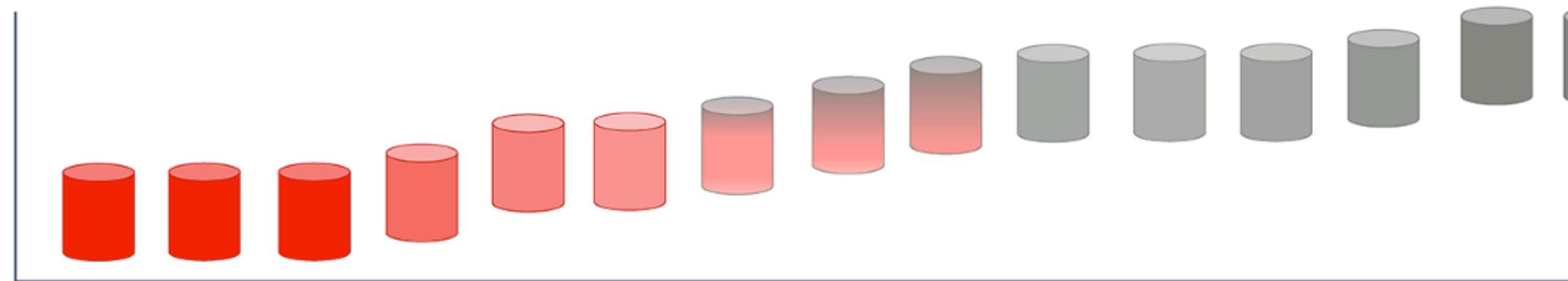
# What is concept drift?

Concept drift occurs when the **patterns**  
**the model learned no longer hold**

# What is concept drift?

Concept drift occurs when the **patterns**  
**the model learned no longer hold**

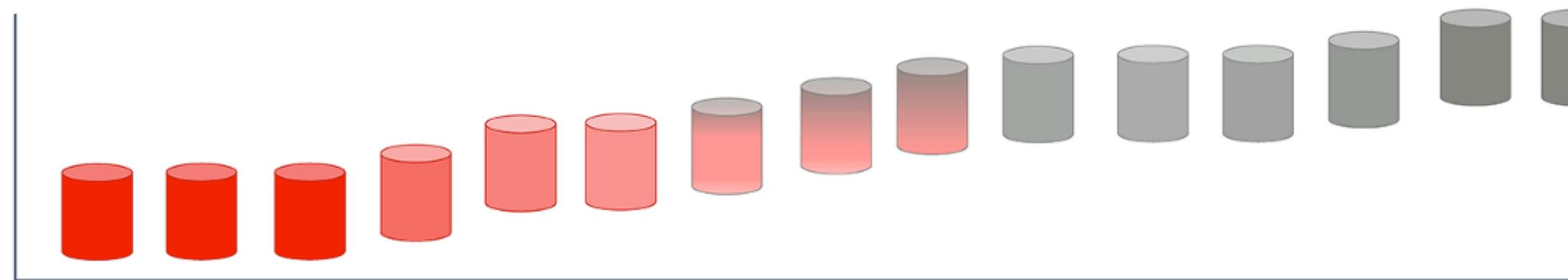
## Gradual concept drift



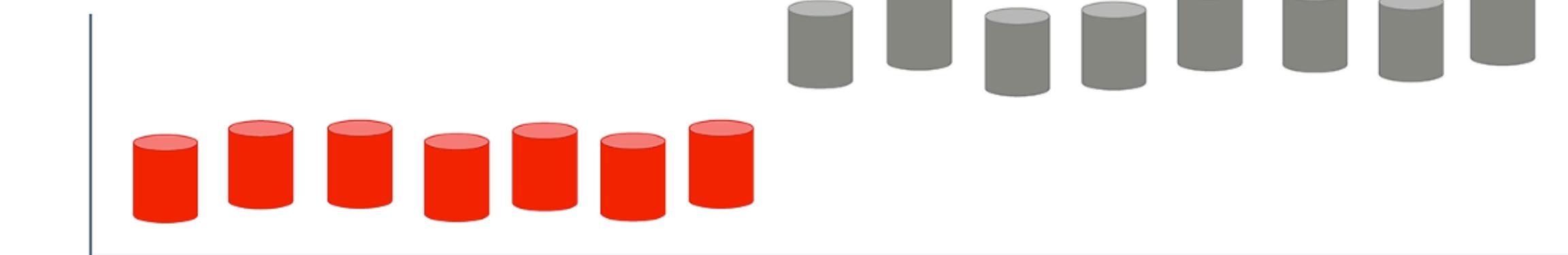
# What is concept drift?

Concept drift occurs when the **patterns**  
**the model learned no longer hold**

**Gradual concept drift**

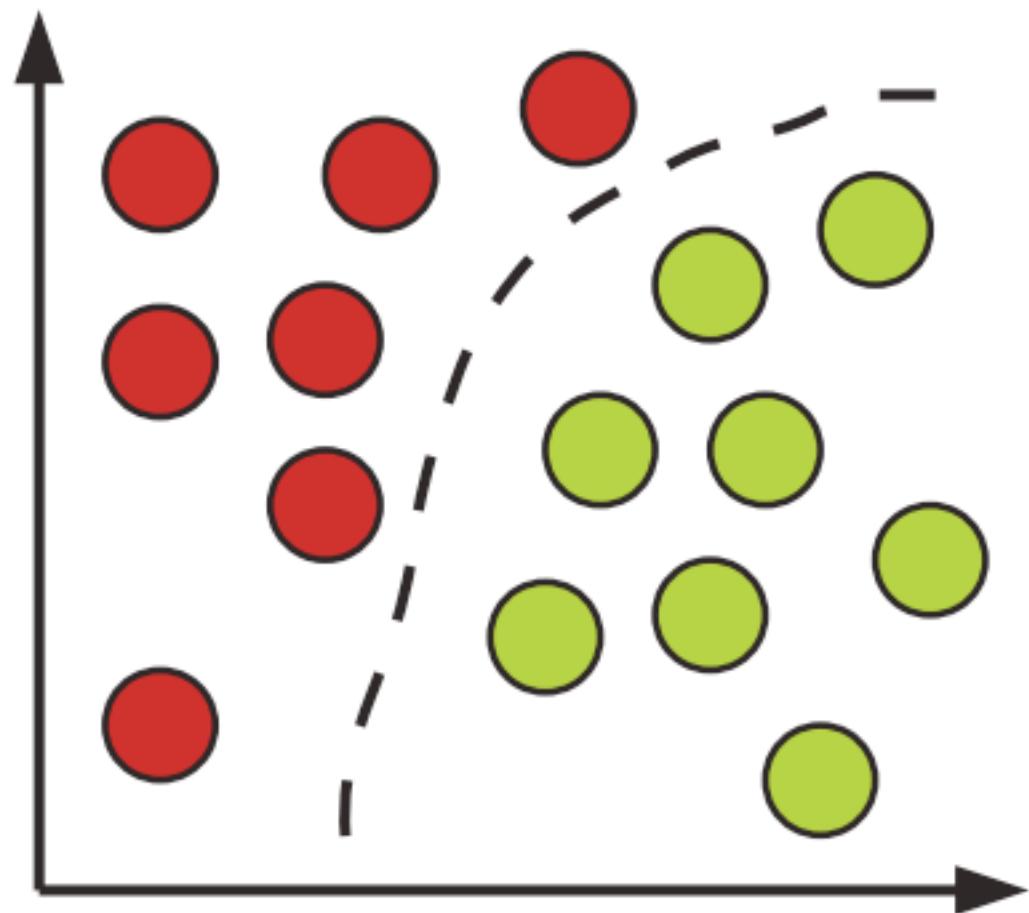


**Sudden concept drift**

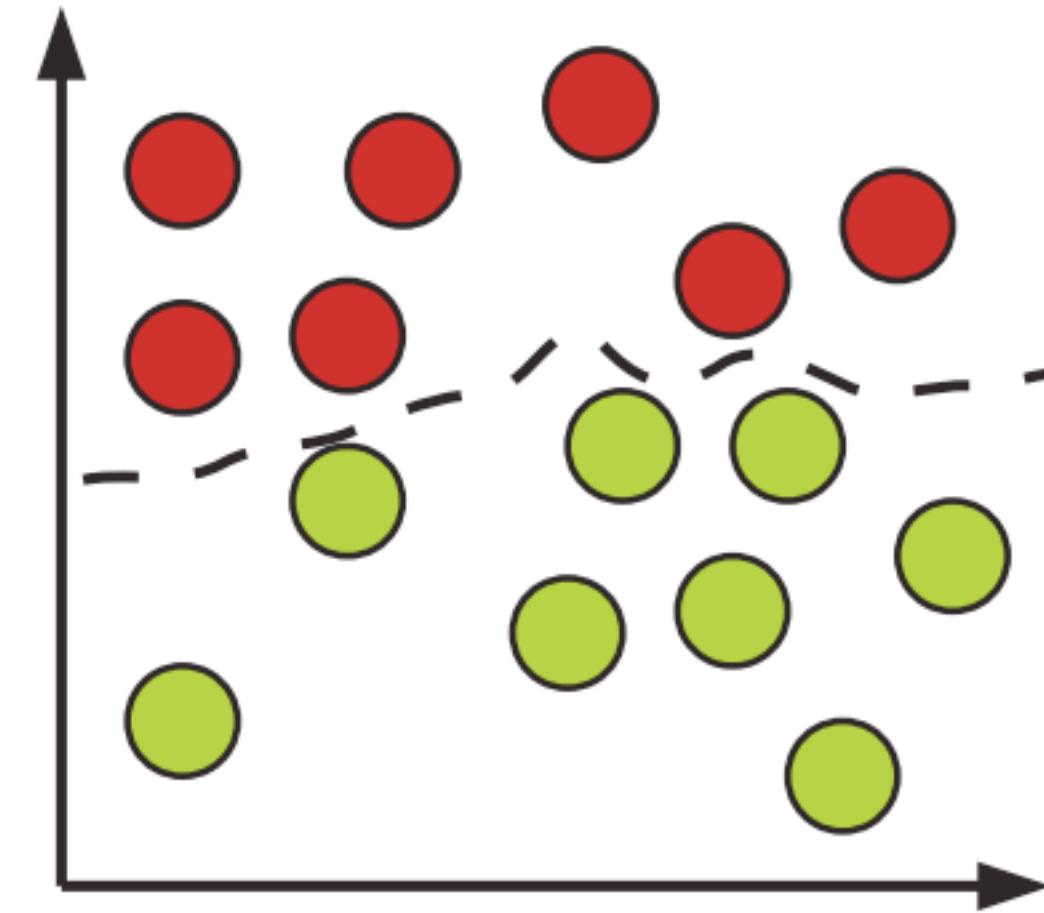


# Data drift vs Concept drift

Orginal data

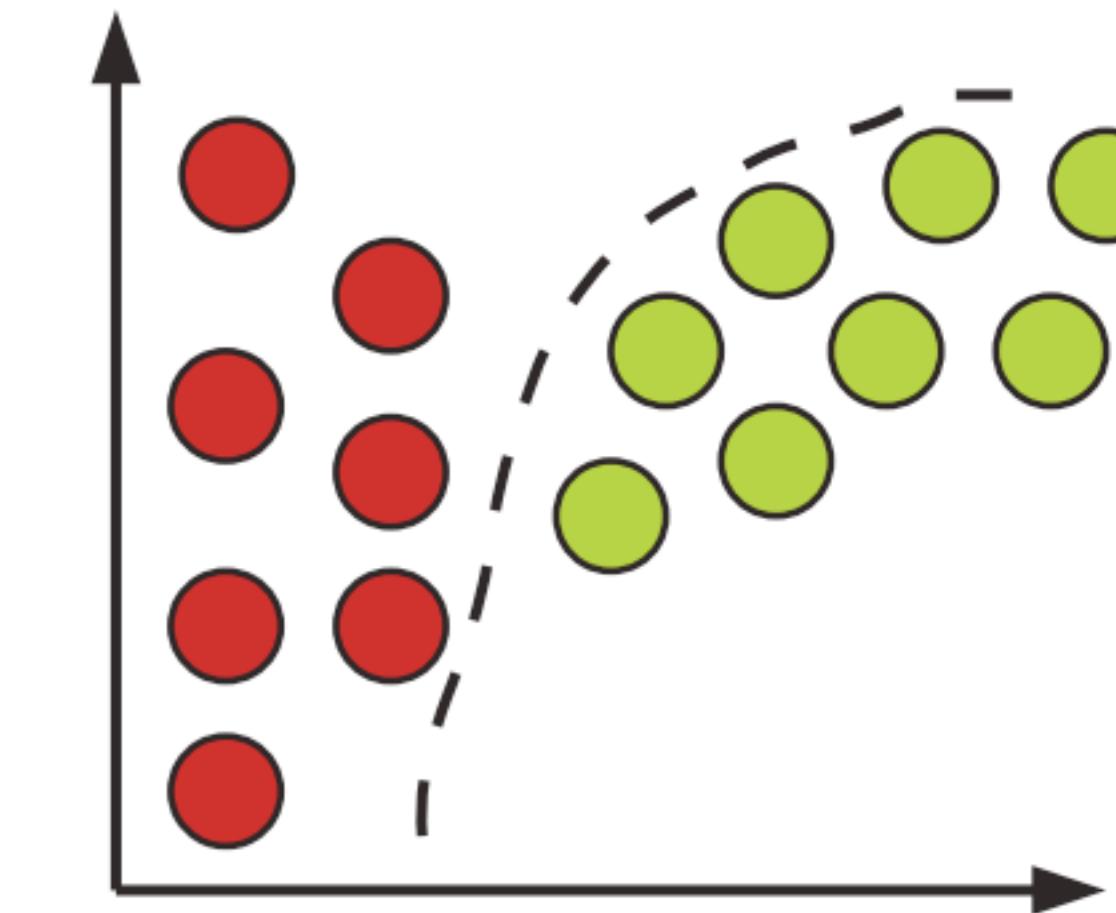


Concept drift



$p(y|X)$  changes

Data drift



$p(X)$  changes, but not  $p(y|X)$

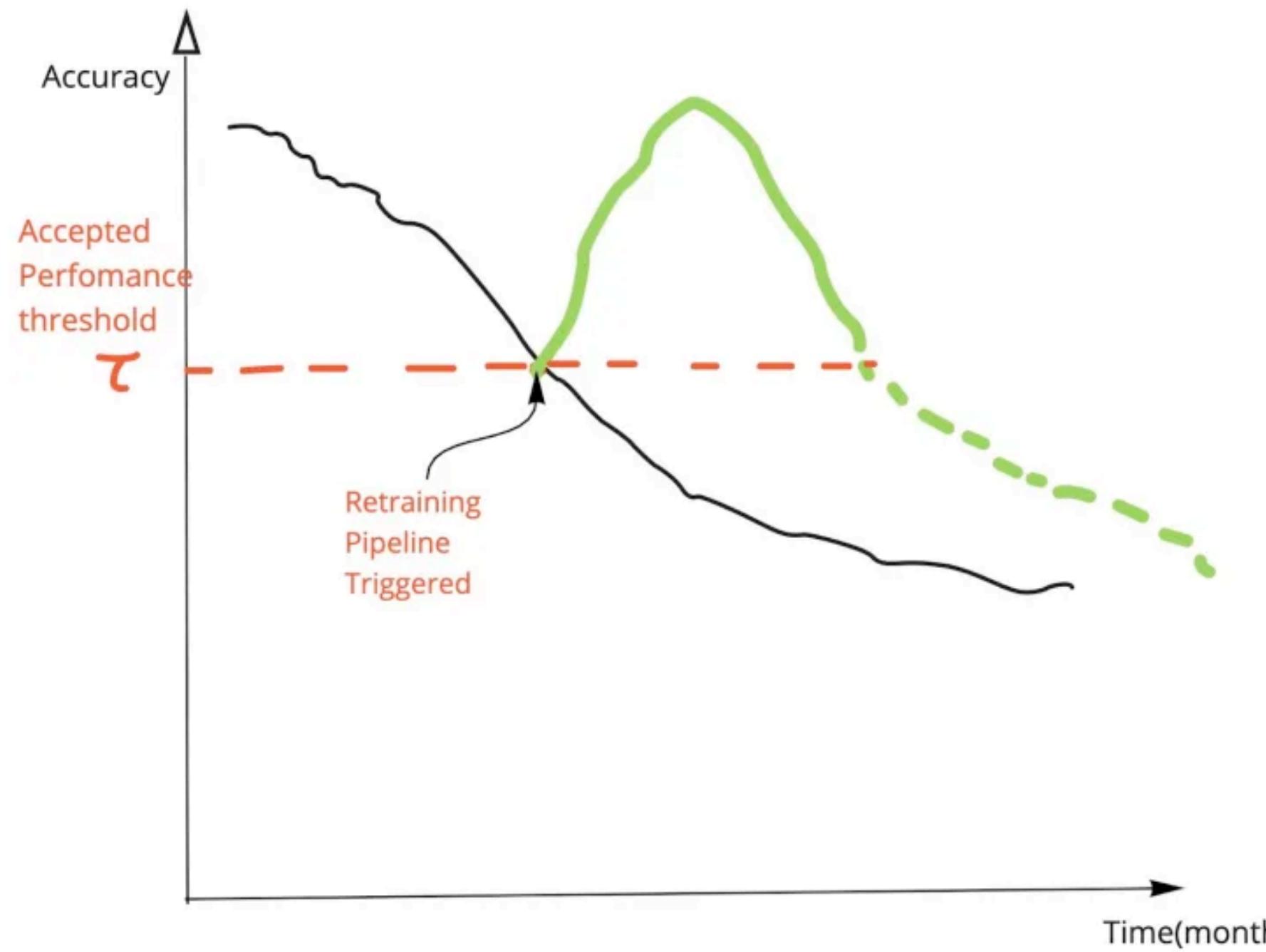
Can you **observe**  
**data drift** in our datasets?

# What is Continuous Training?

Continuous training is an aspect of machine learning operations that **automatically and continuously retrains machine learning models** to adapt to changes in the data before it is redeployed.

# Why is continuous training important?

Machine learning models  
**get stale with time**



# Defining the retraining strategy

# Defining the retraining strategy

When should a model be  
retrained?

# Defining the retraining strategy

When should a model be  
retrained?

How much data is  
needed for retraining?

# Defining the retraining strategy

When should a model be  
retrained?

How much data is  
needed for retraining?

What should be  
retrained?

# Defining the retraining strategy

When should a model be  
retrained?

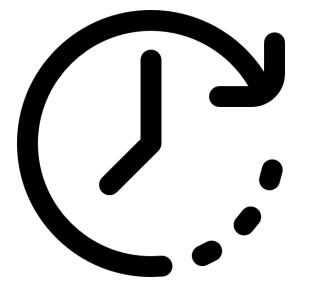
How much data is  
needed for retraining?

What should be  
retrained?

When to deploy your  
model after retraining?

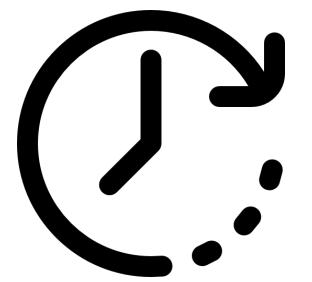
# When should you retrain a model?

# When should you retrain a model?

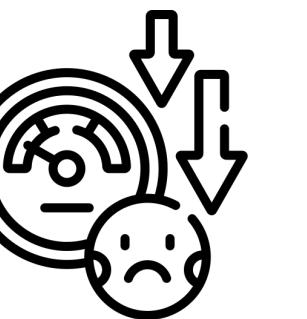


Retraining based on  
**interval**

# When should you retrain a model?

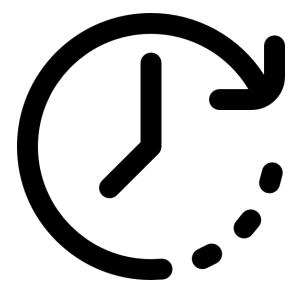


Retraining based on  
**interval**

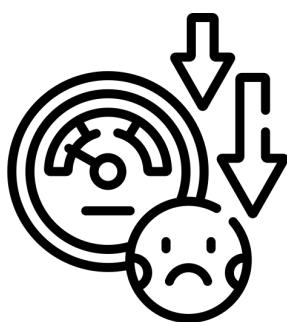


**Performance-based**  
trigger

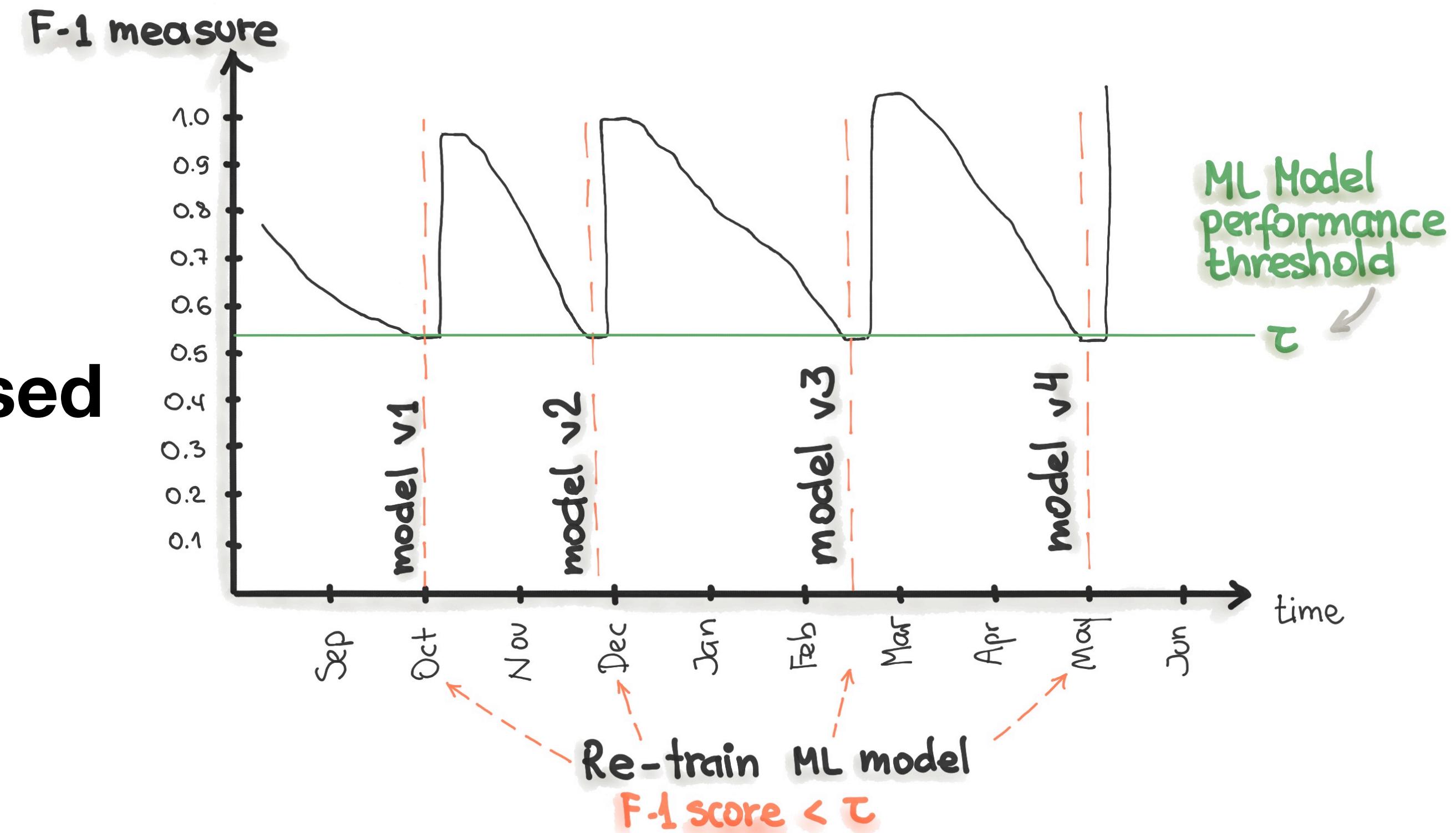
# When should you retrain a model?



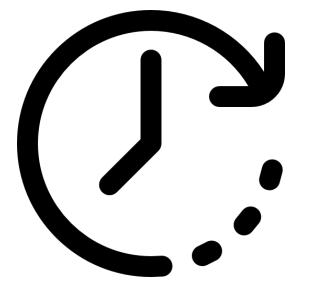
Retraining based on  
**interval**



**Performance-based**  
trigger

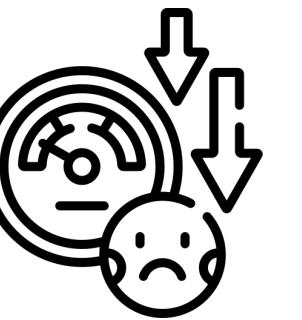


# When should you retrain a model?



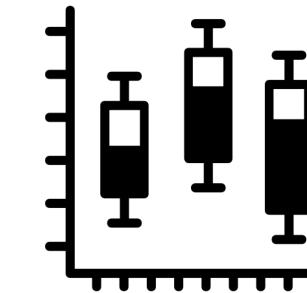
Retraining based on

**interval**



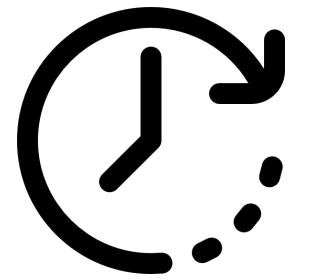
**Performance-based**

trigger

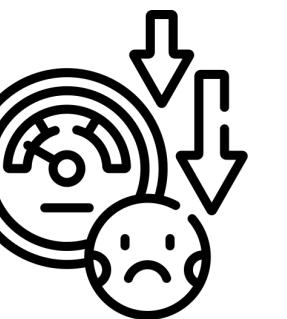


Trigger based on **data  
changes**

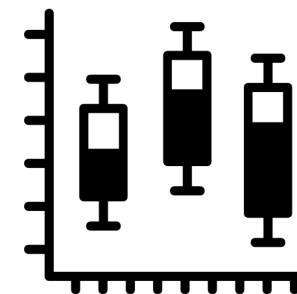
# When should you retrain a model?



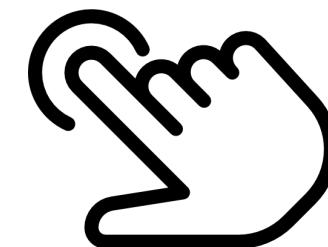
Retraining based on  
**interval**



**Performance-based**  
trigger



Trigger based on **data**  
**changes**



Retraining on **demand**

# How much data is needed for retraining?

# How much data is needed for retraining?

**Three things to consider when choosing the right size of data:**

- What is the size of your data?
- Is your data drifting?
- How often do you get new data?

# How much data is needed for retraining?

**Three things to consider when choosing the right size of data:**

- What is the size of your data?
- Is your data drifting?
- How often do you get new data?

- **Fixed window**

# How much data is needed for retraining?

**Three things to consider when choosing the right size of data:**

- What is the size of your data?
- Is your data drifting?
- How often do you get new data?

- **Fixed window**

- **Dynamic window**

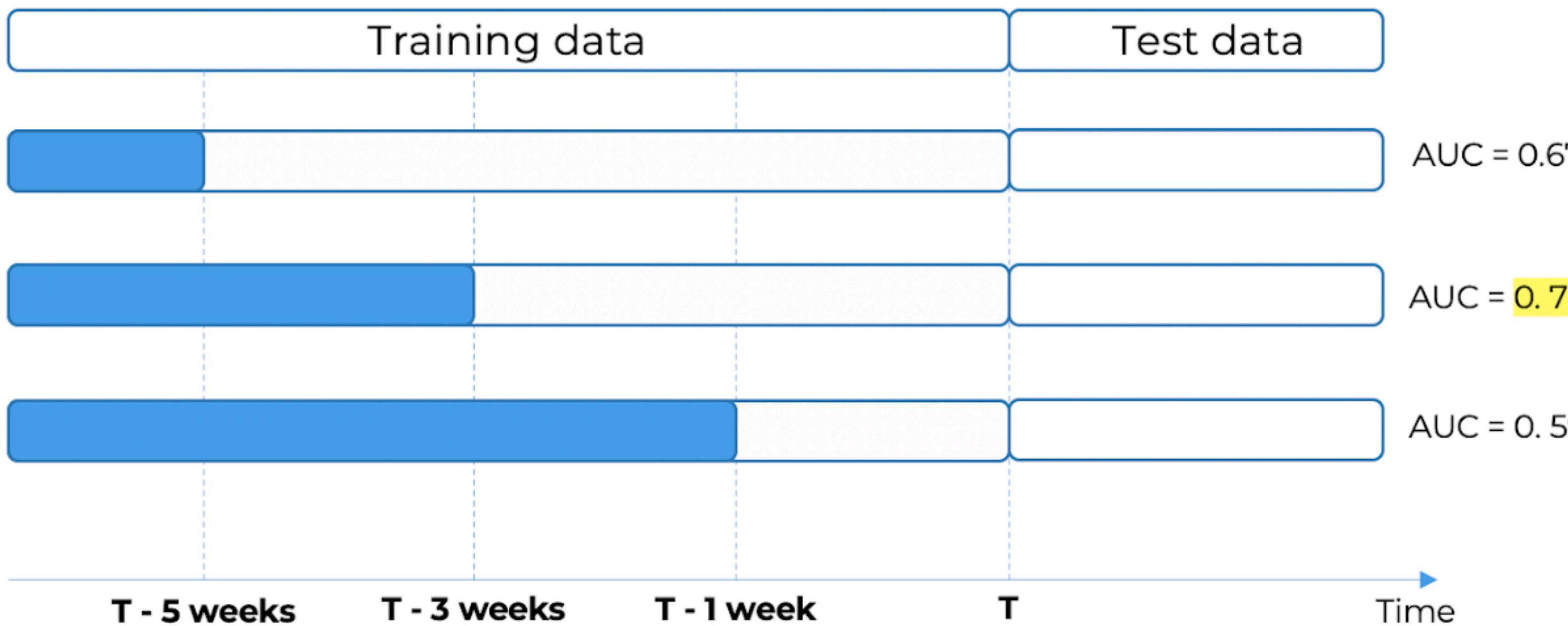
# How much data is needed for retraining?

**Three things to consider when choosing the right size of data:**

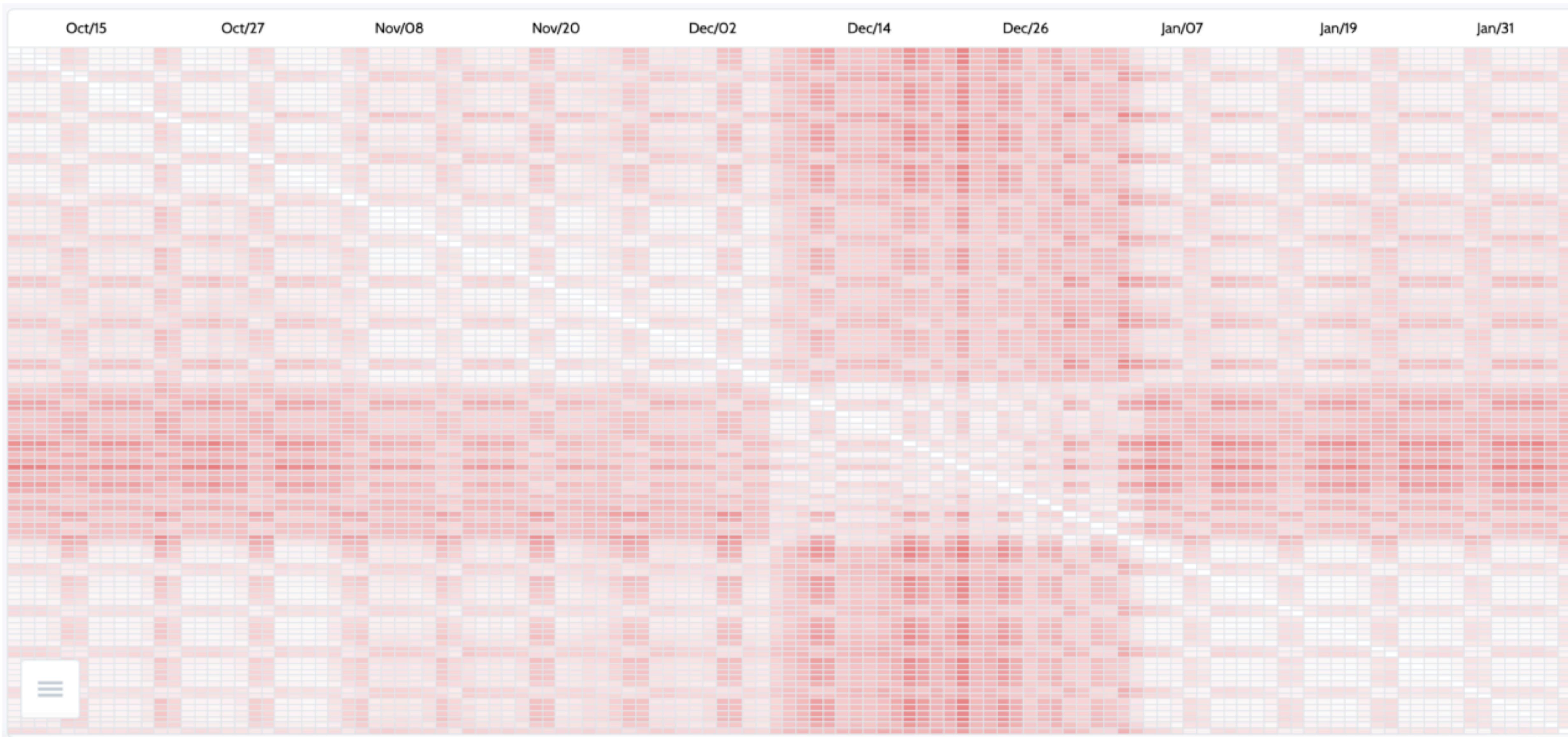
- What is the size of your data?
- Is your data drifting?
- How often do you get new data?

- **Fixed window**
- **Dynamic window**
- **Dynamic data selection**

# Dynamic window size



# Dynamic data selection



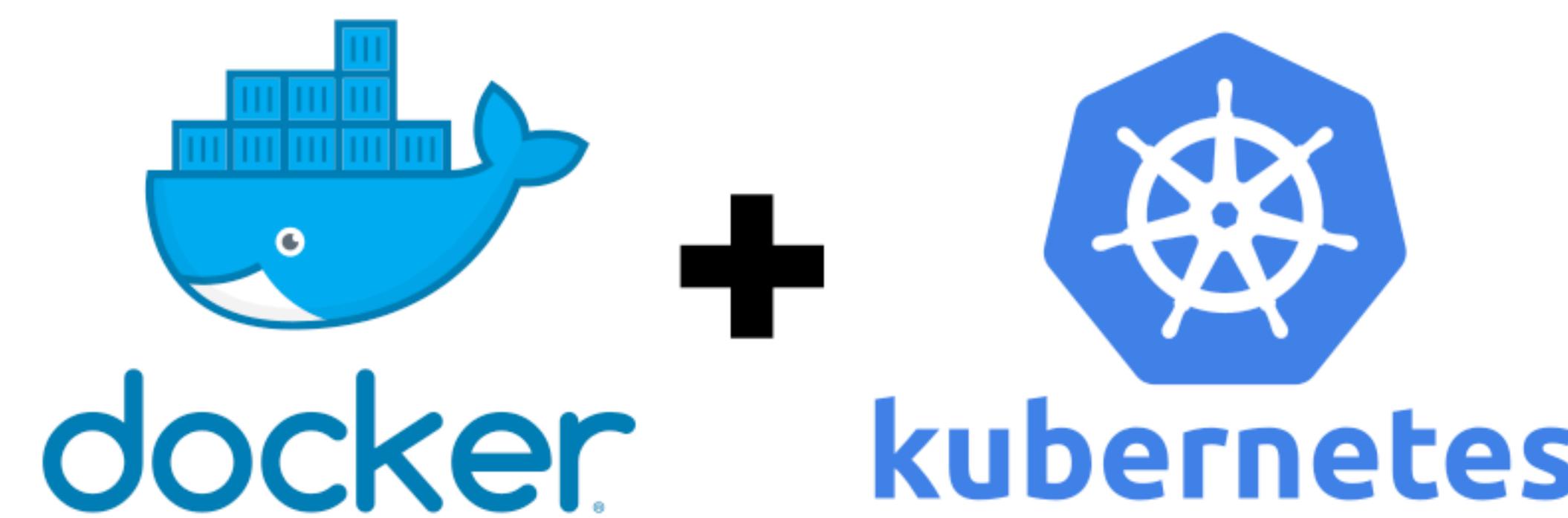
# What should be retrained?

- **Level of retraining:**
  - Are you just refitting the model (or other steps) with the new data?
  - Do you do hyperparameters optimization?
  - Do you challenge the selection of the model itself and test for other model types?

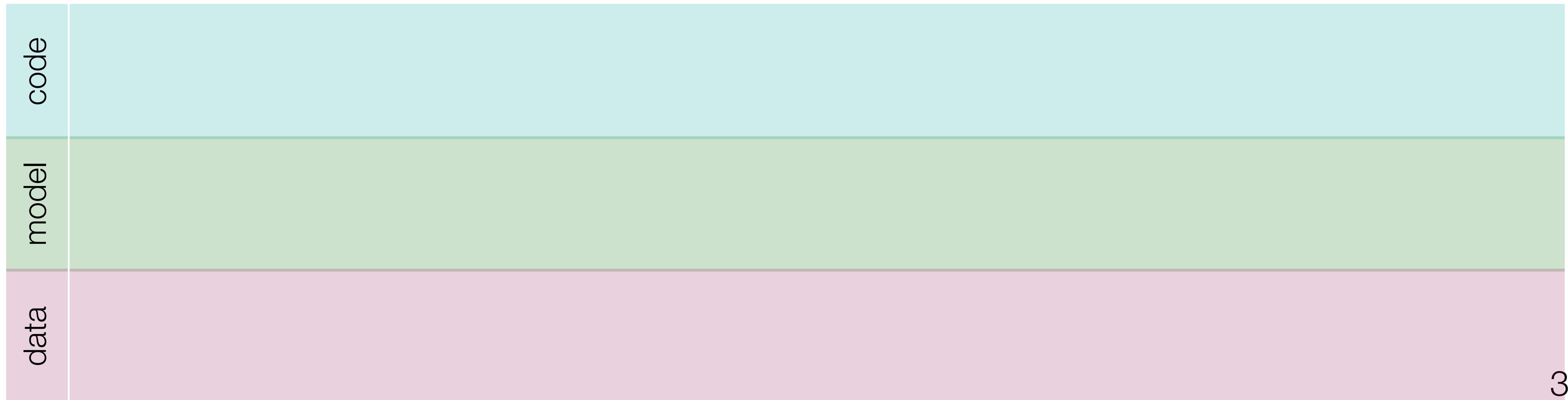
# Deployment after retraining model

**Should you deploy as soon as you retrain?**

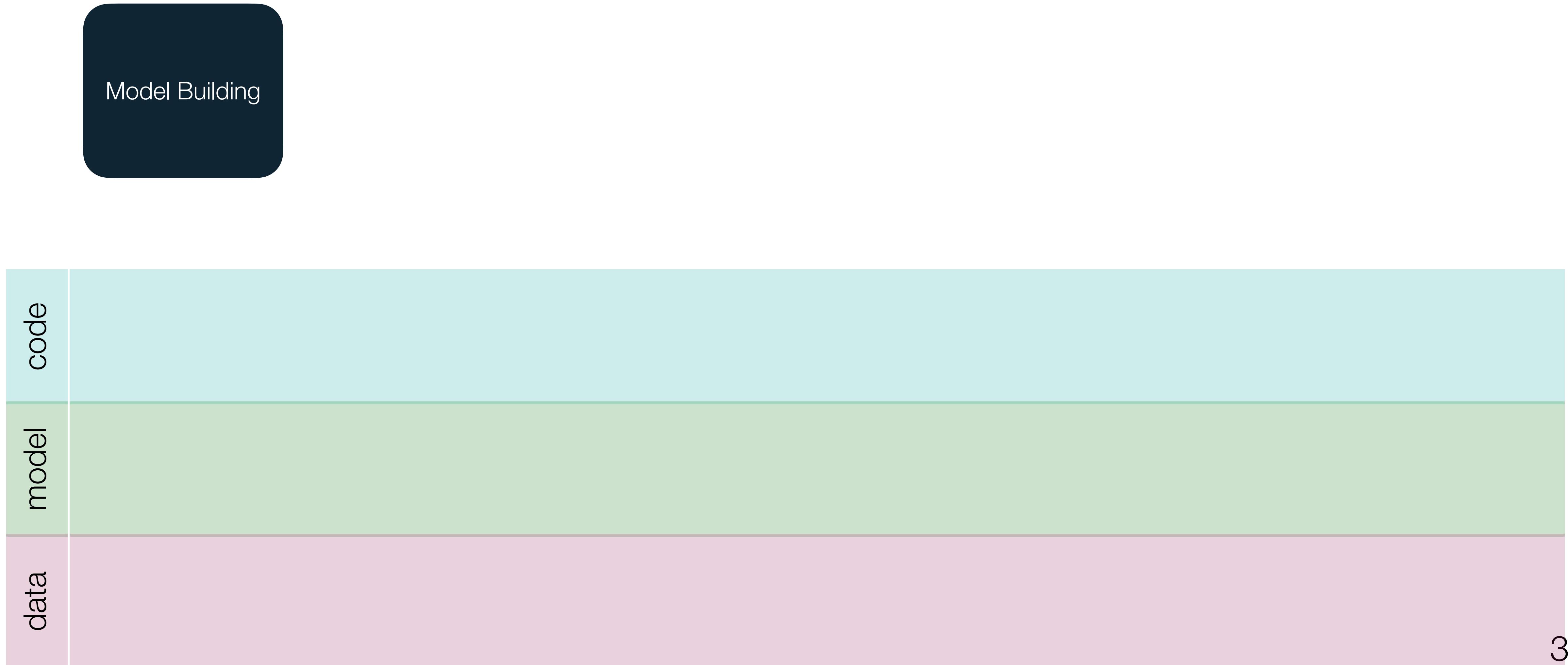
- Competing models
  - Shadow models
  - Multiple models



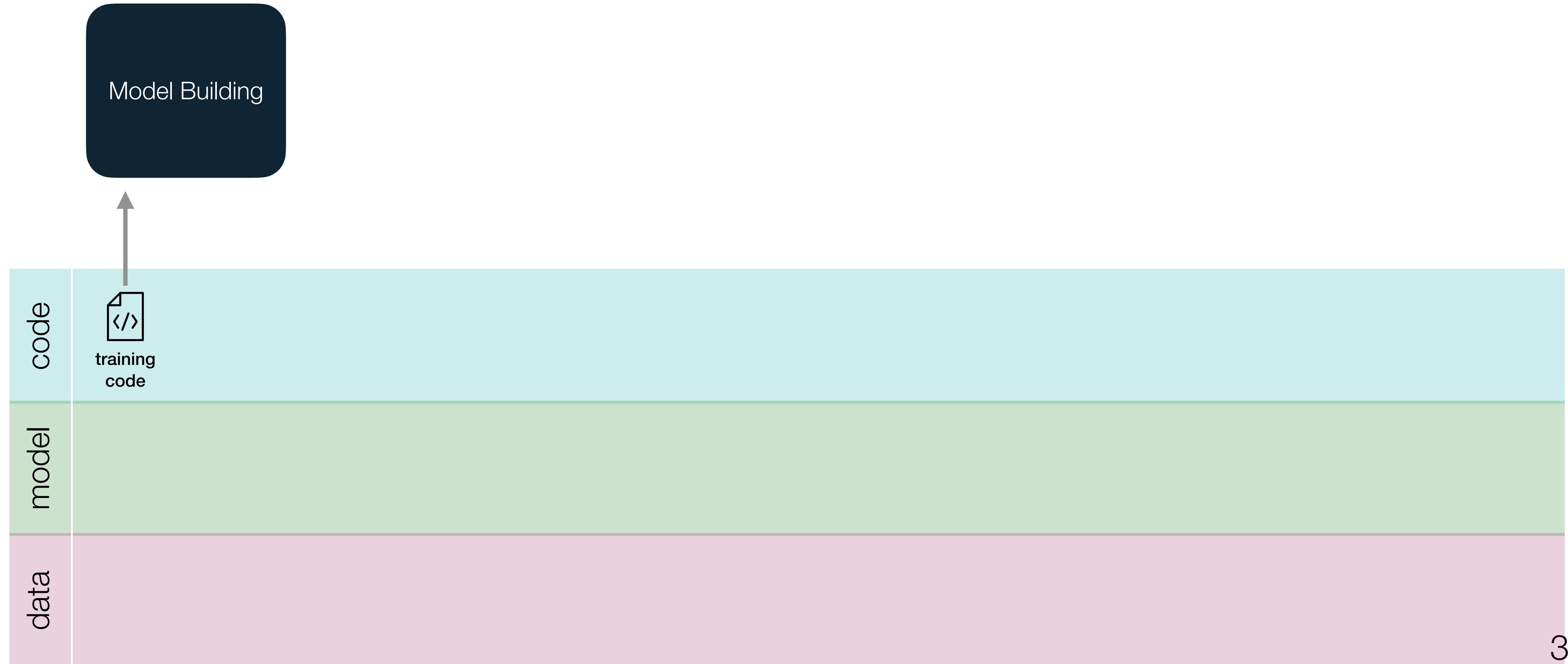
# ML Pipeline in Production



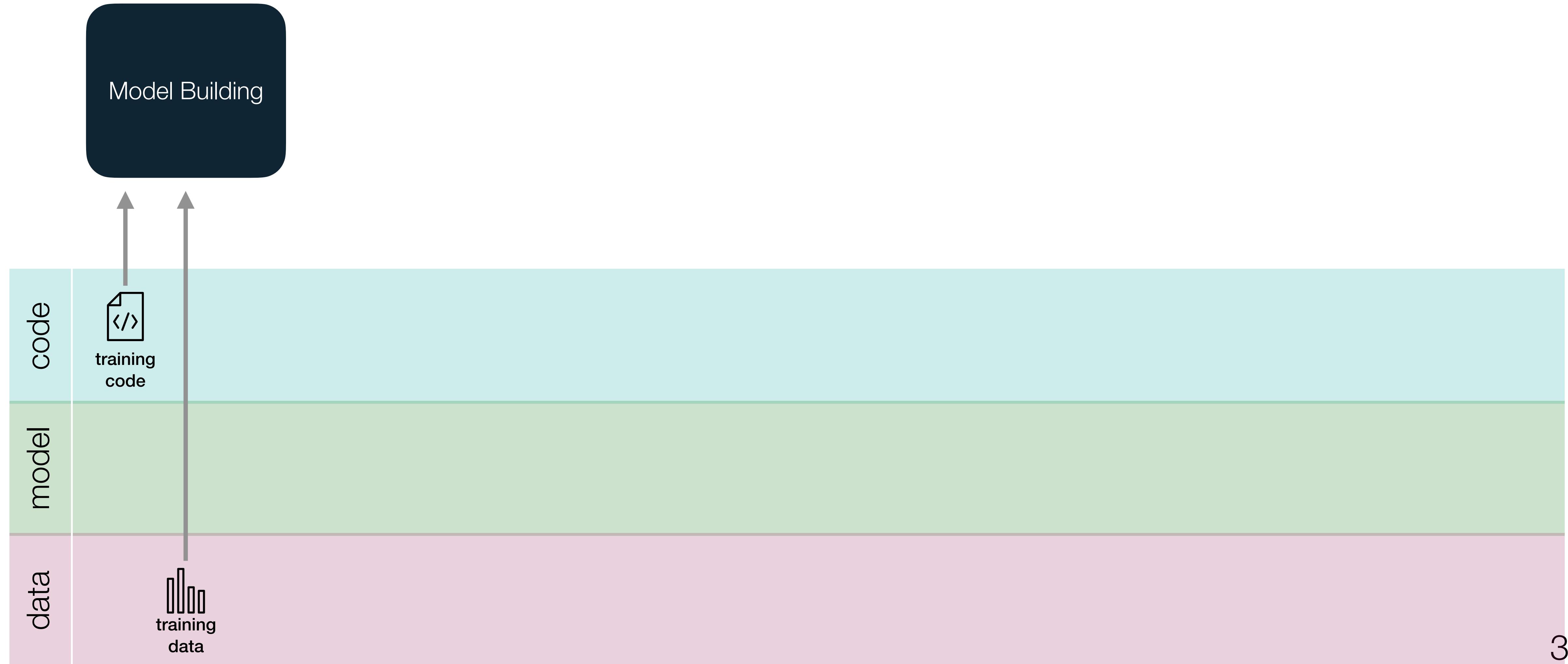
# ML Pipeline in Production



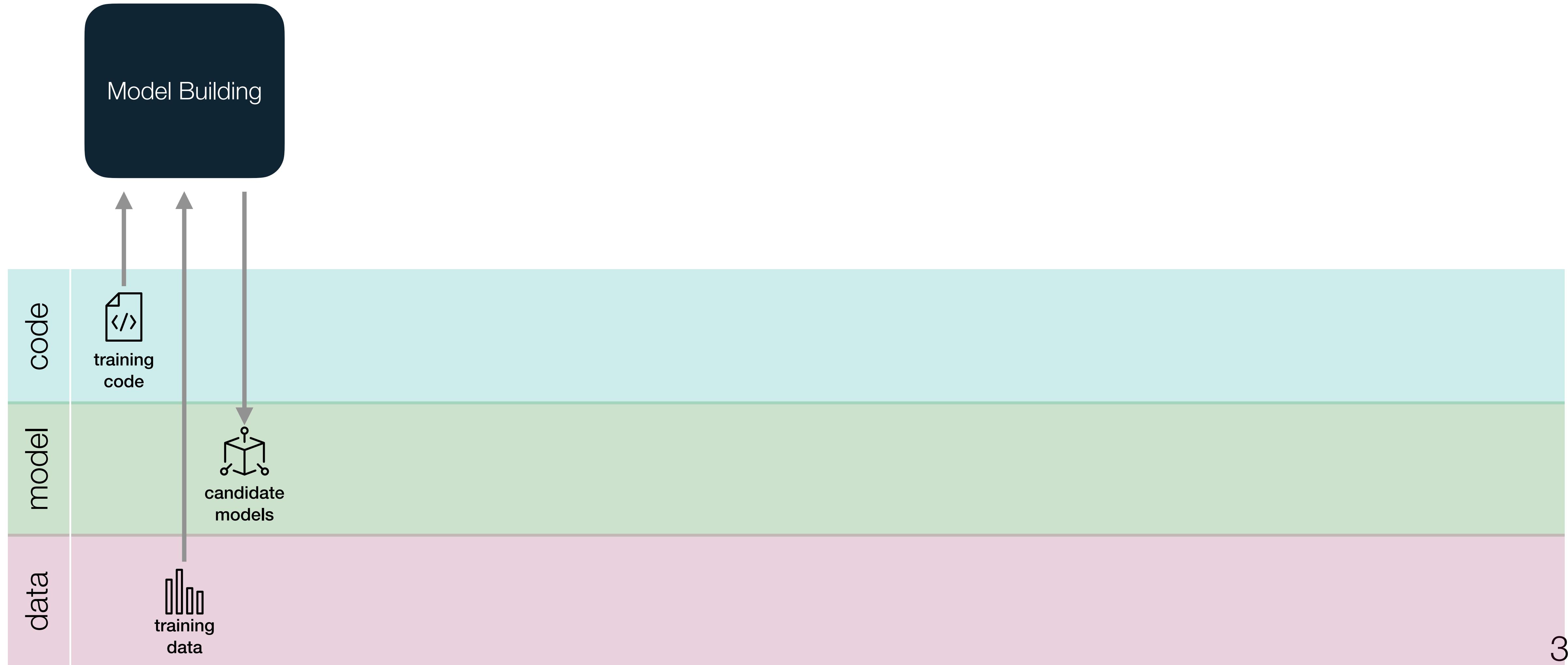
# ML Pipeline in Production



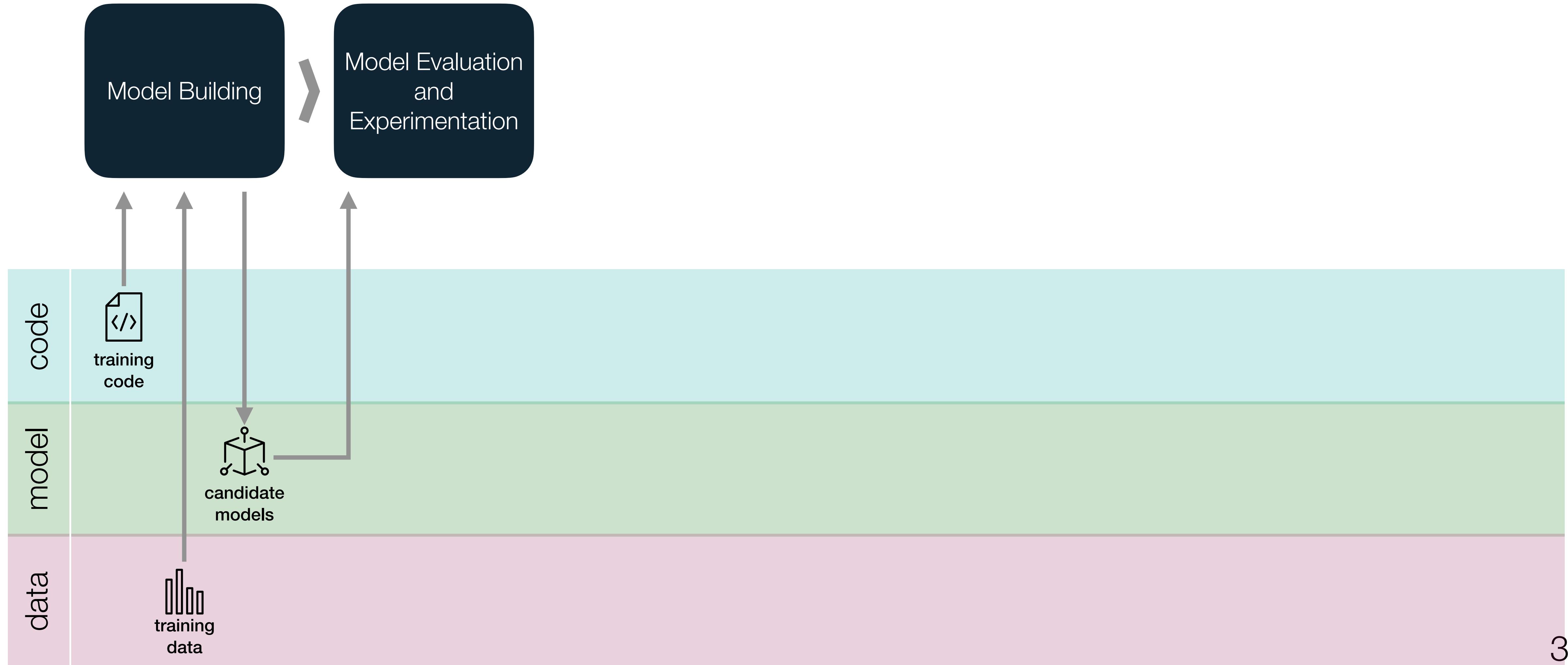
# ML Pipeline in Production



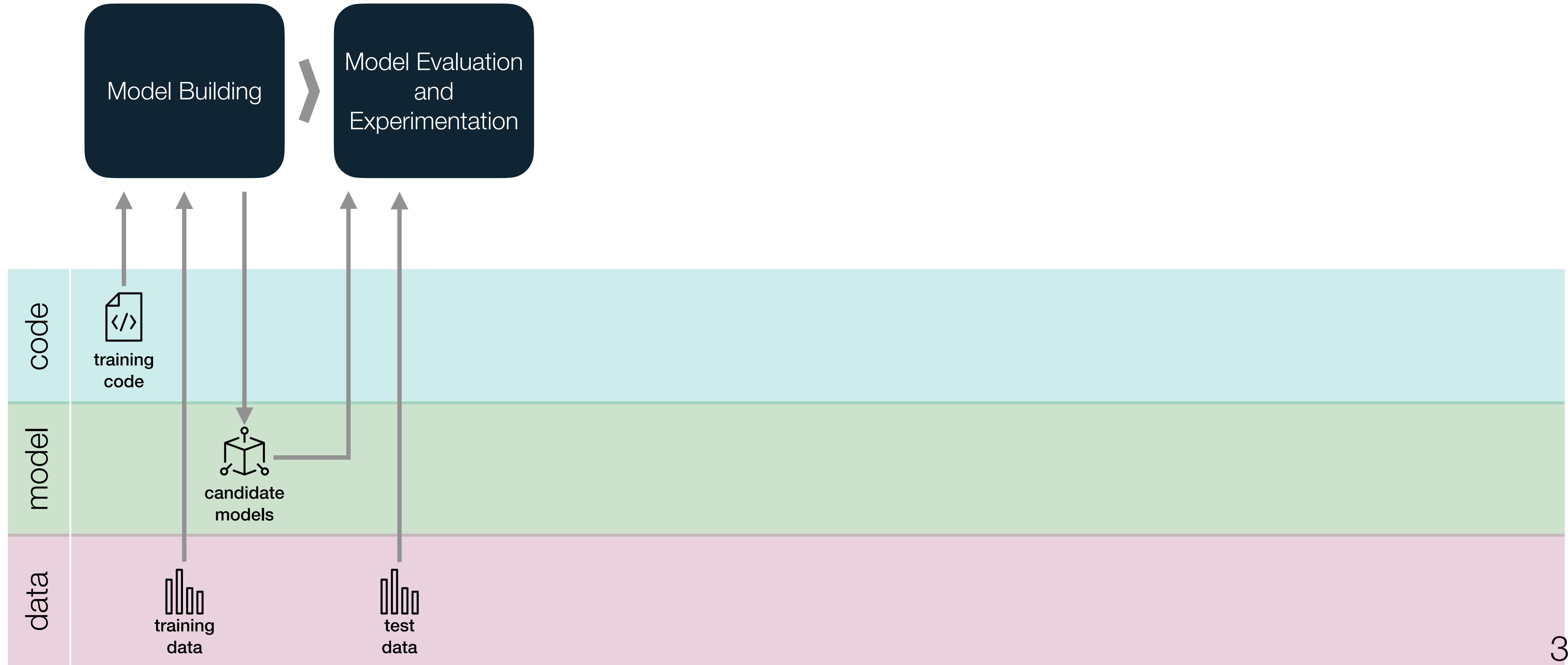
# ML Pipeline in Production



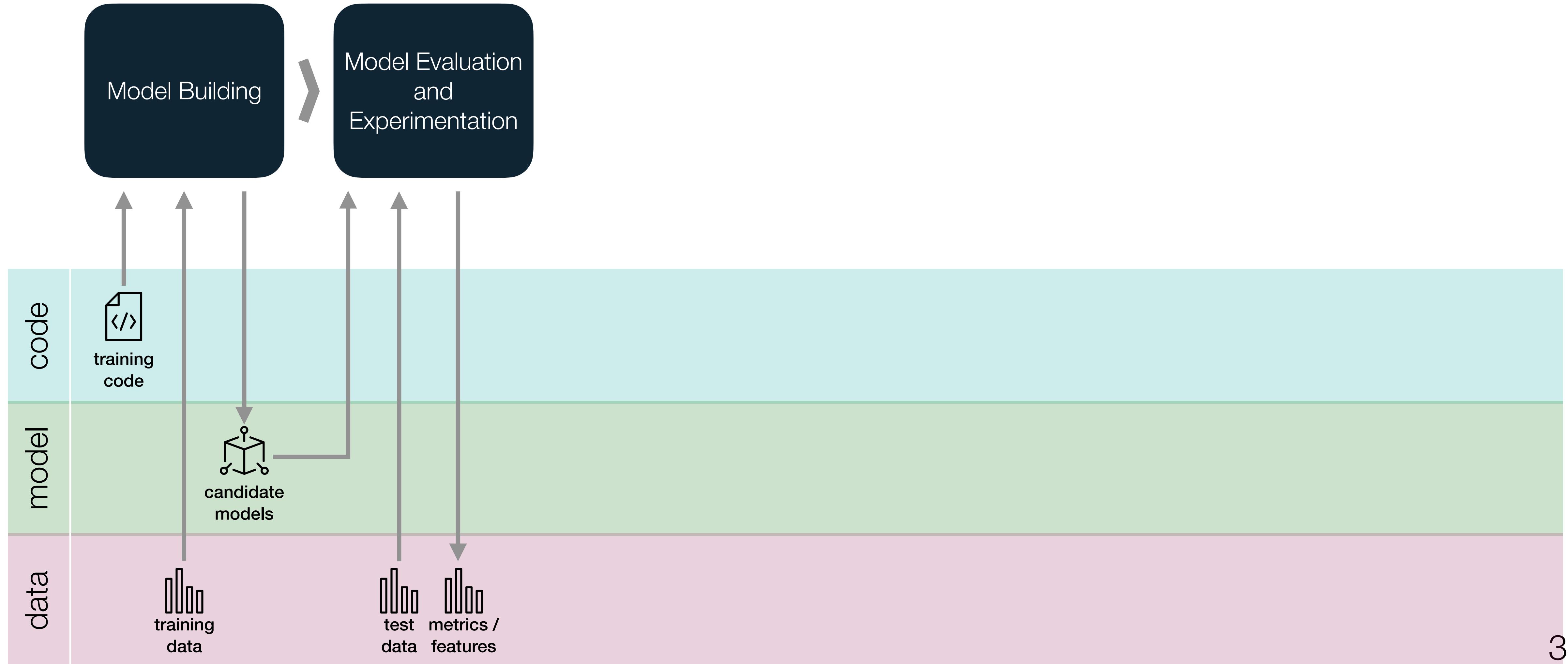
# ML Pipeline in Production



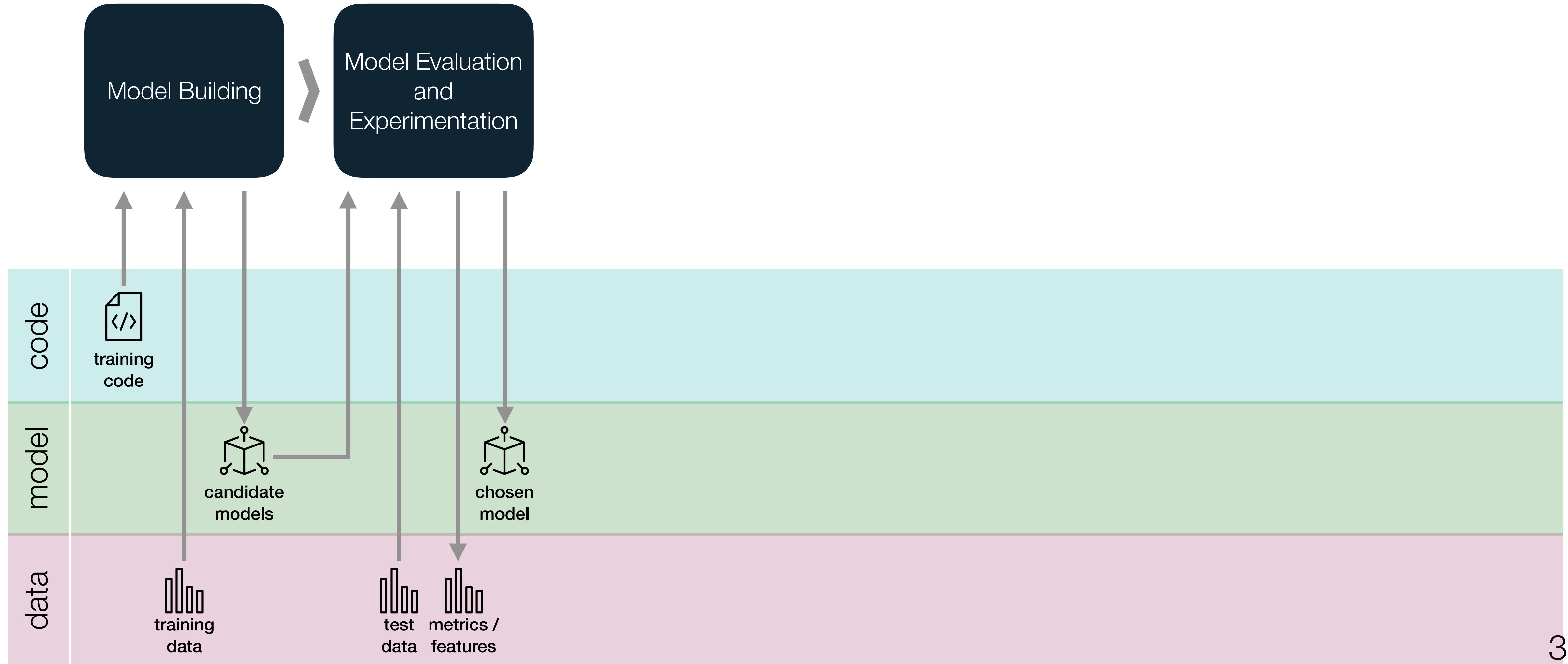
# ML Pipeline in Production



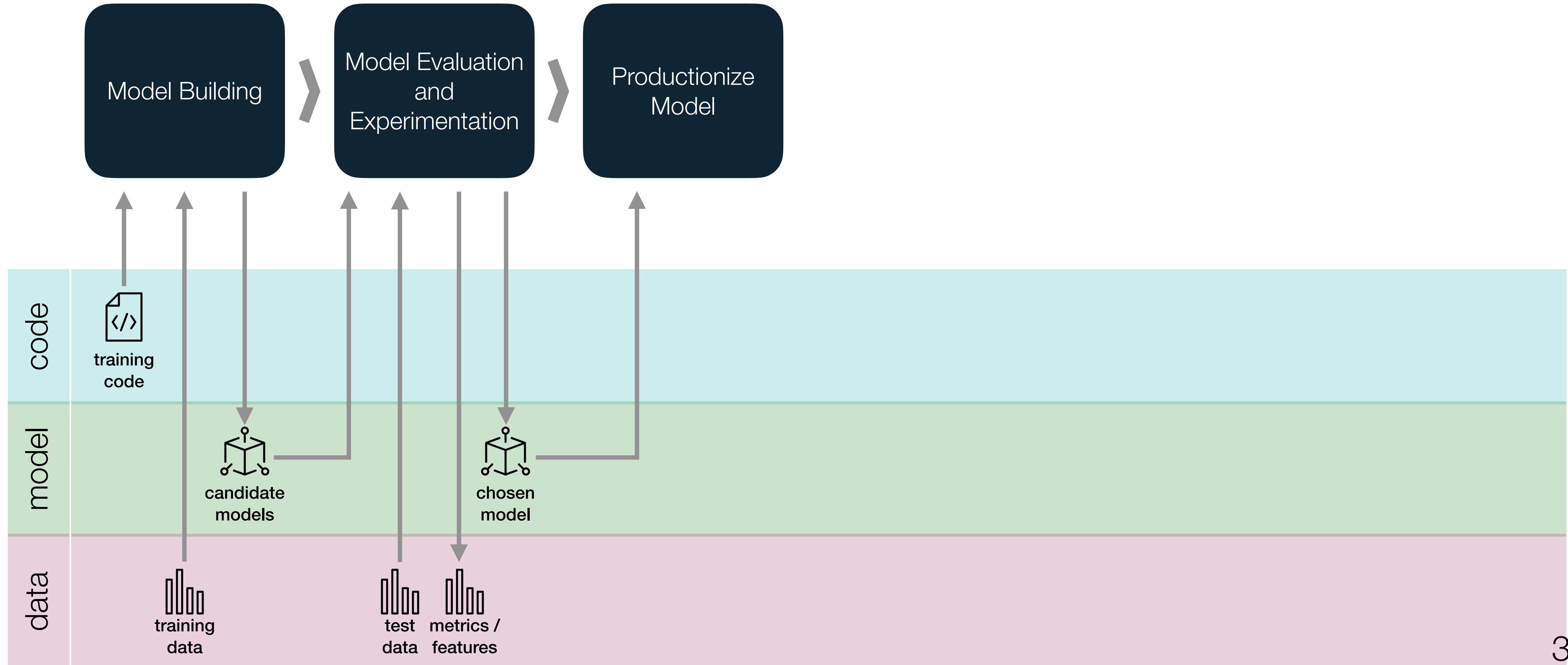
# ML Pipeline in Production



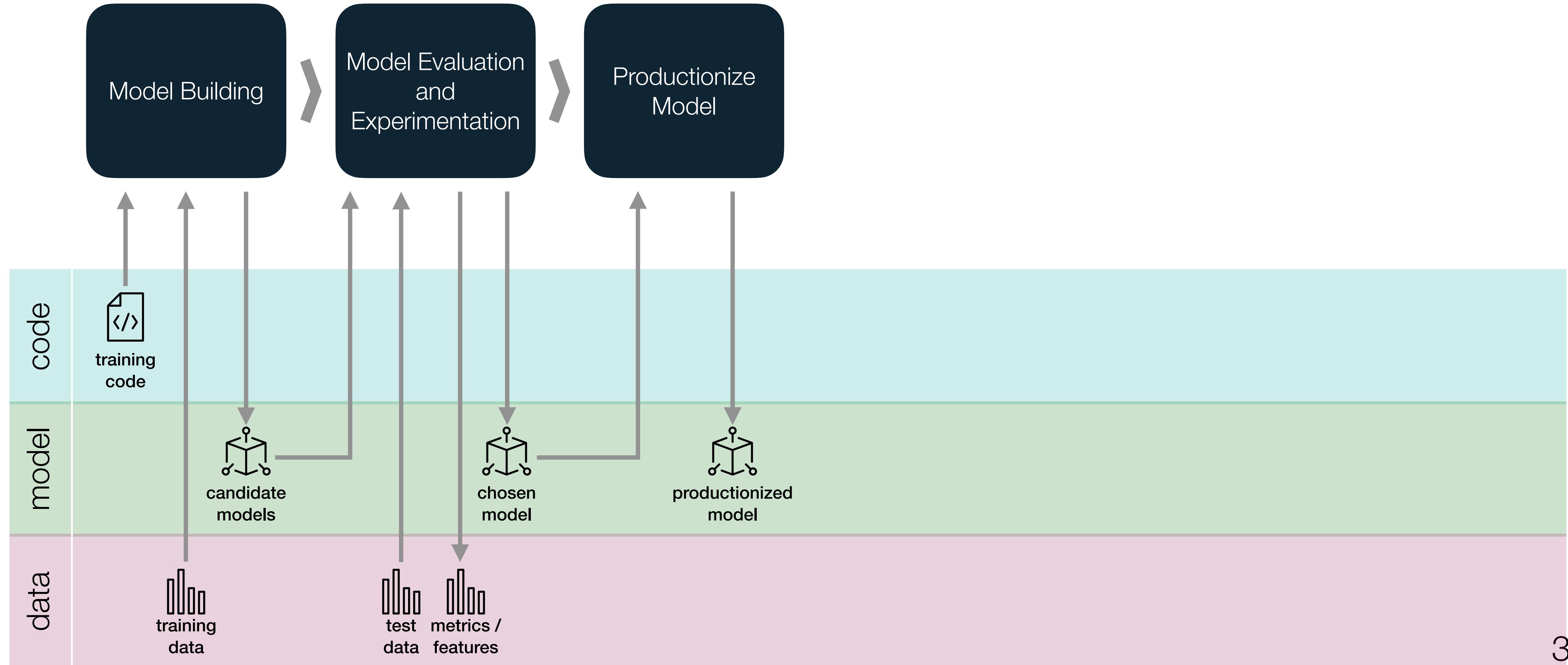
# ML Pipeline in Production



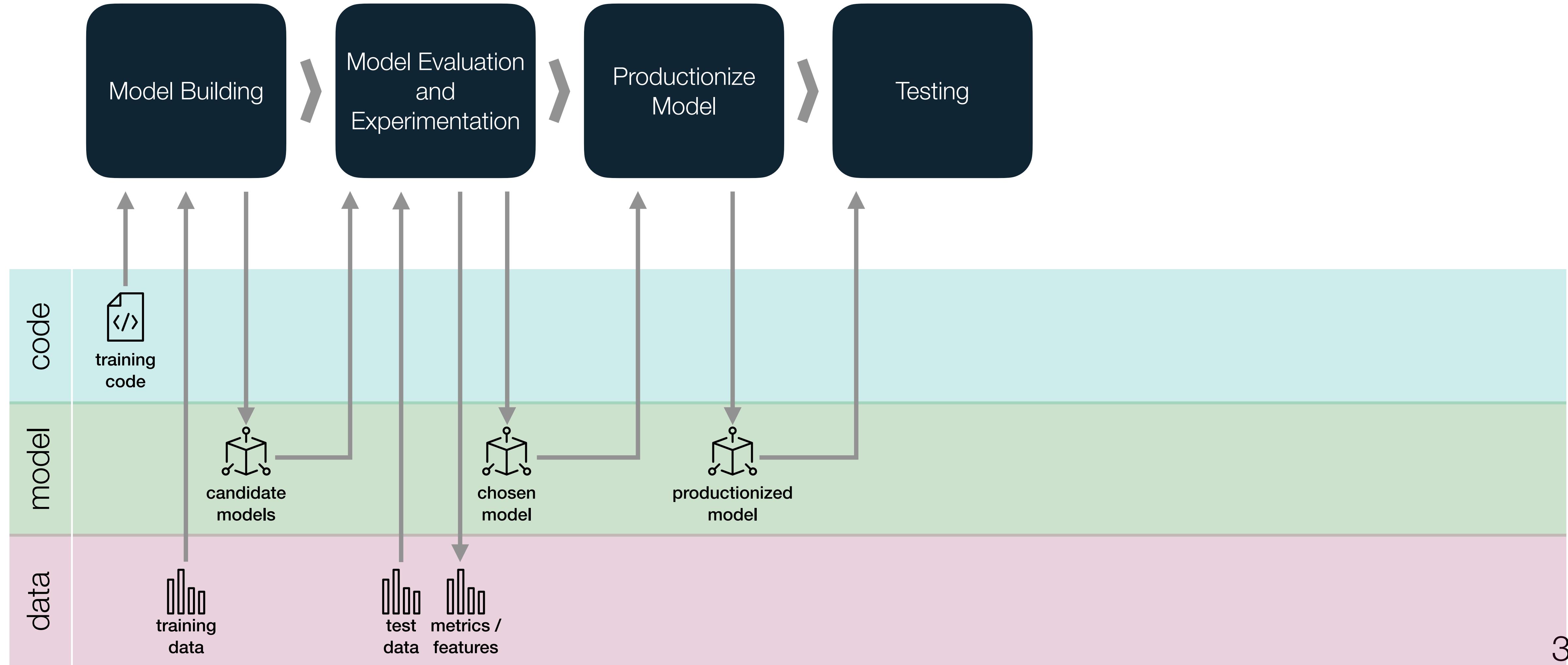
# ML Pipeline in Production



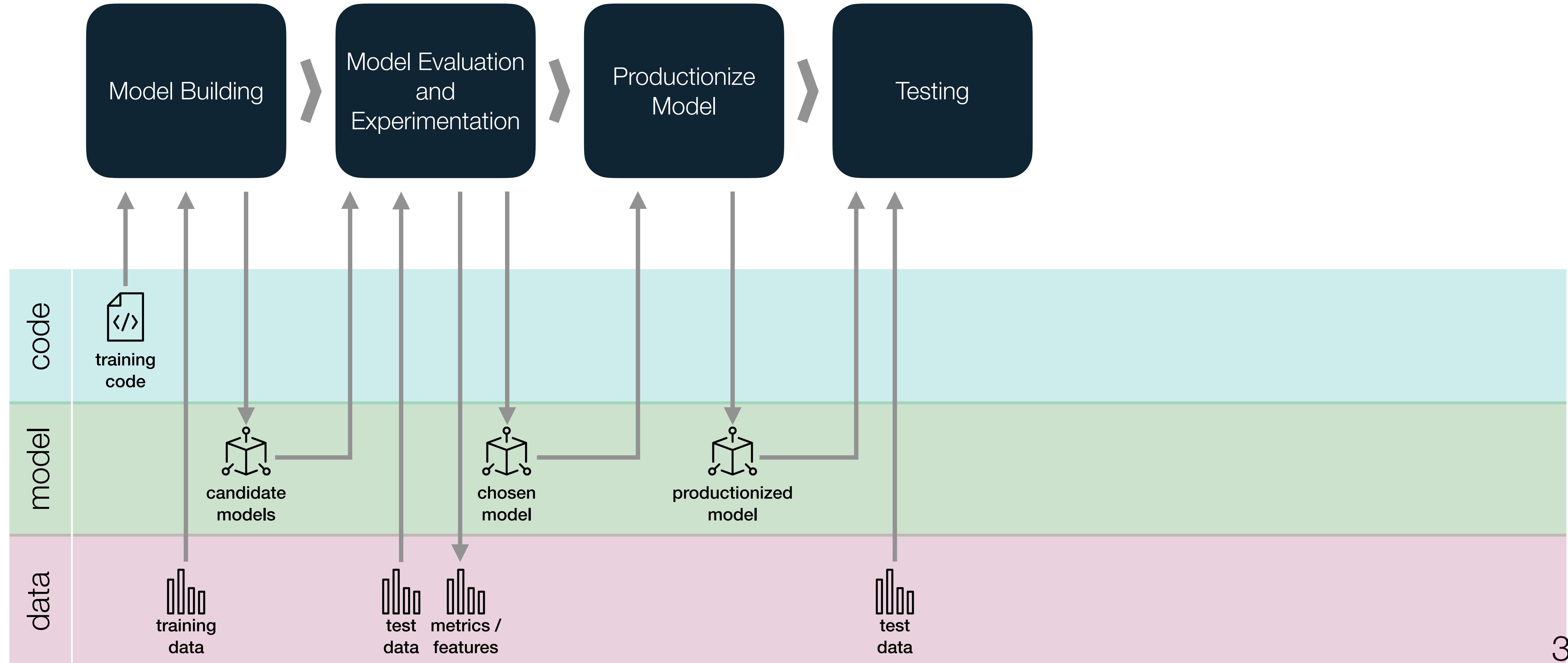
# ML Pipeline in Production



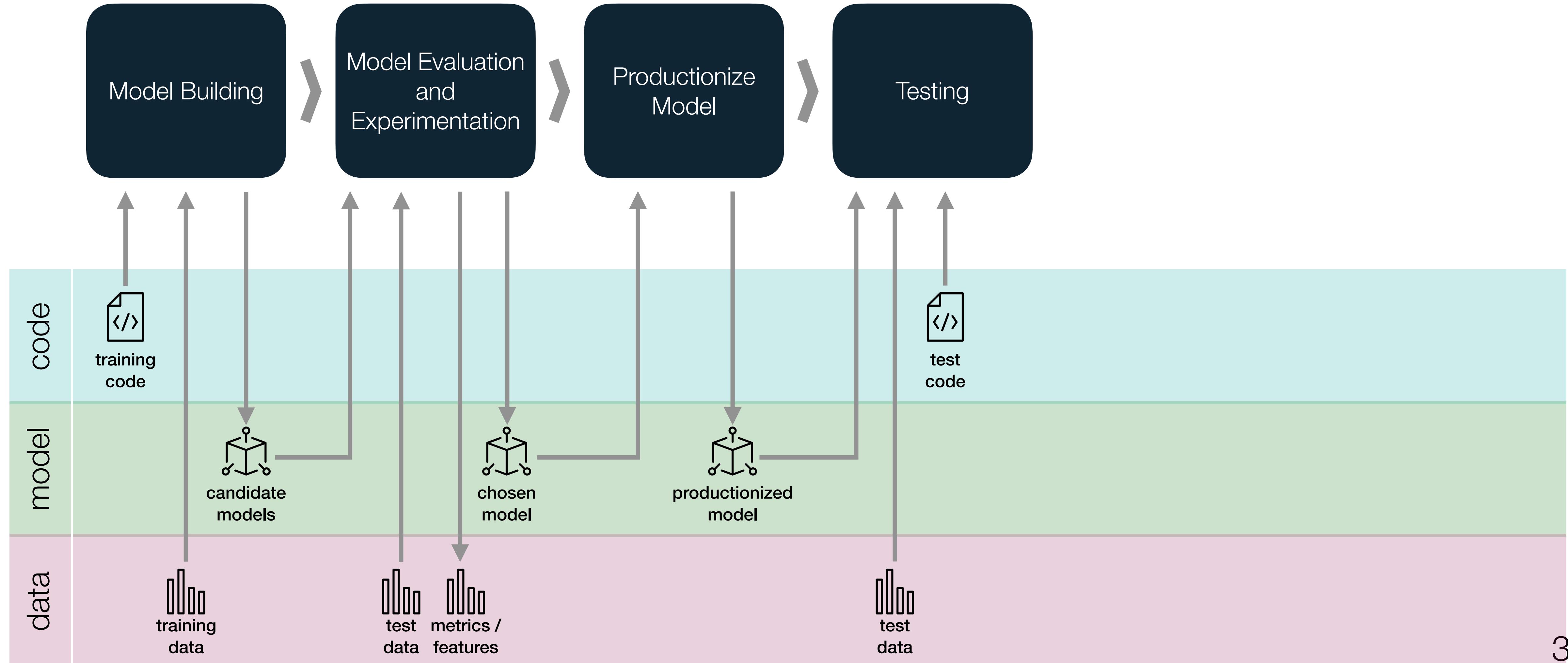
# ML Pipeline in Production



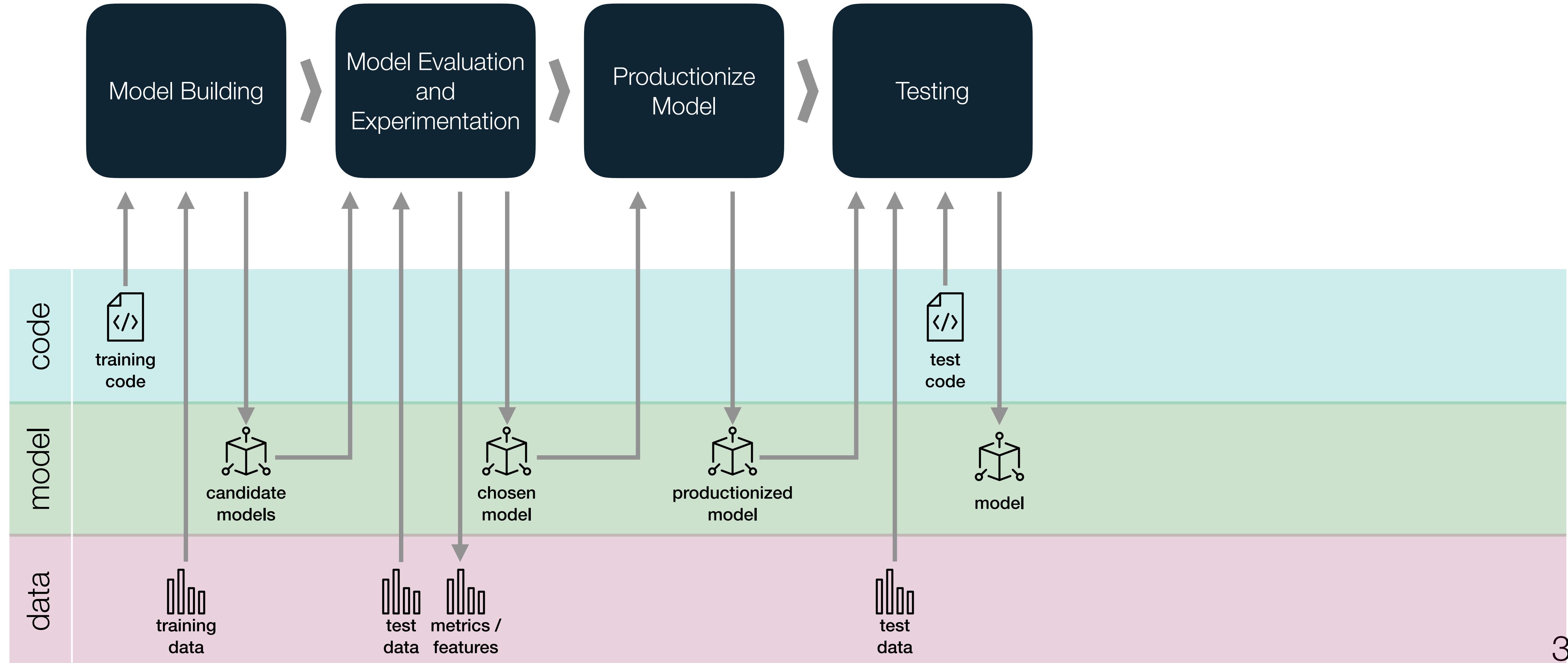
# ML Pipeline in Production



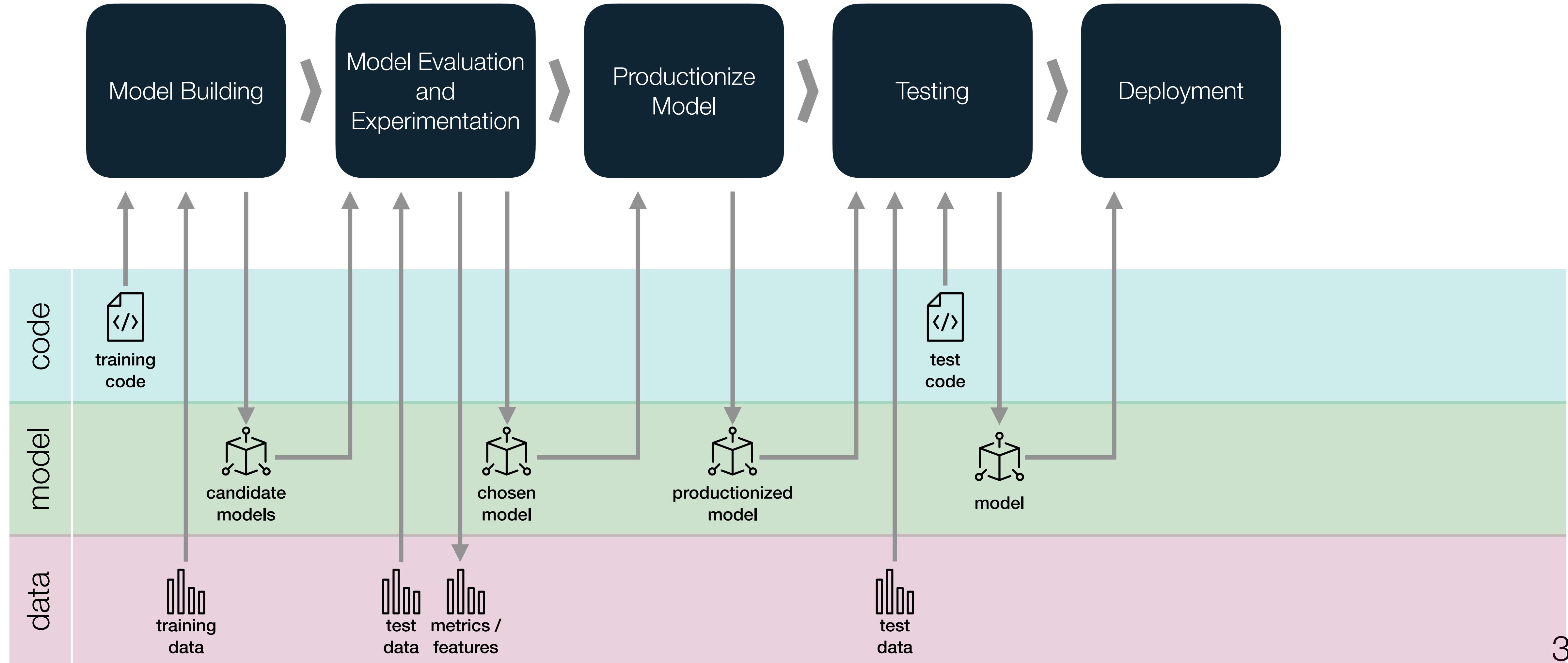
# ML Pipeline in Production



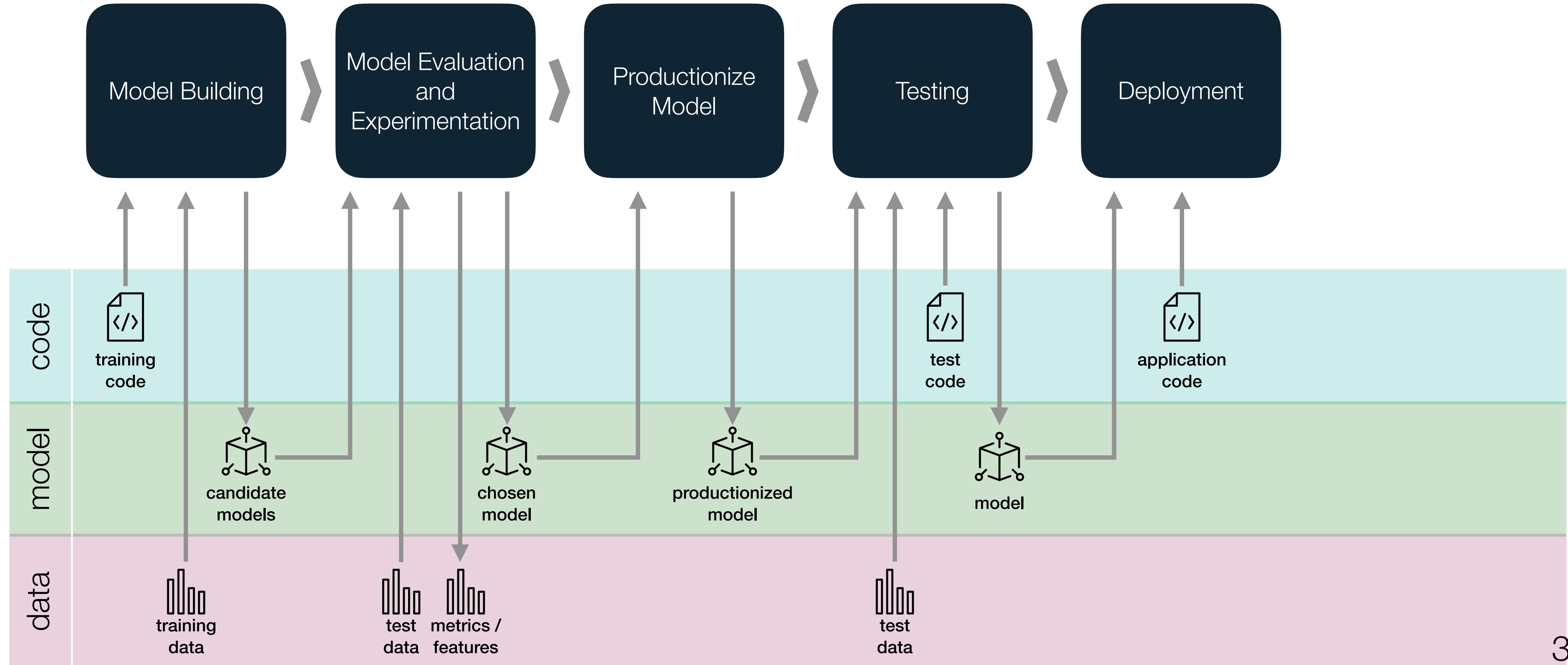
# ML Pipeline in Production



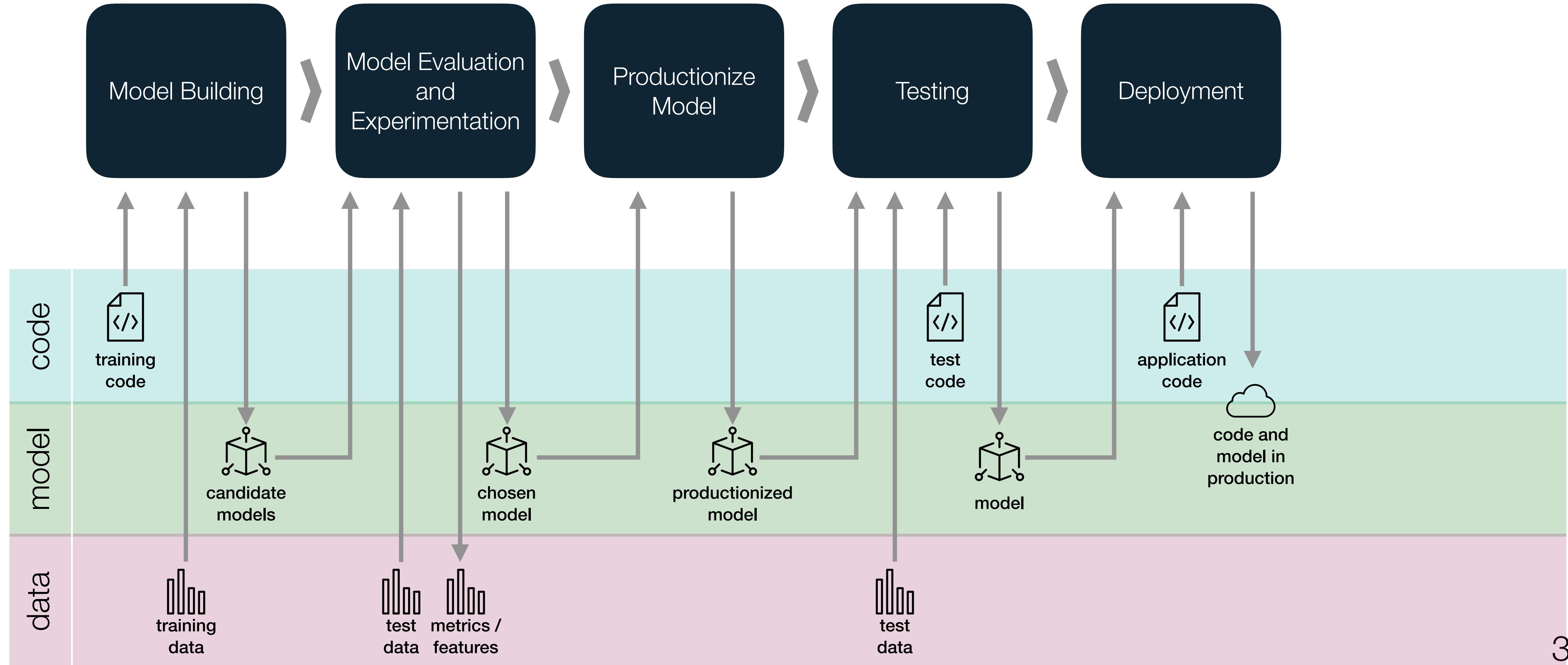
# ML Pipeline in Production



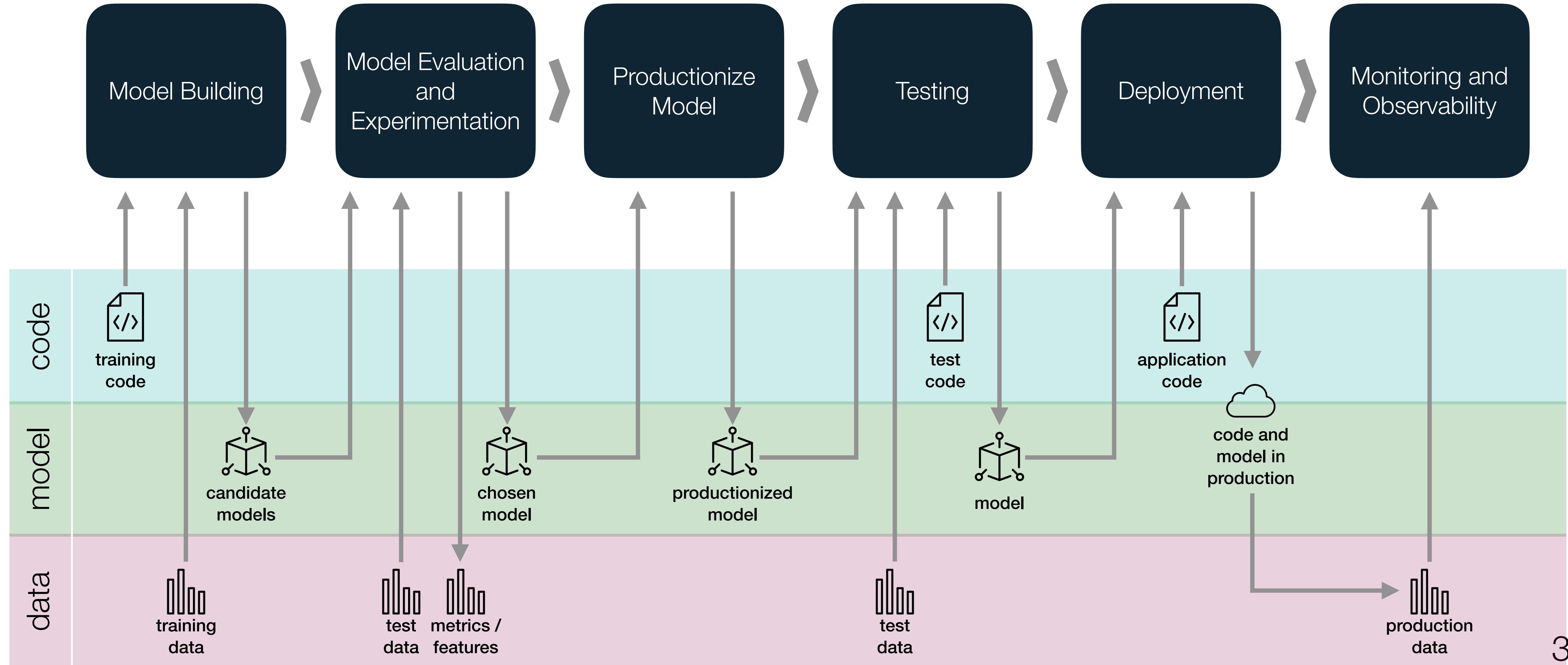
# ML Pipeline in Production



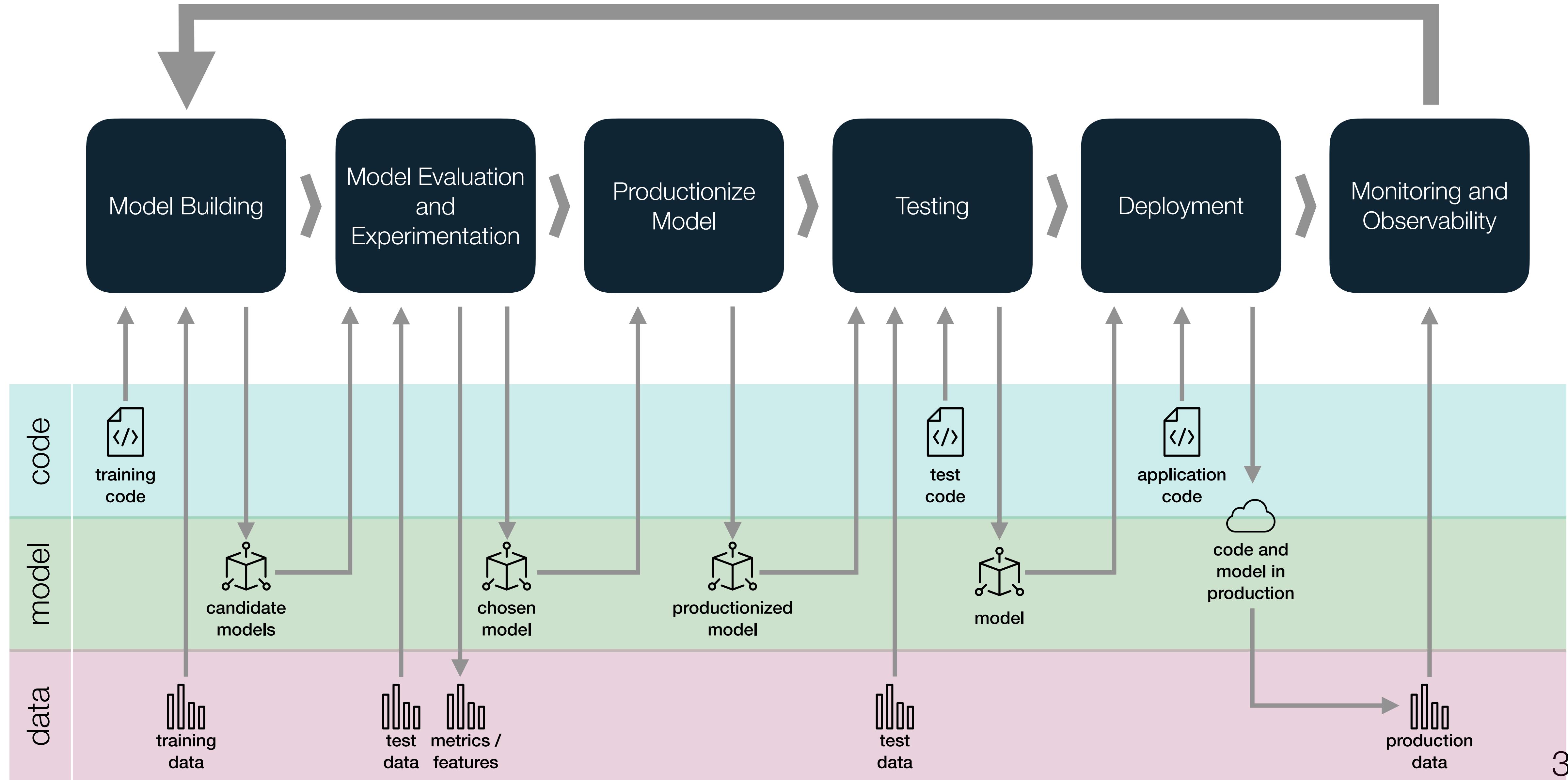
# ML Pipeline in Production



# ML Pipeline in Production



# ML Pipeline in Production



# More to Consider

- Scaling
- Edge deployments
- Optimizations
- Offline learning vs online learning
- Transfer learning
- Progress monitoring and UX