Deploying Machine Learning Solutions

UNDERSTANDING FACTORS THAT IMPACT DEPLOYED MODELS

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Recent challenge posed by underperformance of deployed models

Several possible causes

Overfitting, training-serving skew

Concept drift, concerted adversaries

Need for monitoring and retraining of deployed models

Model development does not end with deployment

Prerequisites and Course Outline

Prerequisites

Basic Python programming

Basic knowledge of machine learning

Basic understanding of cloud computing

Course Outline

Understanding factors that impact model deployment

Deploying to Flask

Deploying to serverless cloud environments

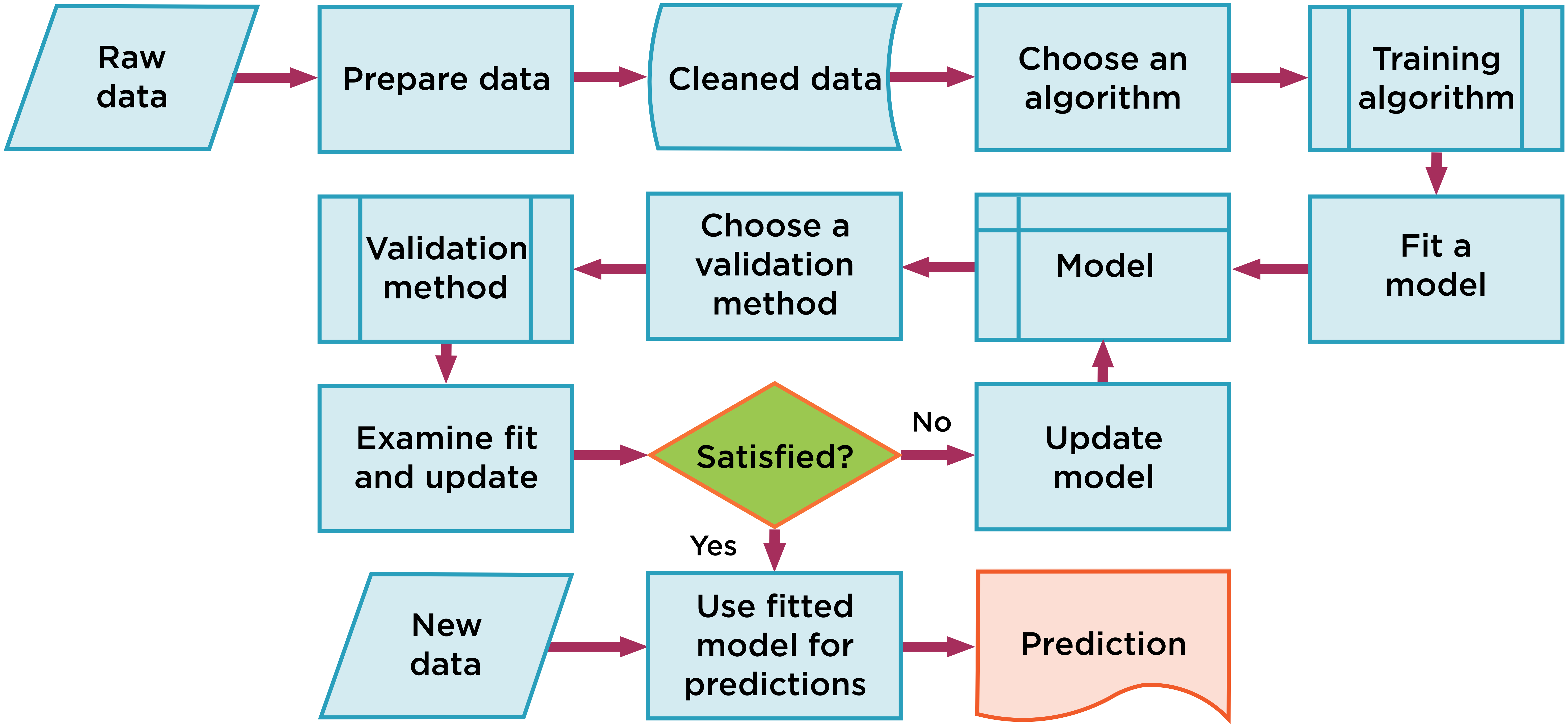
Deploying to Google Cloud AI Platform

Deploying to AWS SageMaker

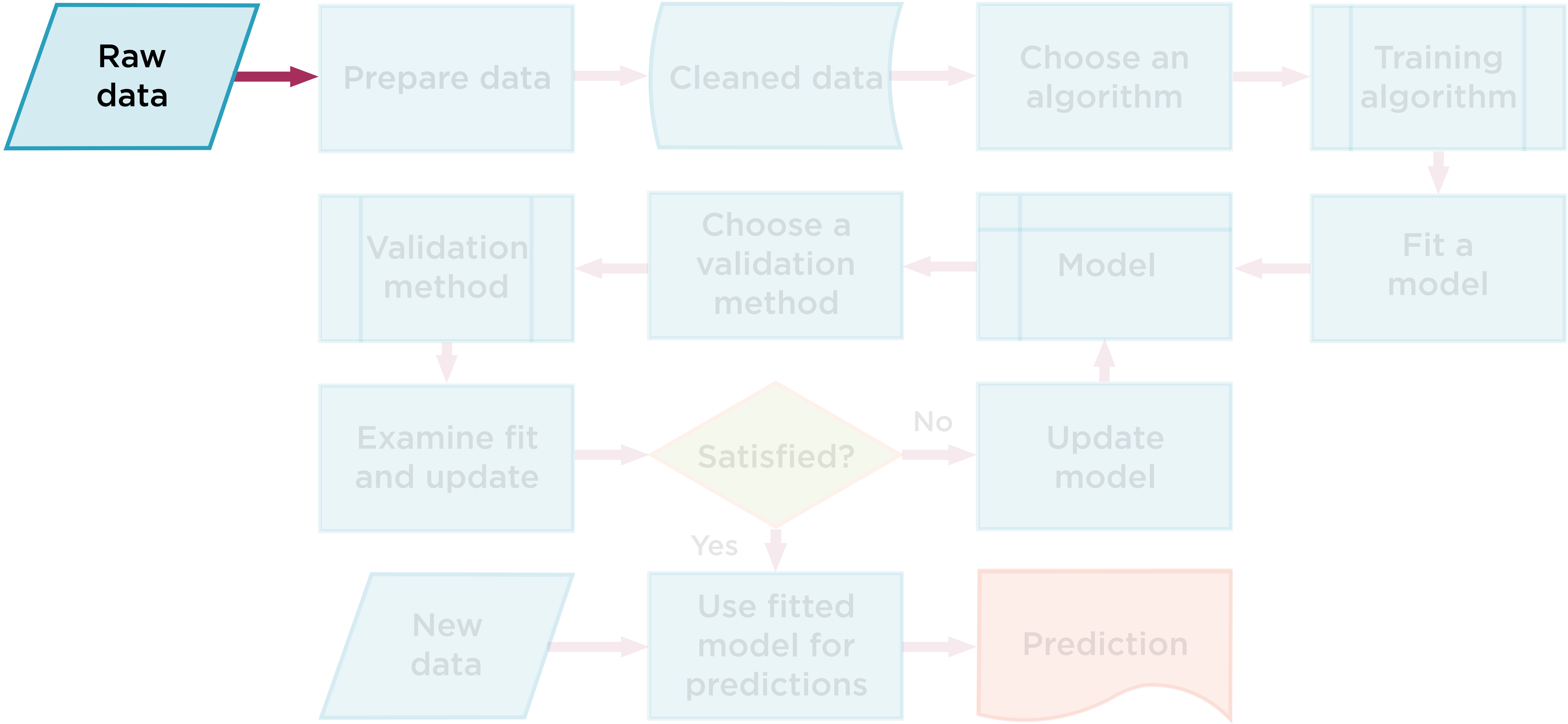
The Classic Machine Learning Workflow

Models are performing **worse in production than in development**, and the classic ML workflow is proving inadequate

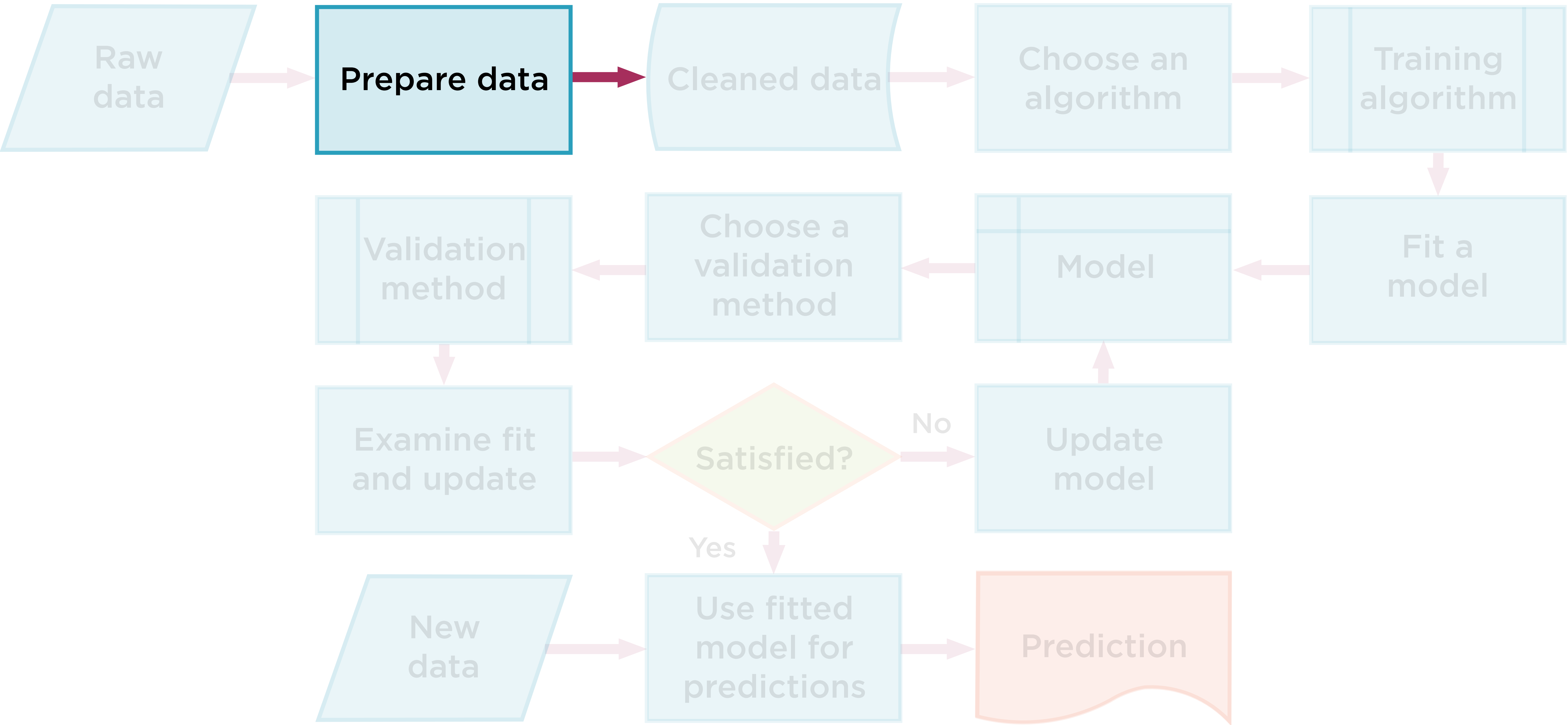
Basic Machine Learning Workflow



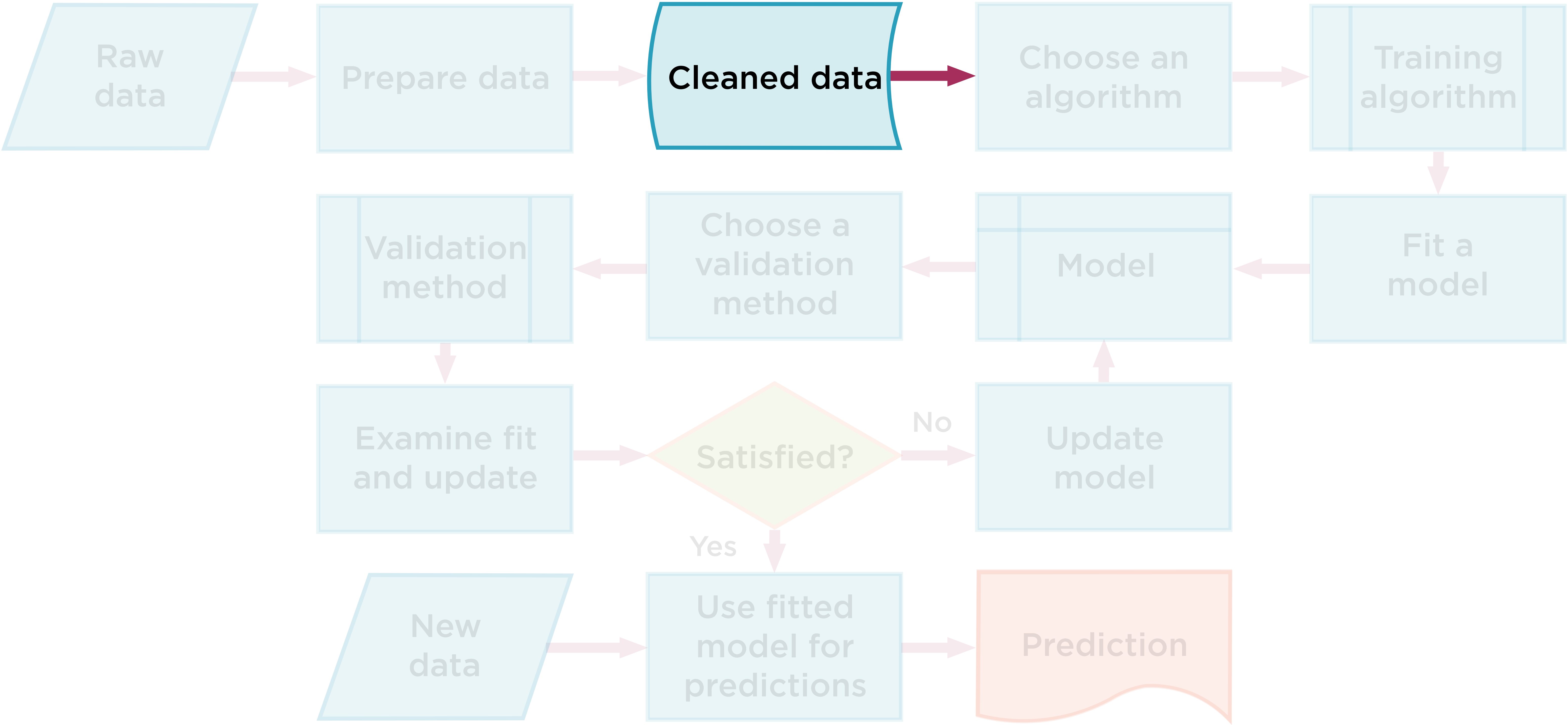
What Data Do You Have to Work With?



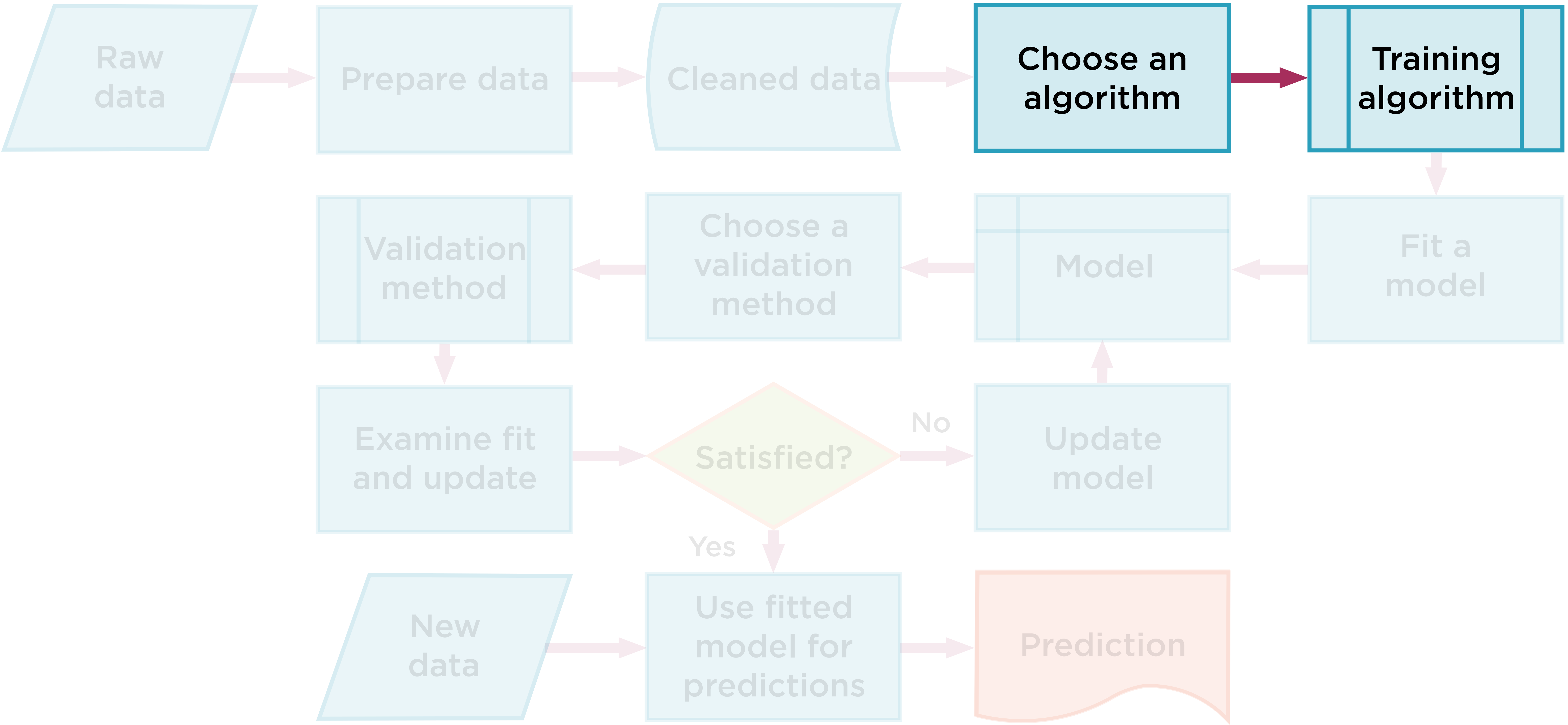
Load and Store Data



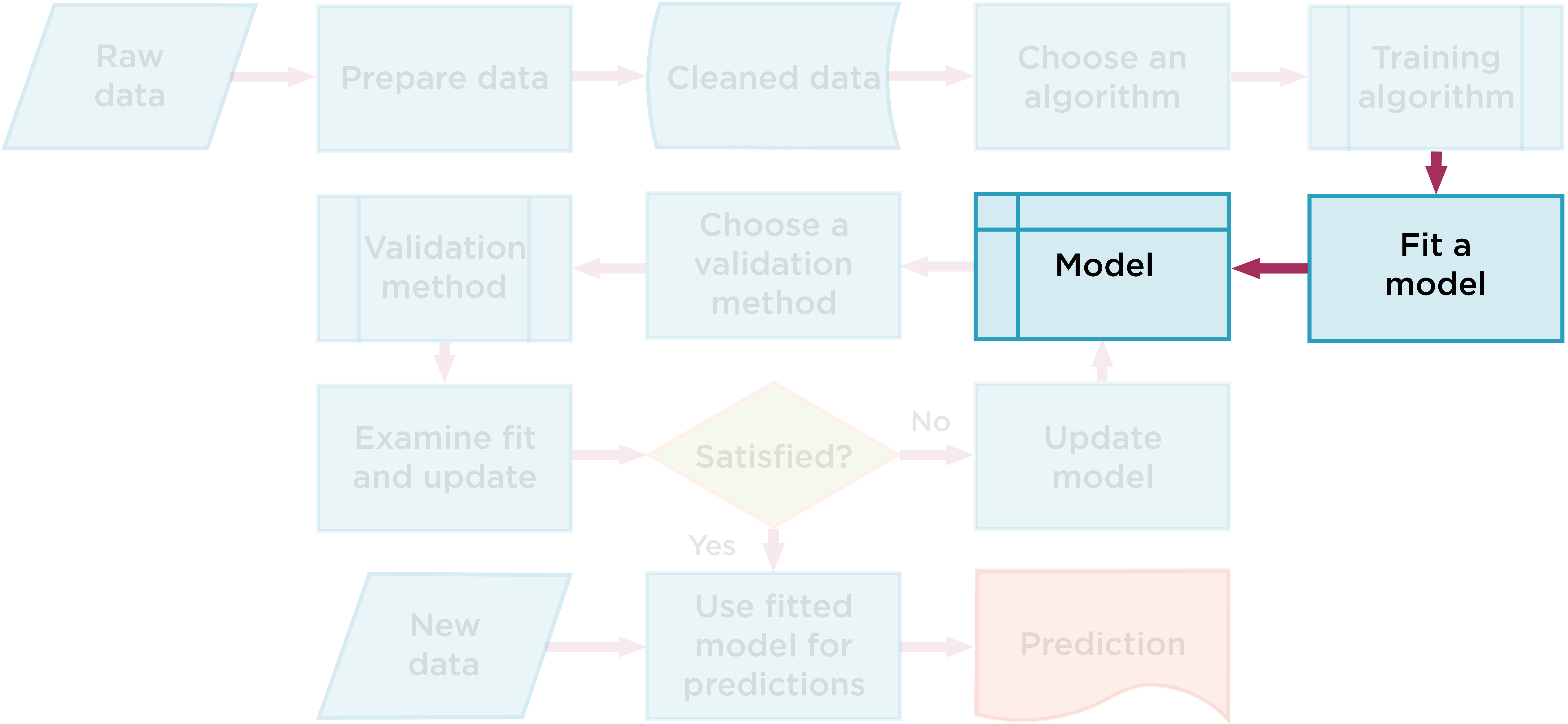
Data Preprocessing



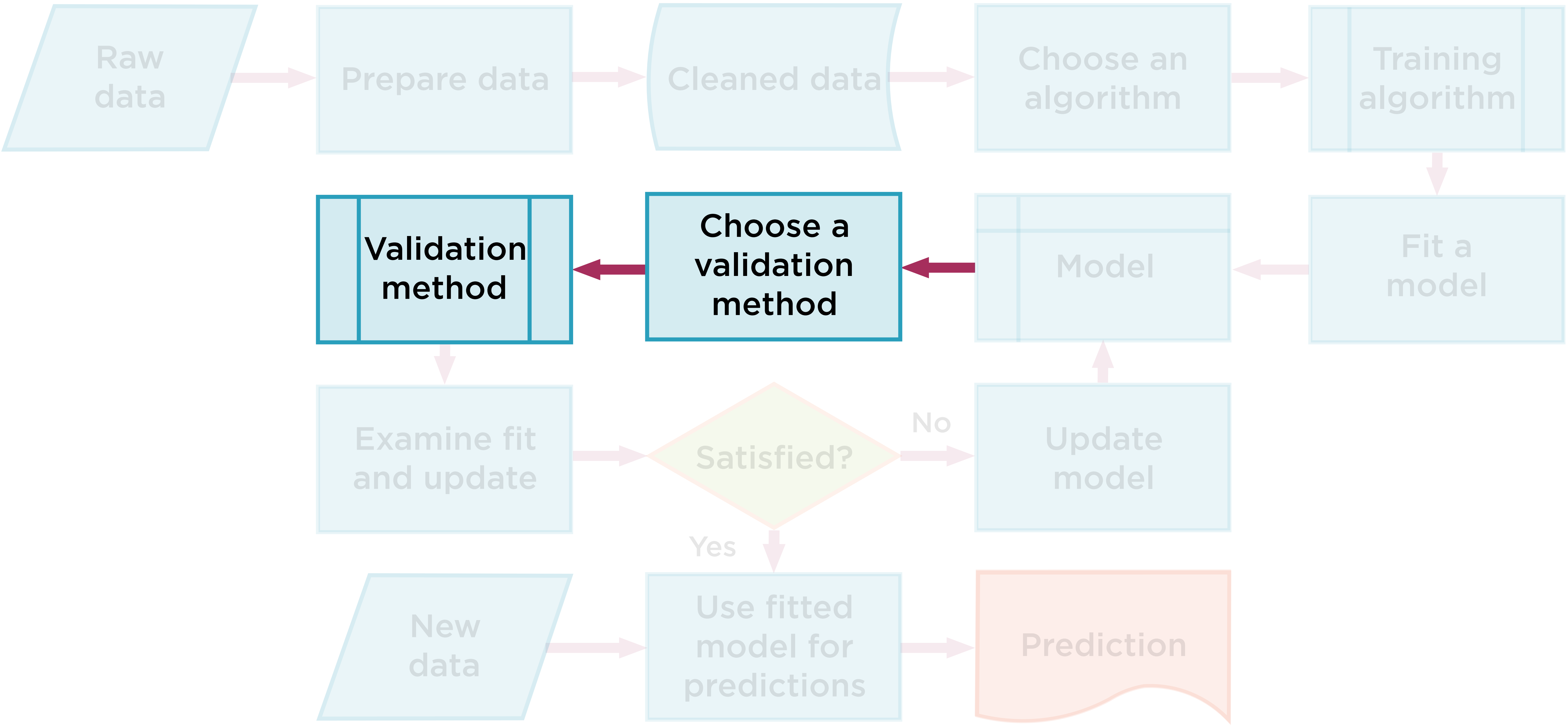
Decision Trees, Support Vector Machines?



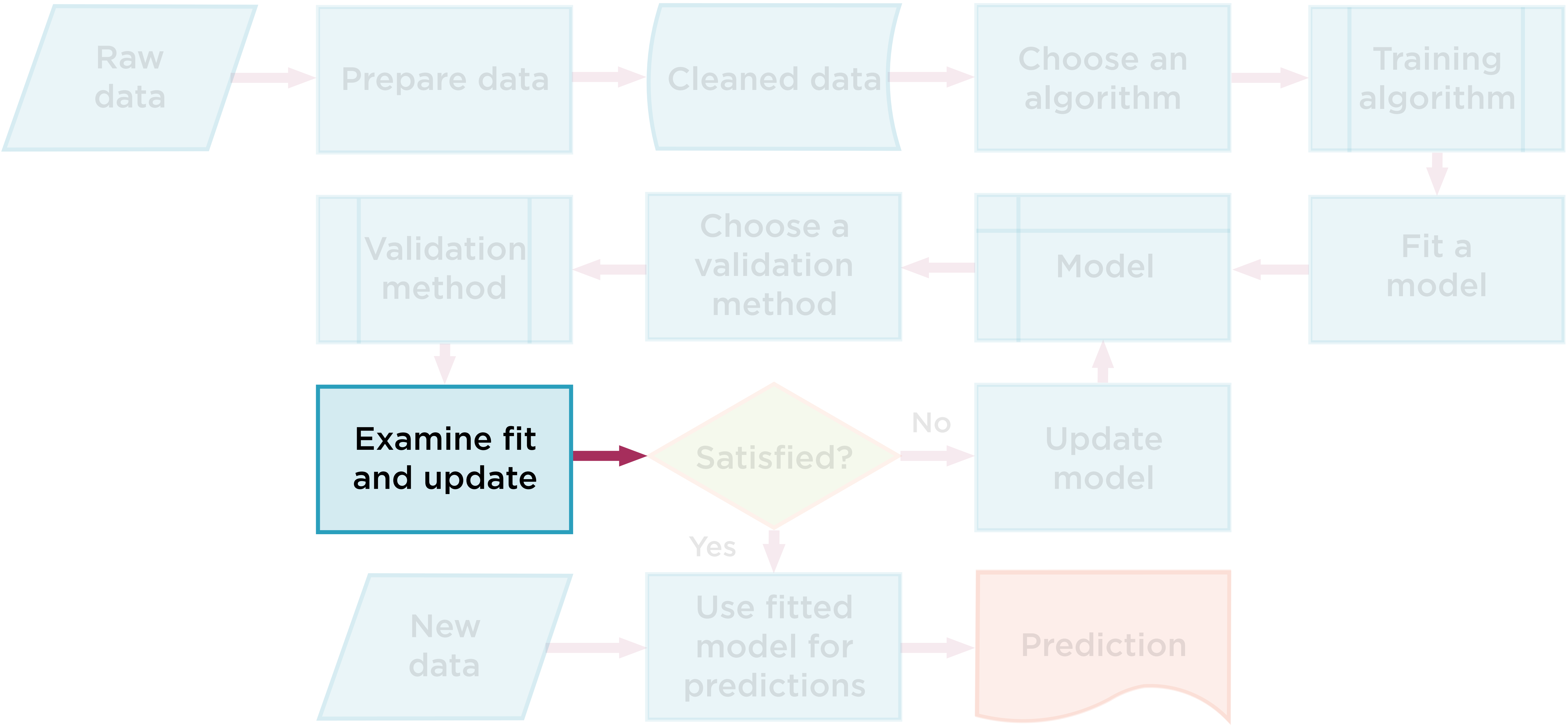
Training to Find Model Parameters



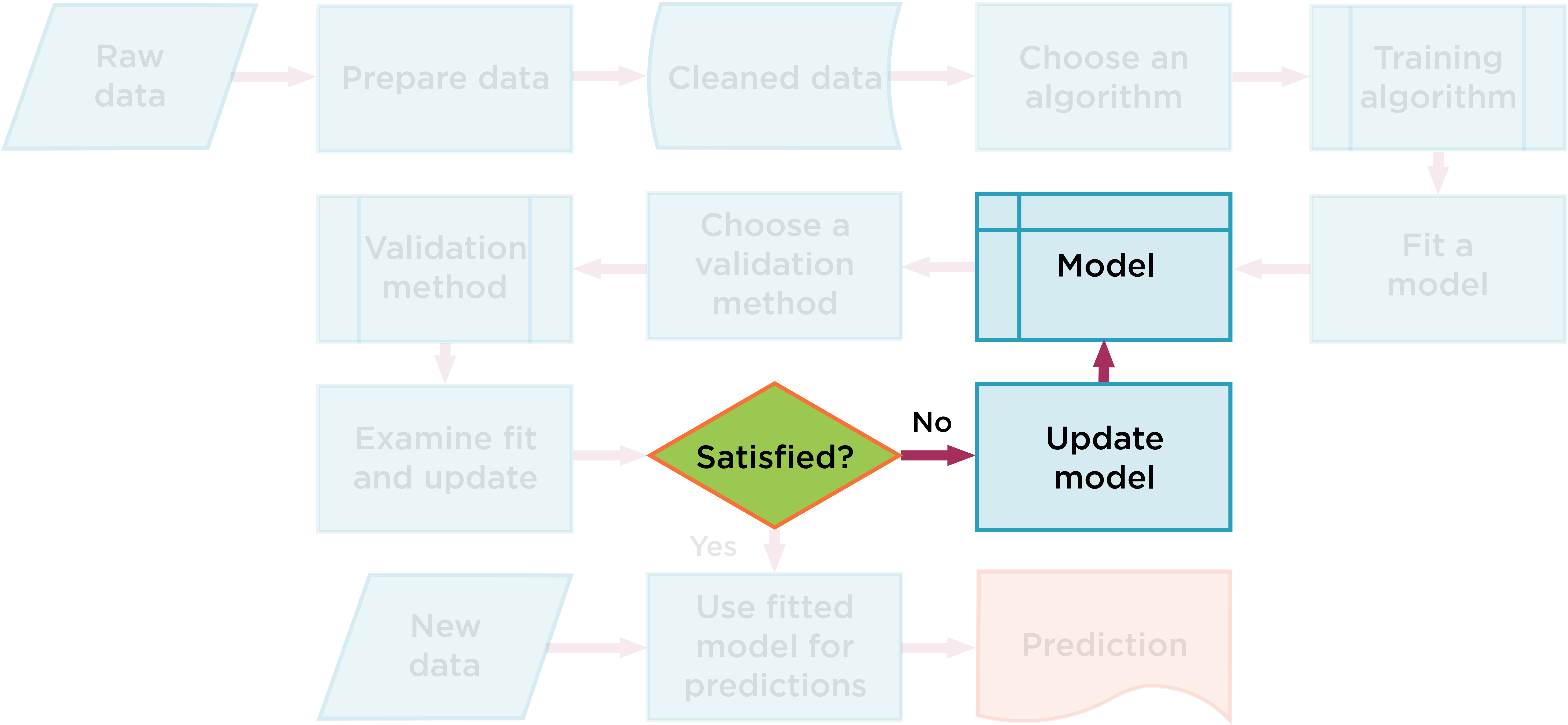
Evaluate the Model



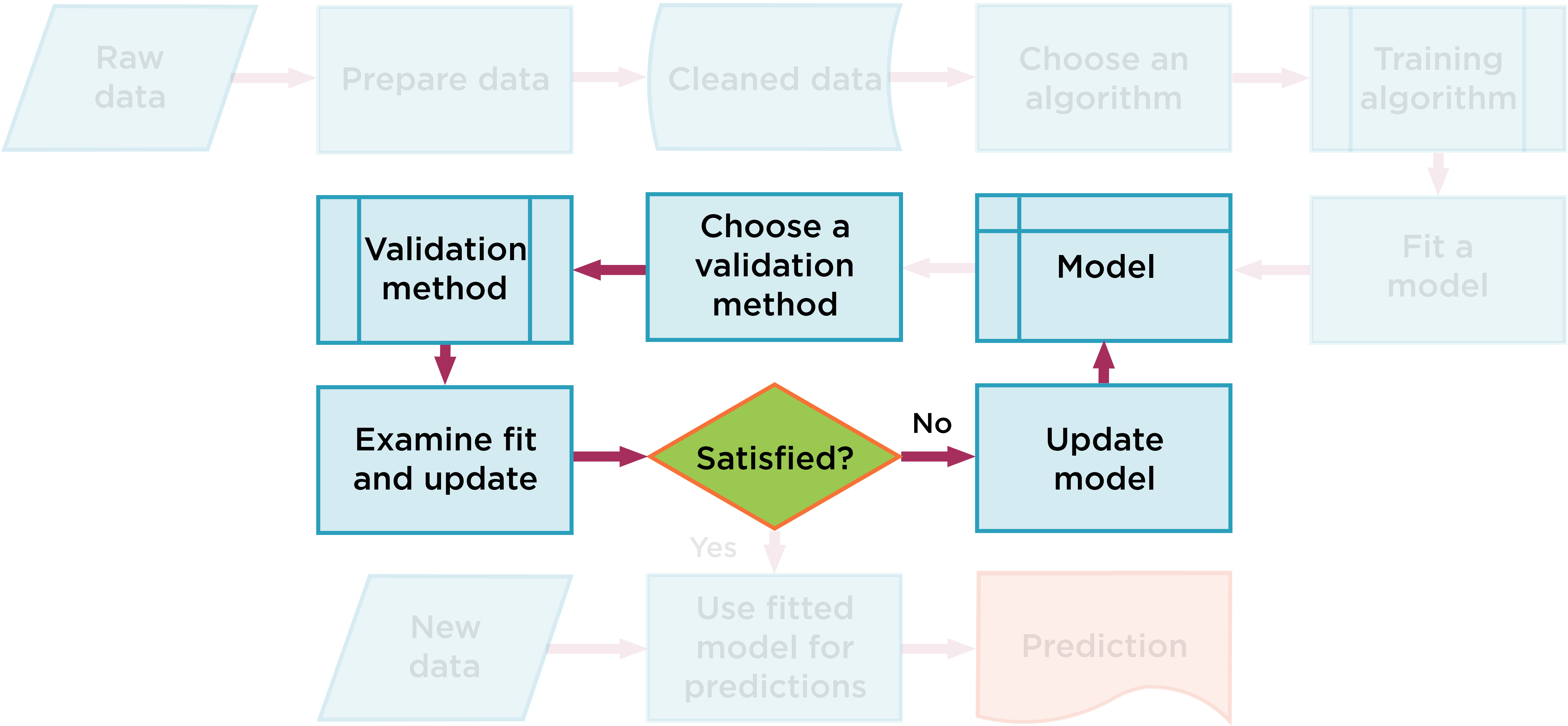
Score the Model



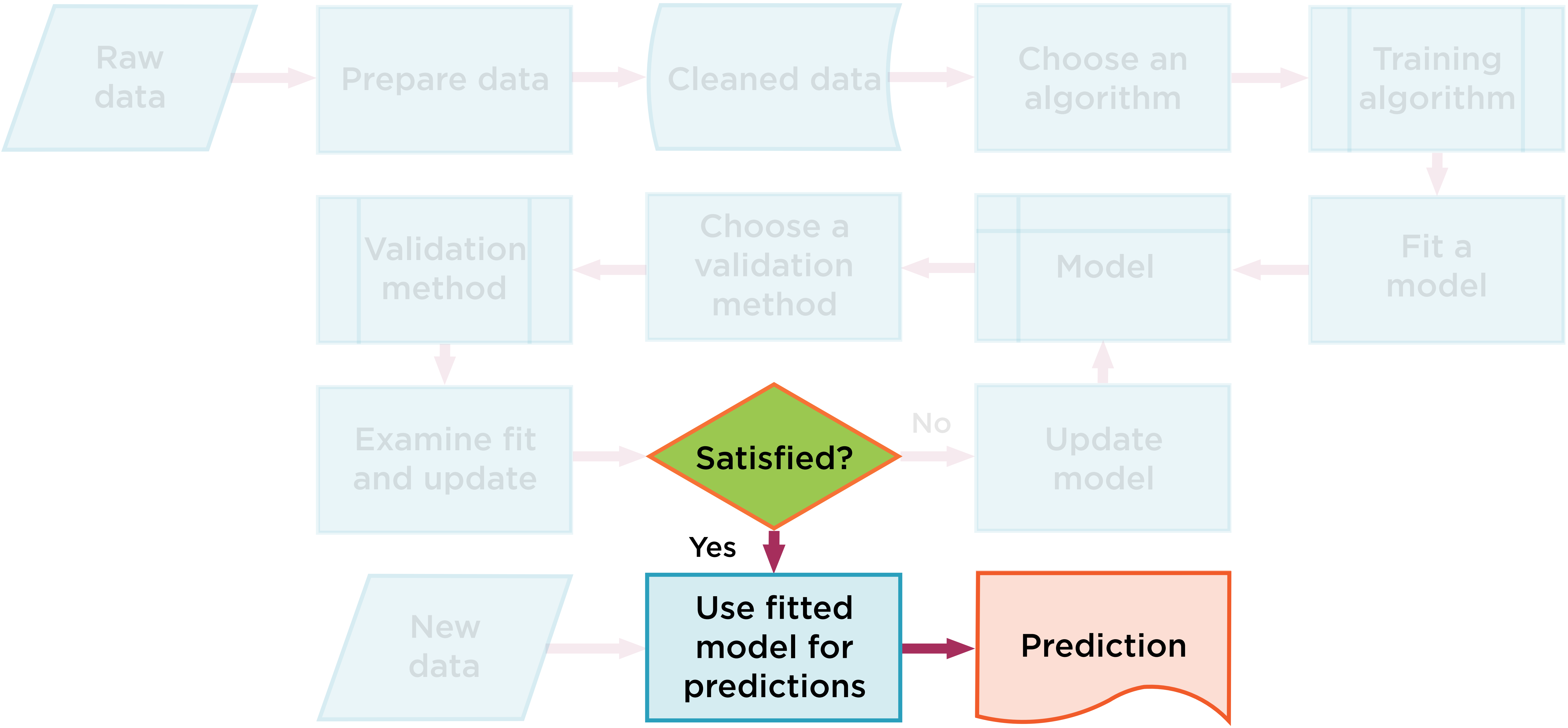
Different Algorithm, More Data, More Training?



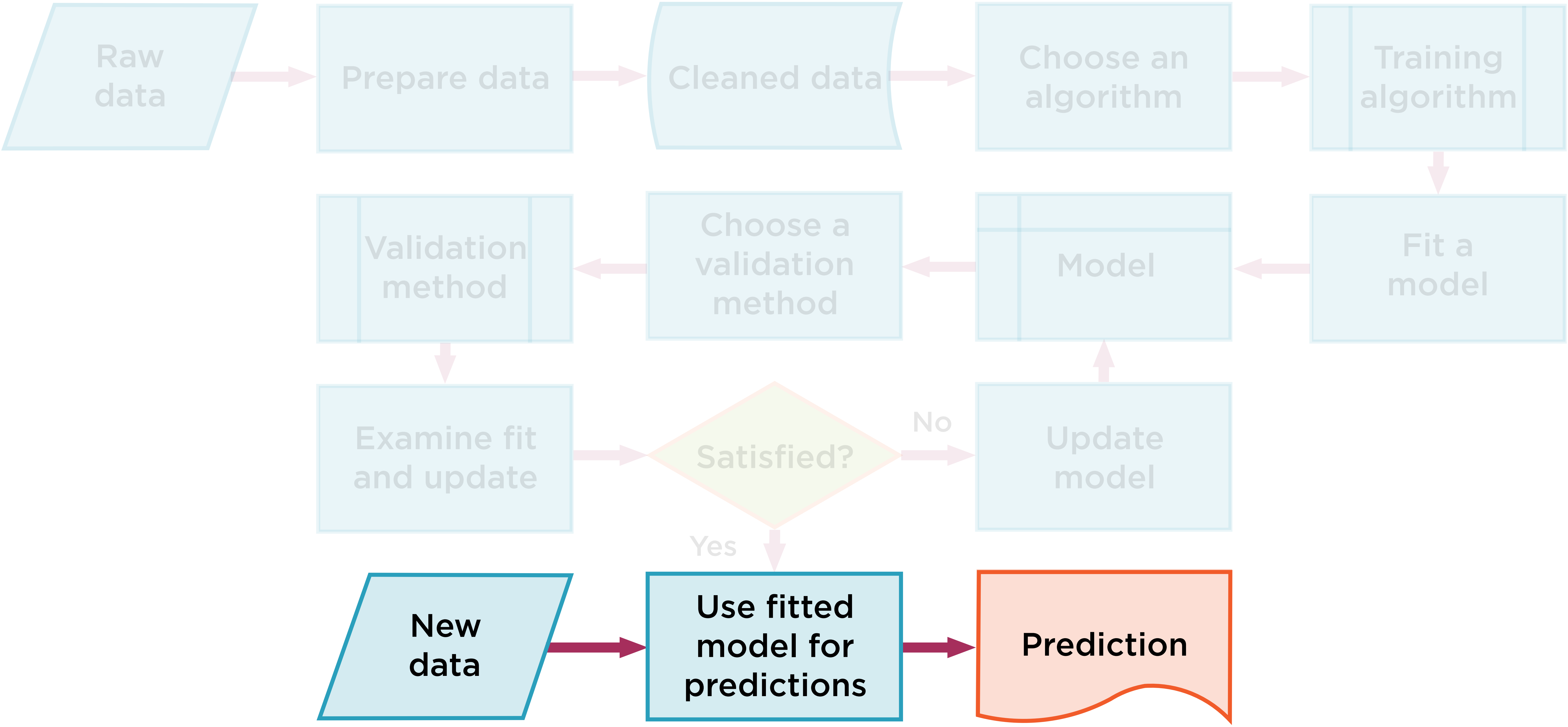
Iterate Till Model Finalized



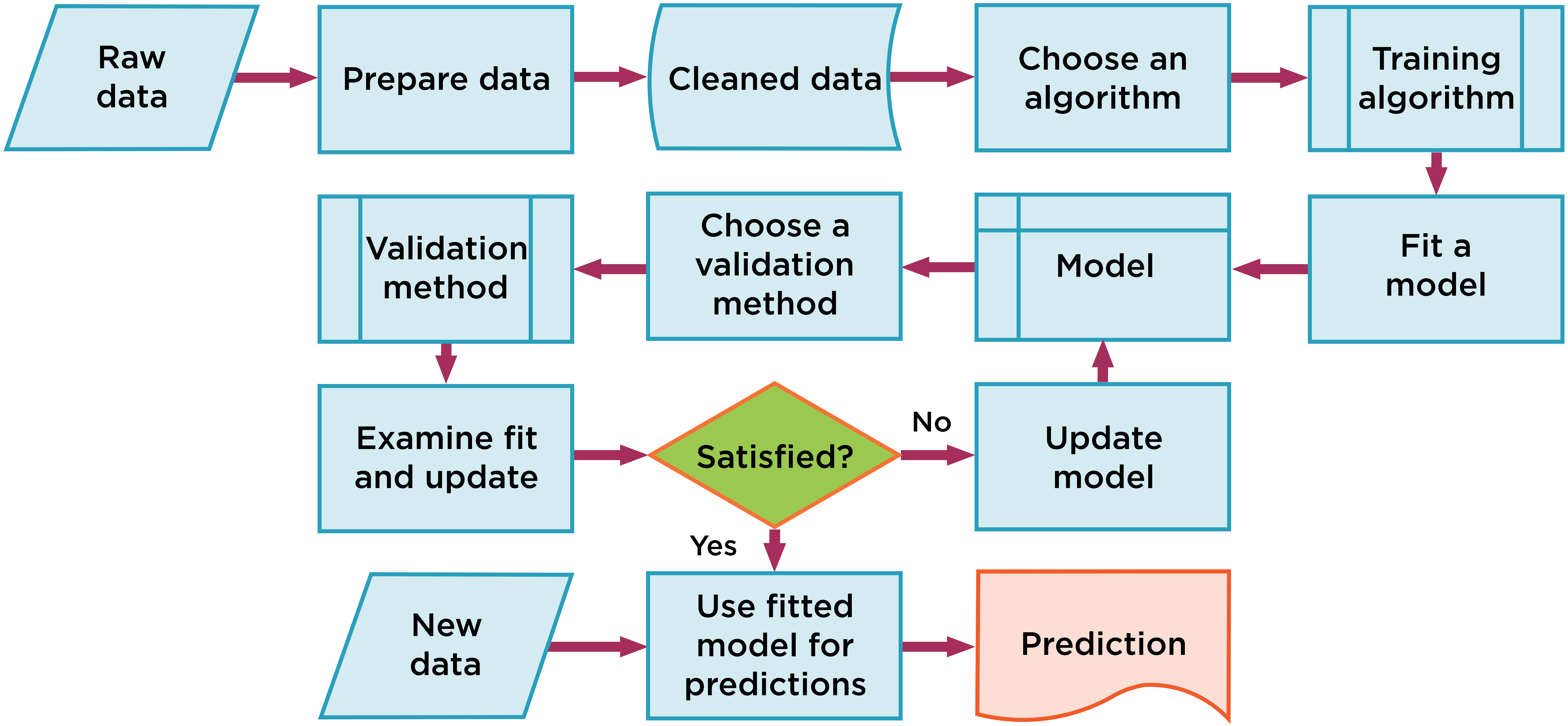
Model Used for Predictions



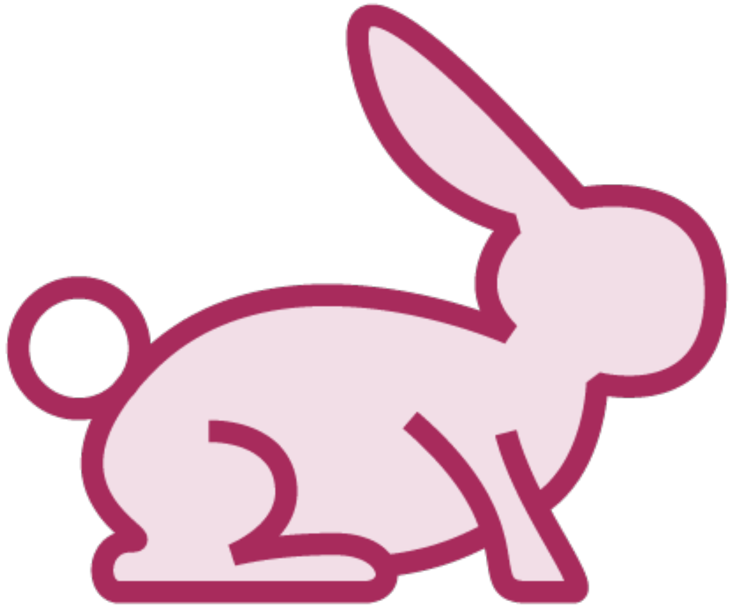
Retrained Using New Data



Basic Machine Learning Workflow

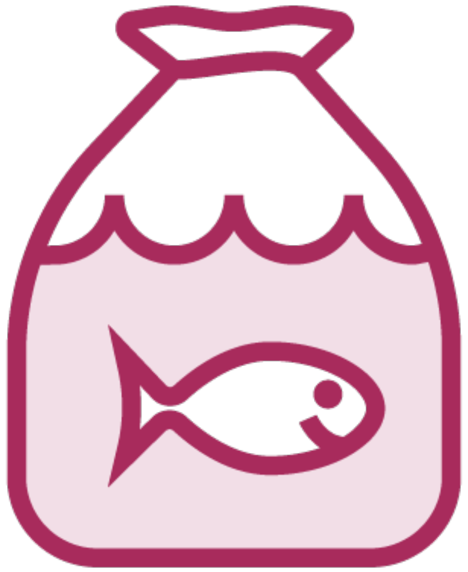


Whales: Fish or Mammals?



Mammals

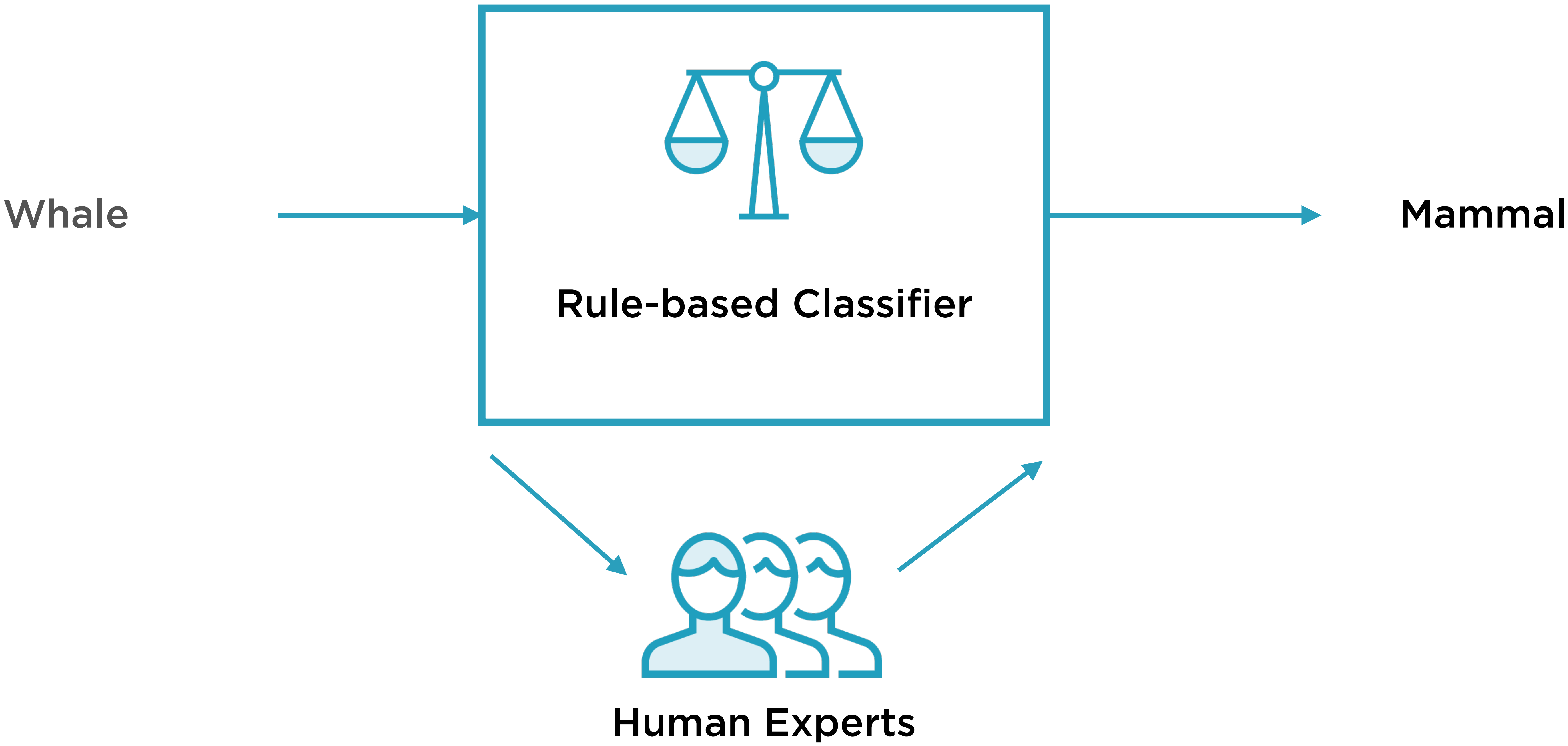
Members of the infraorder *Cetacea*



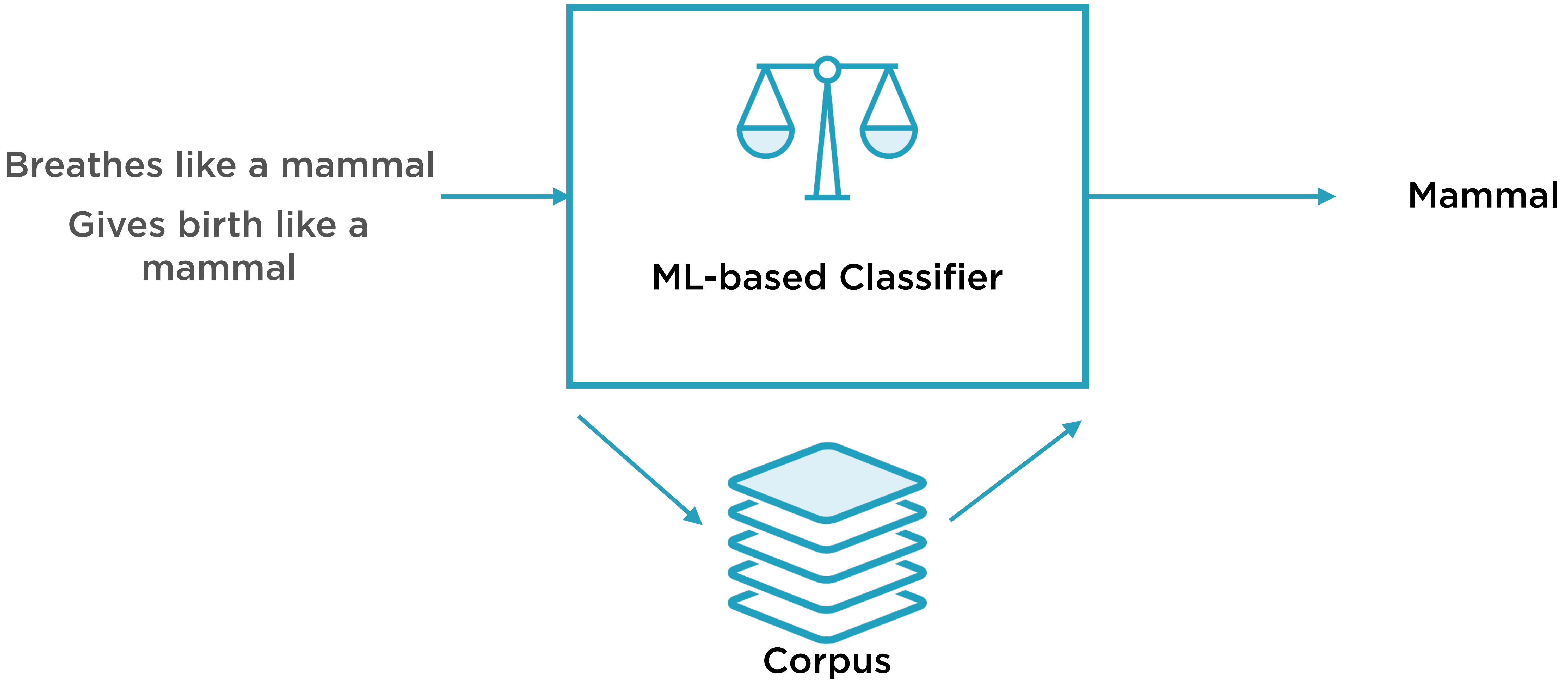
Fish

Look like fish, swim like fish, move with fish

Rule-based Binary Classifier



ML-based Binary Classifier

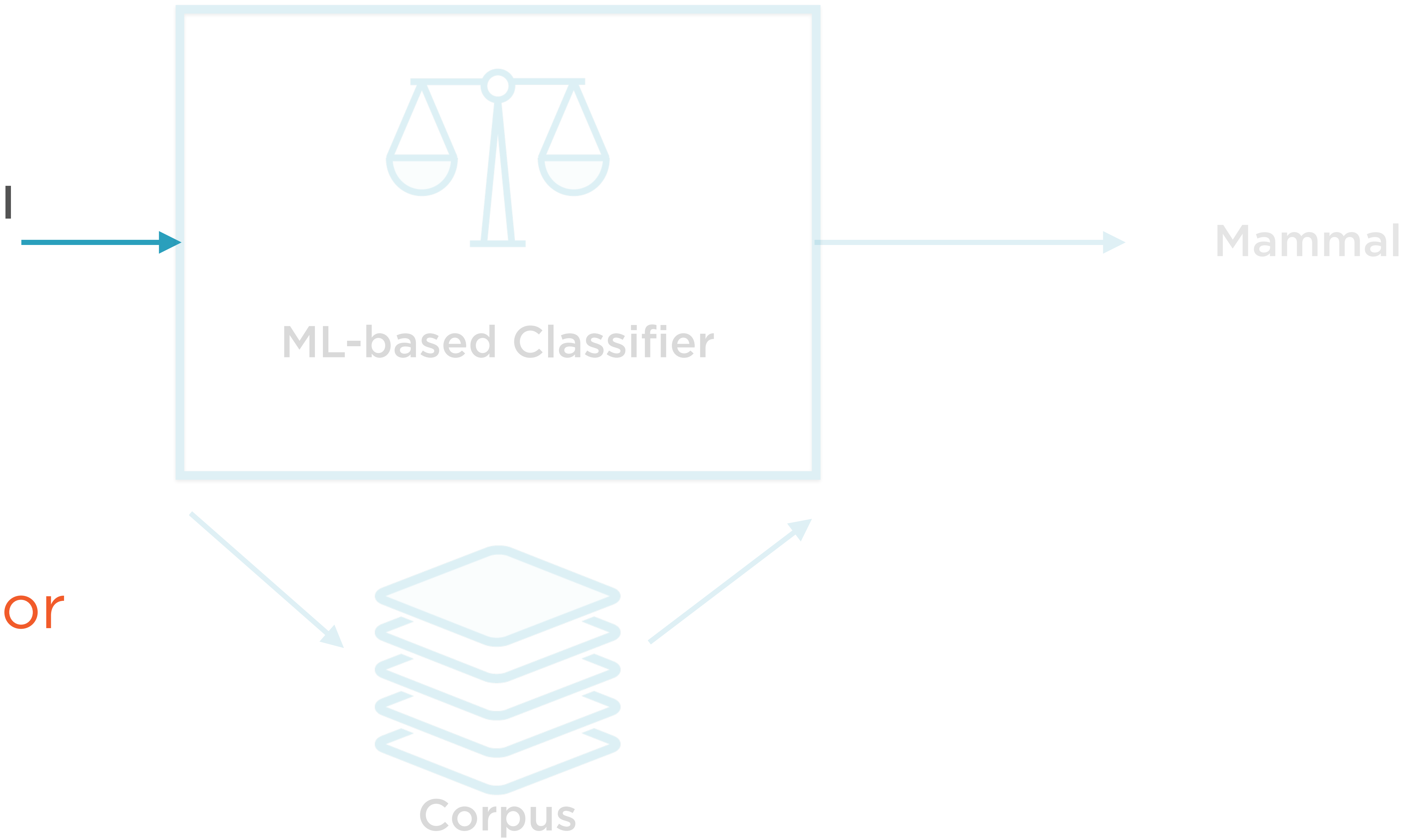


ML-based Binary Classifier

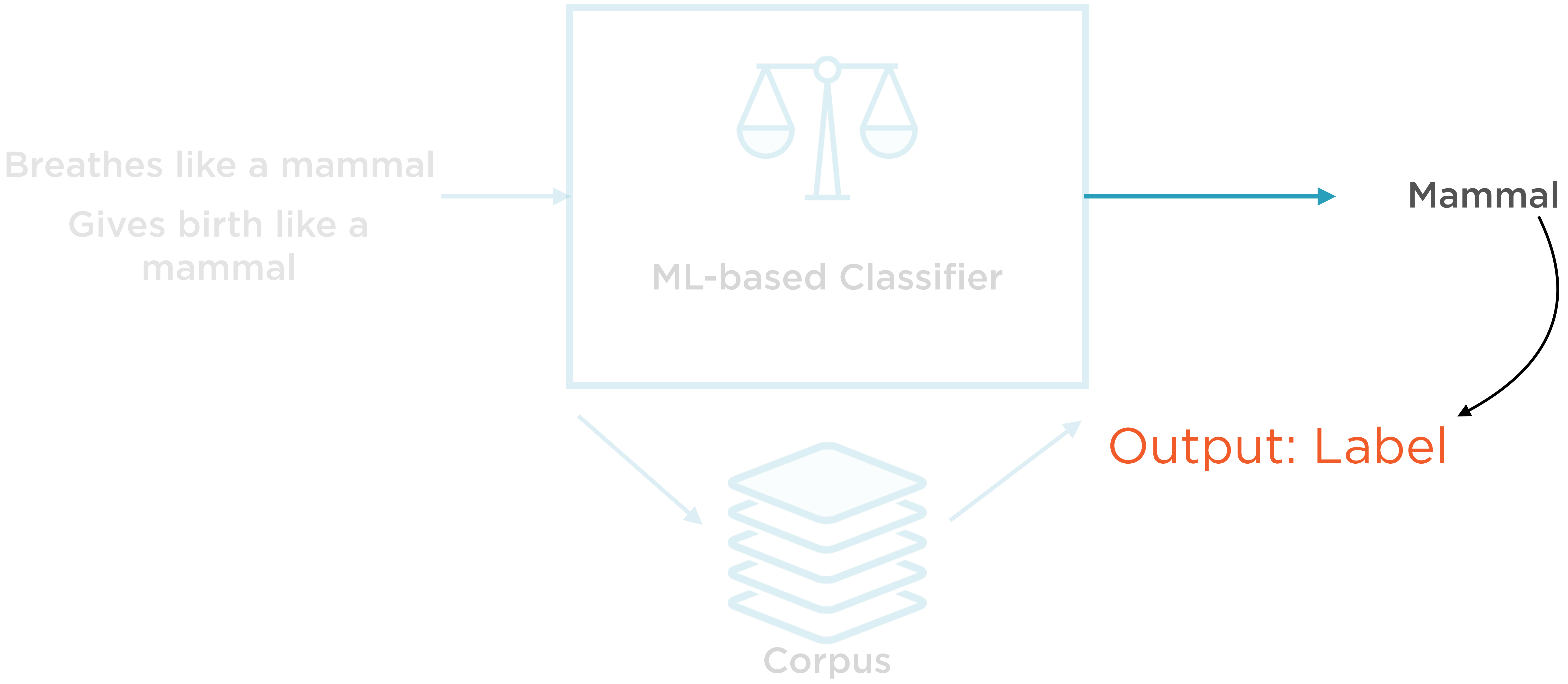
Breathes like a mammal

Gives birth like a mammal

Input: Feature Vector



ML-based Binary Classifier



New Realities of Deployed Models

Models degrade in accuracy as soon as they are deployed in the real world

Degrading Models

A model is at its best just before being deployed to production

Rookie assumption: deployed models work as well as they did in testing

Static machine learning models become less useful over time

Software Development

Software which has been around for a while is more robust

More bug fixes, all code paths tested, usability fixes applied

External changes rarely affect regular software

Can have steady, periodic release cycles

Model Development != Software Development

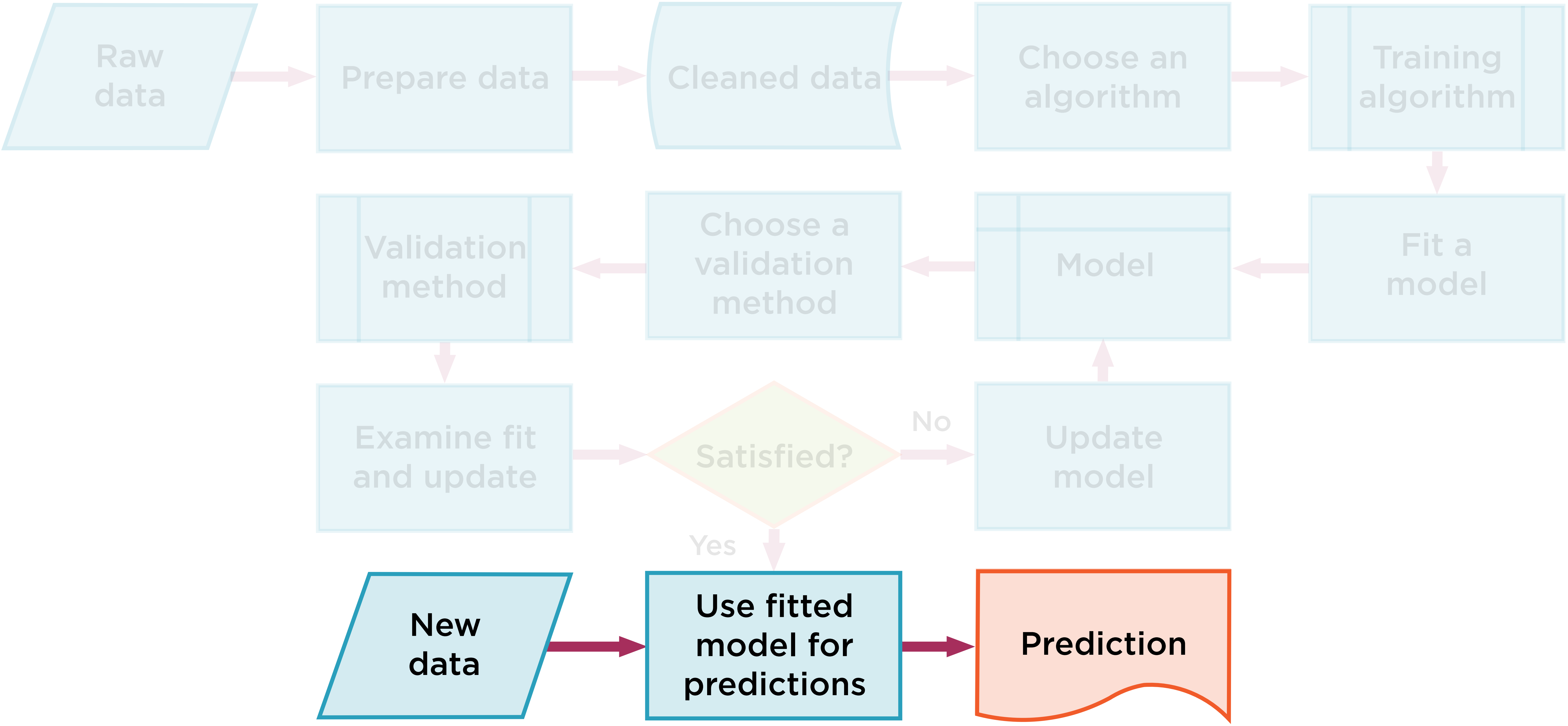
Model development is not exactly the same as software development

A constant stream of new data is needed to keep models working well

Models need to adjust for shifting realities in the real world

Deploying models is just the beginning

Critical Step: Retraining Models



Retraining Models

Based on what the model is used for

Preferences, news, weather, buying behavior, security threats

Constantly need to train models on new data

May need to localize models to take into account geographical differences

Models are performing worse in production than in development, and the **solutions need to be sought in deployment**

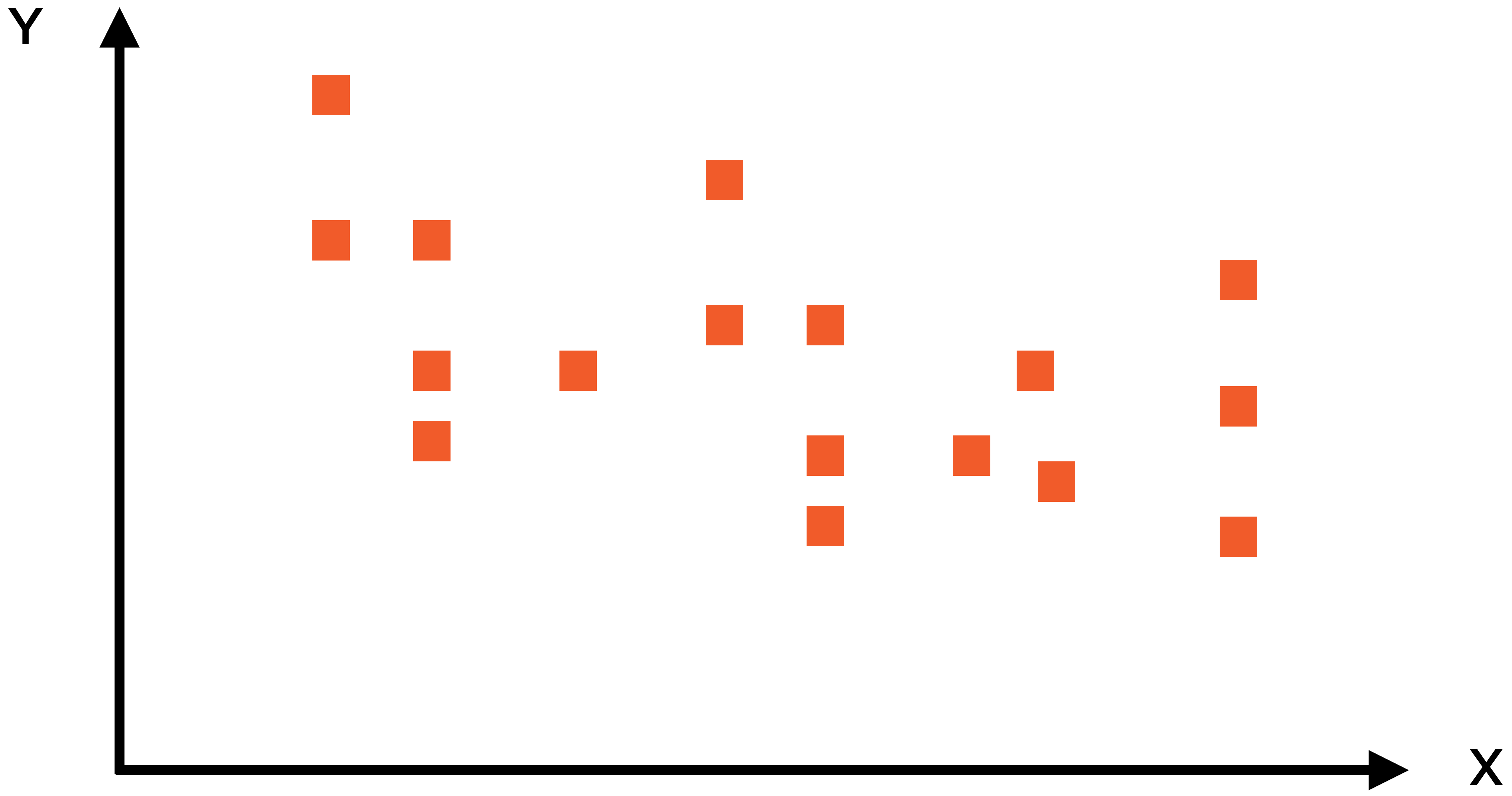
Problems Afflicting AI-based Solutions

Overfitting	Training-serving Skew
Concept Drift	Concerted Adversaries

Problems Afflicting AI-based Solutions

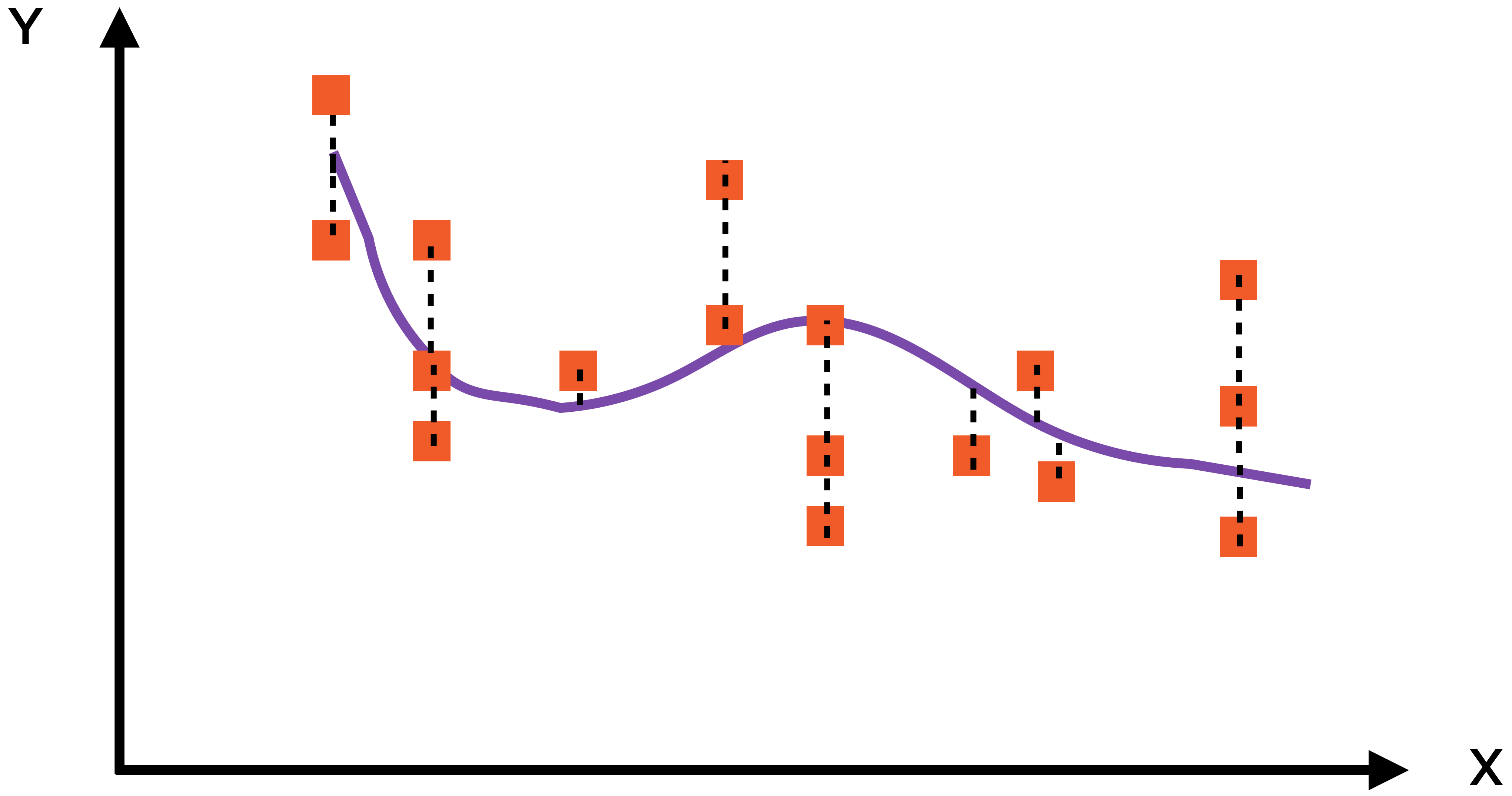
Overfitting	Training-serving Skew
Concept Drift	Concerted Adversaries

Connecting the Dots



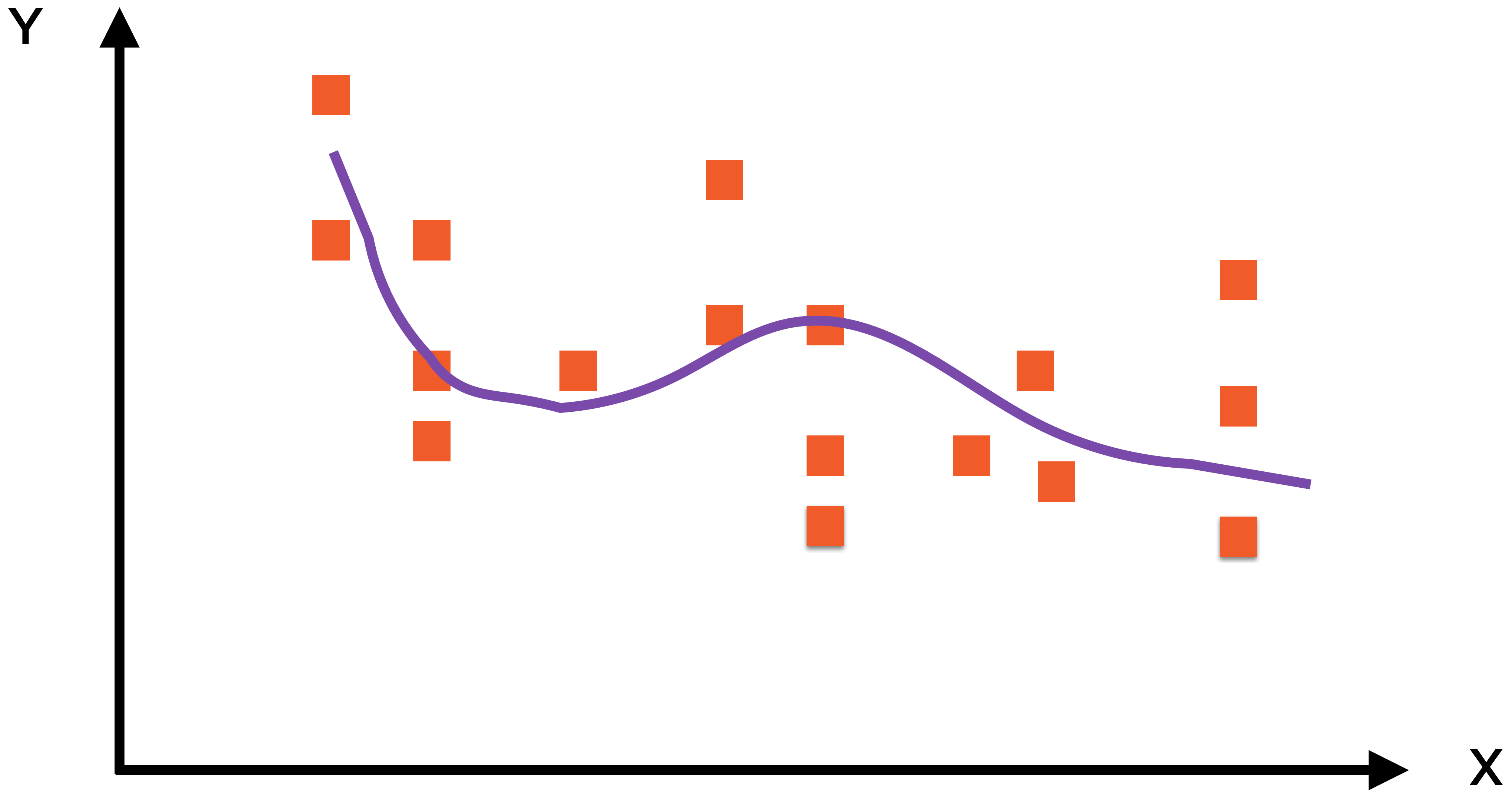
Challenge: Fit the "best" curve through these points

Good Fit?



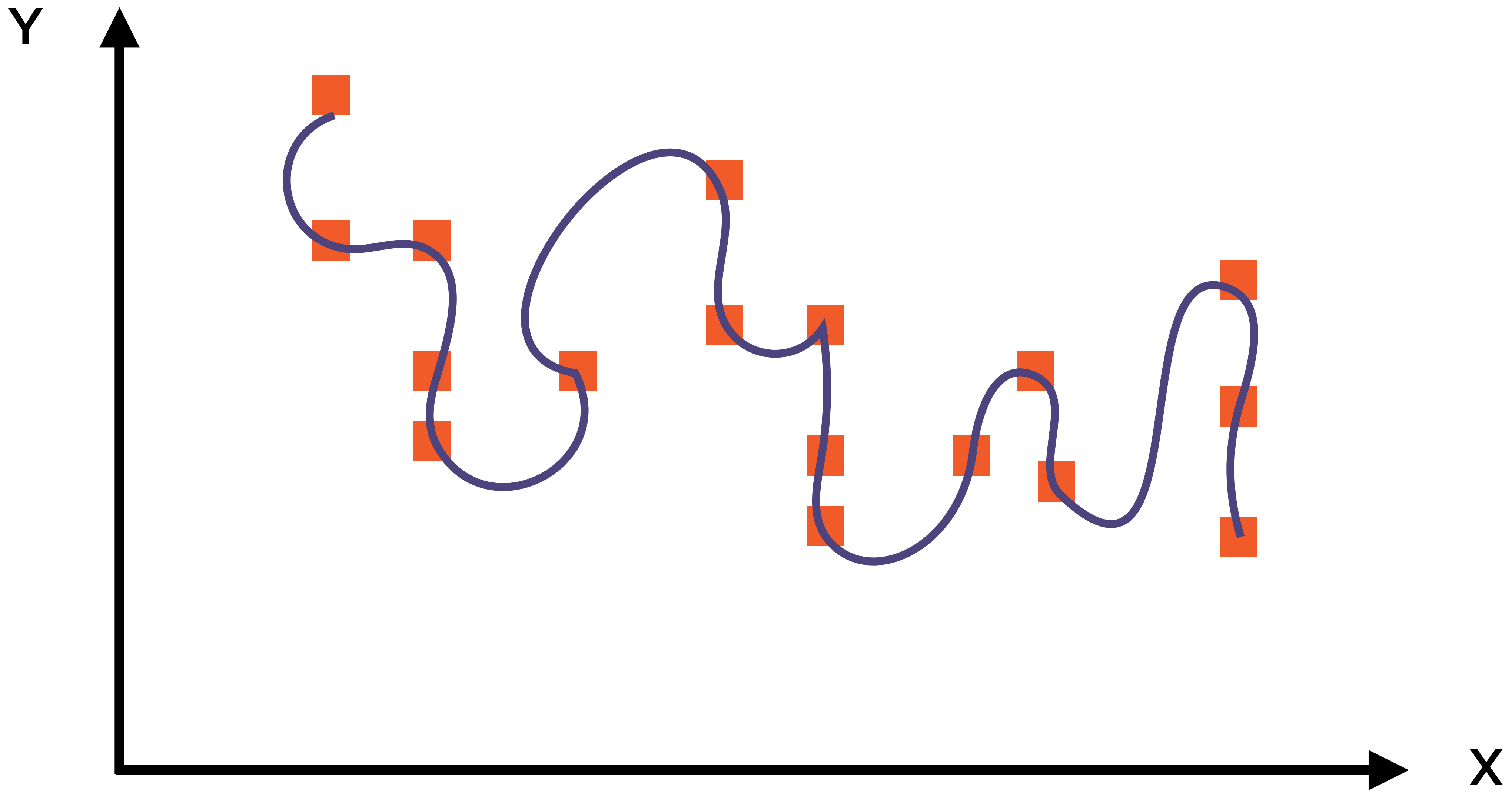
A curve has a “good fit” if the distances of points from the curve are small

Connecting the Dots



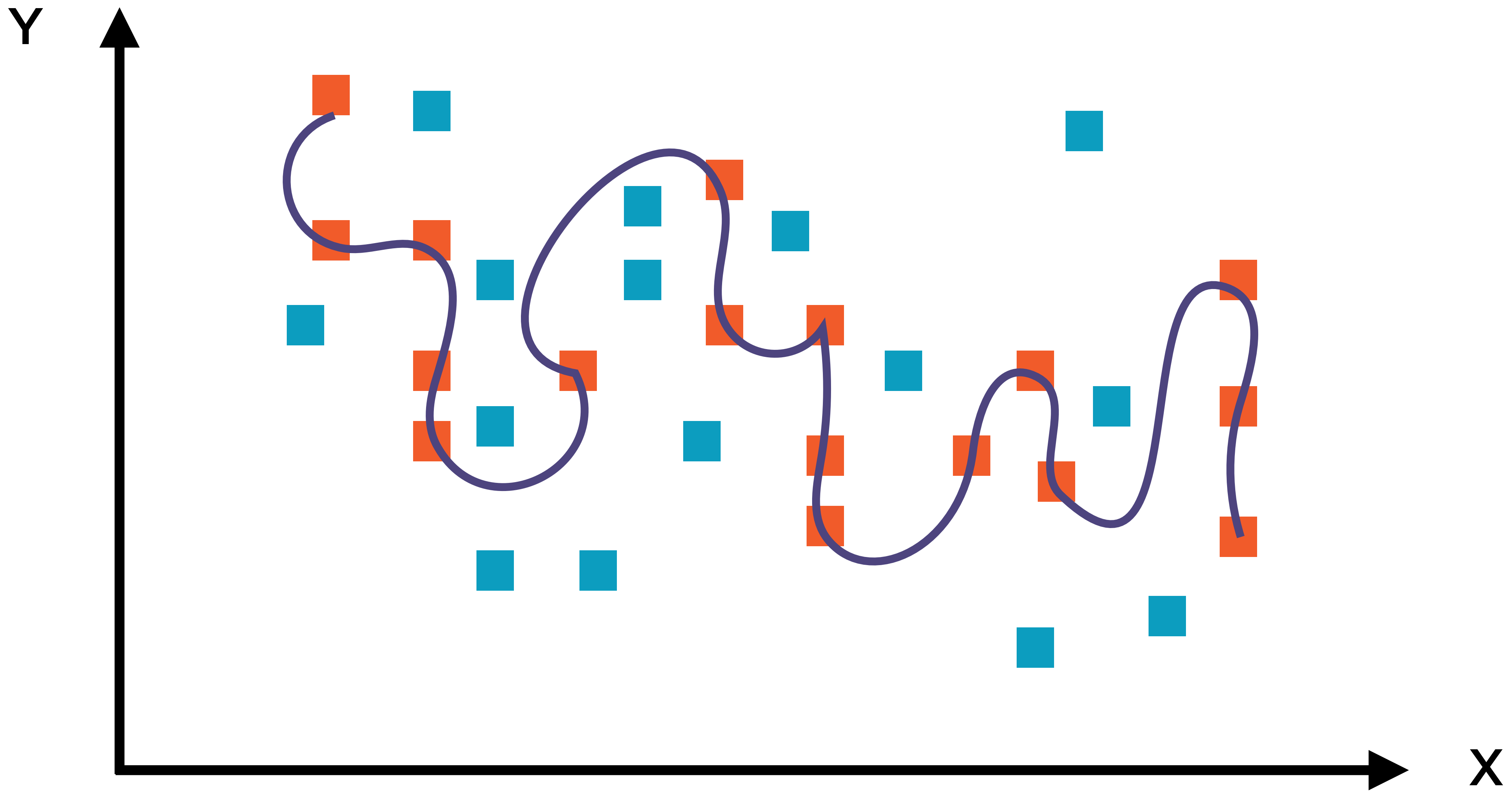
We could draw a pretty complex curve

Connecting the Dots



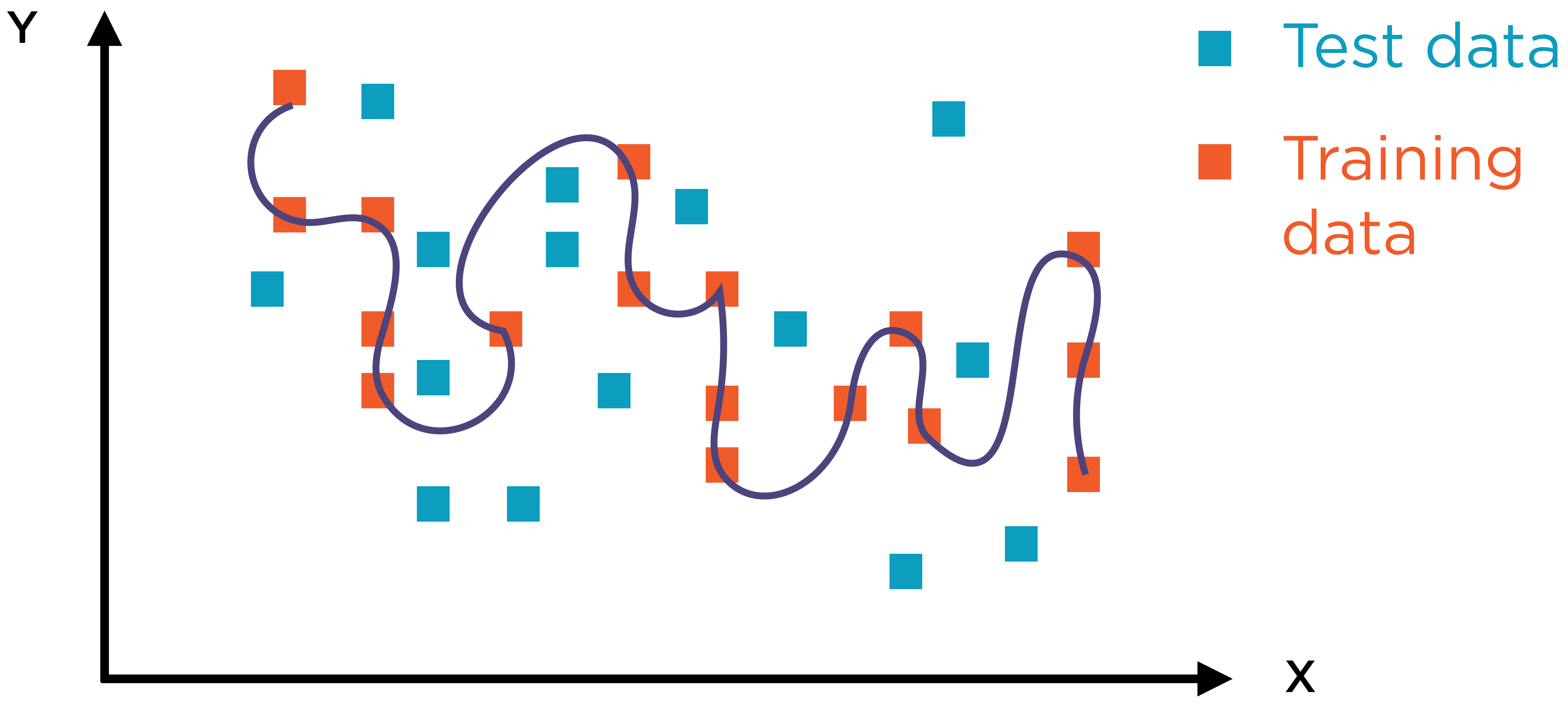
We can even make it pass through every single point

Connecting the Dots



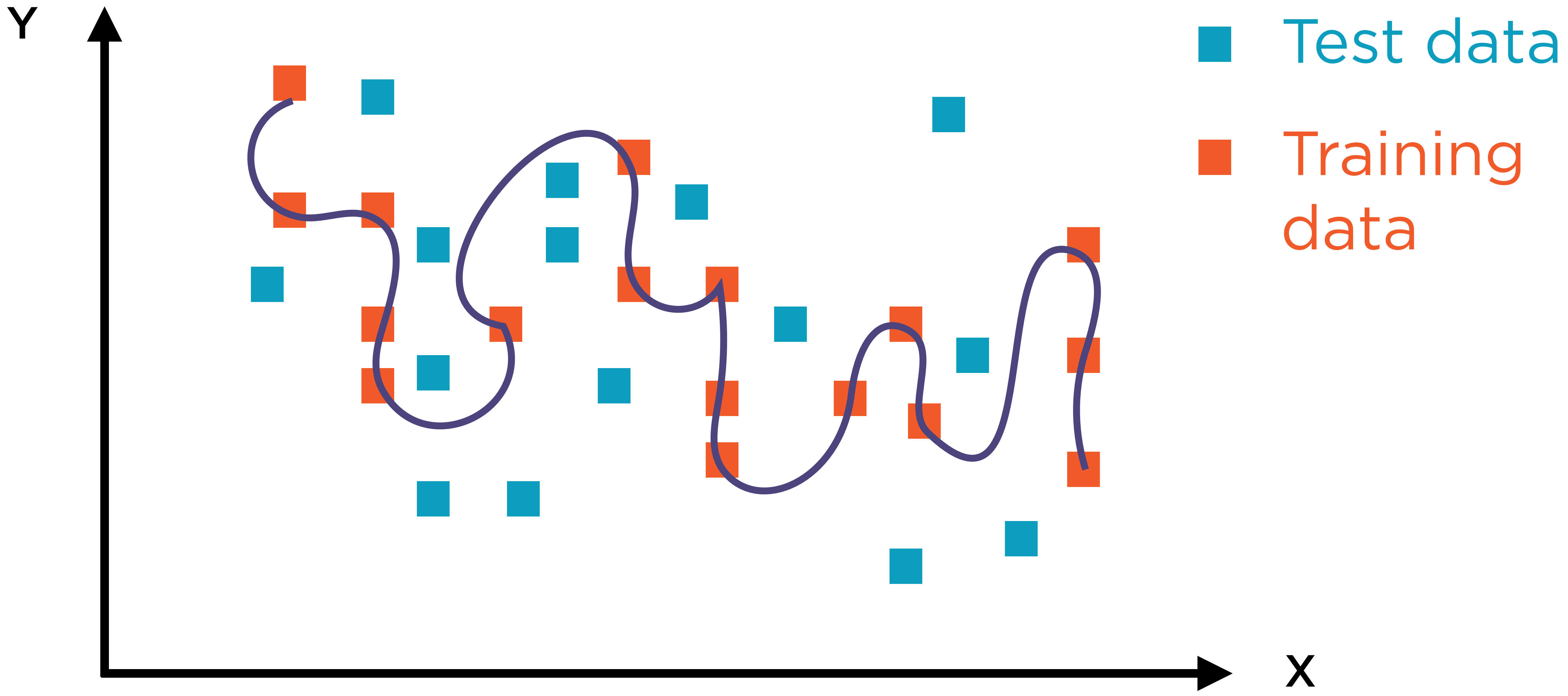
But given a new set of points, this curve might perform quite poorly

Connecting the Dots



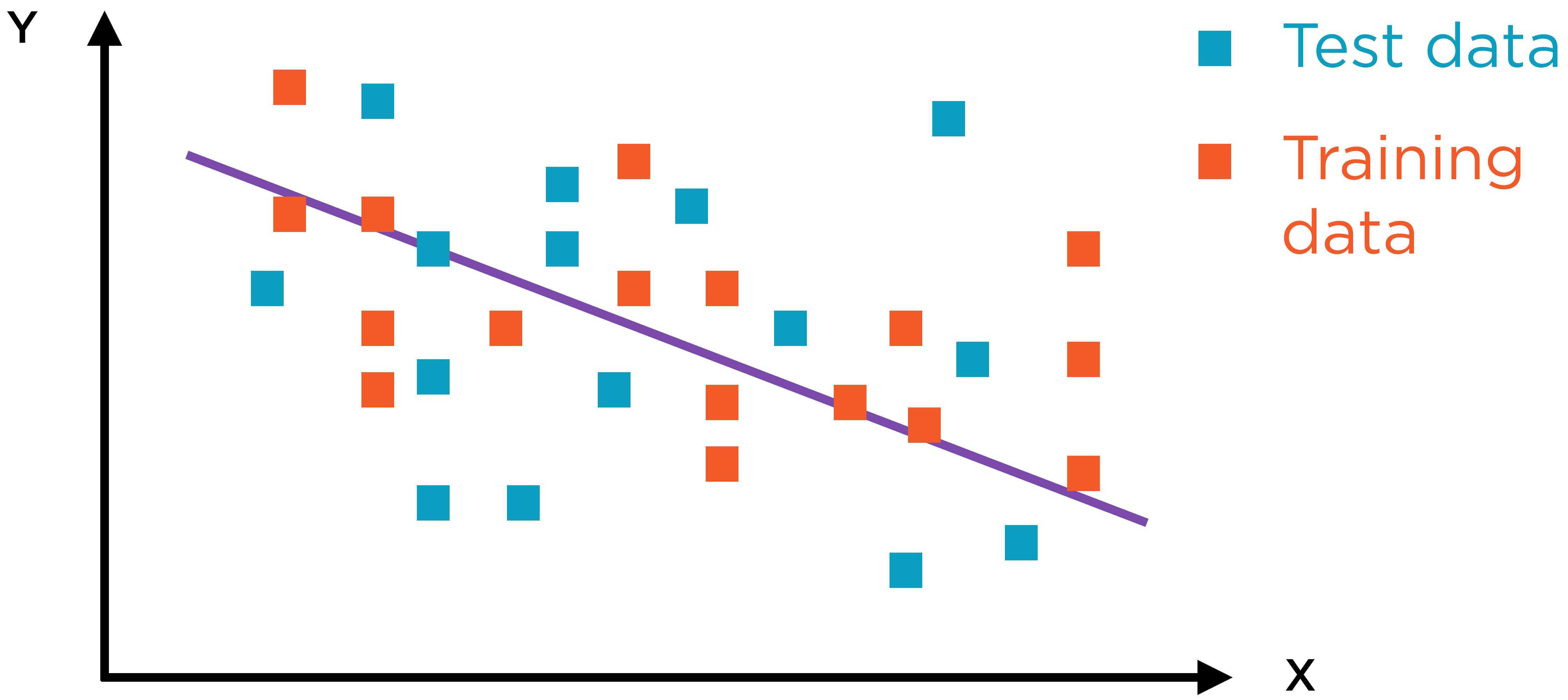
The original points were "training data", the new points are "test data"

Overfitting



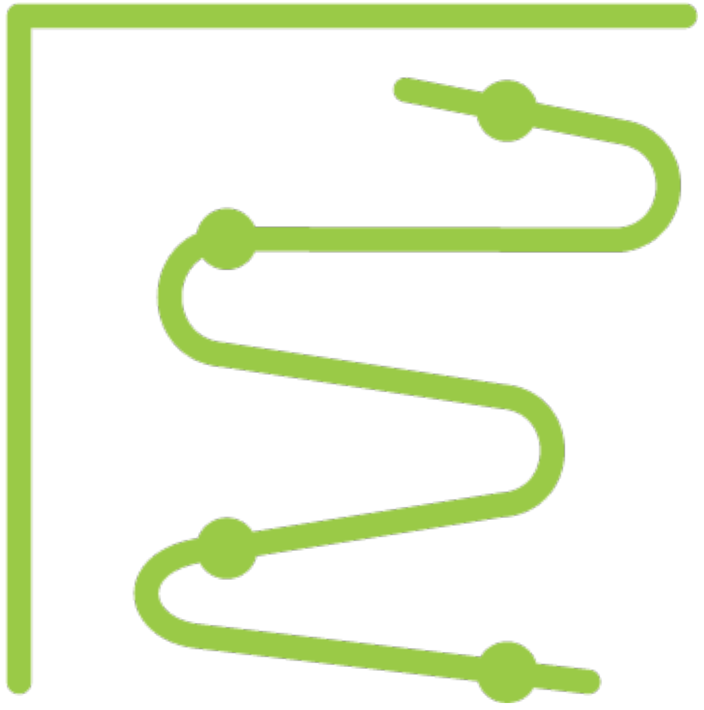
Great performance in training, poor performance in real usage

Connecting the Dots



A simple straight line performs worse in training, but better with test data

Overfitting



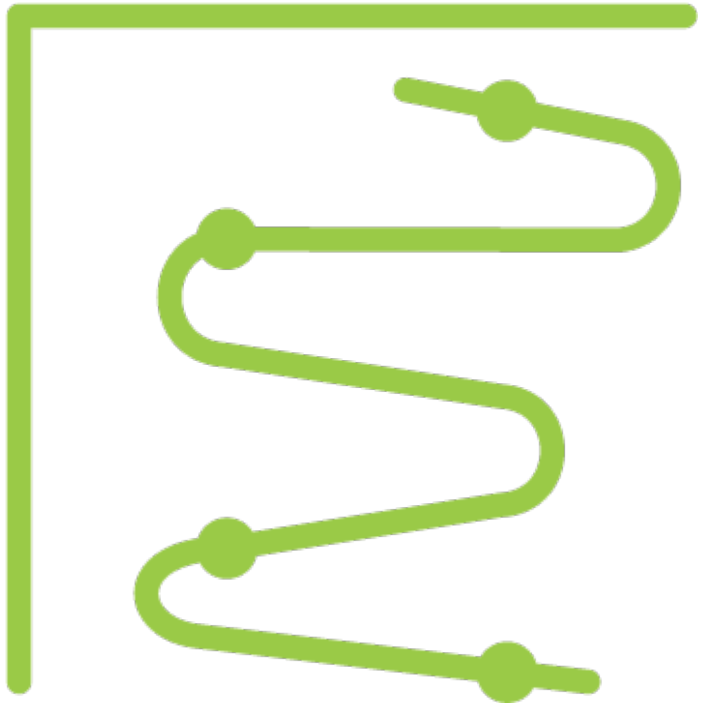
Model has memorized the training data

Low training error

Does not work well in the real world

High test error

Overfitting



Model has not extracted general patterns that exist in the data

The model's ability to adapt to new unseen data is poor

Preventing Overfitting

Regularization - Penalize complex models

Cross-validation - Distinct training and validation phases

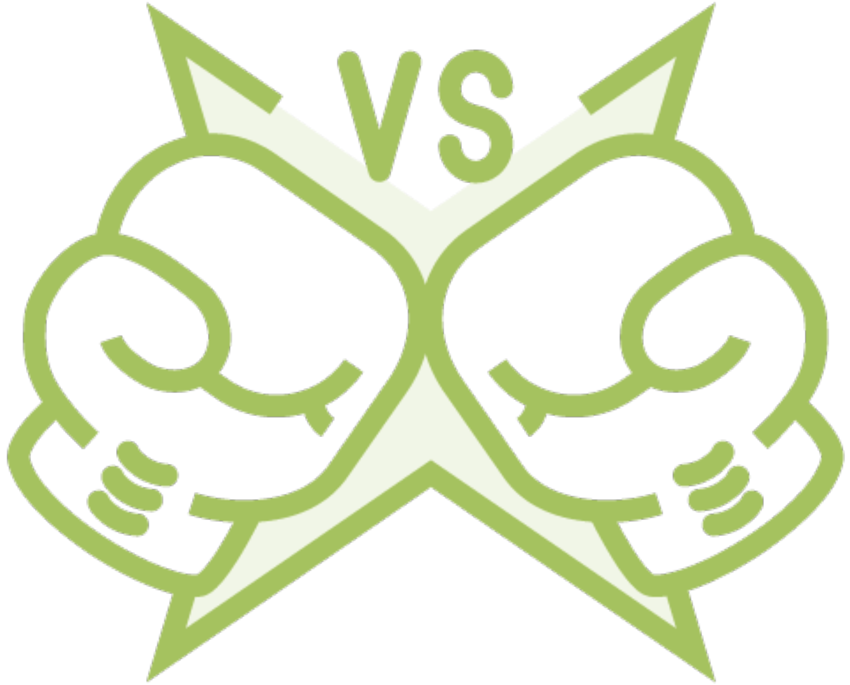
Dropout (NNs only) - Intentionally turn off some neurons during training

Ensemble learning - aggregate predictions from individual learners

Problems Afflicting AI-based Solutions

Overfitting	Training-serving Skew
Concept Drift	Concerted Adversaries

Training-Serving Skew

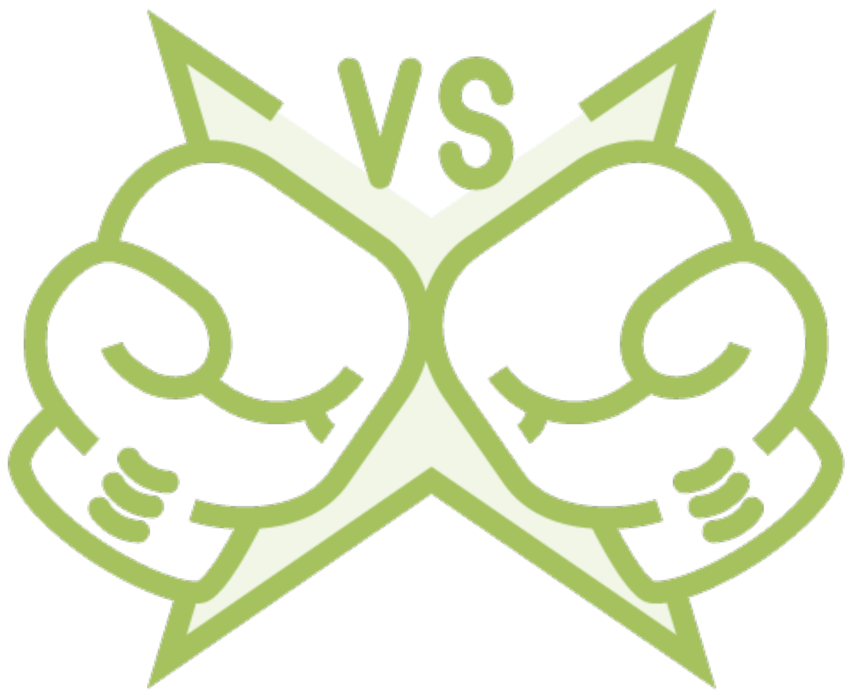


Models are performing well in backtests

But performing poorly in production

Training-serving skew is a big, but neglected cause

Training-Serving Skew



Training data is sourced from batch pipelines

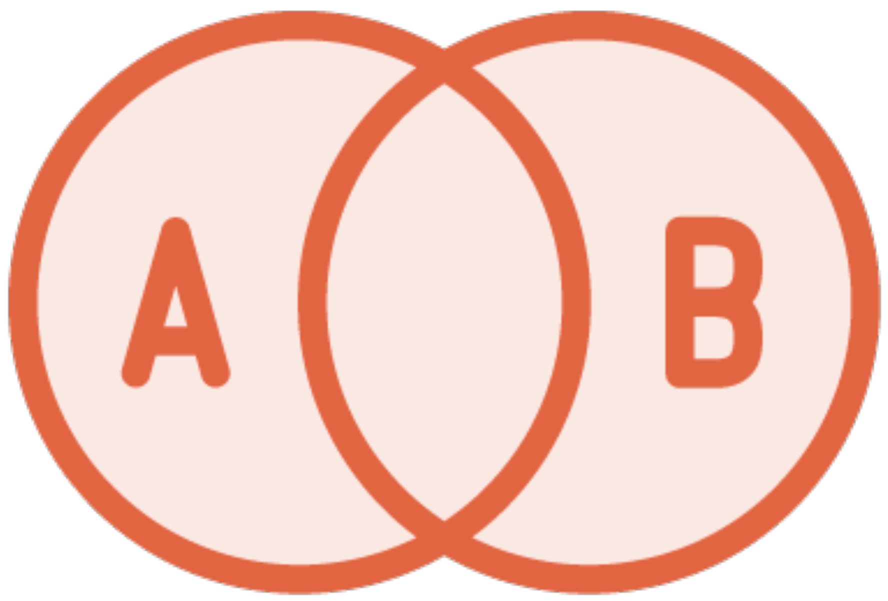
Processed meticulously well

Prediction data is sourced from streaming pipelines

Processed in an ad-hoc manner with many short cuts

Batch and streaming data should be processed in the same manner, using the same pipeline

Lambda or Kappa



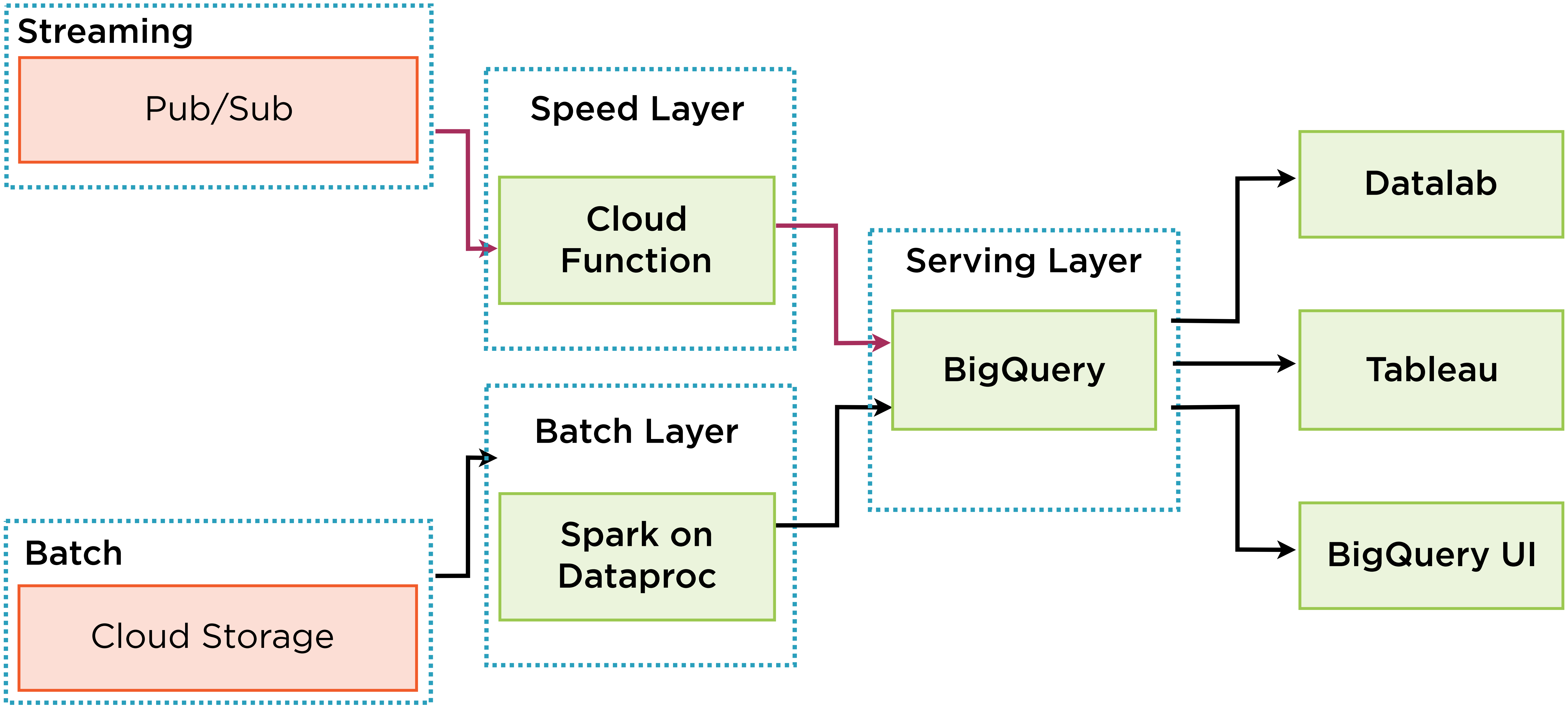
Lambda and Kappa architectures both combine batch and stream data

They do so in different ways

Lambda couples them less tightly

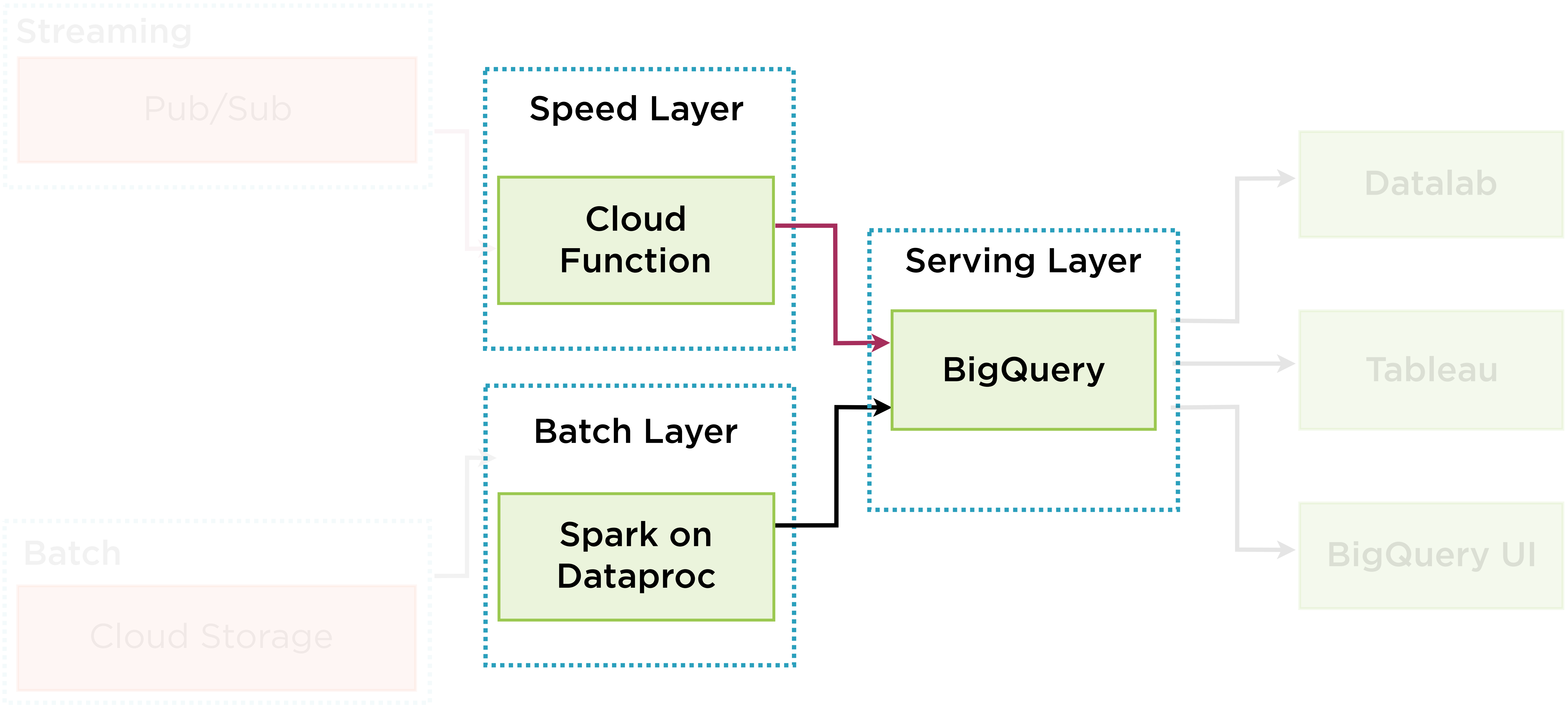
But is more robust

Lambda Architecture



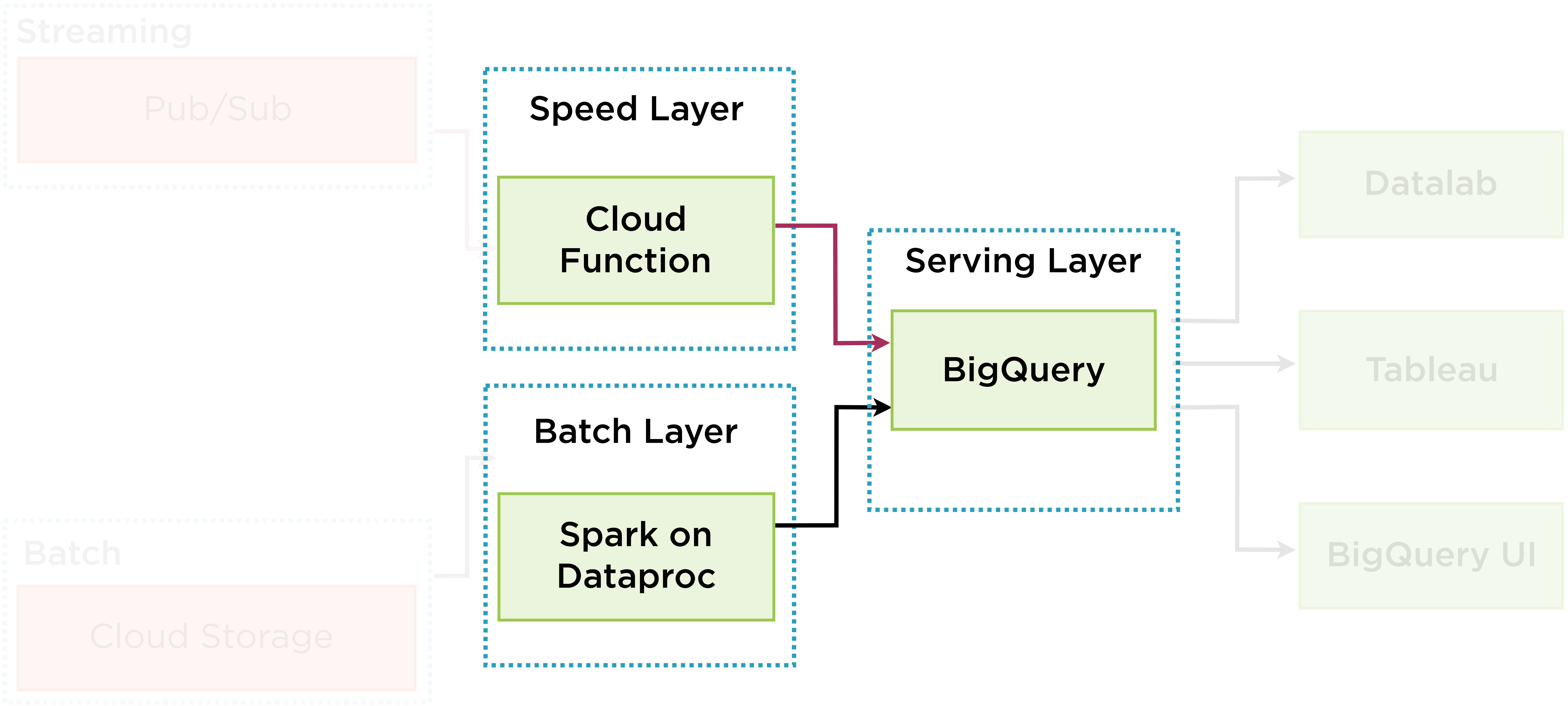
Hybrid approach to batch and near real-time processing

Lambda Architecture



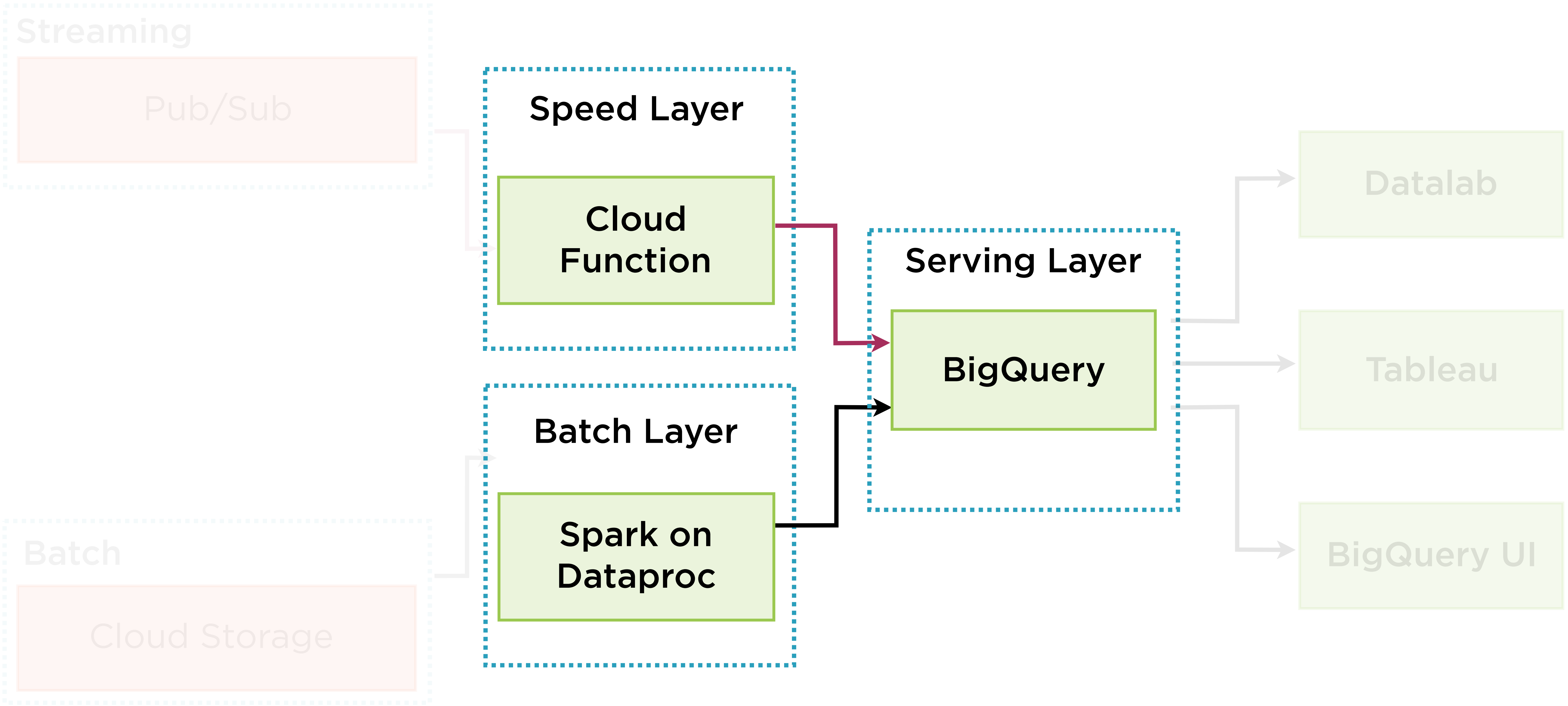
The basic architecture contains these 3 layers - batch is often the source of truth

Lambda Architecture



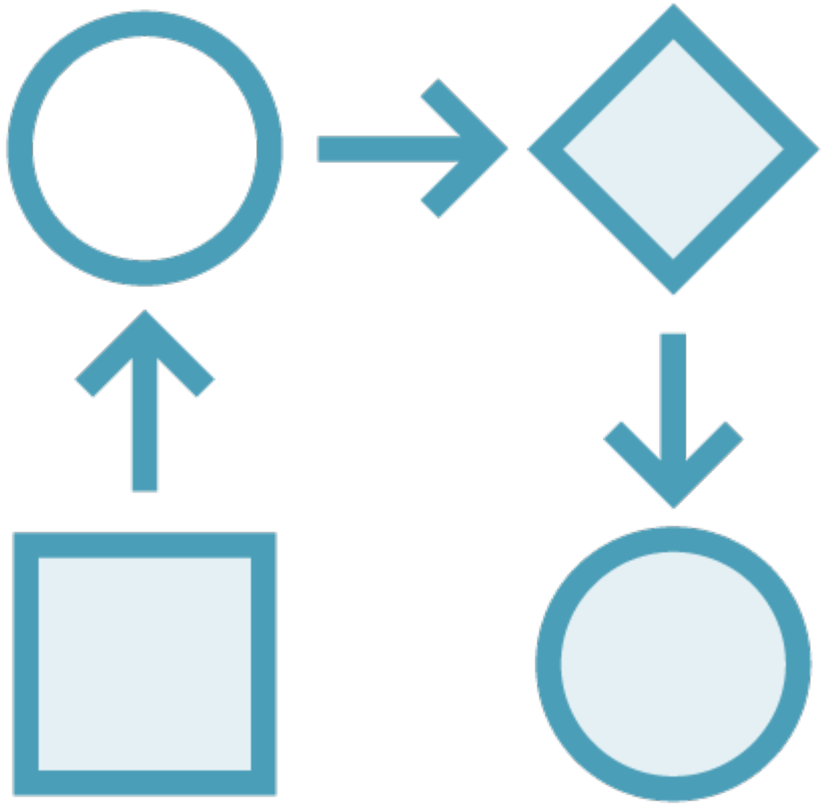
Streaming code may also be stored in the batch layer

Lambda Architecture



Different code to be maintained to process batch and streaming

Kappa Architecture



Original idea proposed by Jay Kreps

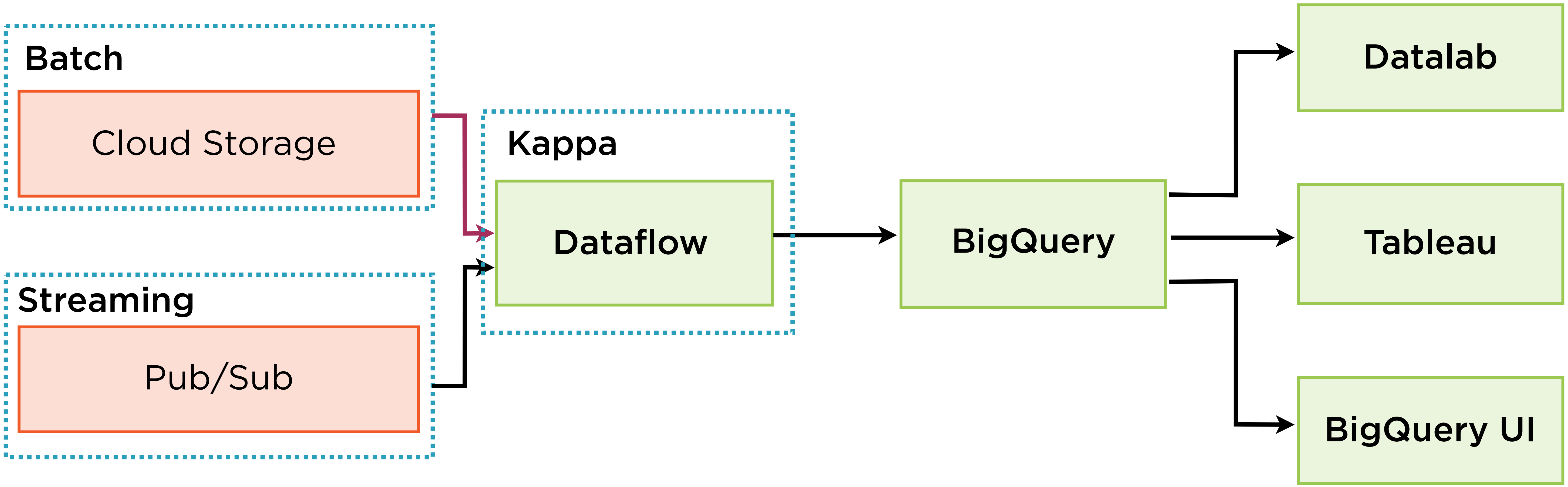
Stream is the source of truth

Use technology which retains a log of all data

Always process from the stream

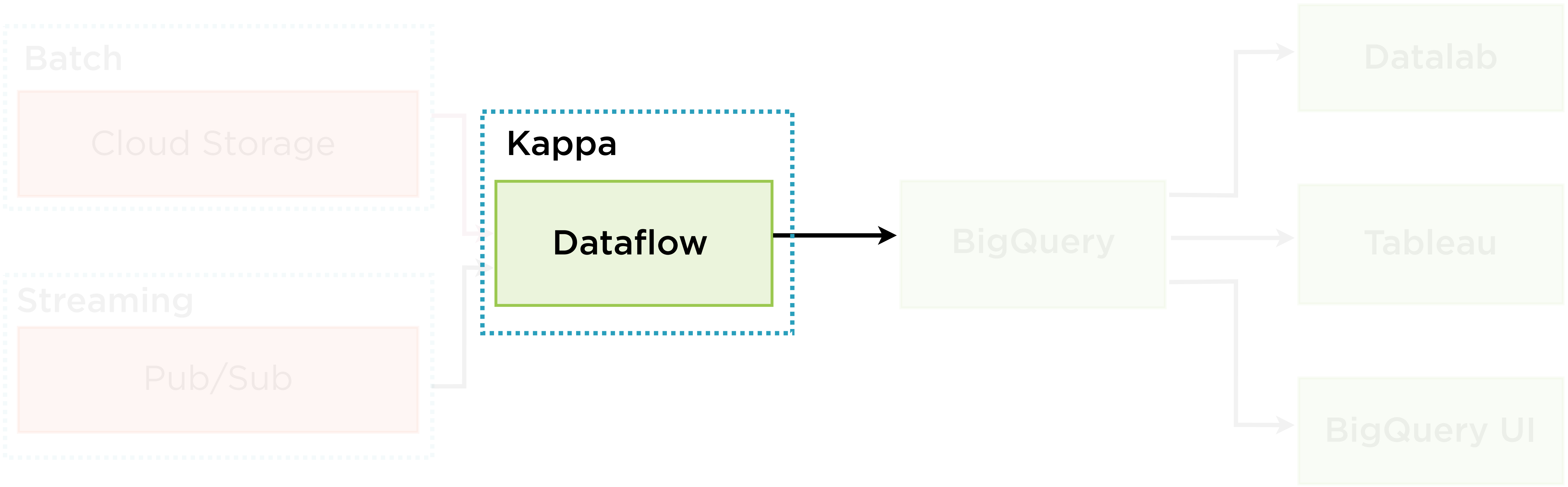
Use a single processing framework

Simple Kappa Architecture



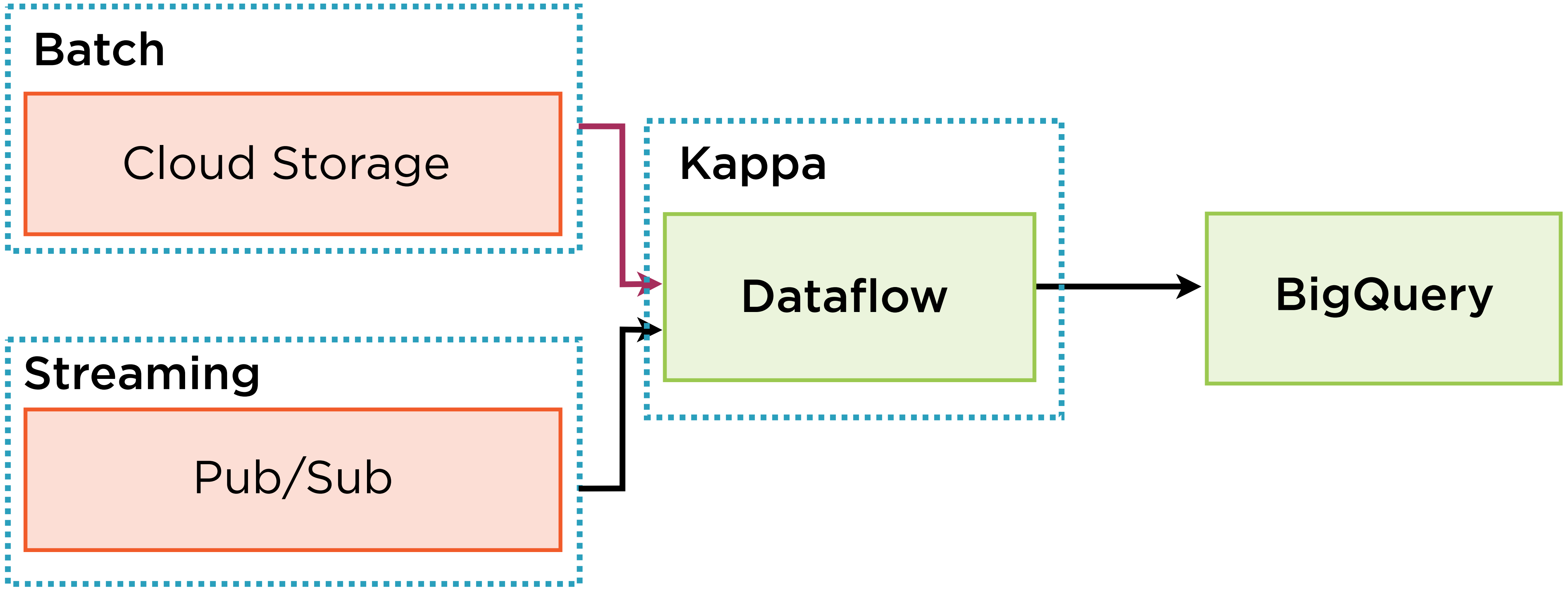
Process batch and streaming data using the same code

Simple Kappa Architecture

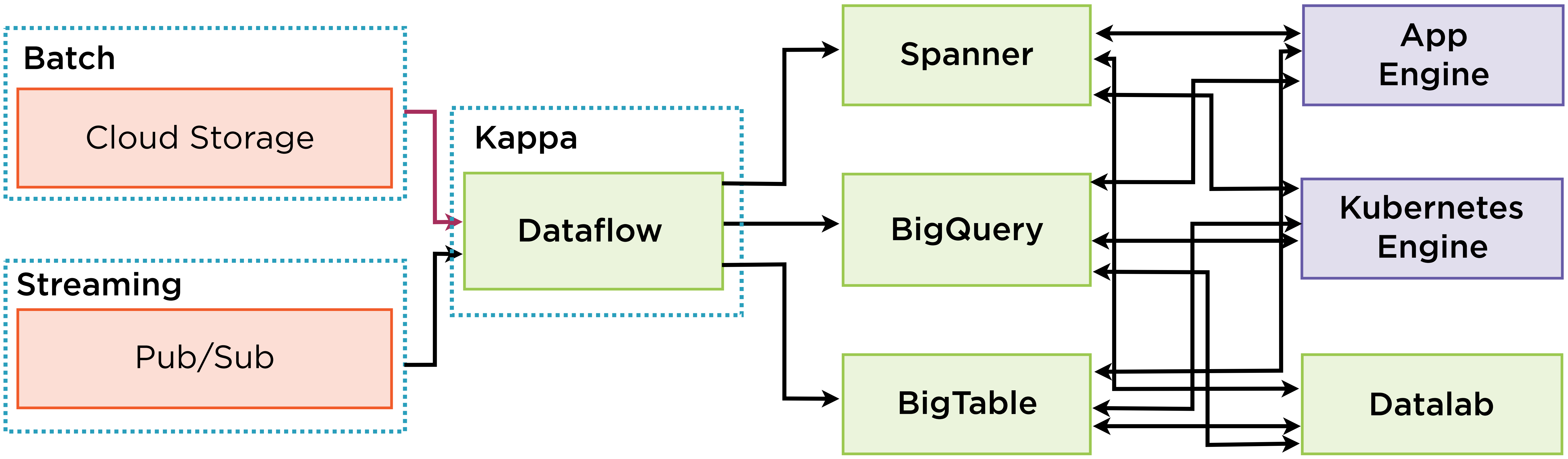


Maintain one codebase for simplicity and robustness

Simple Kappa Architecture

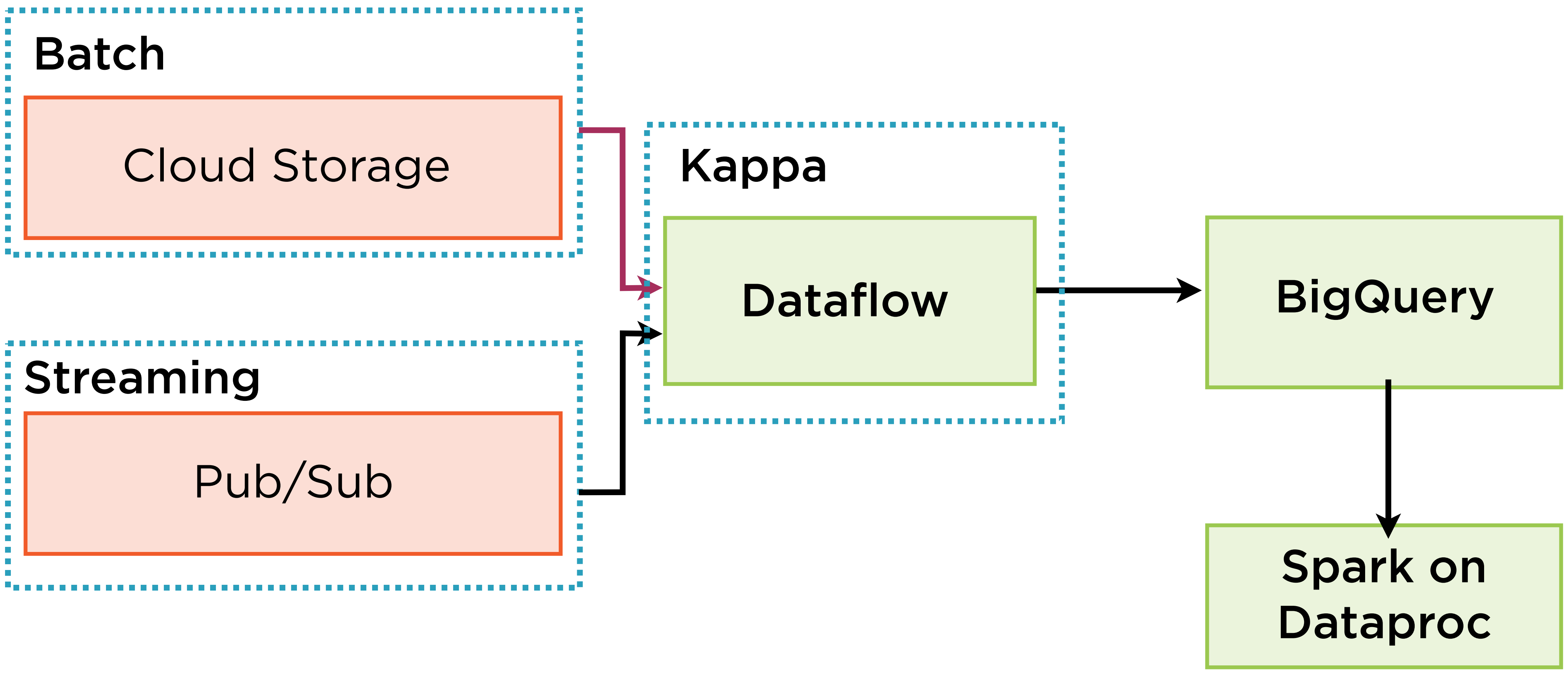


Complex Kappa Architecture



Training and prediction data ought to follow identical code paths

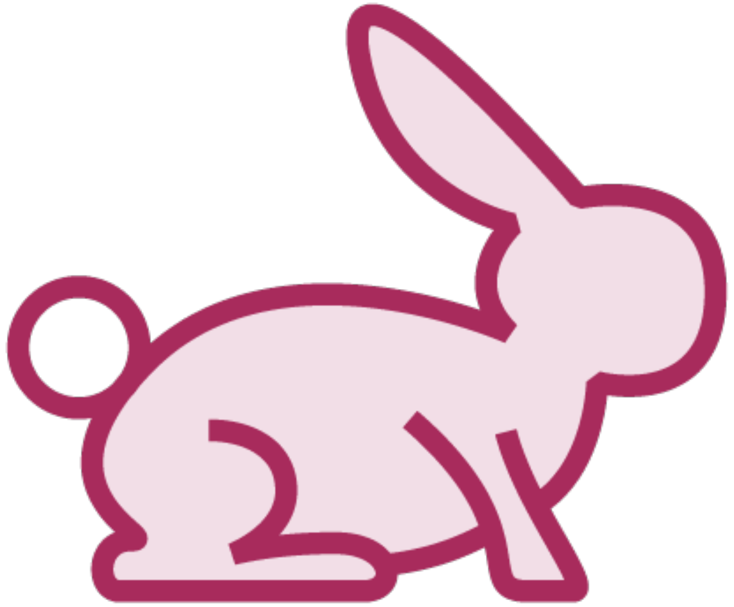
Complete Big Data Pipeline



Problems Afflicting AI-based Solutions

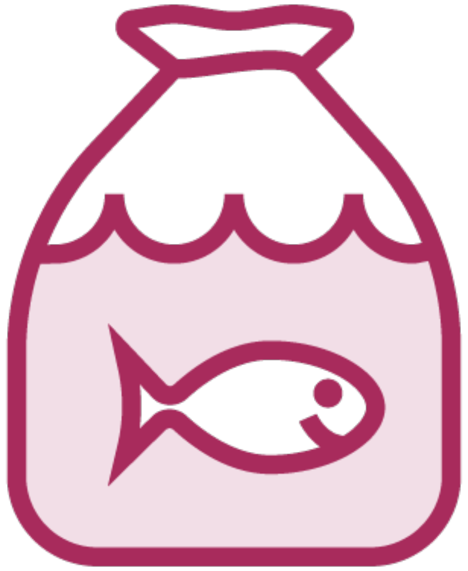
Overfitting	Training-serving Skew
Concept Drift	Concerted Adversaries

Whales: Fish or Mammals?



Mammals

Members of the infraorder *Cetacea*



Fish

Look like fish, swim like fish, move with fish

Whales: Fish or Mammals?

ML-based Classifier

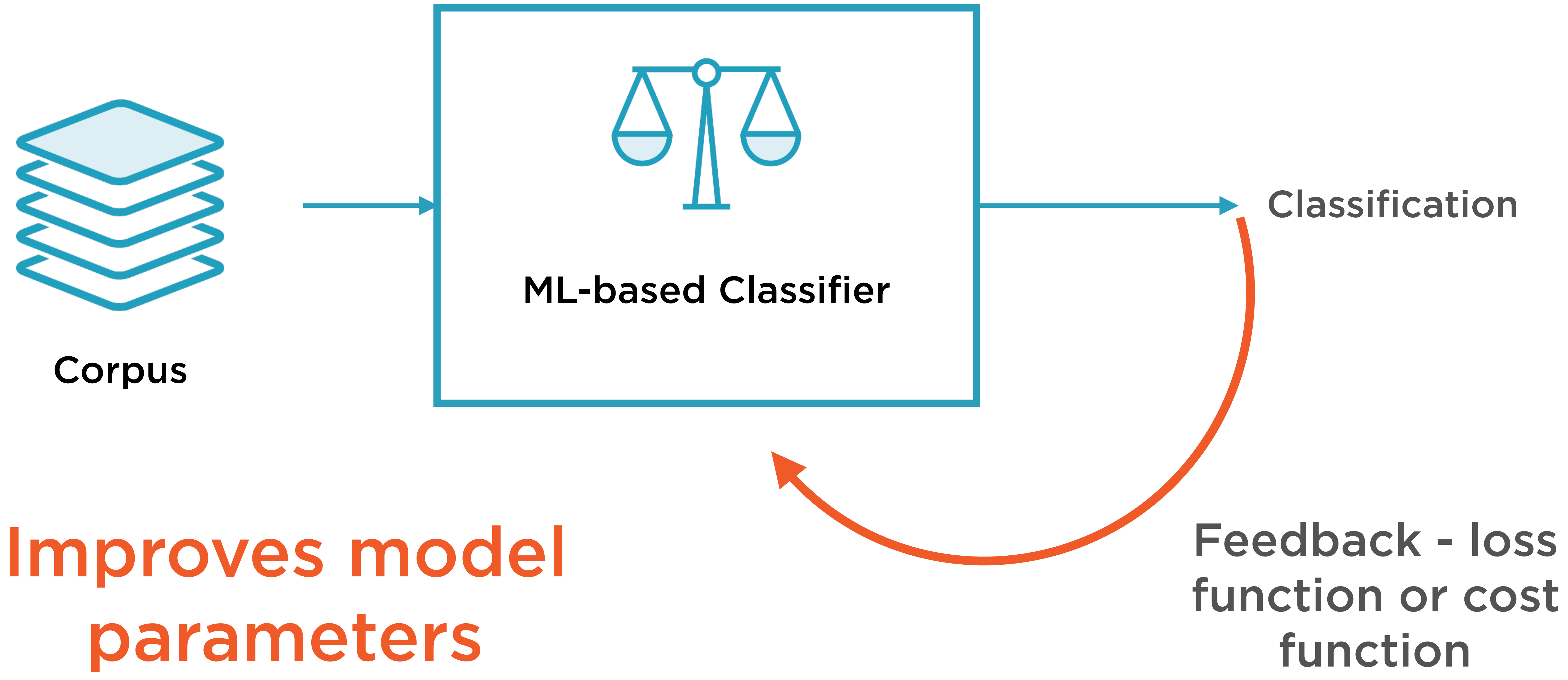
Training

Feed in a large corpus of data classified correctly

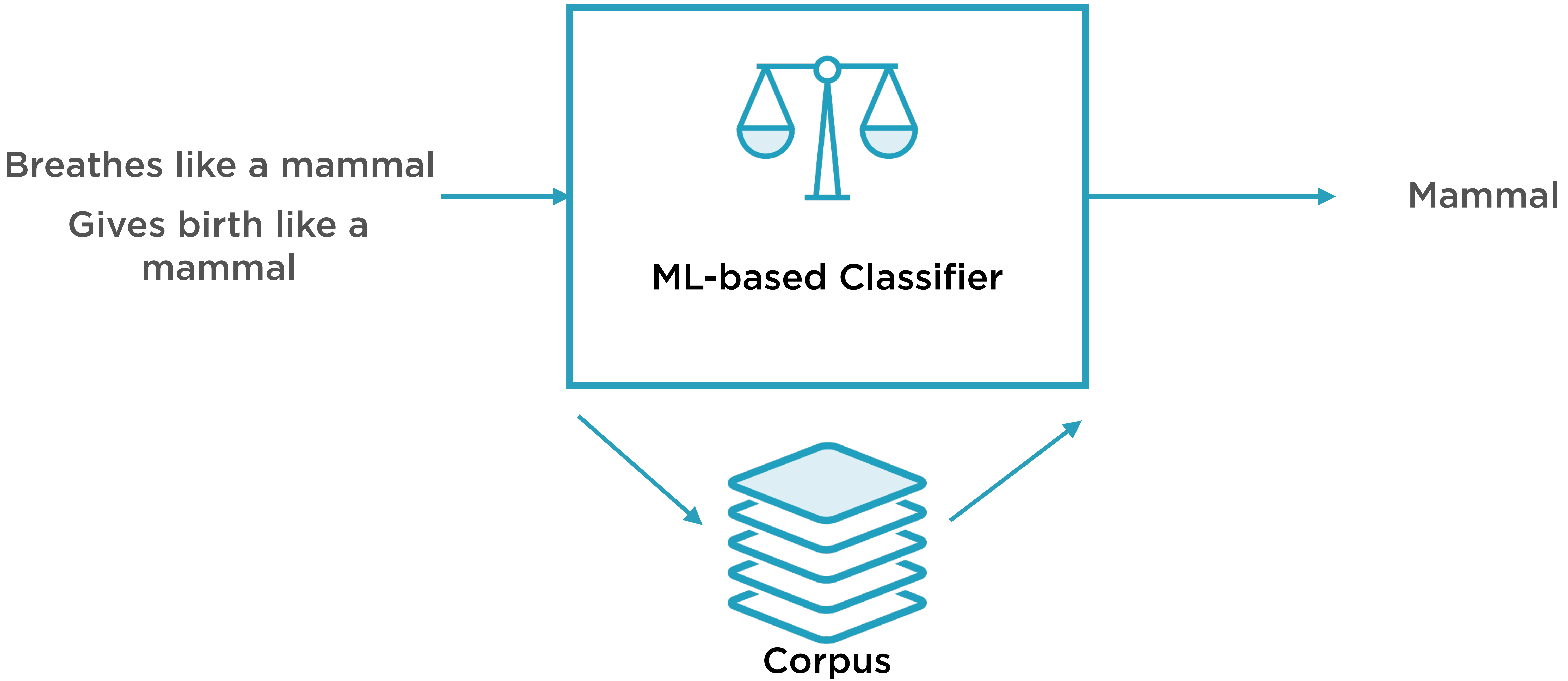
Prediction

Use it to classify new instances which it has not seen before

Training the ML-based Classifier



ML-based Binary Classifier



```
y = f(x)
```

Supervised Machine Learning

Most machine learning algorithms seek to “learn” the function f that links the features and the labels

Concept Drift

The relationship between features (X-variables) and labels (Y-variables) changes over time; ML models fail to keep up, and consequently their performance suffers

Concept Drift

Concept drift happens because the world changes

Relationships are dynamic, not static

Same reason why rule-based systems degrade faster than ML-based

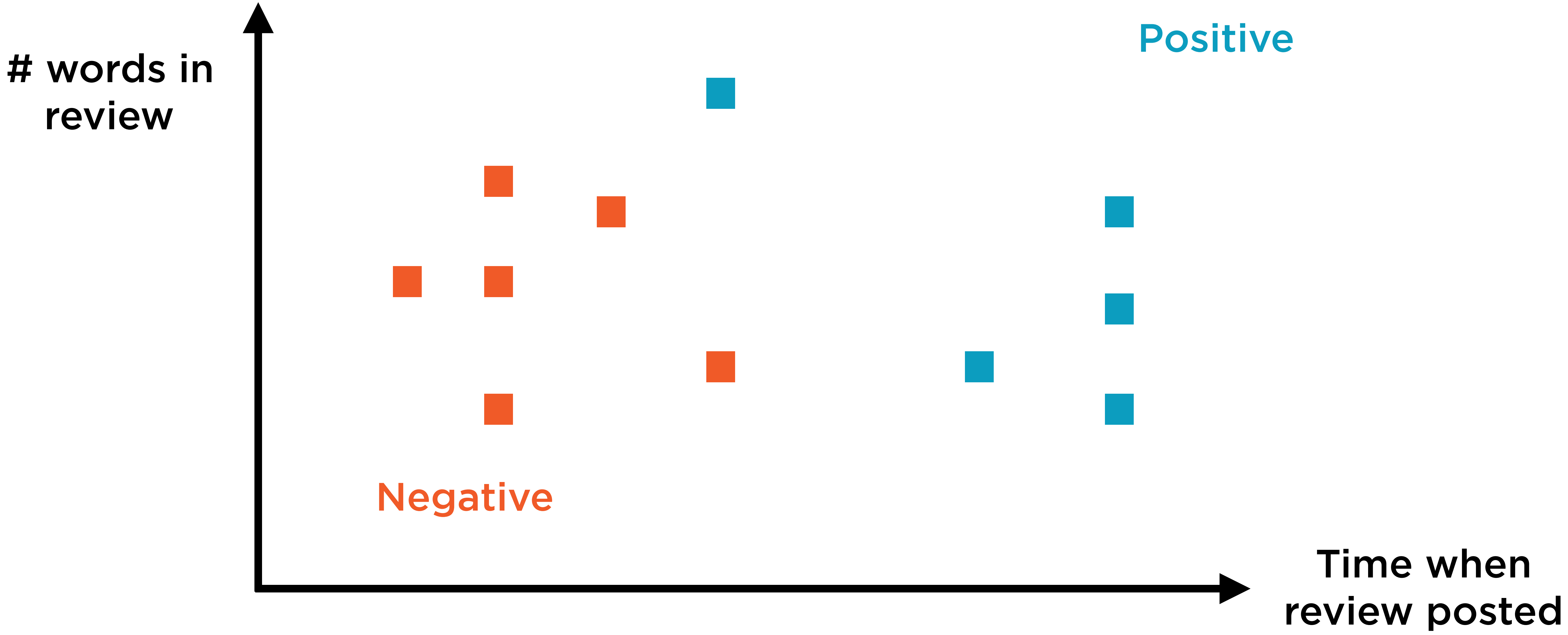
Concept Drift

Related to “regime changes” or “structural breaks” in statistics

Solution #1: Keep monitoring and re-training deployed models

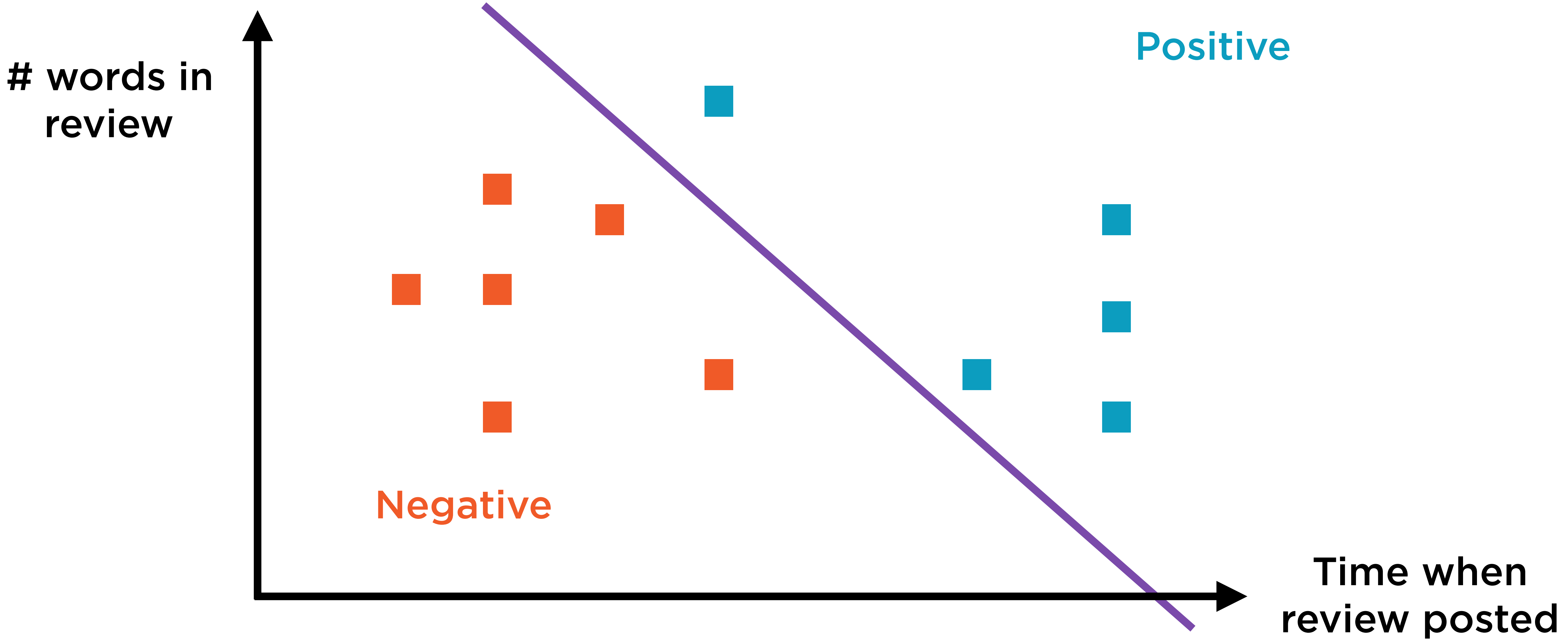
Solution #2: Re-develop models from scratch (if re-training won’t suffice)

Classification Decision Boundary



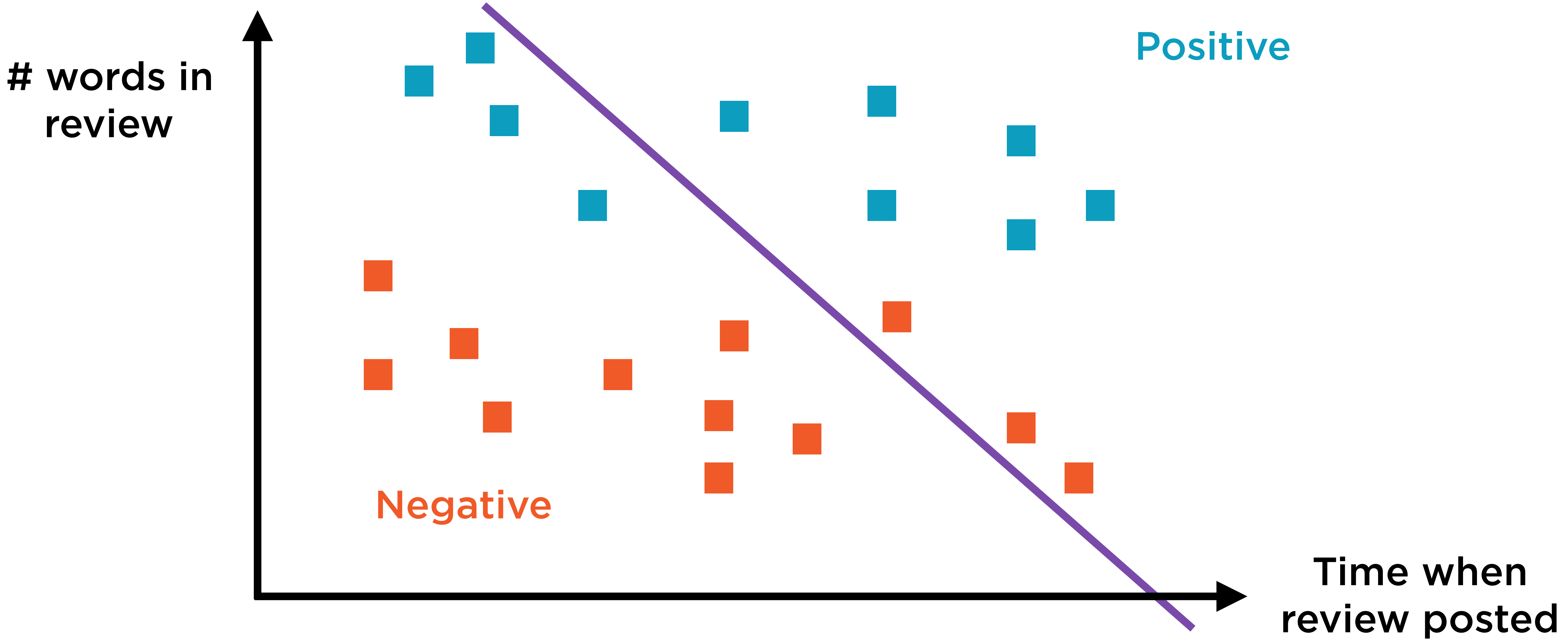
Classify reviews as negative or positive

Classification Decision Boundary



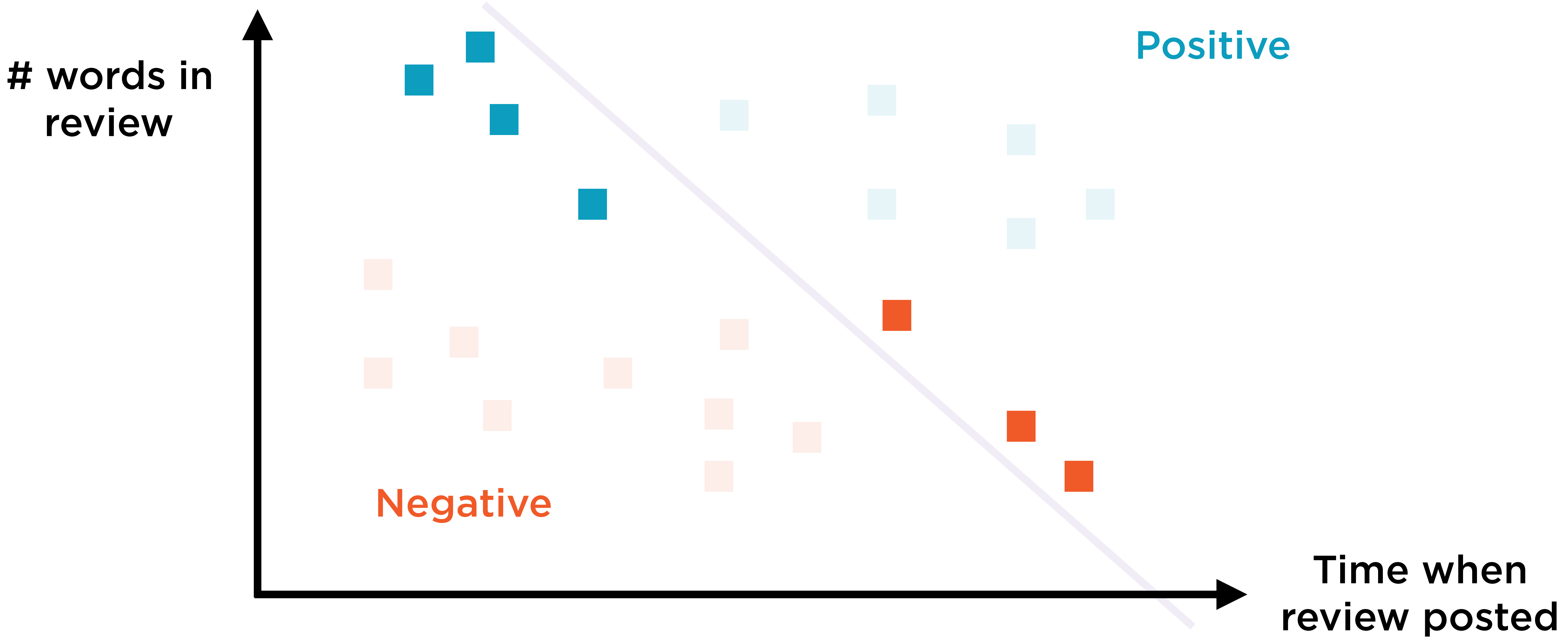
Original decision boundary based on training and test data available at model deployment

Classification Decision Boundary

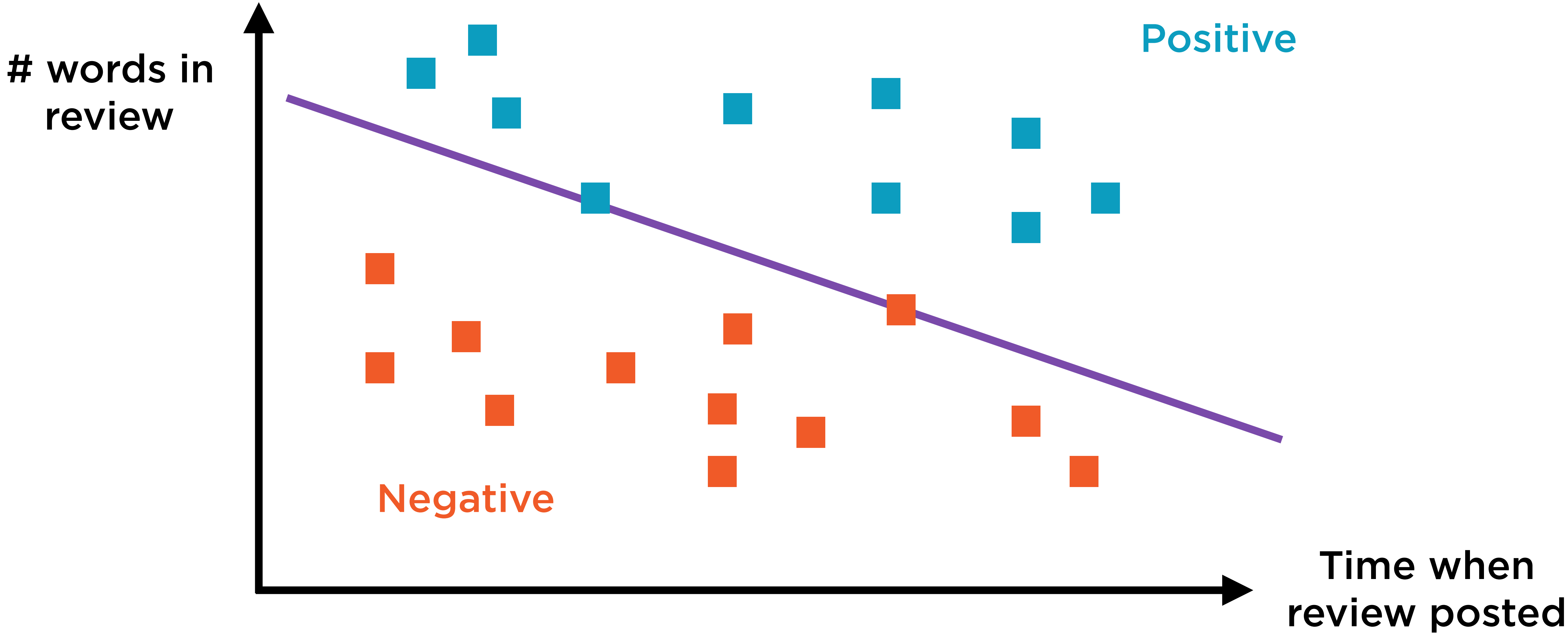


New points available as model used for prediction

Classification Decision Boundary

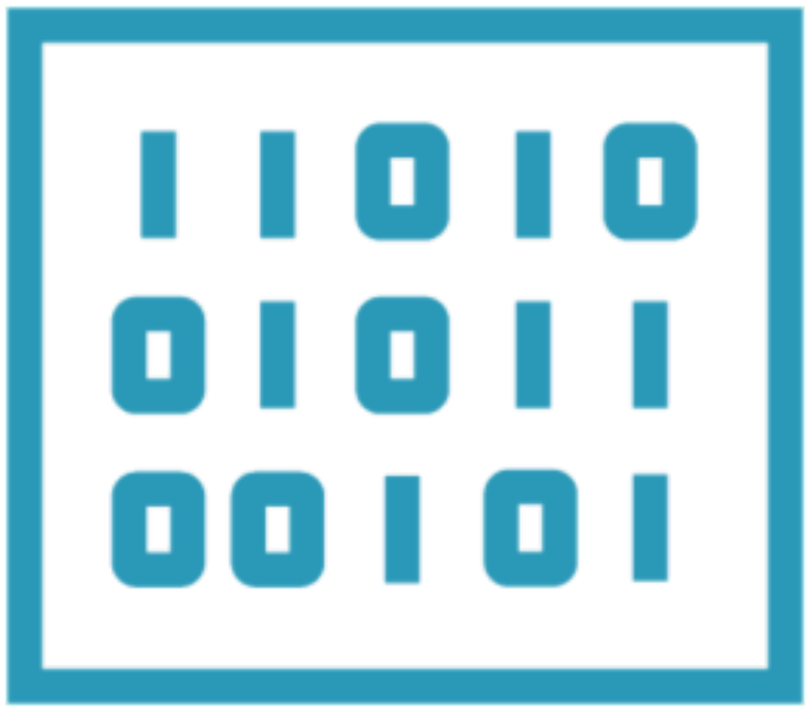


Classification Decision Boundary



Decision boundary has changed - model needs to be updated on the new data

Concept Drift



The data itself has changed

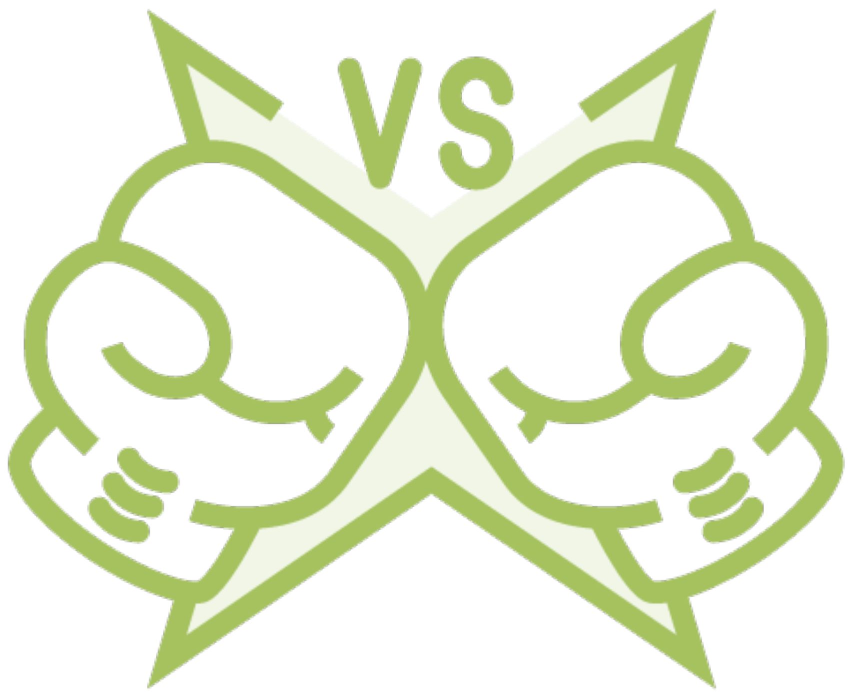
Our interpretation of the data has changed

Concept Drift can be mitigated by constantly monitoring and re-training deployed models

Problems Afflicting AI-based Solutions

Overfitting	Training-serving Skew
Concept Drift	Concerted Adversaries

Concerted Adversaries

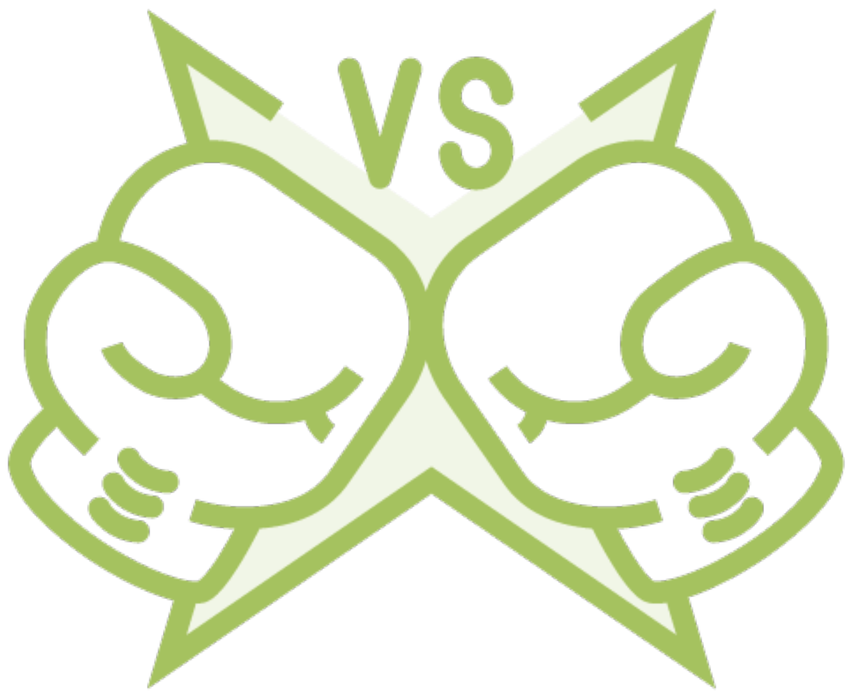


Consider common AI use-cases

- Fraud detection
- Fake news detection
- Quantitative trading

Concerted adversaries try to confuse, mislead models

Concerted Adversaries



Human minders for models becoming more important

Human minders learn from cases where models got it wrong

Back to the future: From machine learning to manual learning

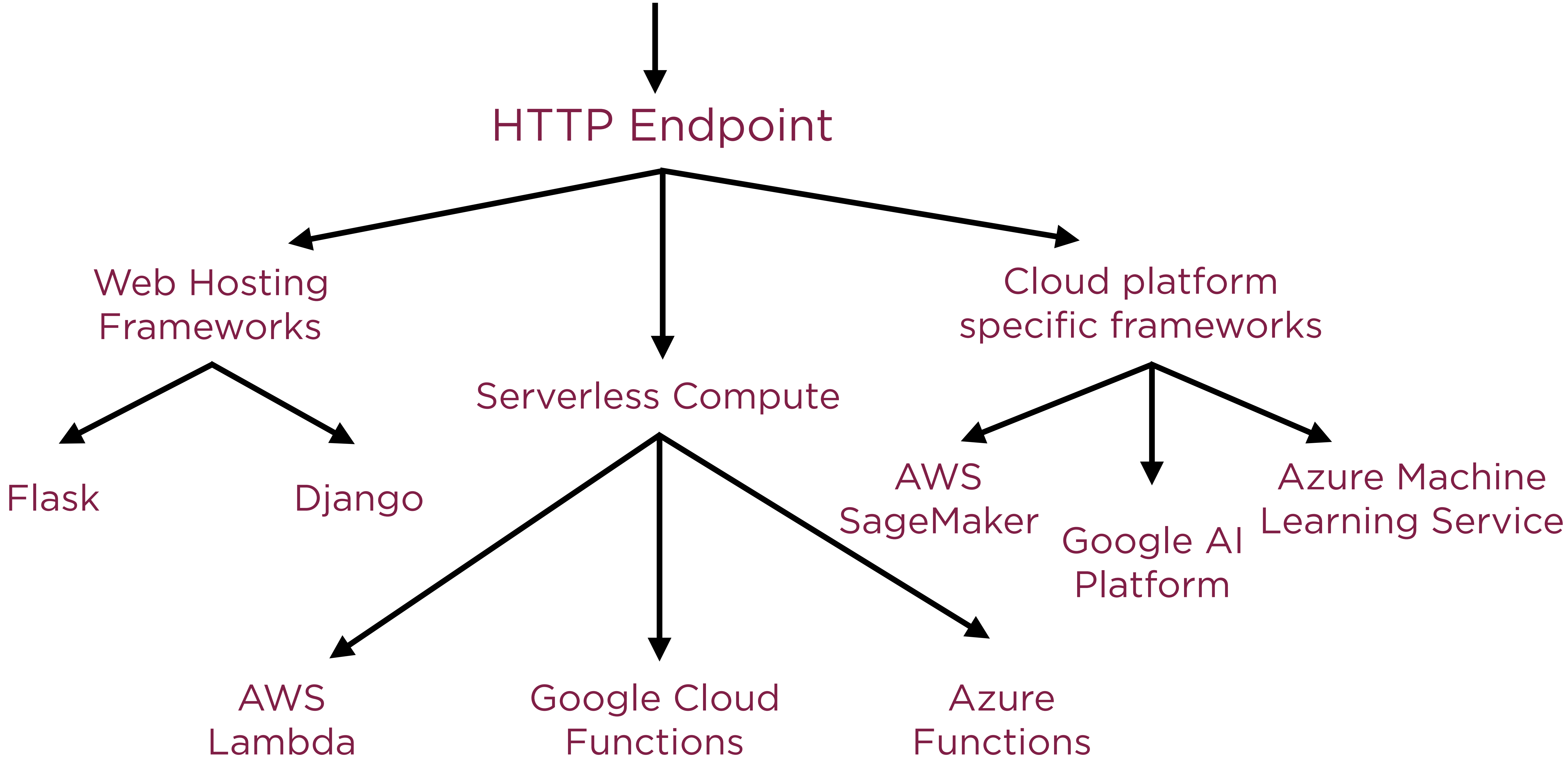
Thwarting concerted adversaries requires continuous **manual learning** to complement machine learning

Problems Afflicting AI-based Solutions

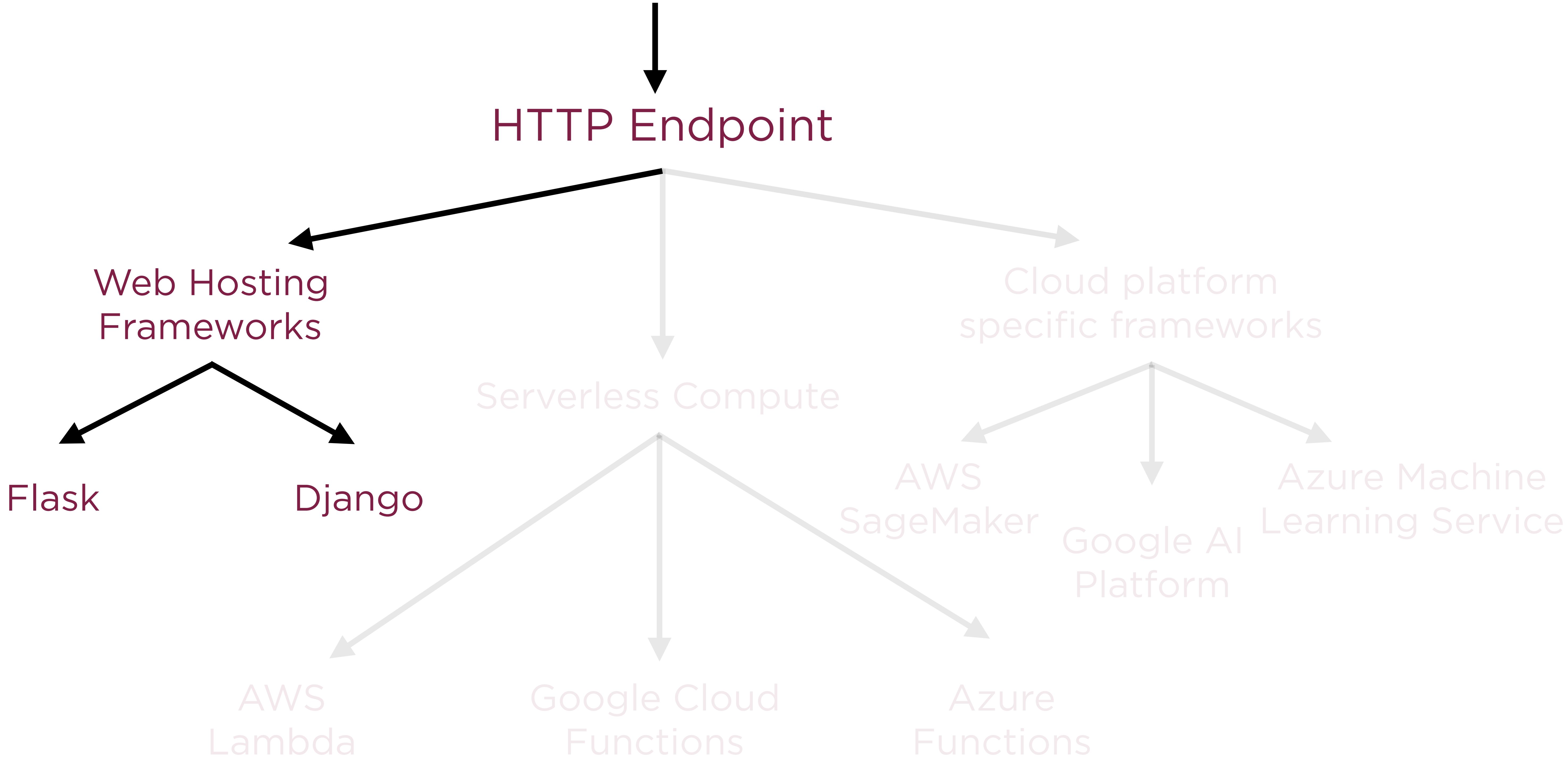
Overfitting	Training-serving Skew
Concept Drift	Concerted Adversaries

Deploying Models for Prediction

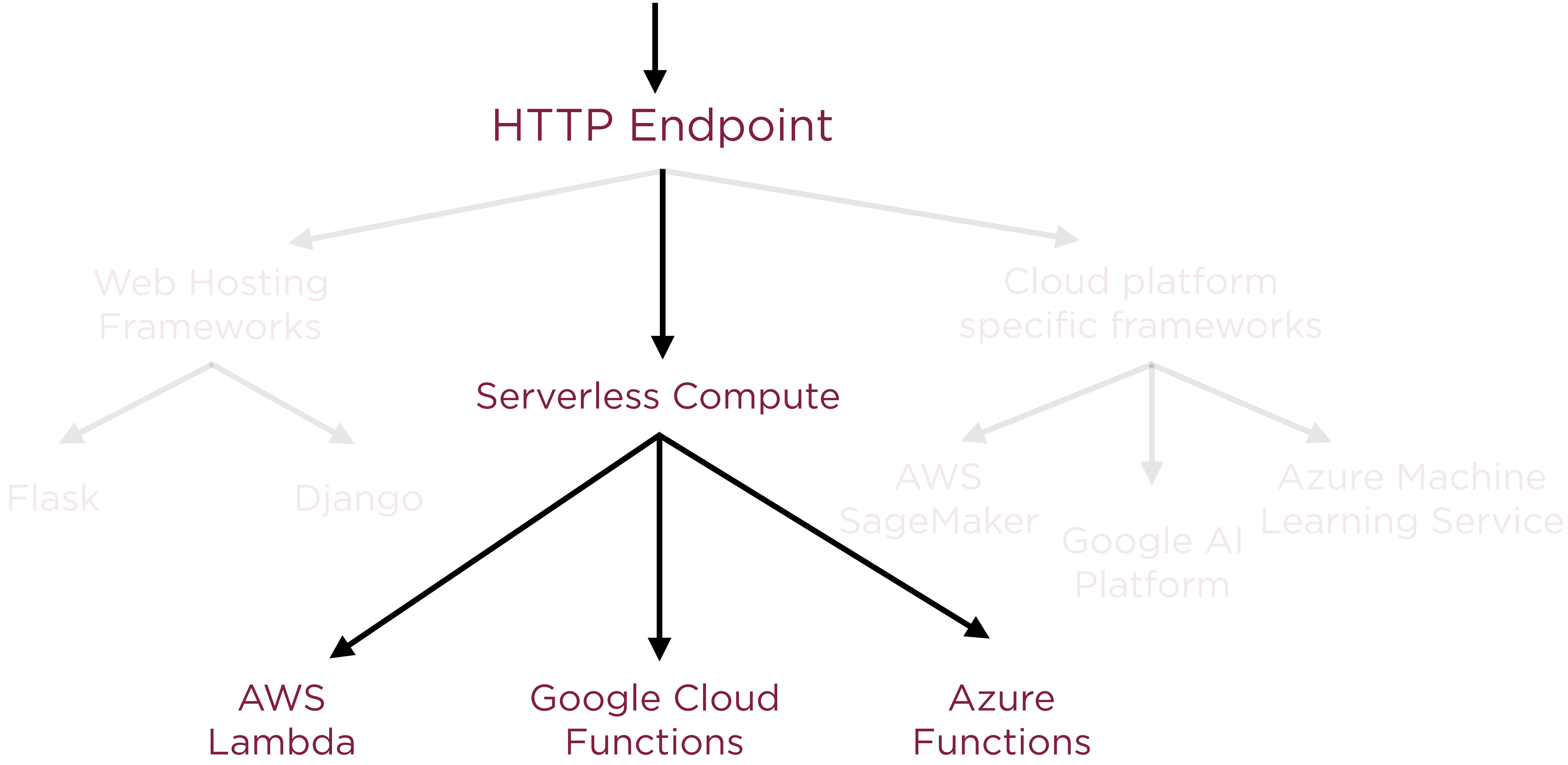
Deploying Models for Prediction



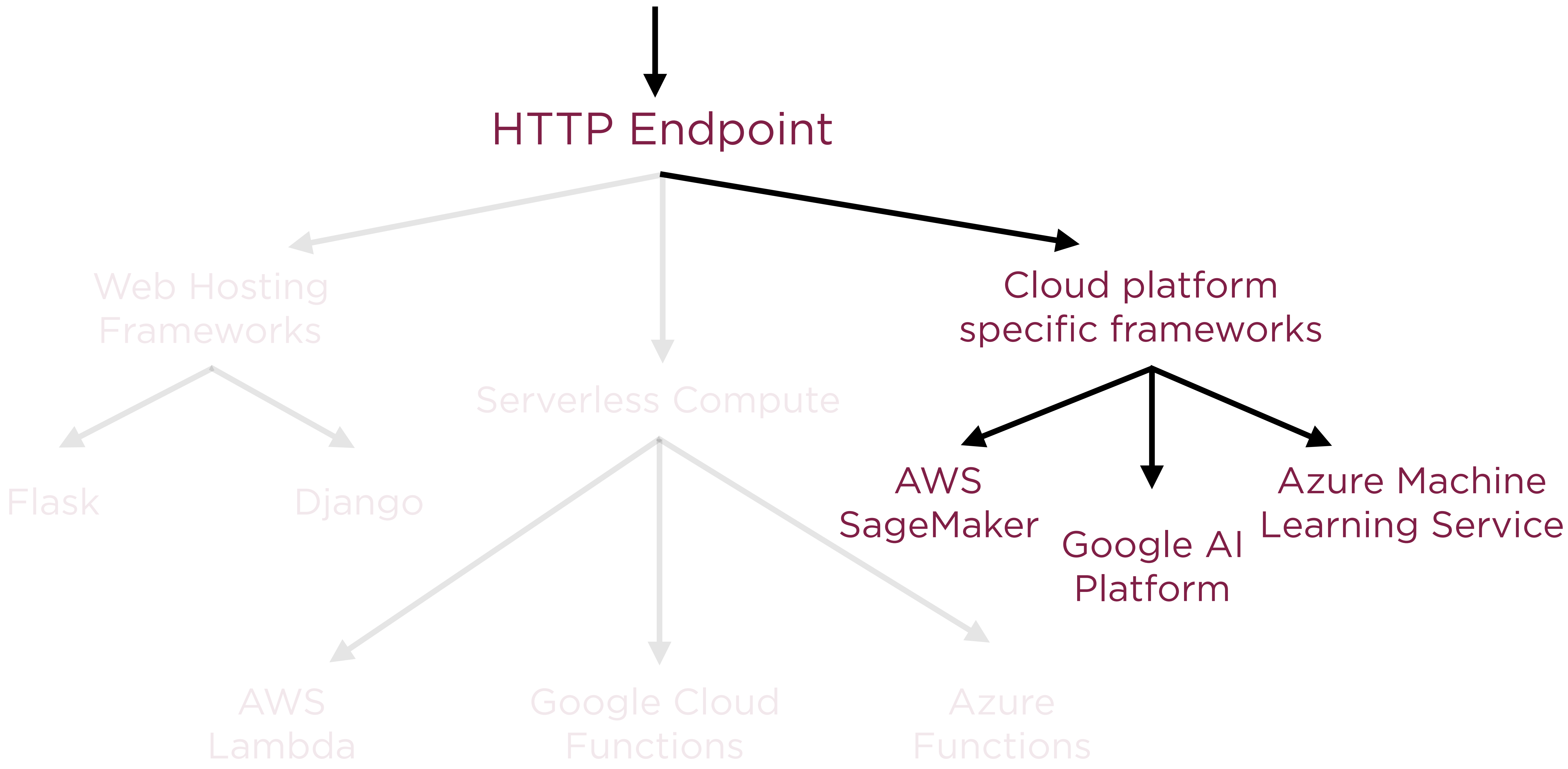
Deploying Models for Prediction



Deploying Models for Prediction



Deploying Models for Prediction



Summary

Recent challenge posed by underperformance of deployed models

Several possible causes

Overfitting, training-serving skew

Concept drift, concerted adversaries

Need for monitoring and retraining of deployed models

Model development does not end with deployment