Deploying ML in Production

Suhaib Mujahid

smujahid@mozilla.com

Outline

Outline

- **Use Case:** Software Defect Prediction

Outline

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

Outline

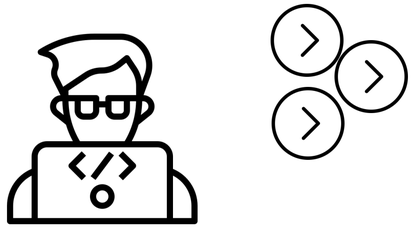
- **Use Case:** Software Defect Prediction
- **ML architecture**
- **Monitoring ML in production**

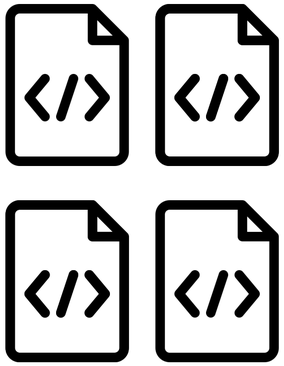
Outline

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

Software Development Process

Develop

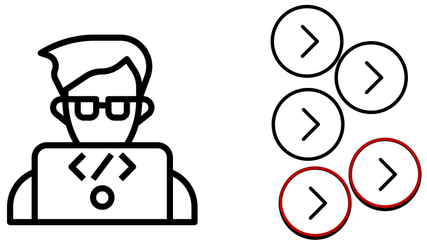


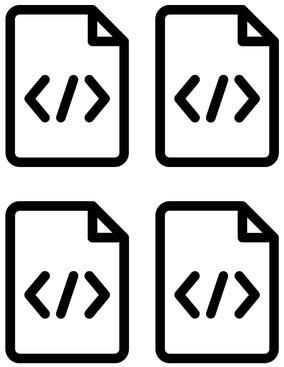


Review & Test

Software Development Process

Develop

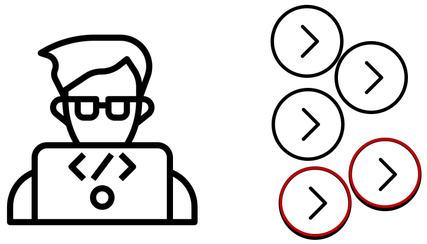


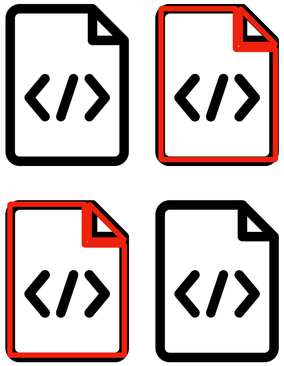


Review & Test

Software Development Process

Develop

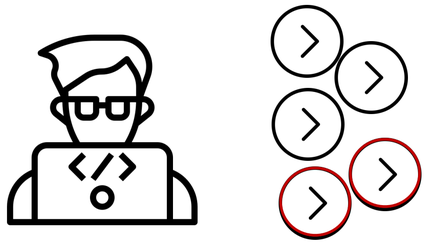


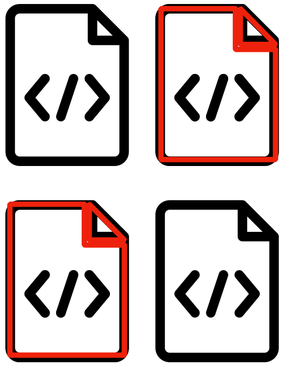


Review & Test

Software Development Process

Develop



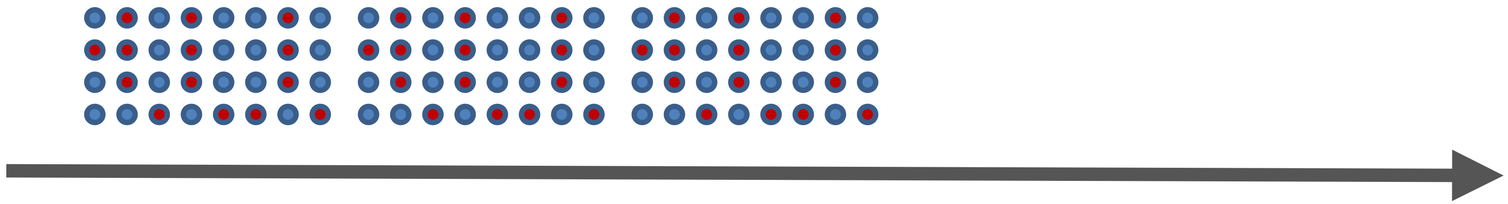


Review & Test

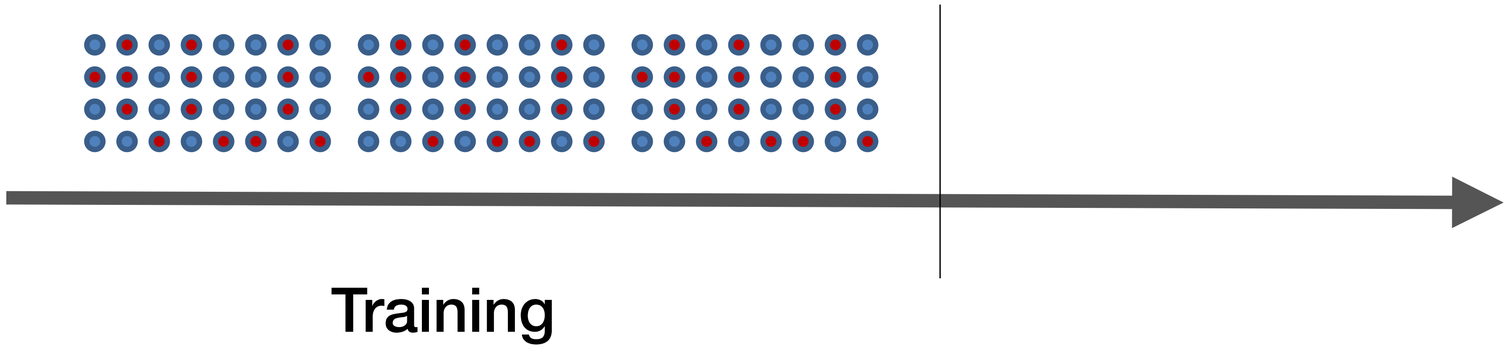
Defect Prediction

How it works…

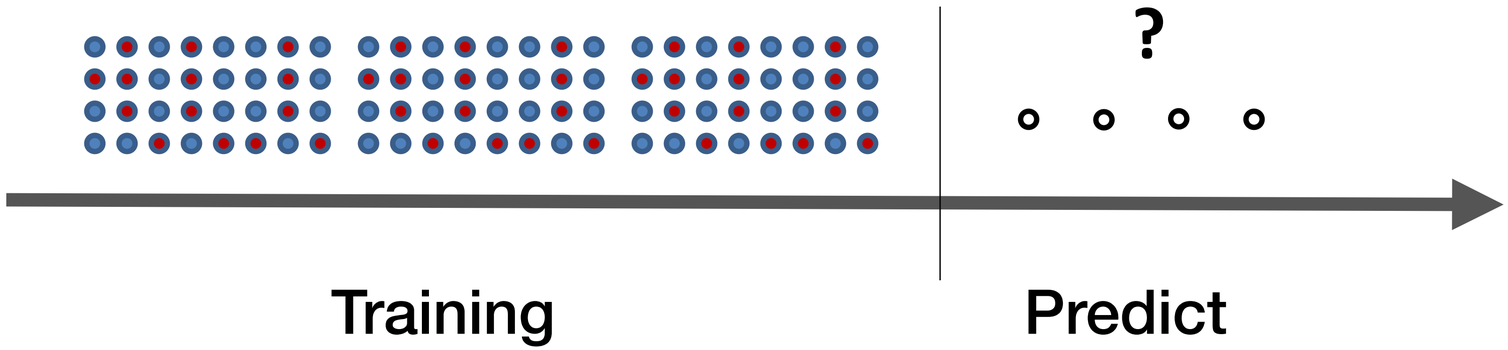
How it works…



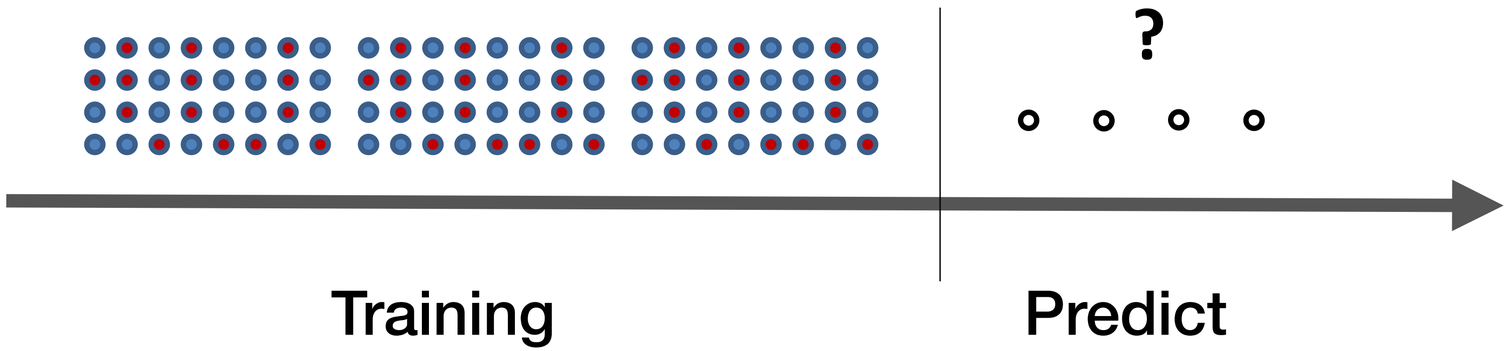
How it works…



How it works…



How it works…



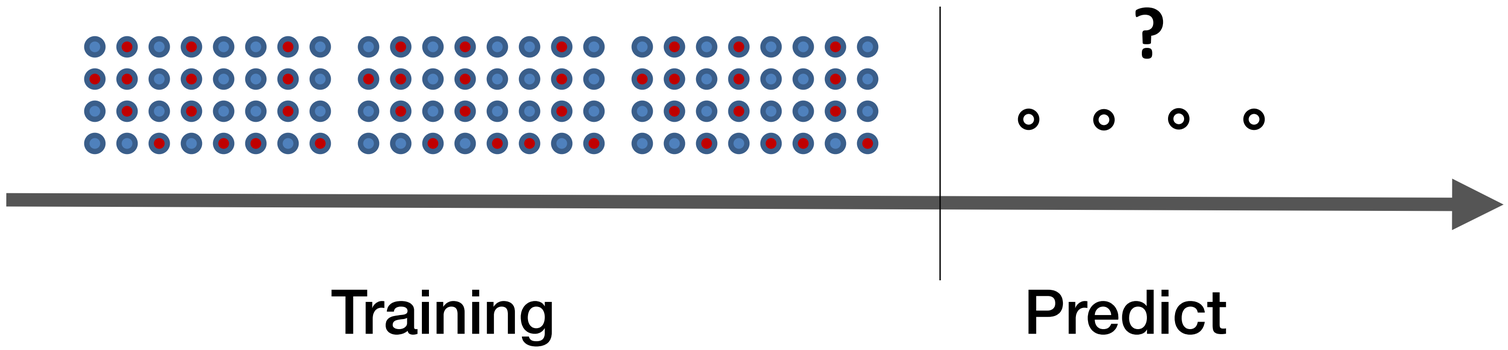

Size


History


Diffusion


Experience

How it works…

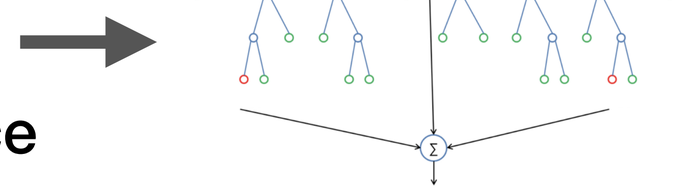


Size

History

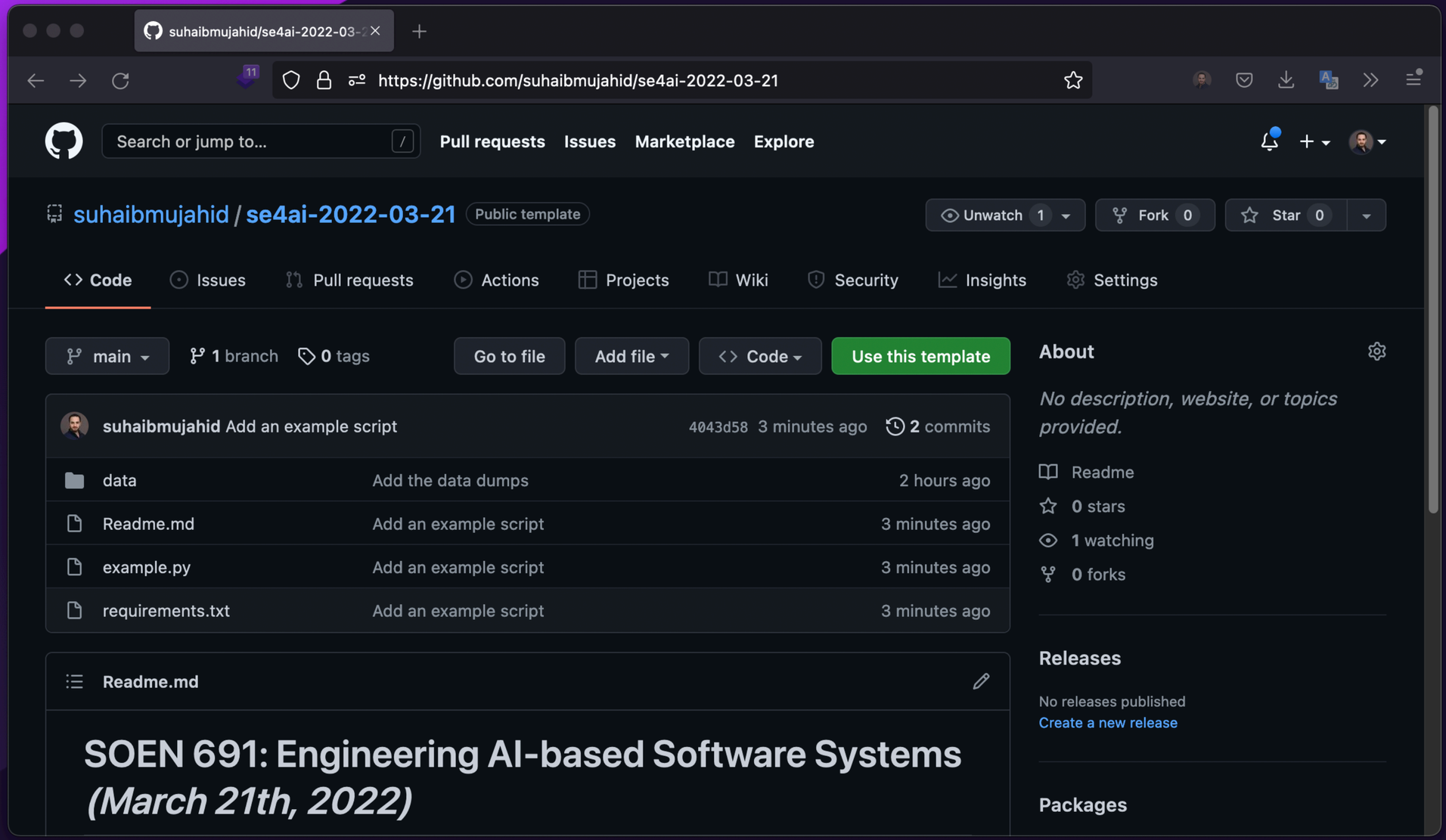
Diffusion

Experience



Software Defect Prediction Use Case

https://**bit.ly/se4ai-repo**



Metrics

```
type ChangeMeasures struct {
    // NS is the number of modified subsystems
    NS
    // NS 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:

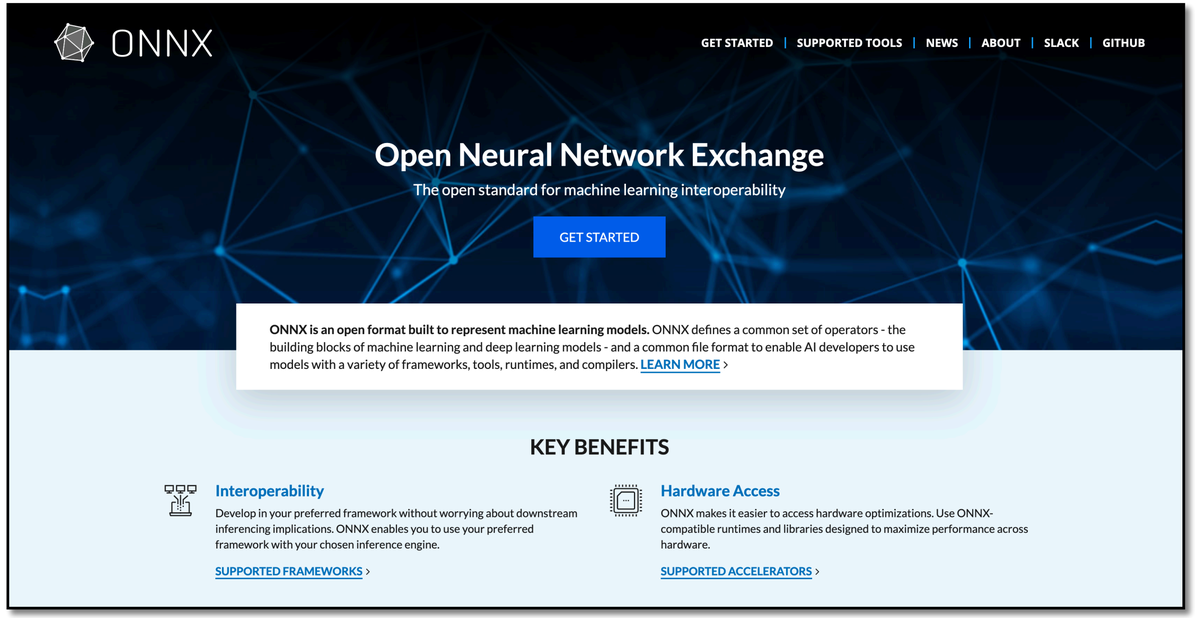
- **Reproducibility**
  - Dependency management
    1. Constrain the dependencies of the model
    2. Use containers
- **Scalability**
- **Performance (latency)**

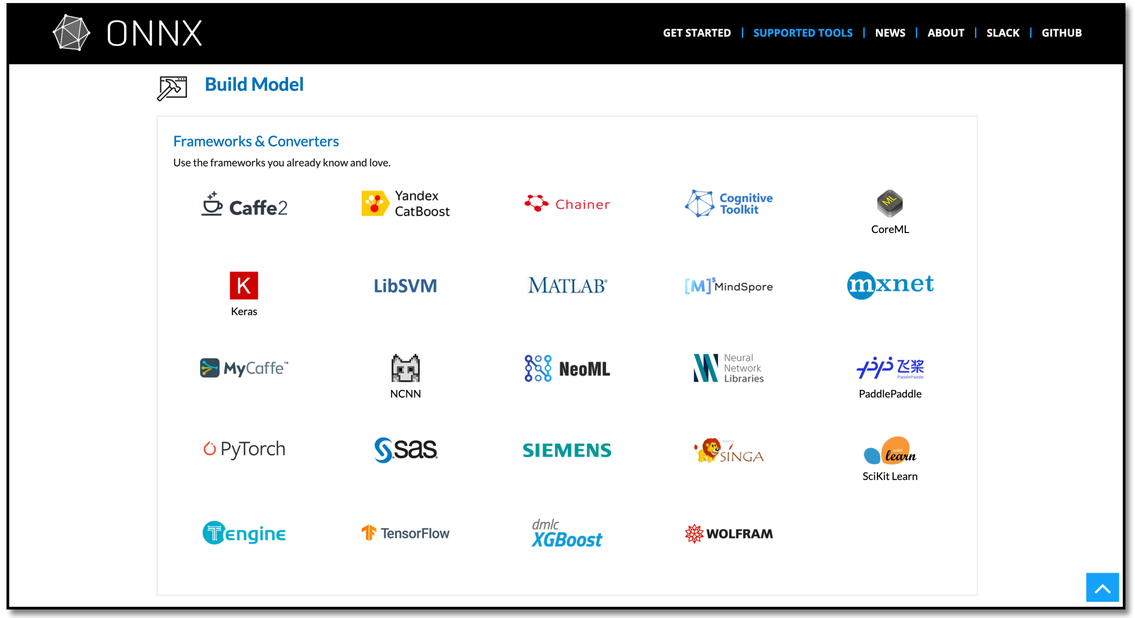
Framework Serving Tools

(TorchServe)

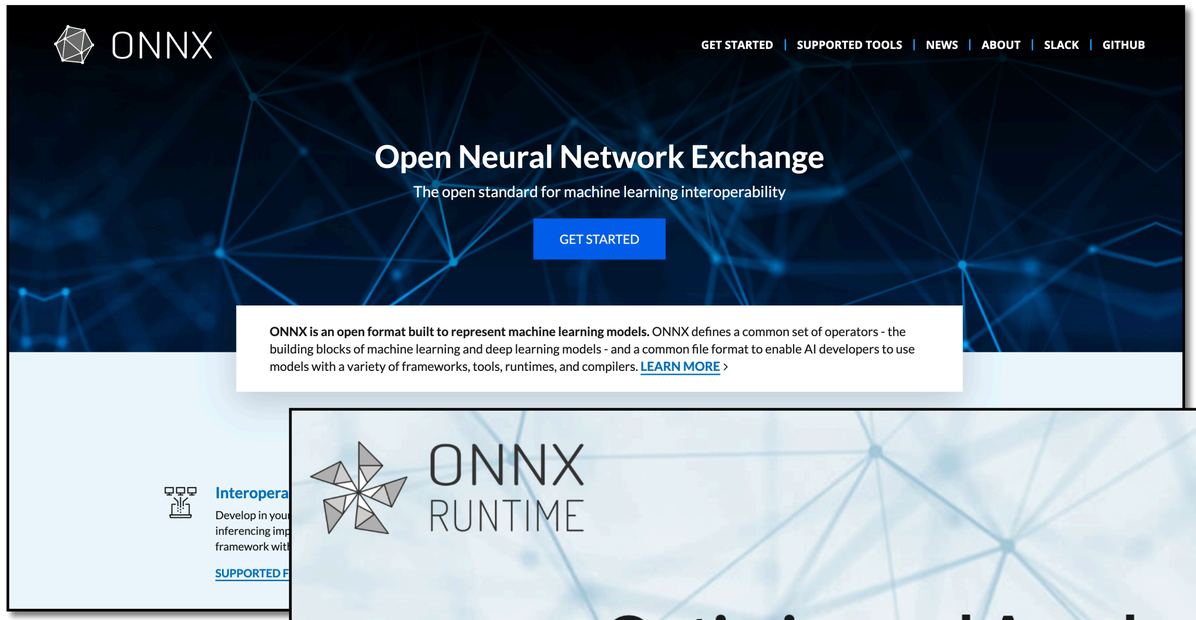
(TensorFlow Serving)

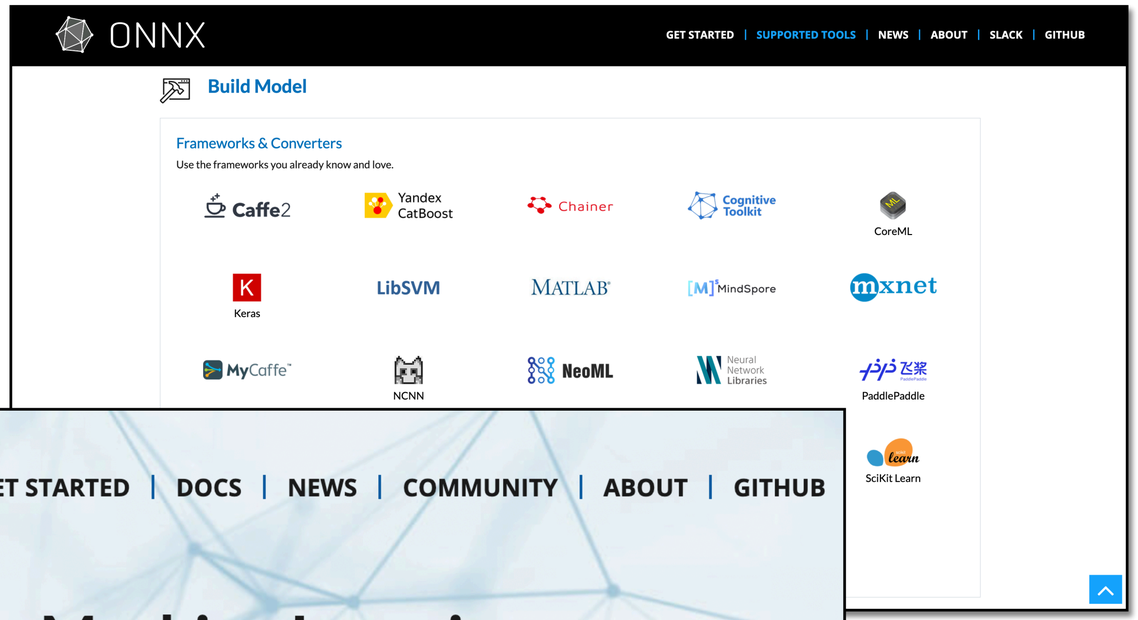
Framework Interoperability

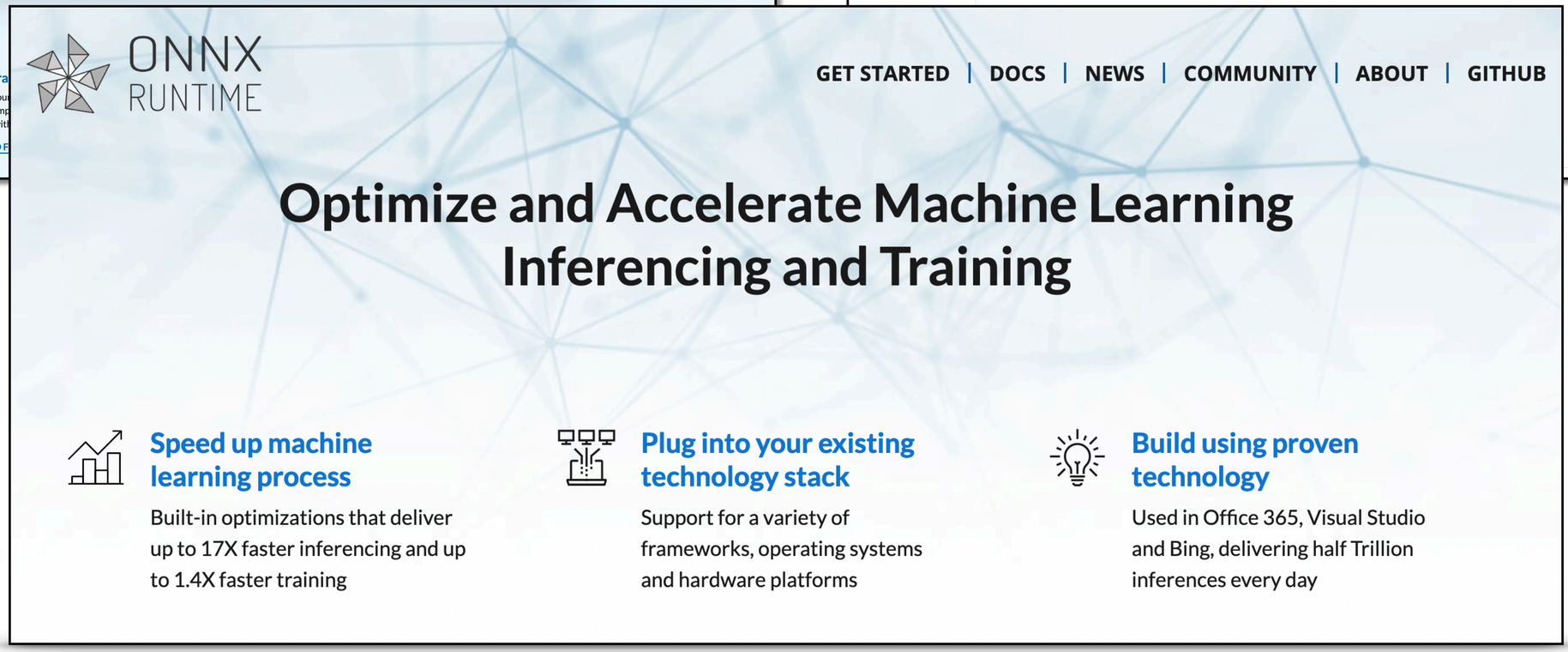




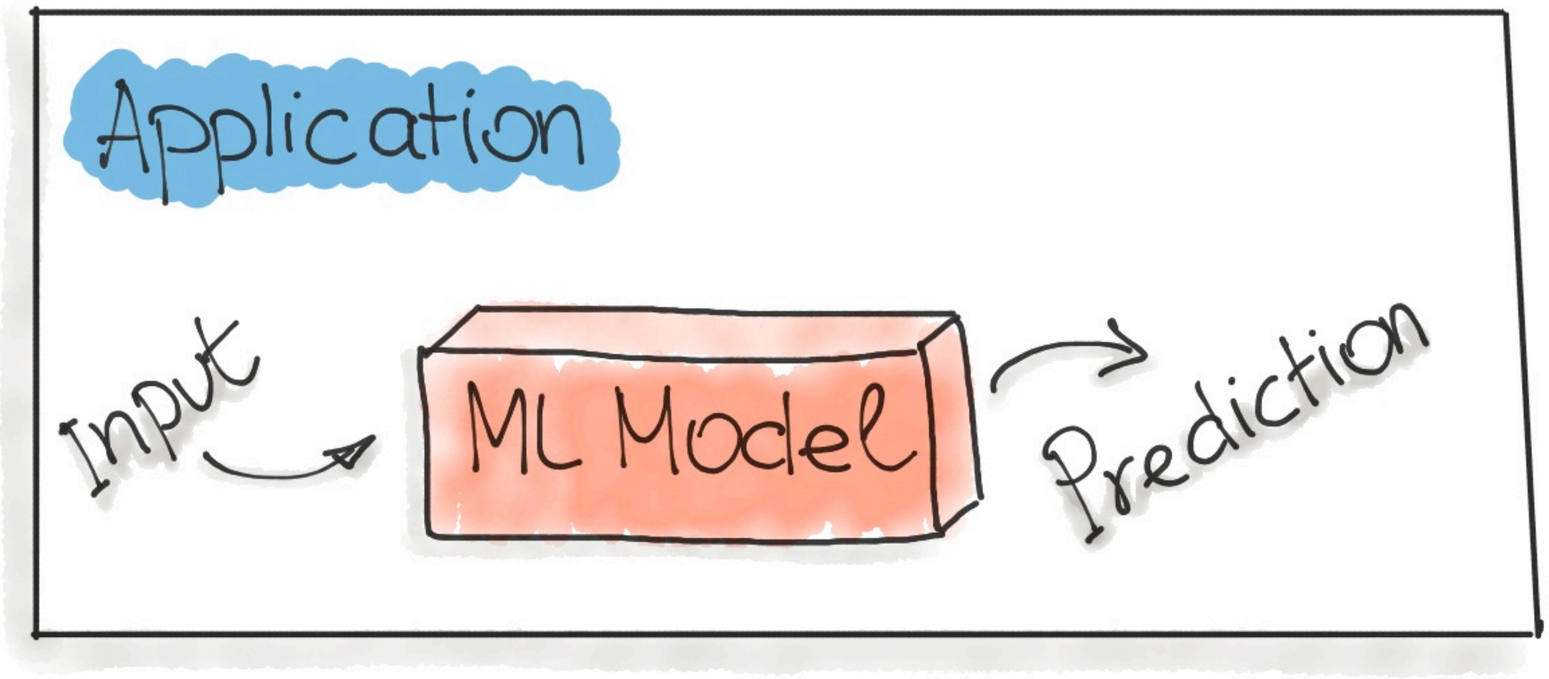
Framework Interoperability



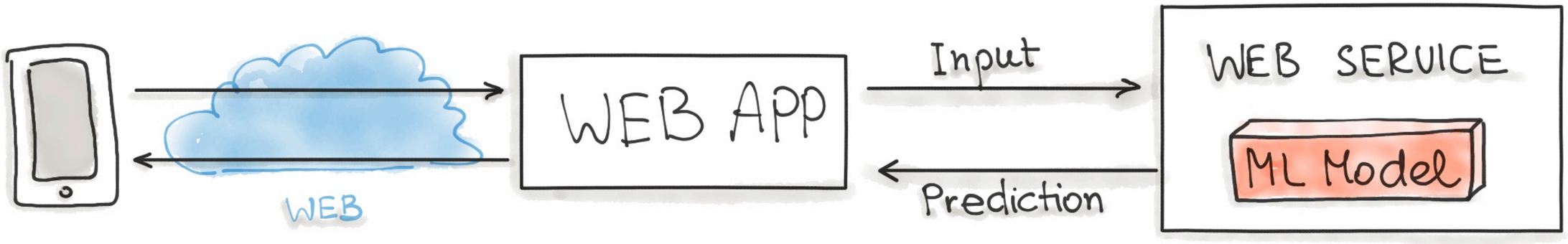




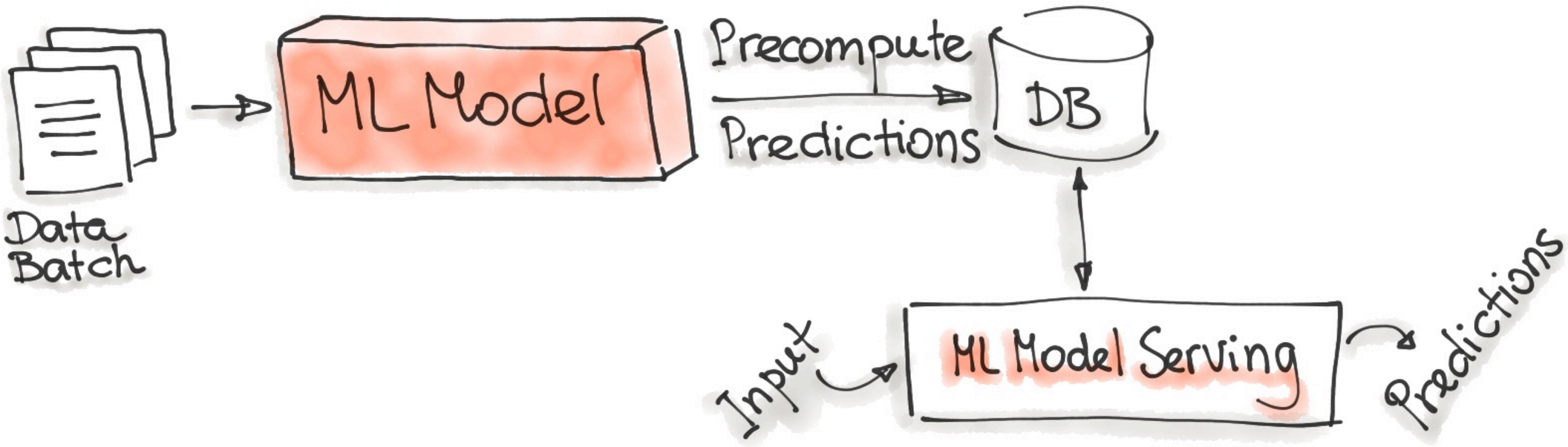
Model-as-Dependency



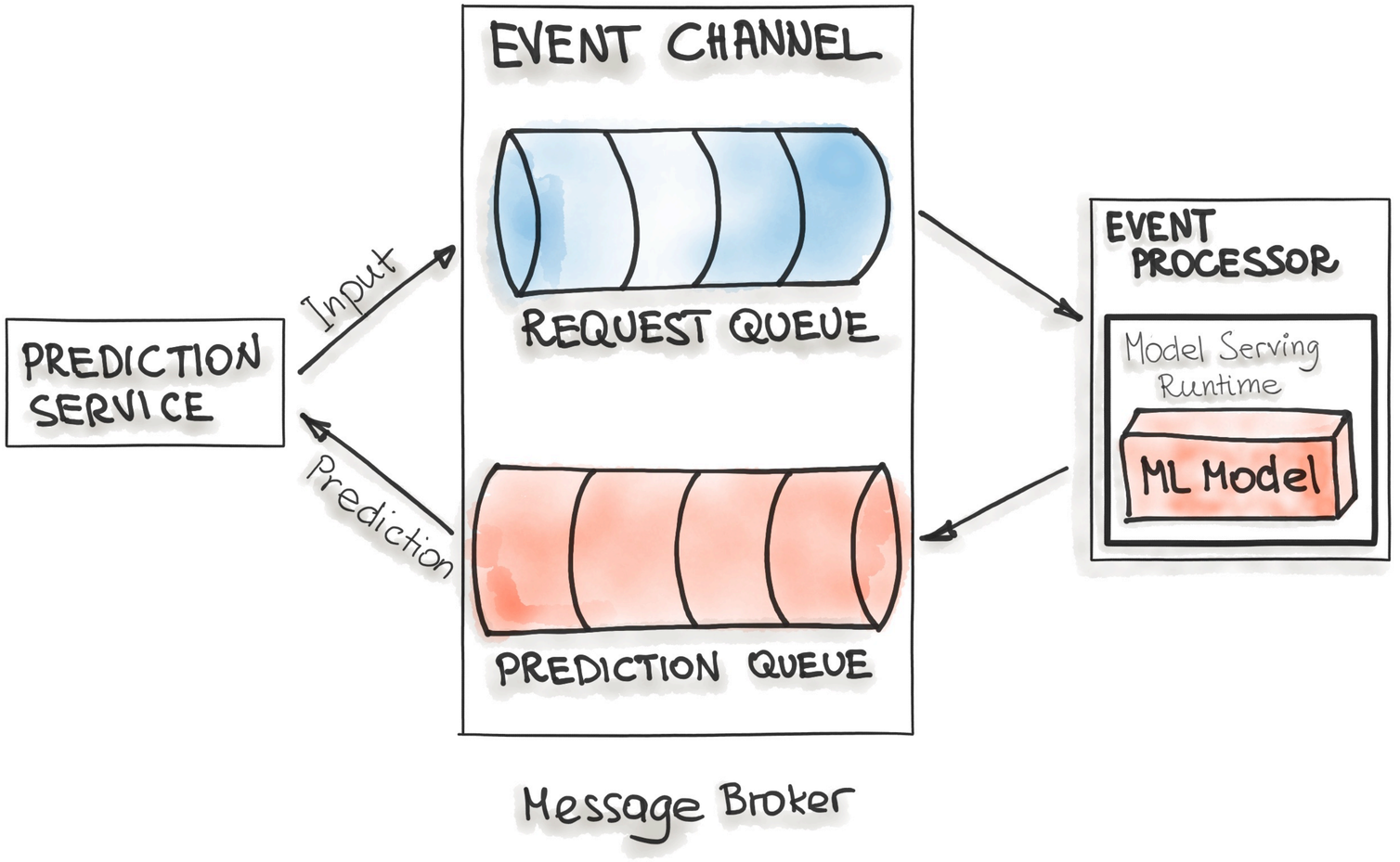
Model-as-Service



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

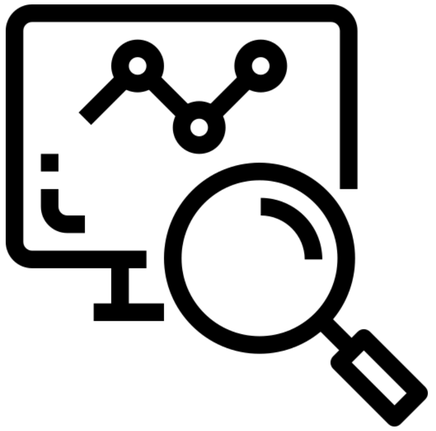
Deployment is not the last stage in model management,

it's just the beginning

What we should monitor?

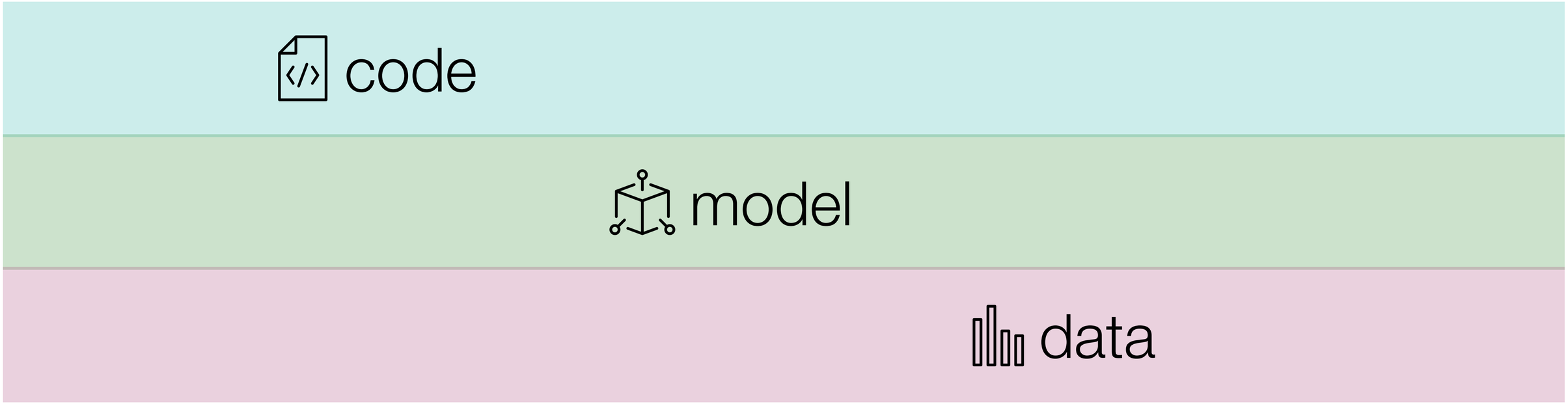
A good model monitoring pipeline should monitor:

- 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**



Monitoring is more central to ML than for traditional software

Add MLflow to the Example

example.py

```
@@ -5,6 +5,9 @@ from sklearn.model_selection import RandomizedSearchCV
 from sklearn.model_selection import train_test_split
 from sklearn.metrics import classification_report

+import mlflow
+import mlflow.sklearn
+
 max_training_rows = 10000
 target = 'buggy'
 date_column = 'author_date'
@@ -66,6 +69,13 @@ def main():
     report = classification_report(y_test, y_pred)
     print(report)

+    mlflow.sklearn.autolog()
+    mlflow.log_param("ignored_days", ignored_days)
+    mlflow.log_metric("training_rows", len(X_train))
+    mlflow.log_metric("testing_rows", len(X_test))
+    mlflow.sklearn.log_model(model, "model")
+    mlflow.sklearn.eval_and_log_metrics(model, X_test, y_test, prefix="val_")
+

 if __name__ == '__main__':
     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**

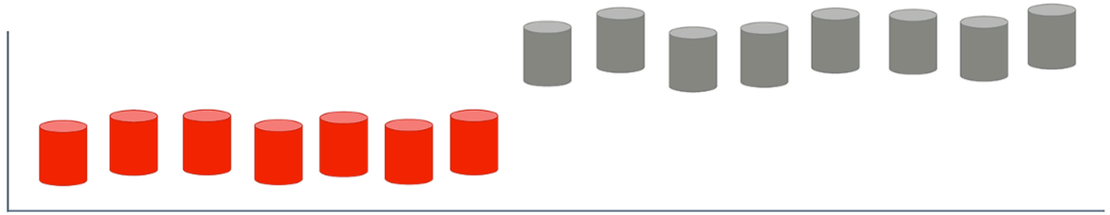
Gradual concept drift

What is concept drift?

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

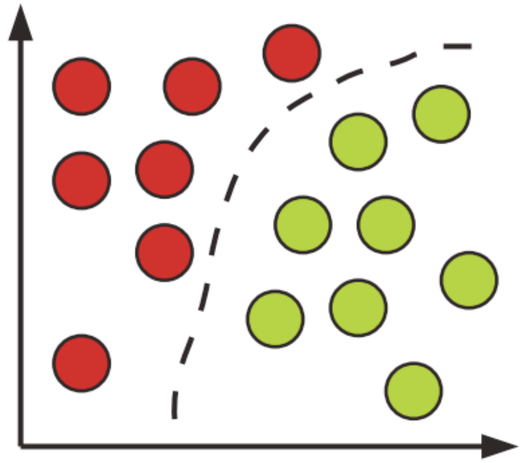
Gradual concept drift

Sudden concept drift

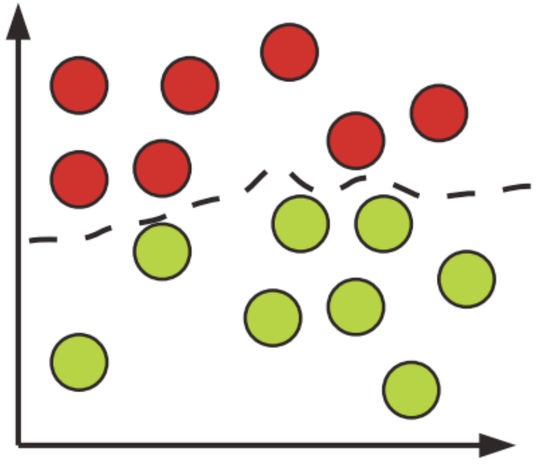


Data drift vs Concept drift

Orginal data

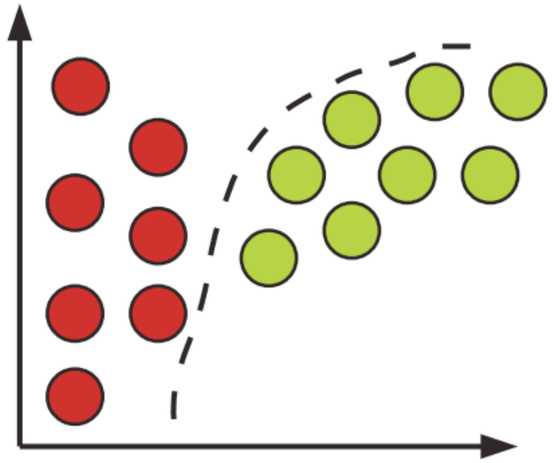


Concept drift



p(y|X) changes

Data drift



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

https://dl.acm.org/doi/10.1145/2523813

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

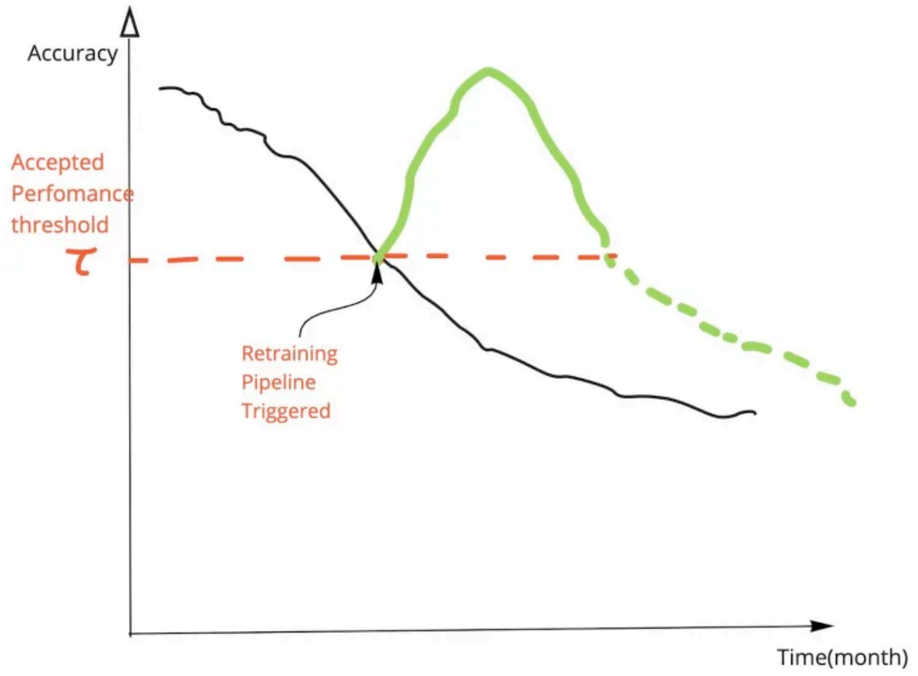
What is Continuous Training?

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?

Figure source: https://ml-ops.org/content/mlops-principles

When should you retrain a model?

Retraining based on
interval

Figure source: https://ml-ops.org/content/mlops-principles

When should you retrain a model?

Retraining based on
interval

Performance-based
trigger

Figure source: https://ml-ops.org/content/mlops-principles

When should you retrain a model?

Retraining based on
interval

Performance-based
trigger

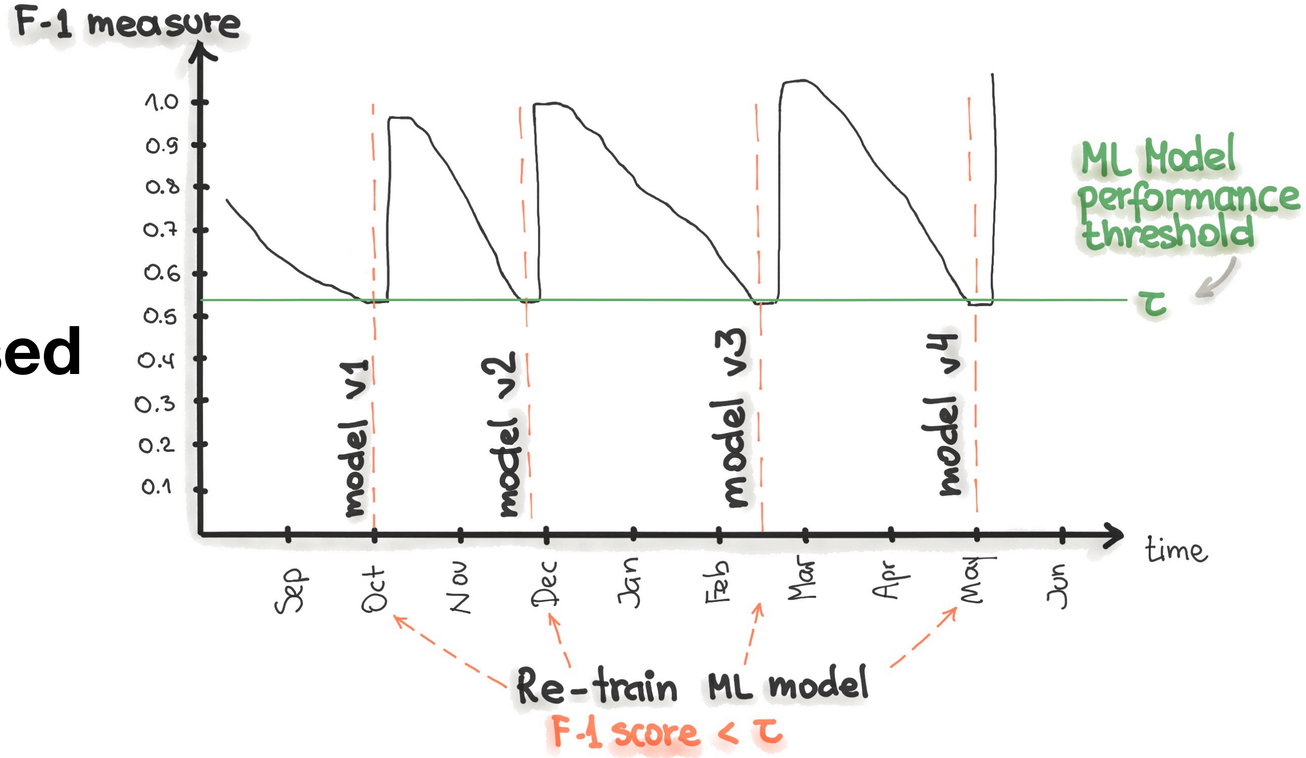
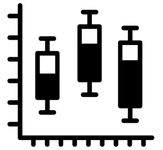


Figure source: https://ml-ops.org/content/mlops-principles

When should you retrain a model?

Retraining based on **interval**

Performance-based trigger



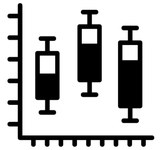
Trigger based on **data changes**

Figure source: https://ml-ops.org/content/mlops-principles

When should you retrain a model?

Retraining based on **interval**

Performance-based trigger



Trigger based on **data changes**

Retraining on **demand**

Figure source: https://ml-ops.org/content/mlops-principles

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**

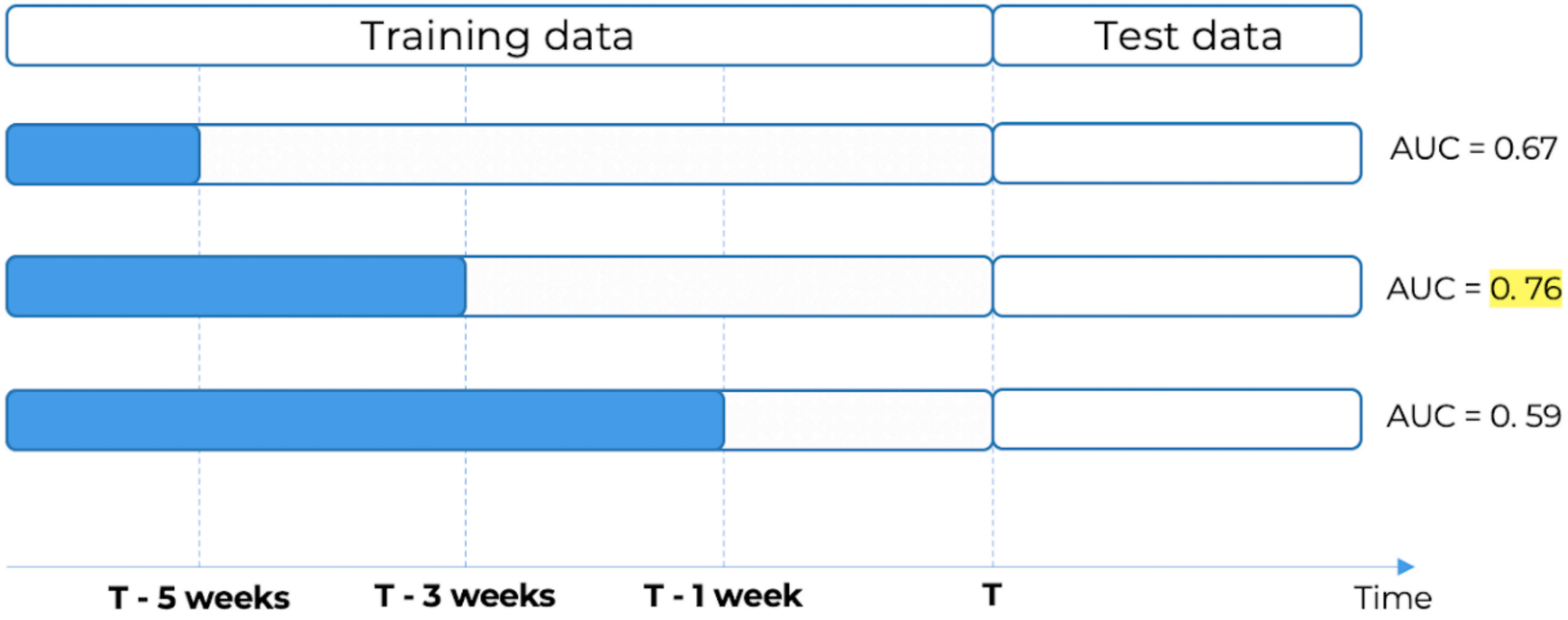
How much data is needed for retraining?

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

https://towardsdatascience.com/framework-for-a-successful-continuous-training-strategy-8c83d17bb9dc

What should be retrained?

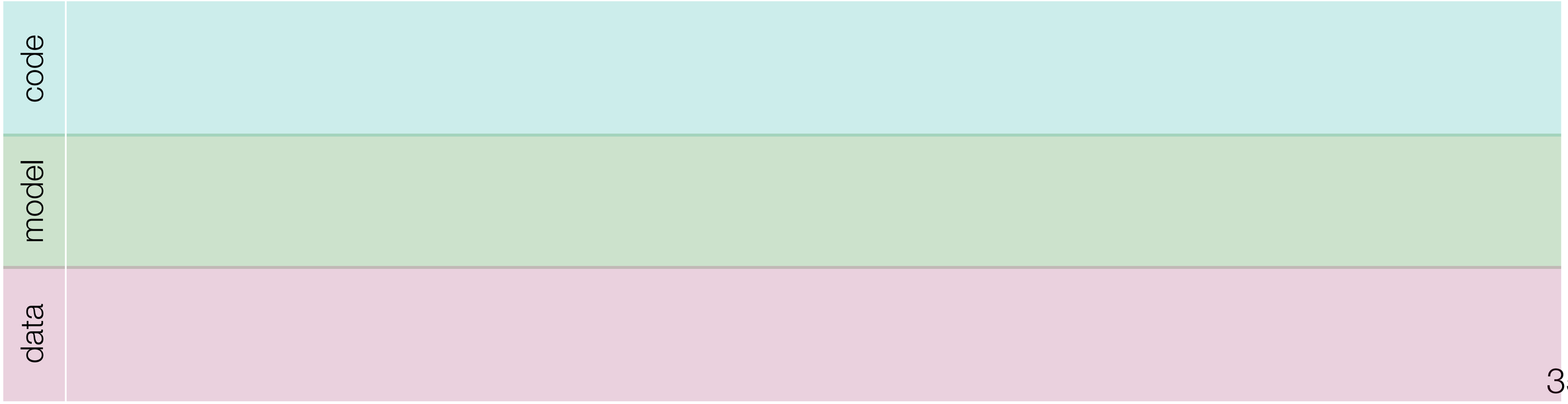
- **Level of retraining:**
  - Are you just refitting the model (or other steps) with the new data?
  - Do you do hyperparameters optimization?
  - Do you challenge the selection of the model itself and test for other model types?

Deployment after retraining model

Should you deploy as soon as you retrain?

- Competing models
  - Shadow models
    - Multiple models

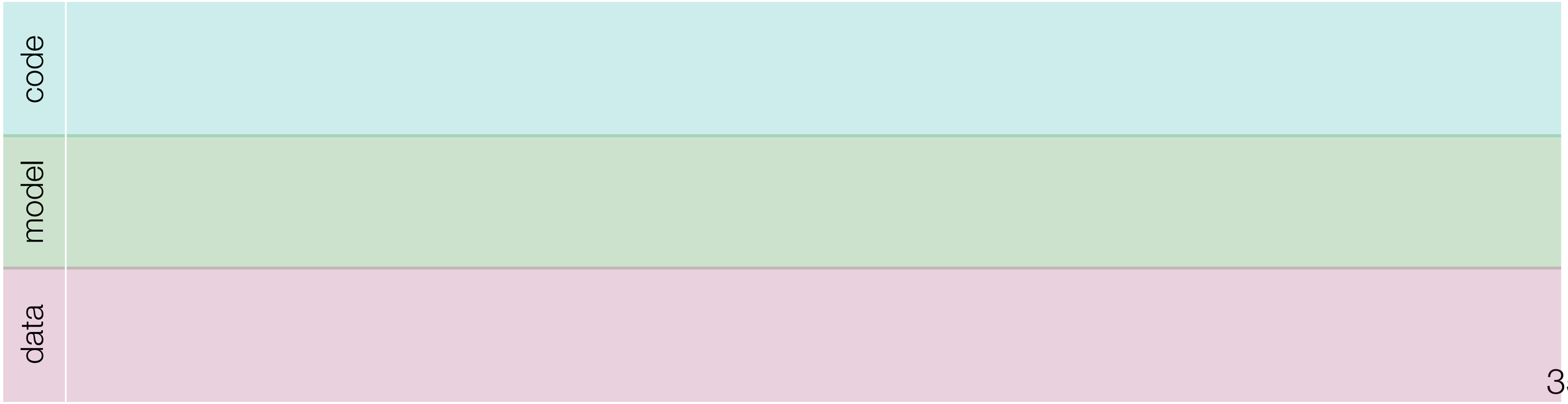
ML Pipline in Porduction



https://martinfowler.com/articles/cd4ml.html

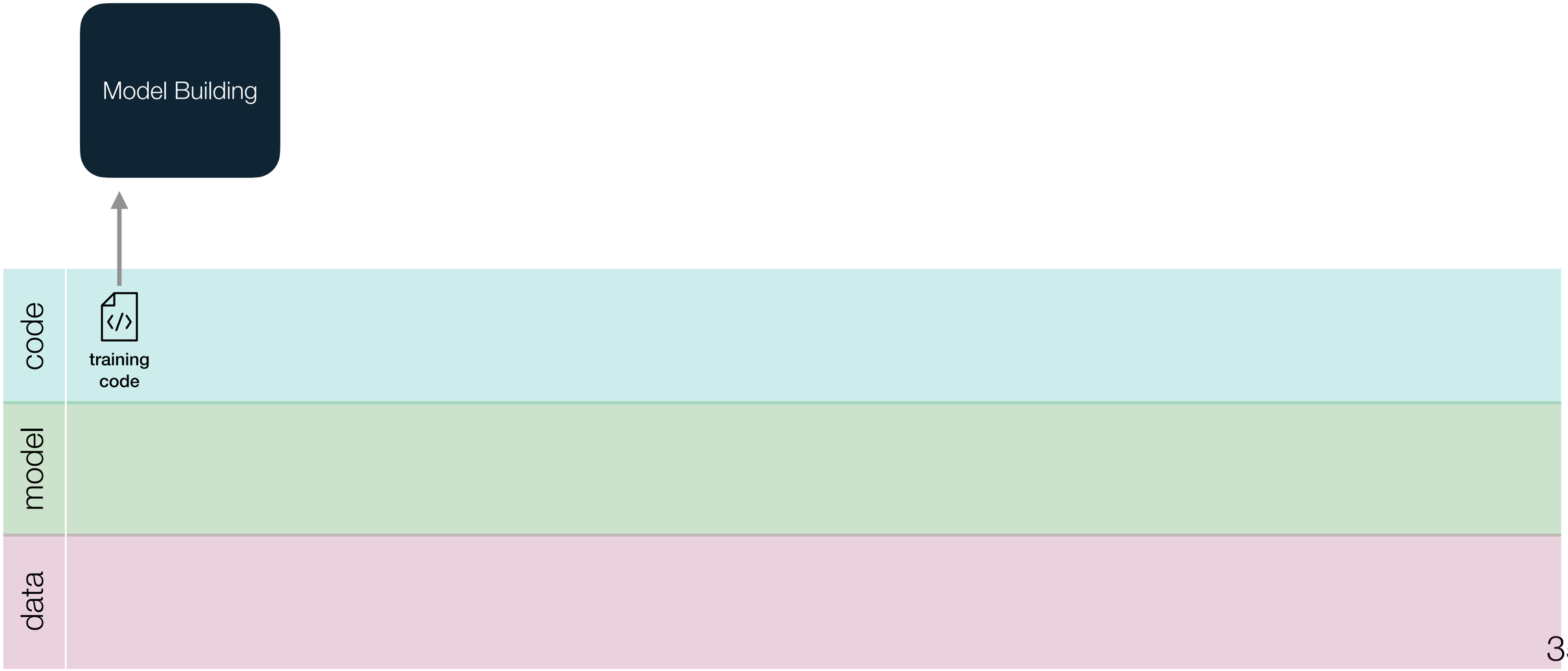
ML Pipline in Porduction

Model Building



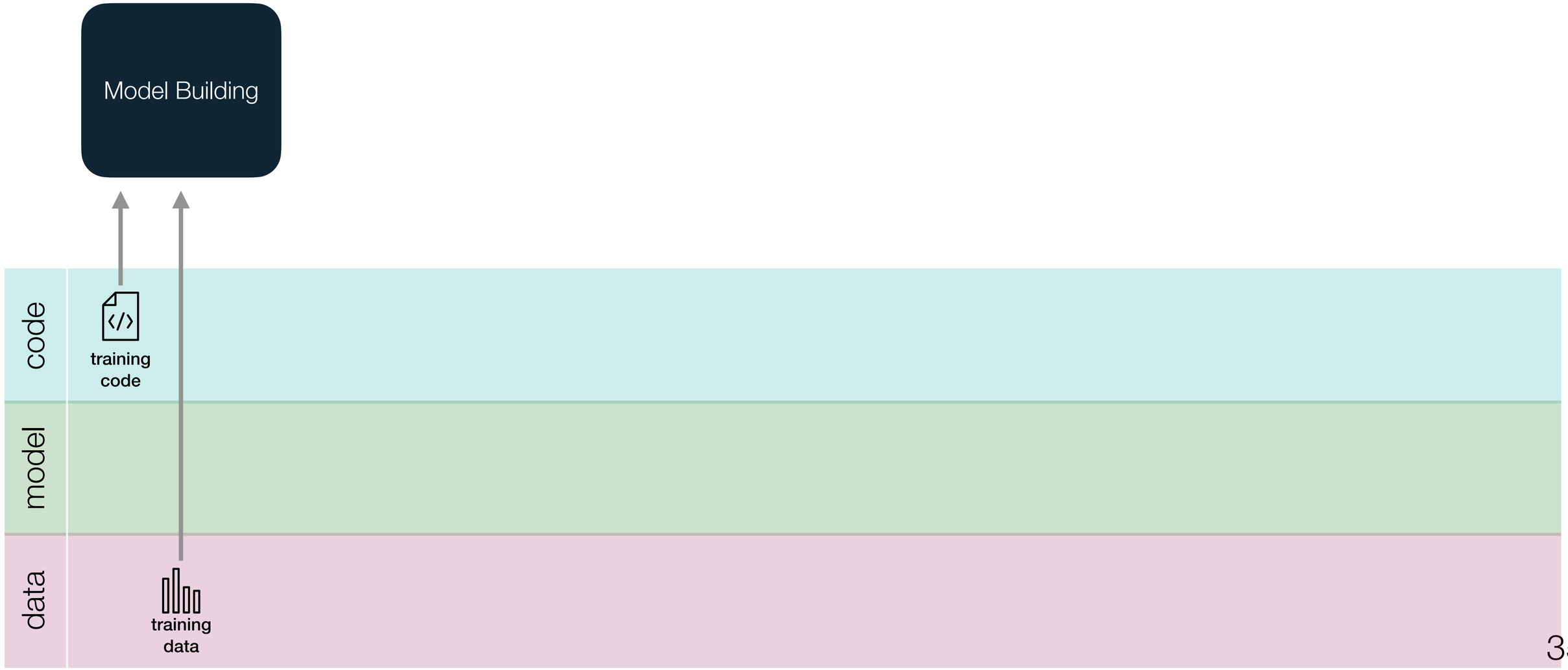
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



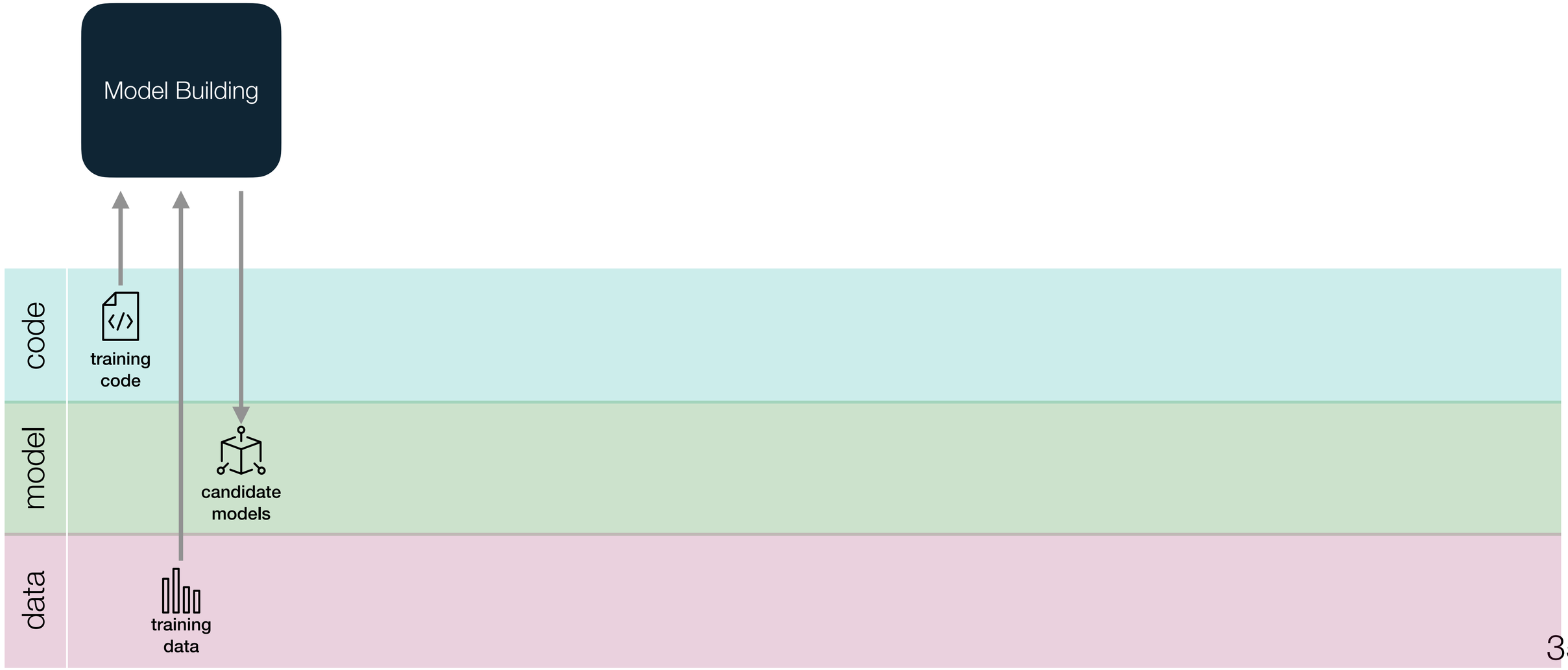
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



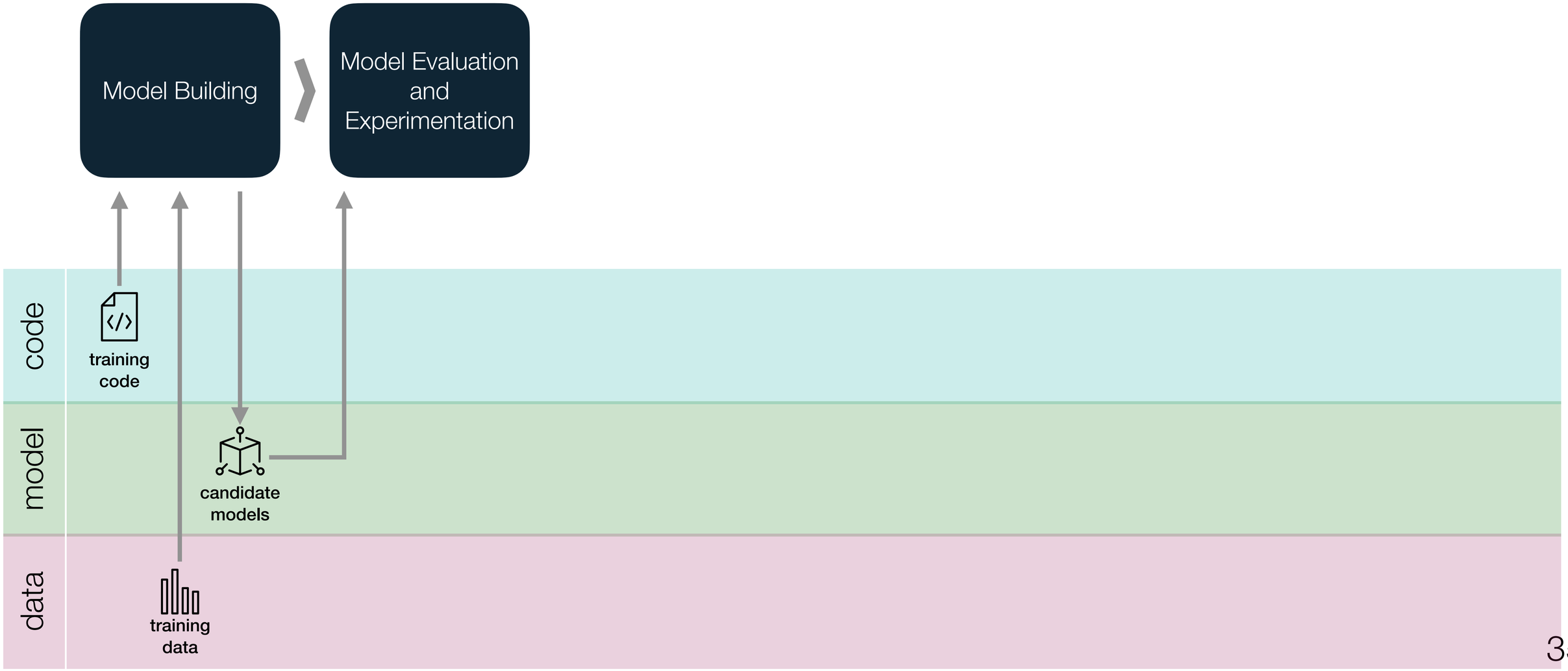
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



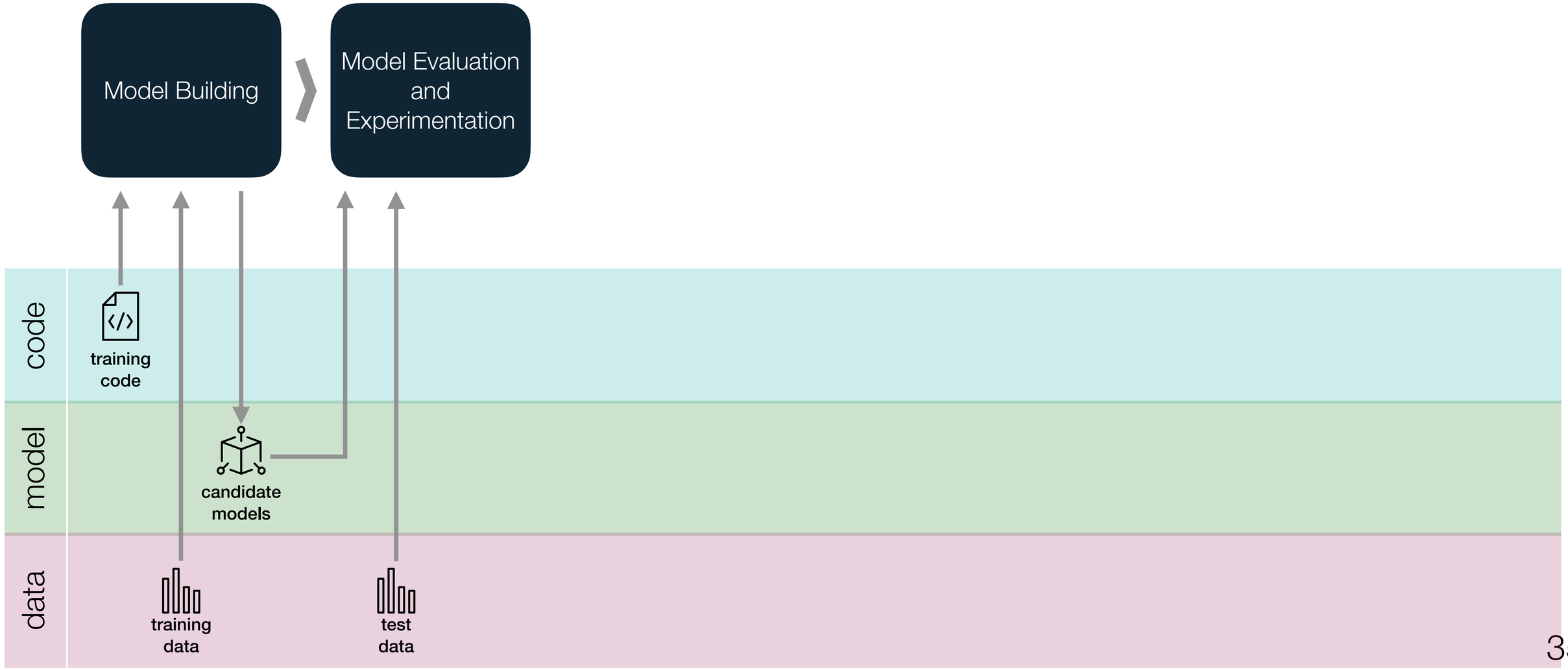
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



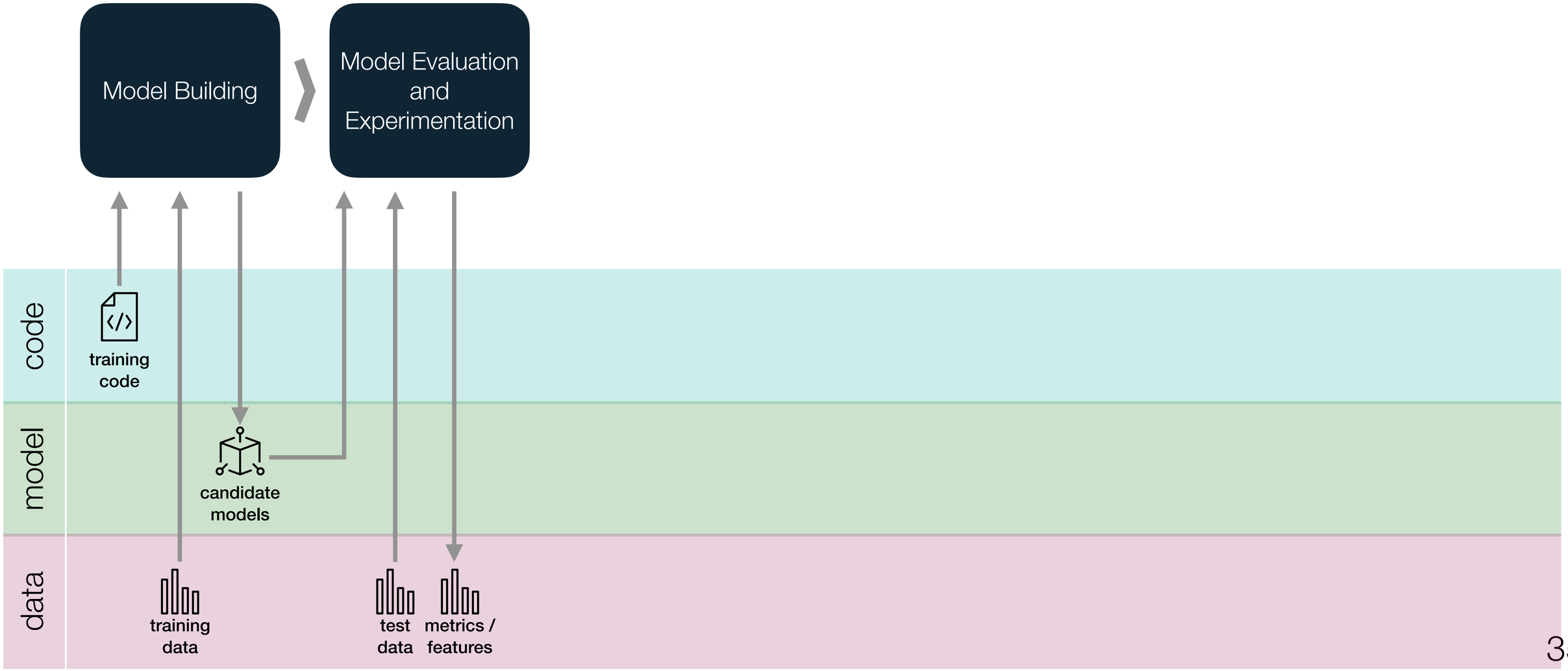
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



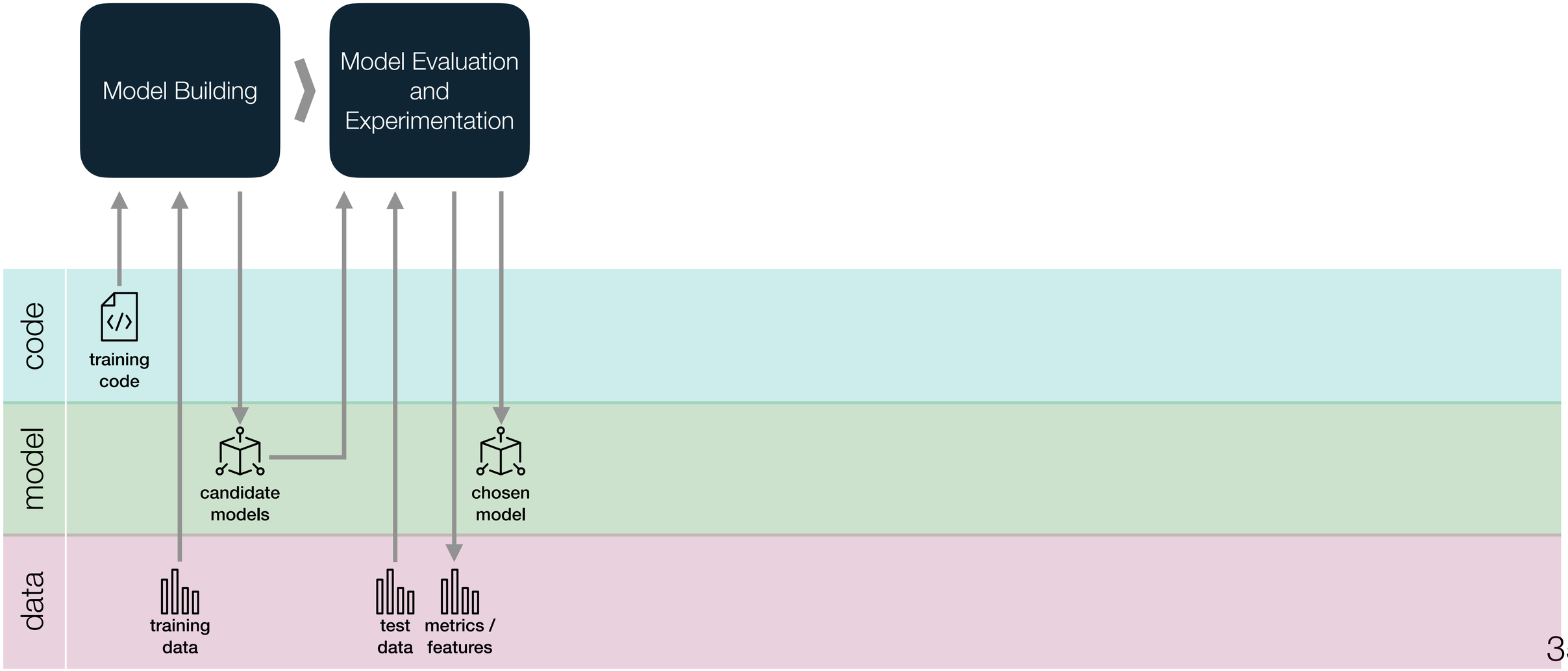
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



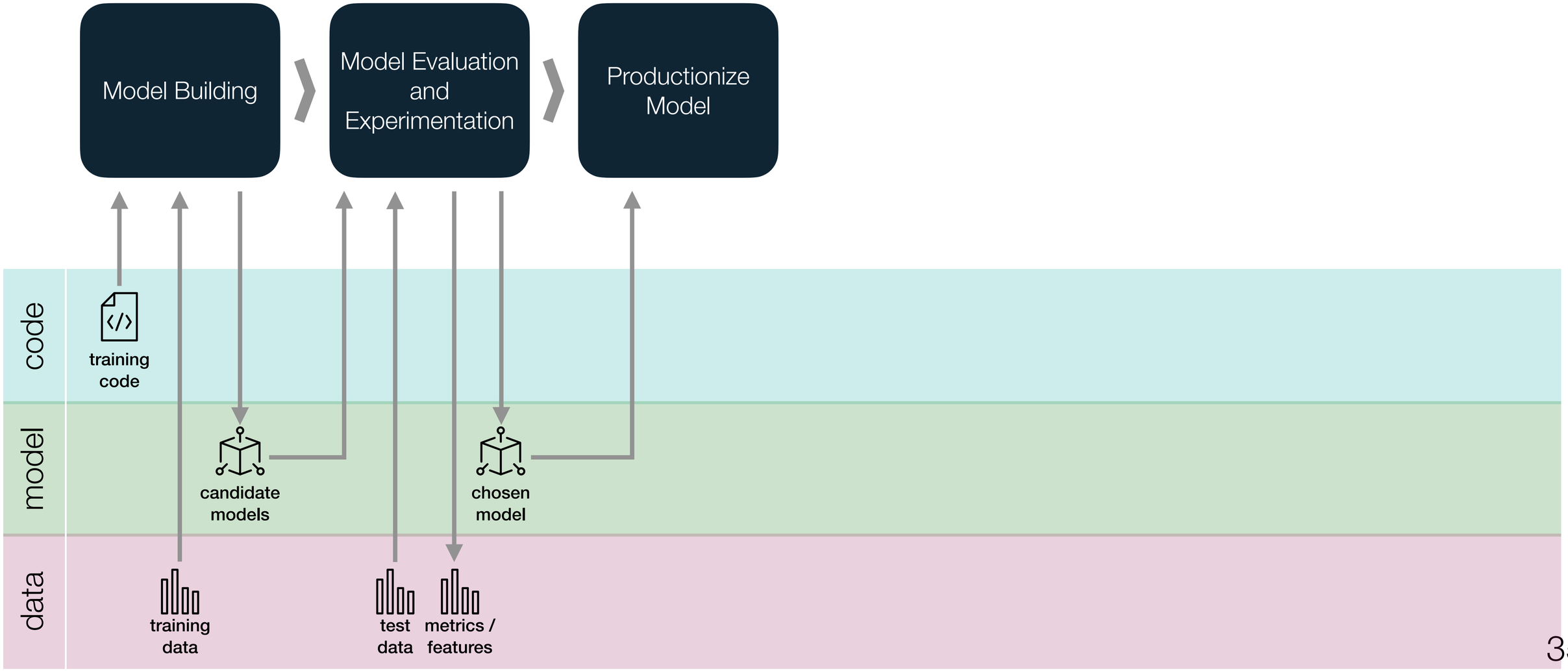
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



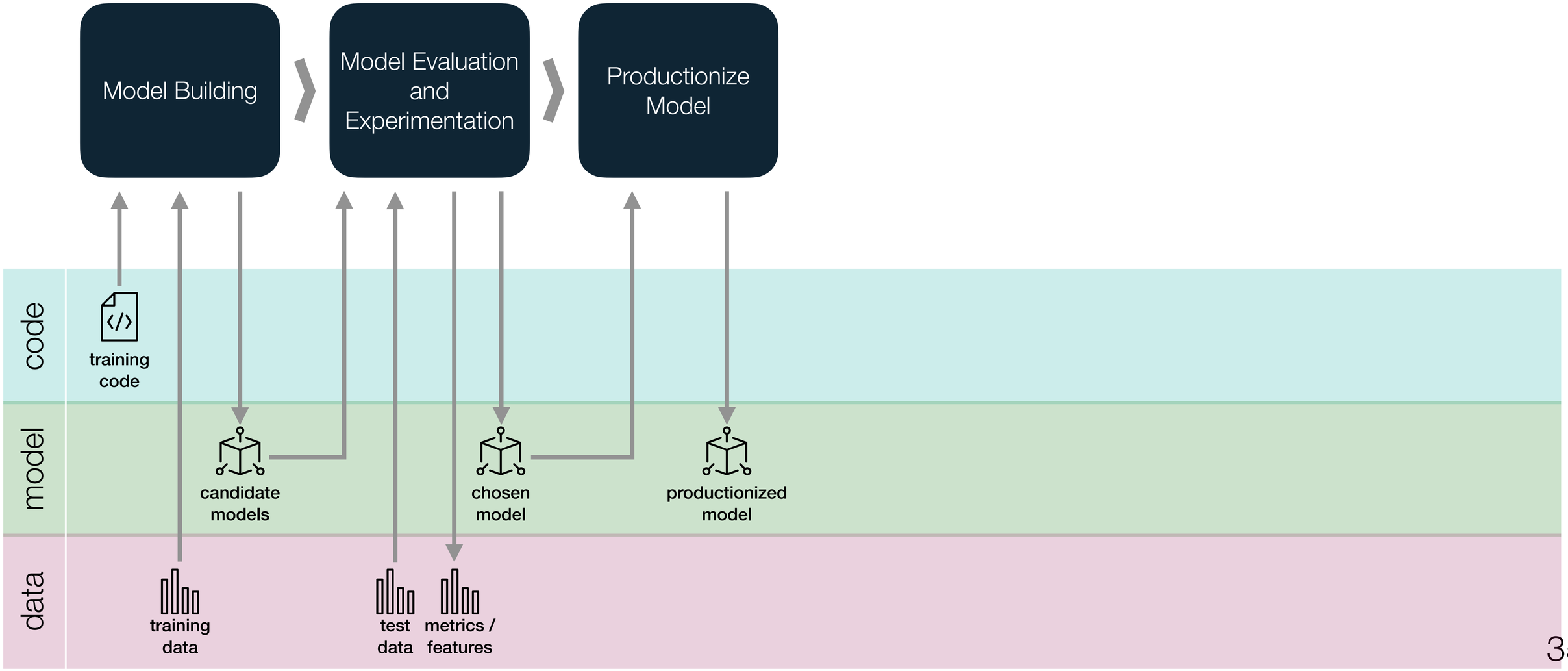
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



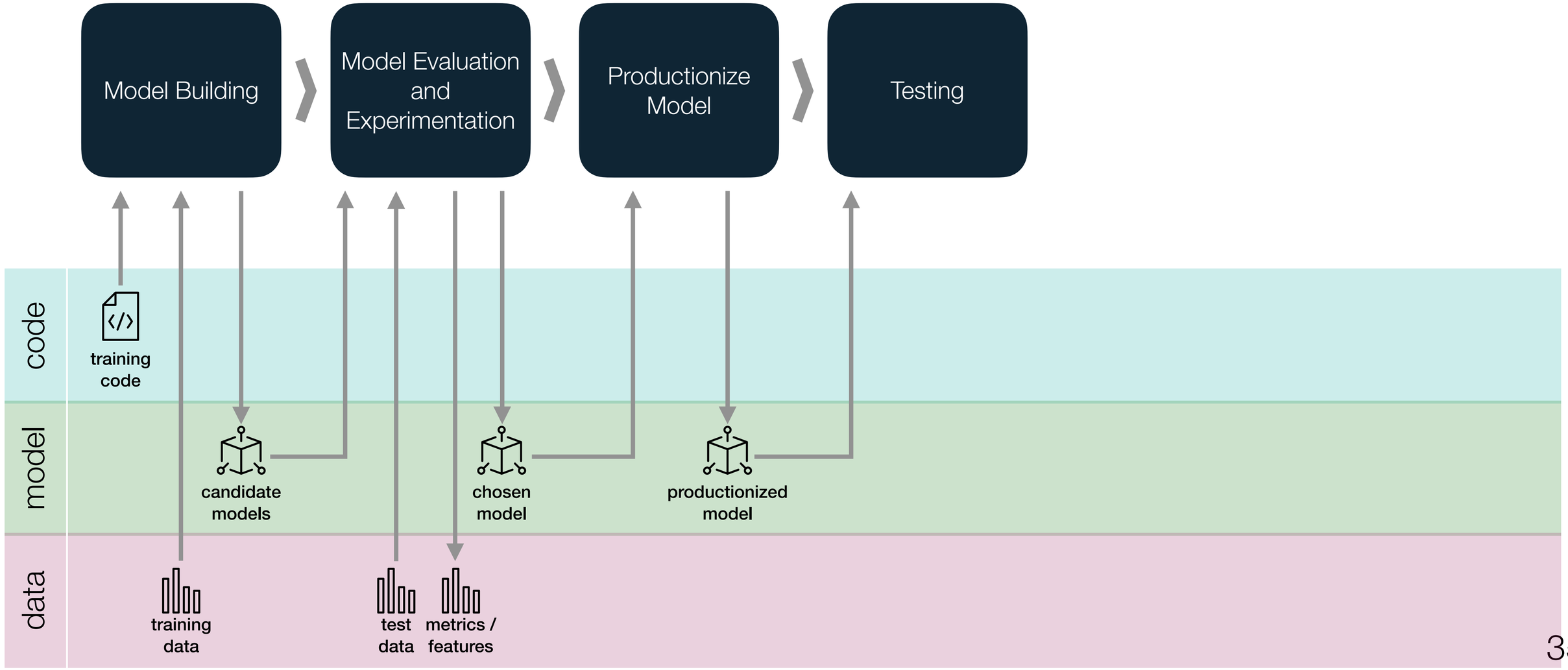
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



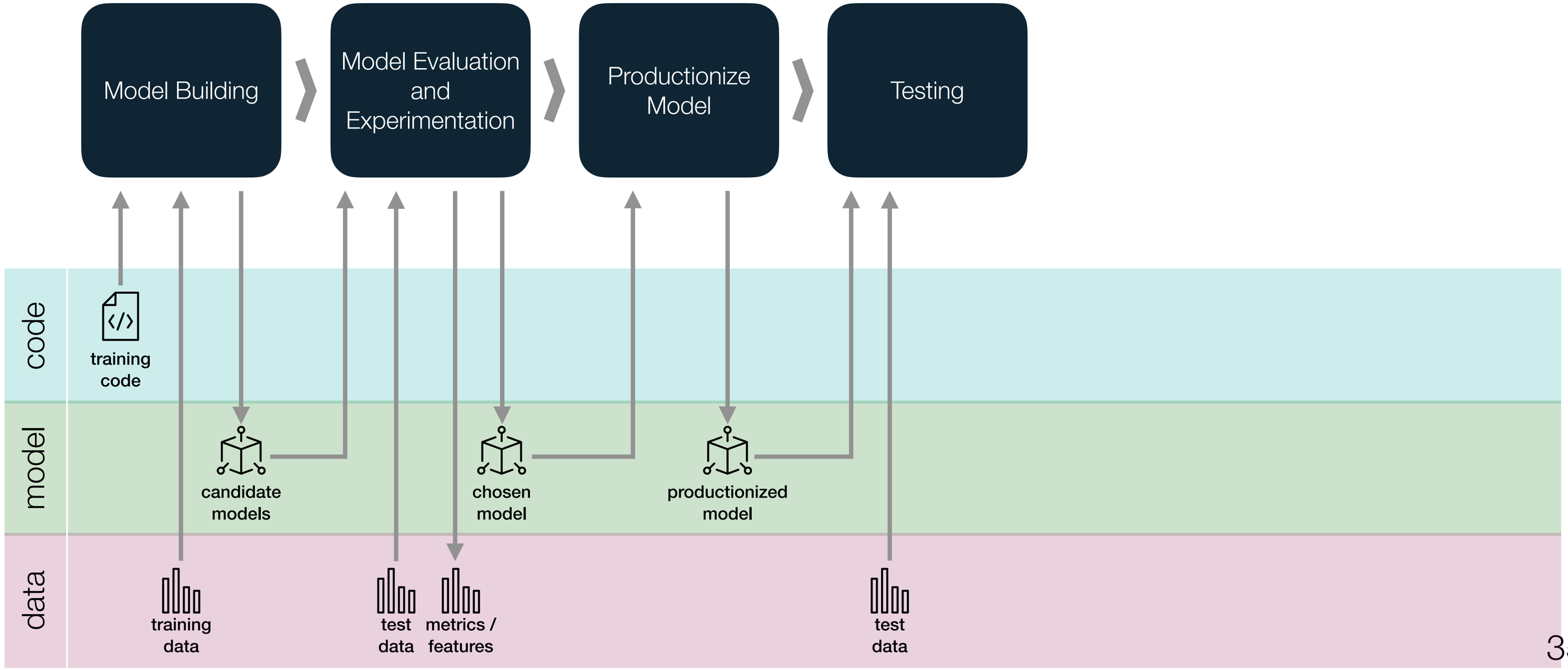
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



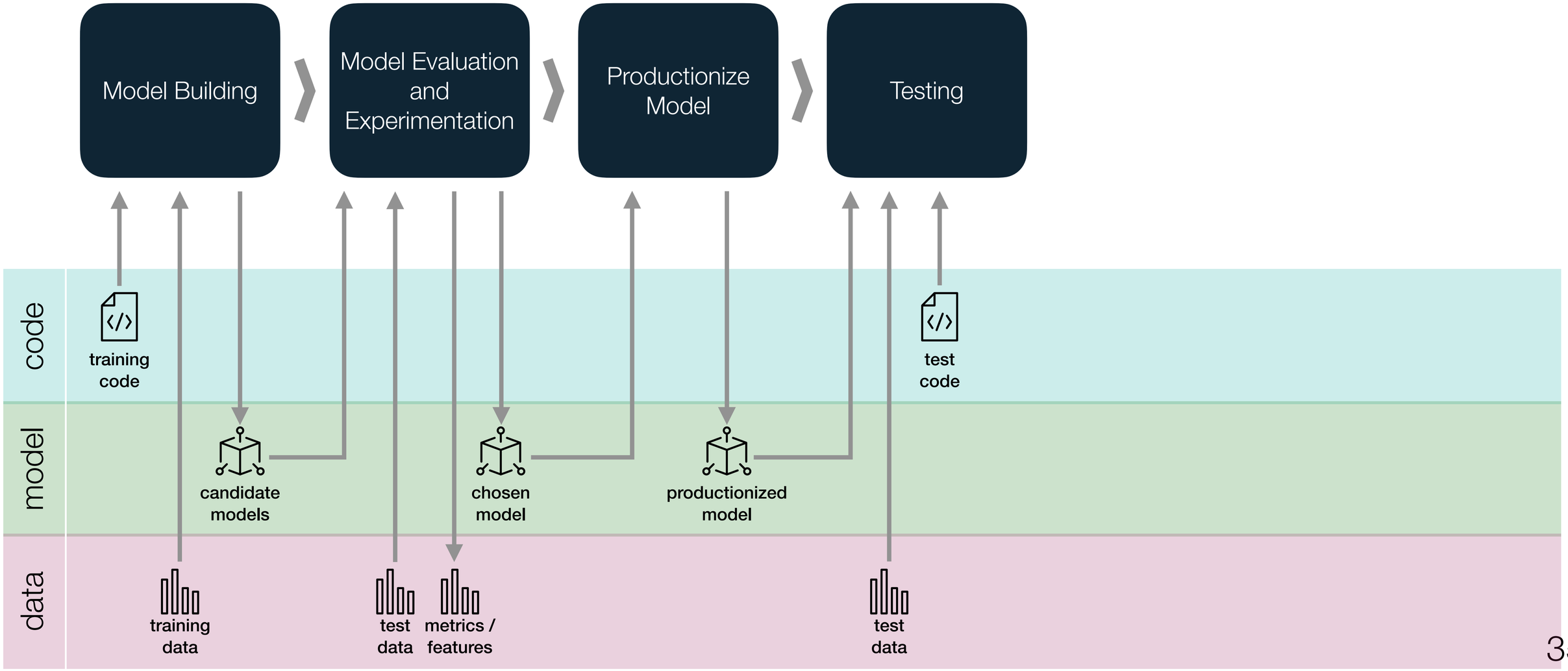
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



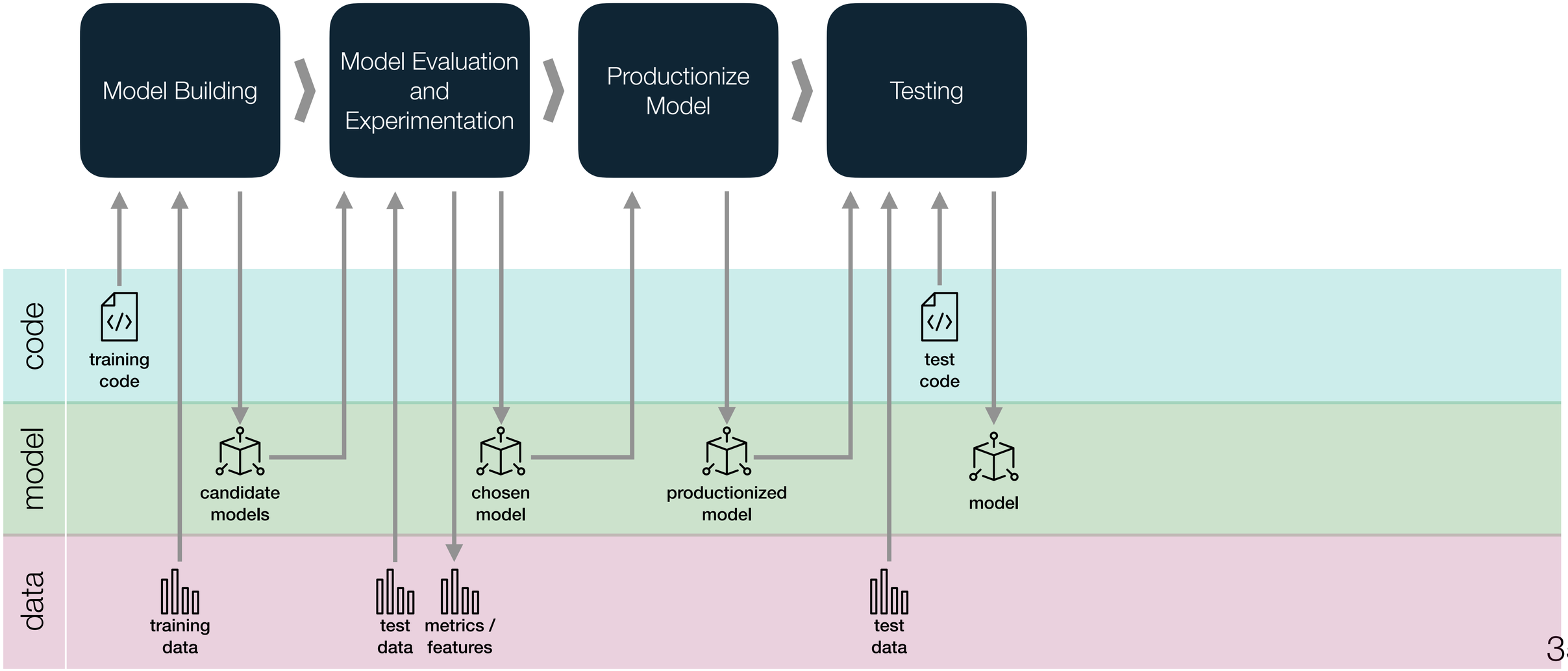
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



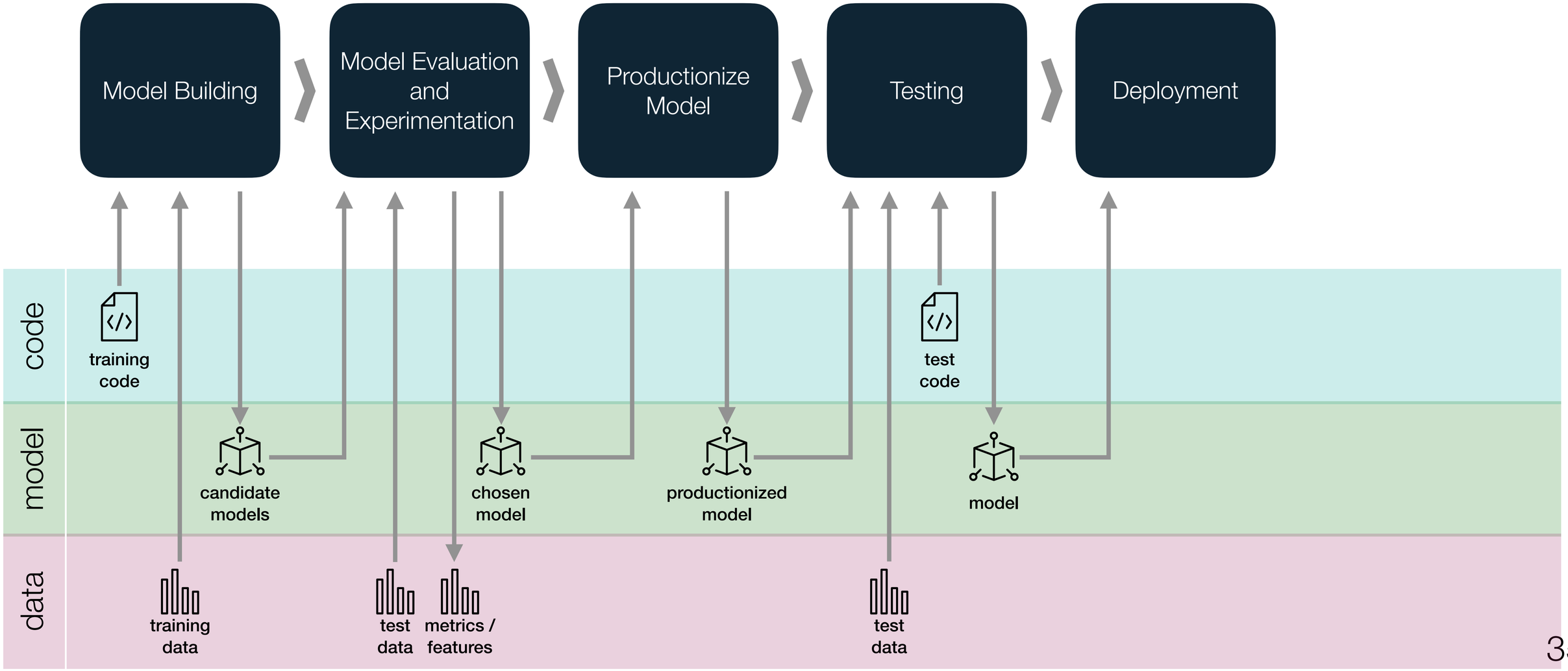
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



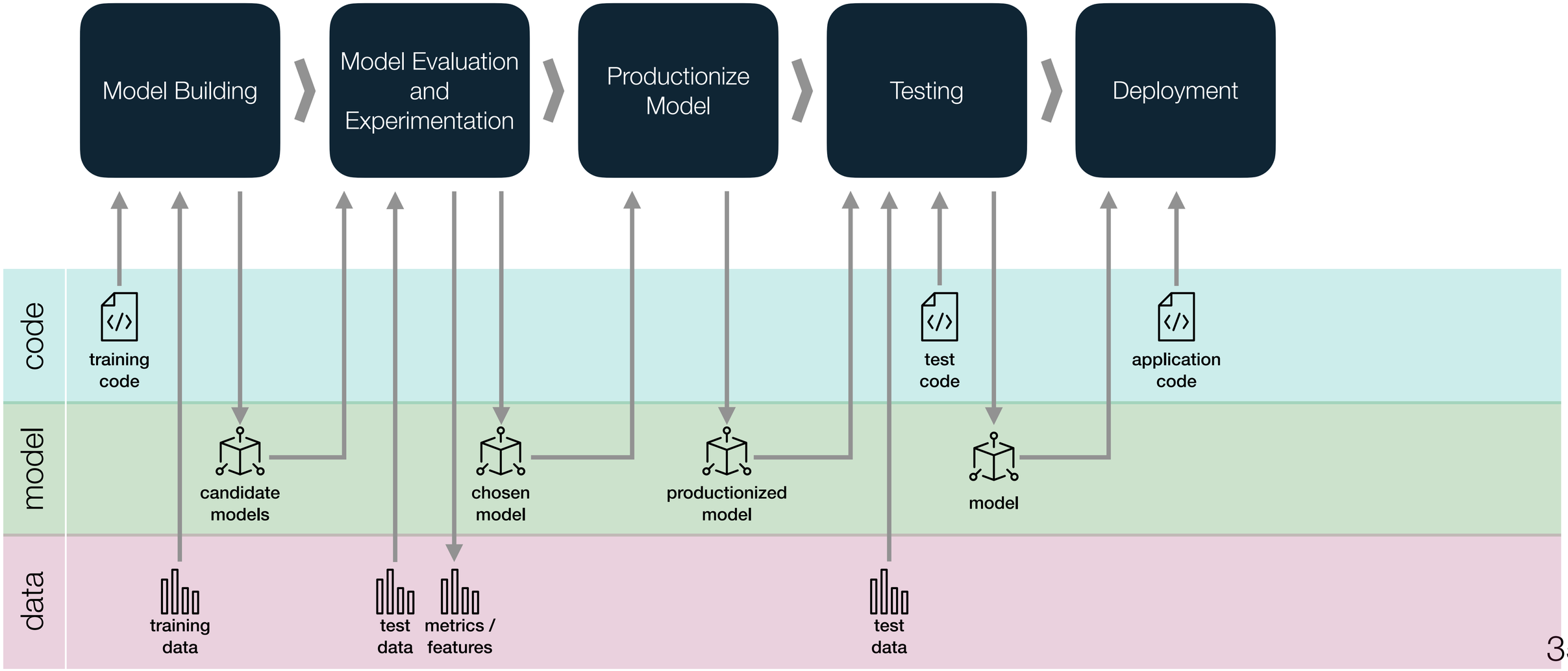
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



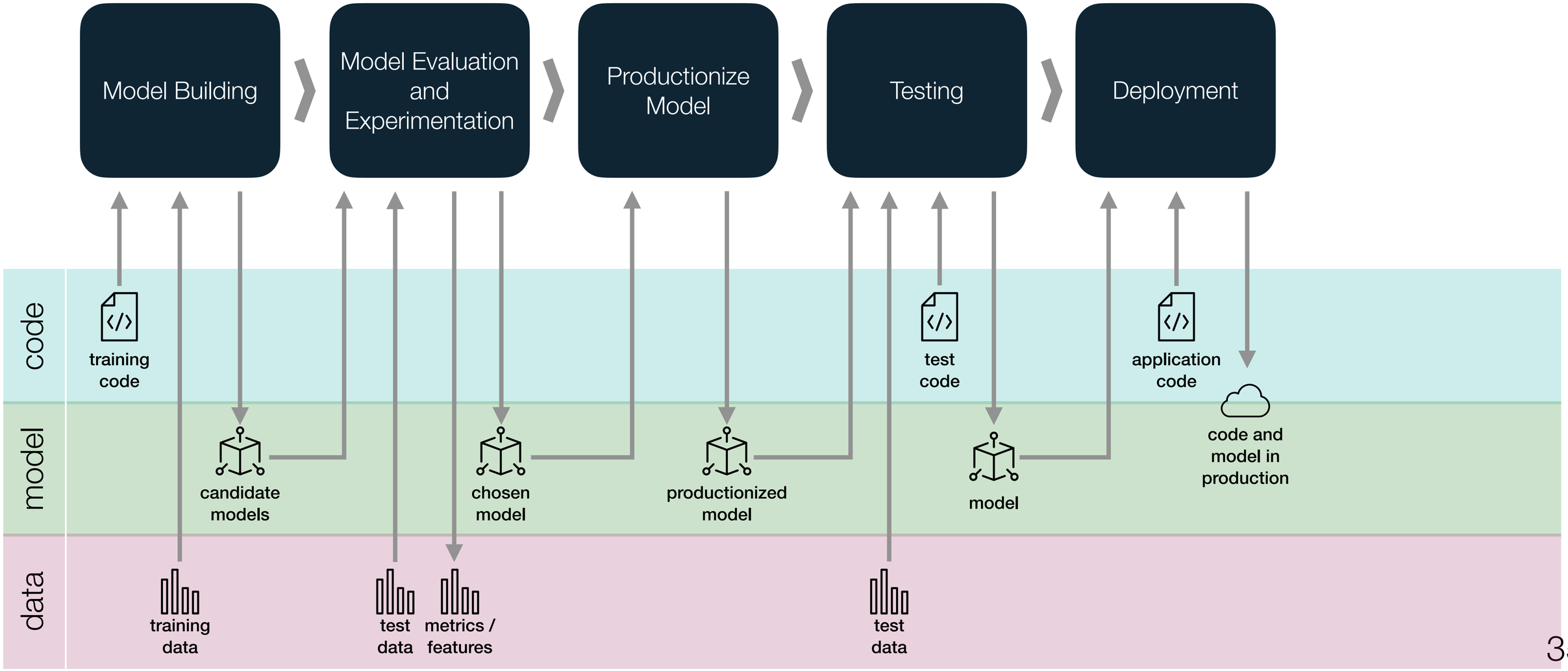
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



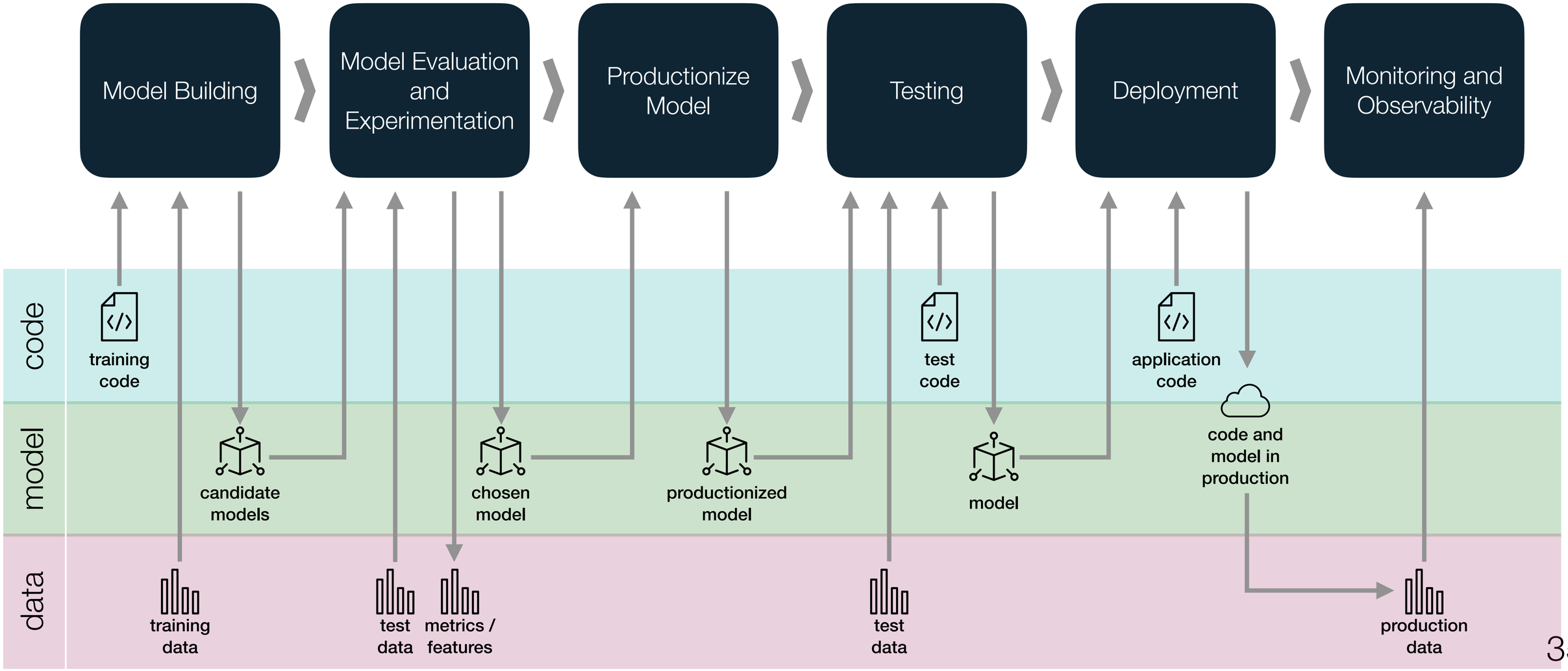
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



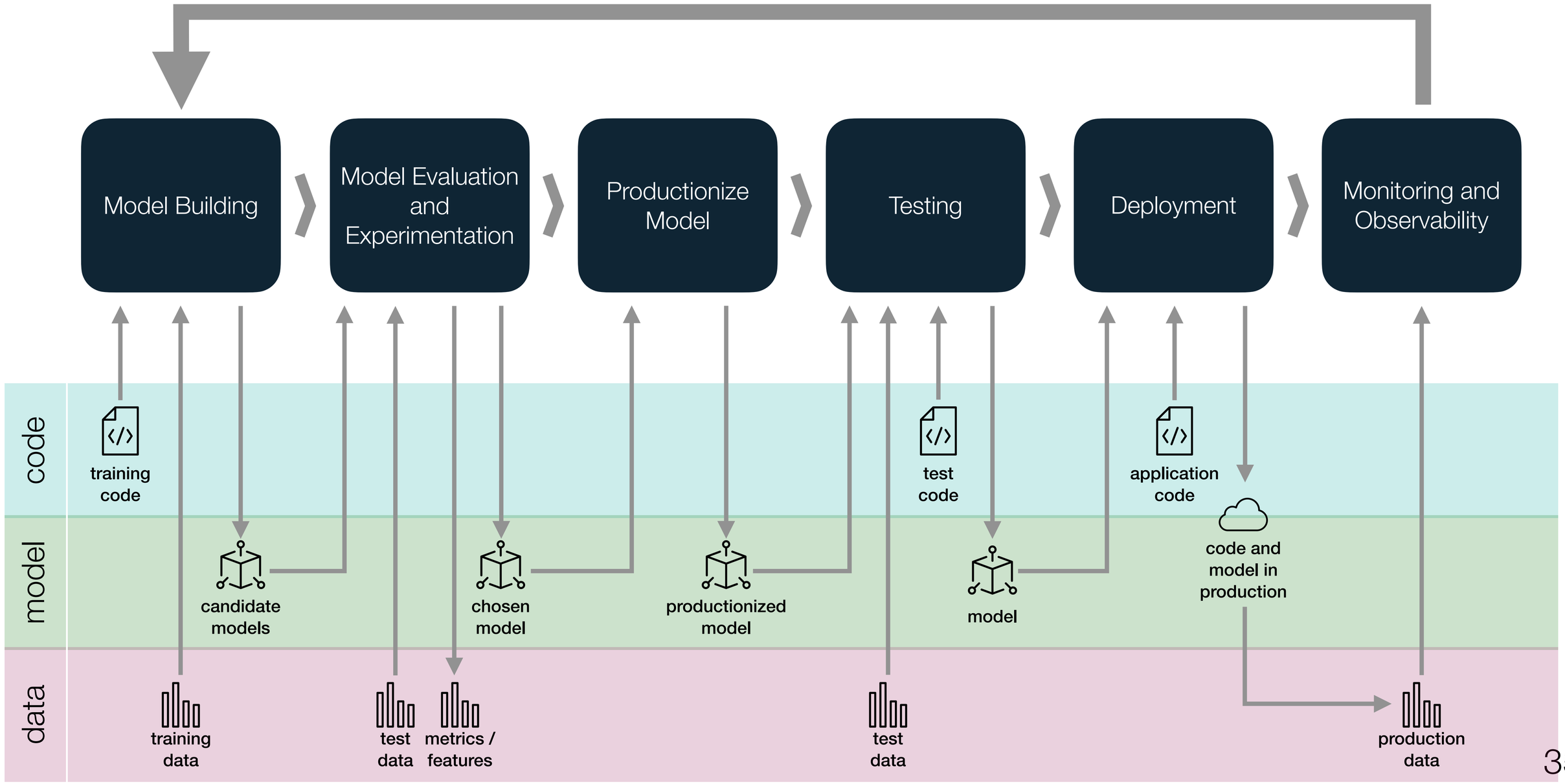
https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



https://martinfowler.com/articles/cd4ml.html

ML Pipline in Porduction



https://martinfowler.com/articles/cd4ml.html

More to Consider

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