Machine Learning Systems for Engineers

Where Data Science Meets Engineering

Who Am I?

- I'm Cameron!
  - https://www.linkedin.com/in/cameron-joannidis/
- Consult across a range of areas and have built many big data and machine learning systems
- Specialise in several areas
  - Big Data / Data Engineering
  - Machine Learning / Data Science
  - Scala / Functional Programming

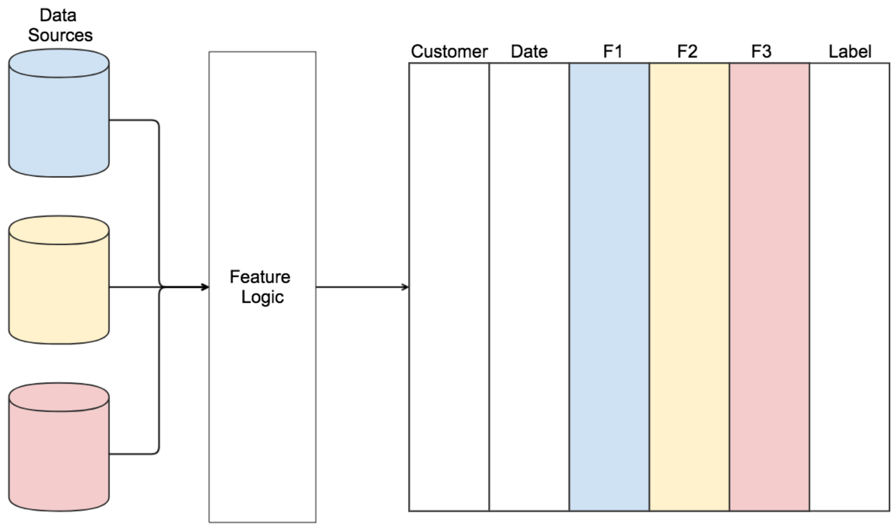
Agenda

- Data
- Deployment
- Metrics
- Big Data Iteration Speed

Data

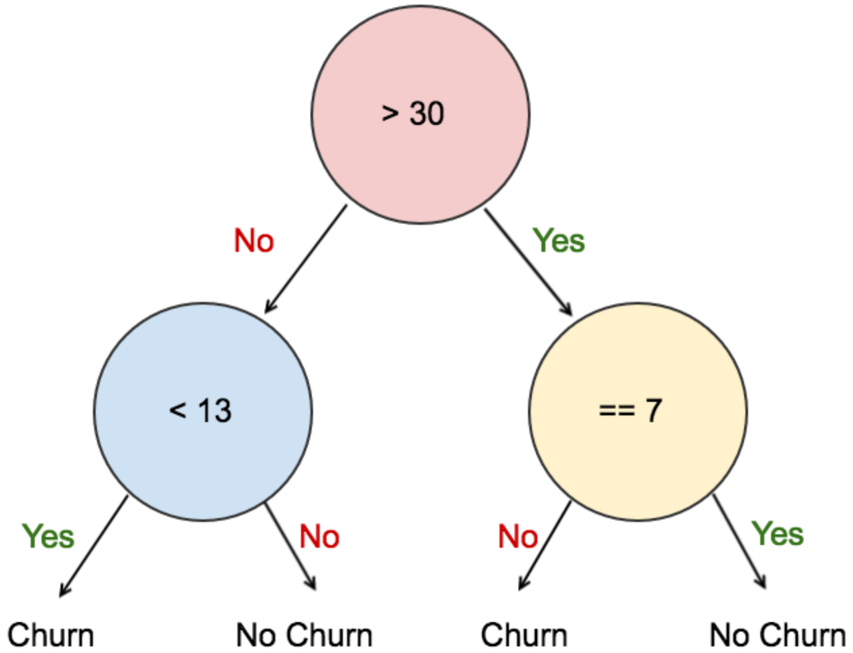
Example Use Case: Churn Prediction

We want to predict which users are likely to leave our service soon so that we can try and give them reasons to stay



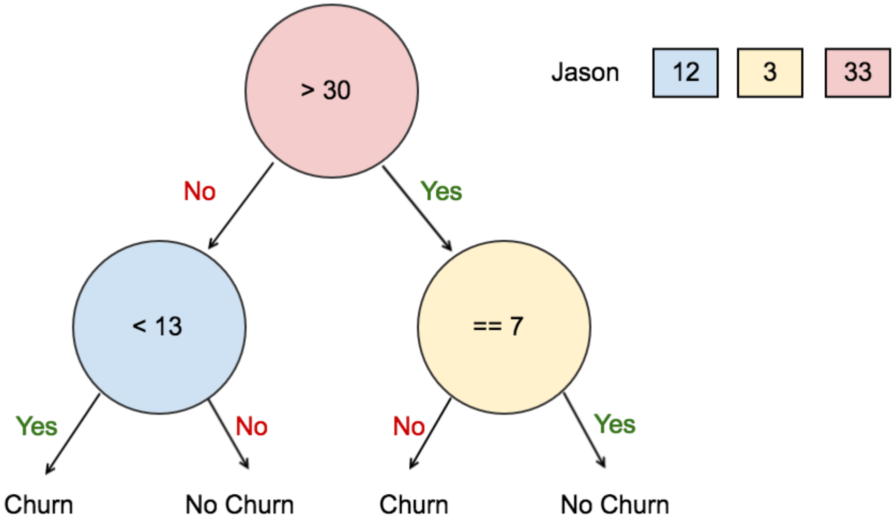
Training Data Creation

- Historical Data (need actual churn events as examples)
- We know the labels at train time
- Produce Features to try and predict the label



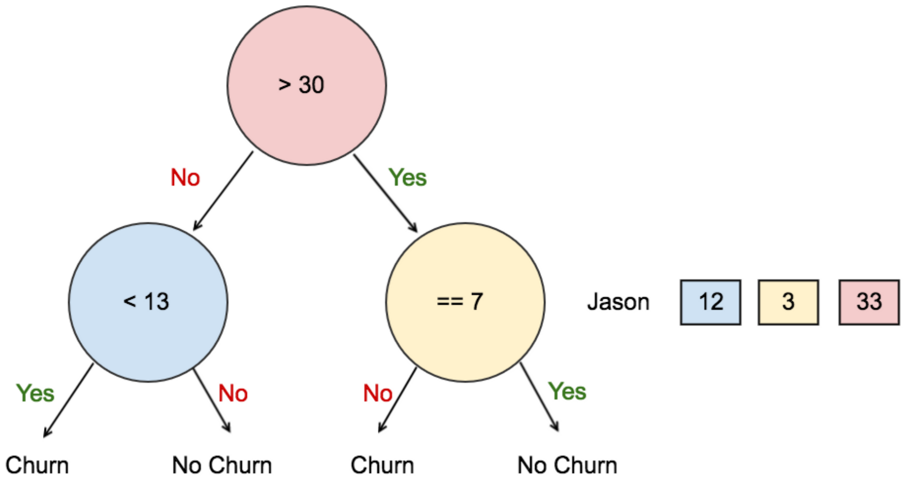
Train Our Model

- Minimise our loss function to best predict out labels (Churn/No Churn)



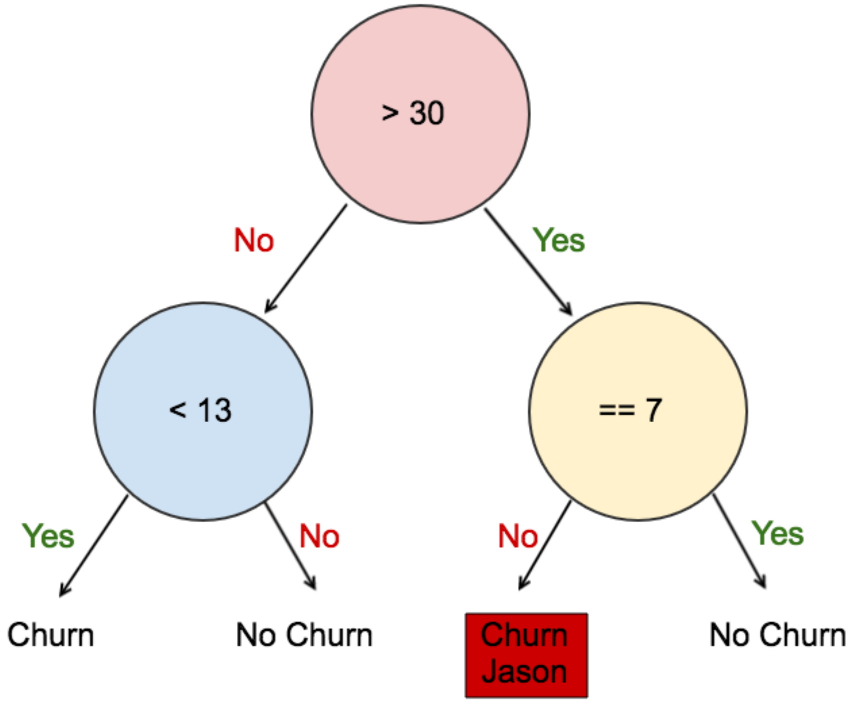
Prediction Time

- Jason's red feature value > 30



Prediction Time

- Jason's red feature value > 30
- Jason's yellow feature value != 7



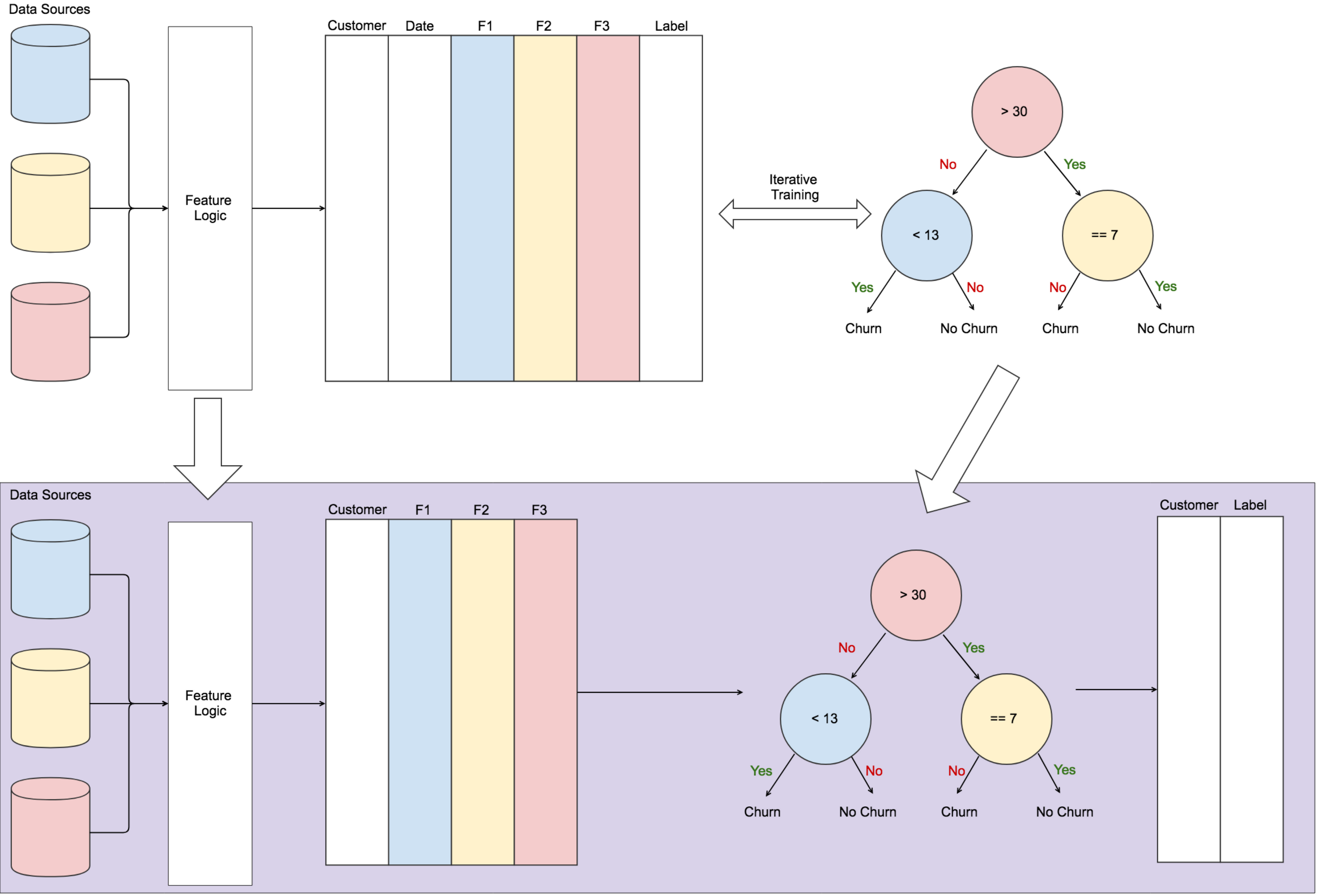
Prediction Time

- Jason's red feature value > 30
- Jason's yellow feature value != 7
- We predict Jason will churn

Moving to Production

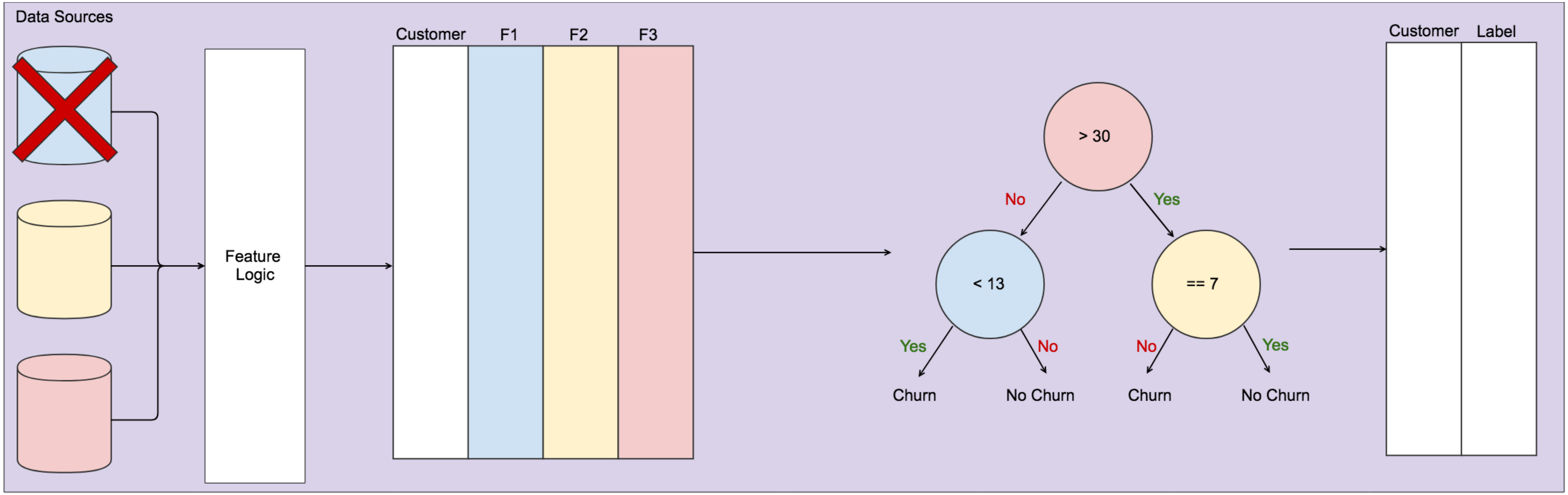
Moving to Production

- Training data - Historical
- Scoring data - New data
- Model and feature logic remains the same



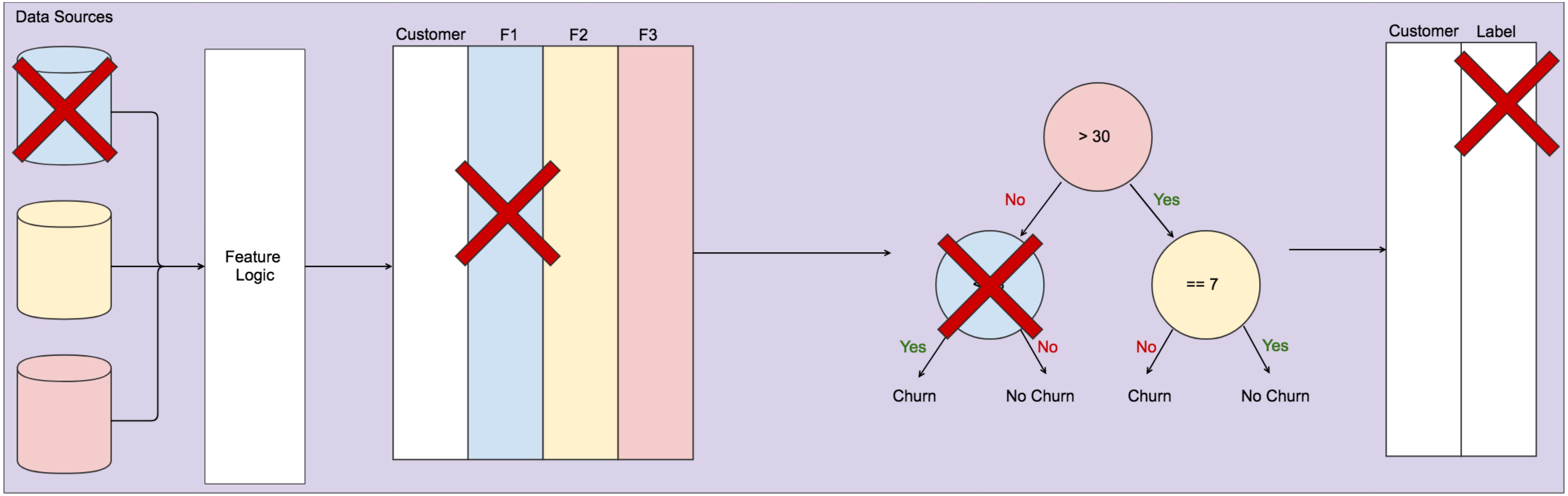
Data Issues

- Data ingestion lags (systematic) or failures (random)
- Data is incorrect



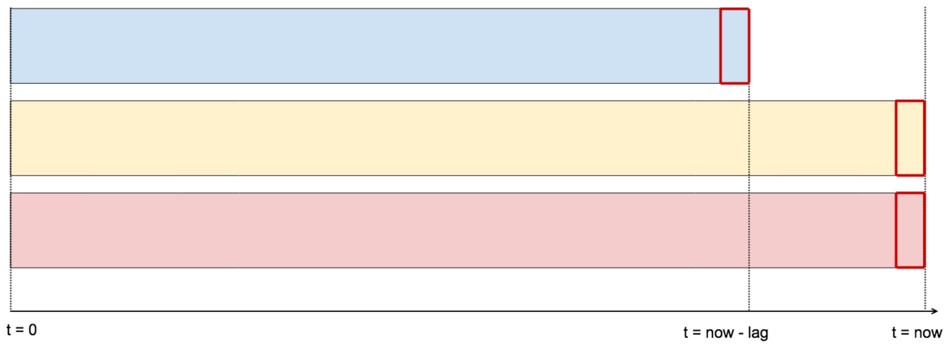
Data Issues

- Data ingestion lags (systematic) or failures (random)
- Data is incorrect



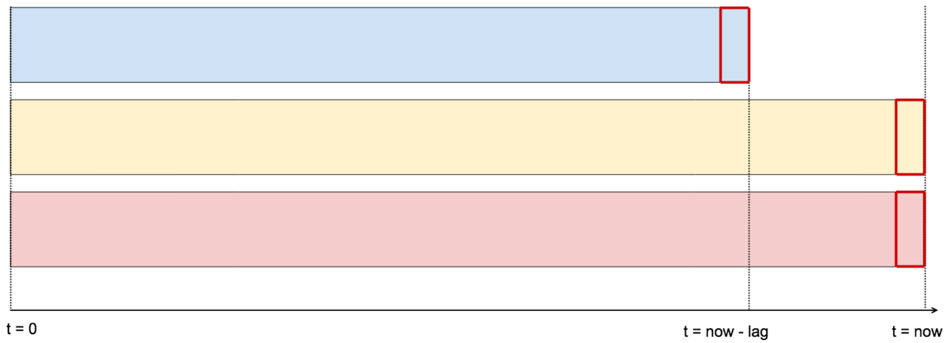
Before we change the system

- Fix the data source if thats an option
- Measure the importance of the feature in the model to quantify the cost/effort



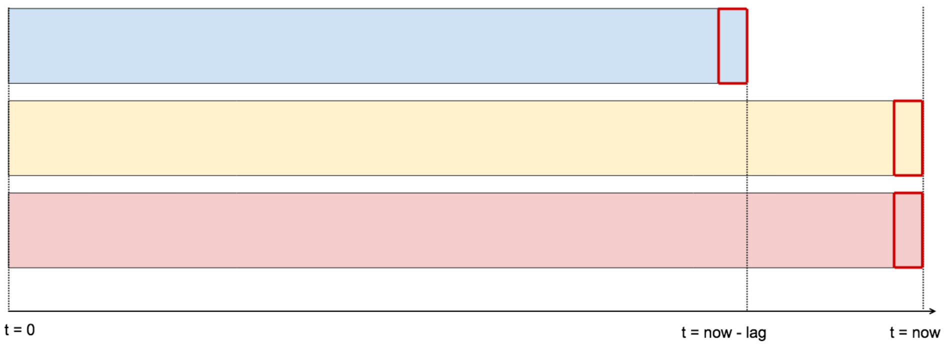
Use most recent data from each source?

- Will allow your system to function in the face of data lag



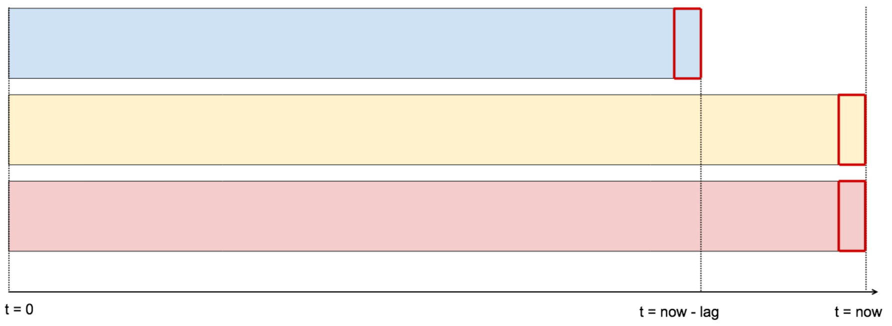
Problems?

- May introduce significant error into the model - especially if the lagged feature is highly predictive and changes quickly



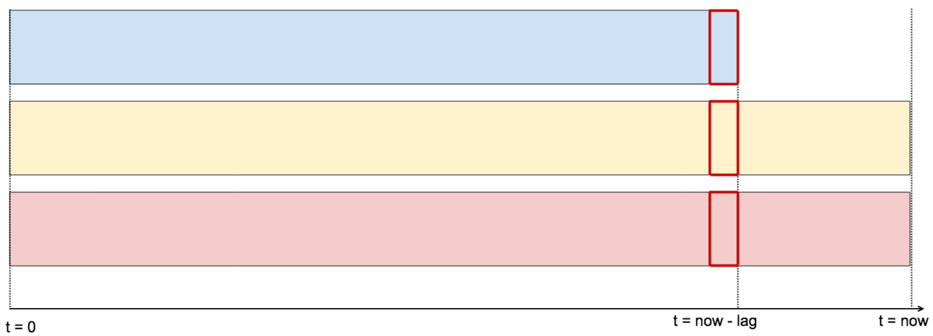
Retrain your model with this data lag?

- e.g. To model a 2 week lag on feature A: for each data point, get whatever the value of feature A was 2 weeks ago



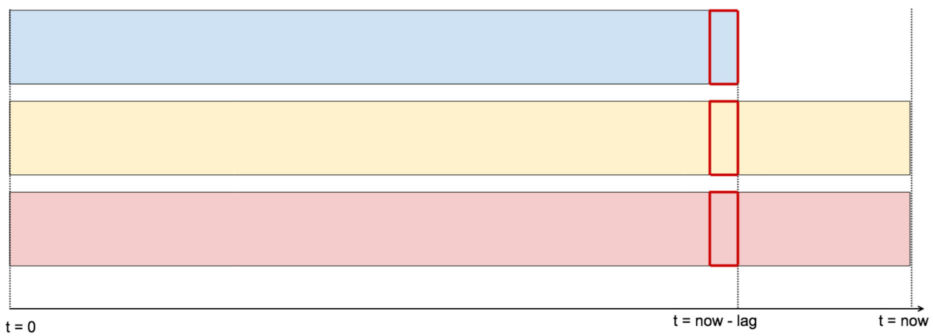
Problems?

- May lose too much information to be predictive
- End up tightly coupling your model to the data lag itself



Use the most recent consistent data

- Means the predictions will behave as expected

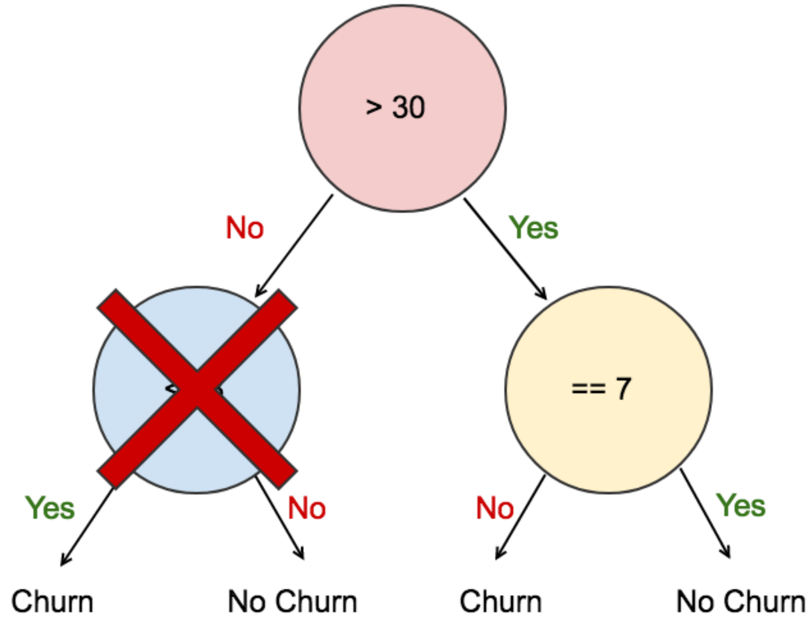


Problems?

- Predictions will be outdated equal to the slowest data source lag

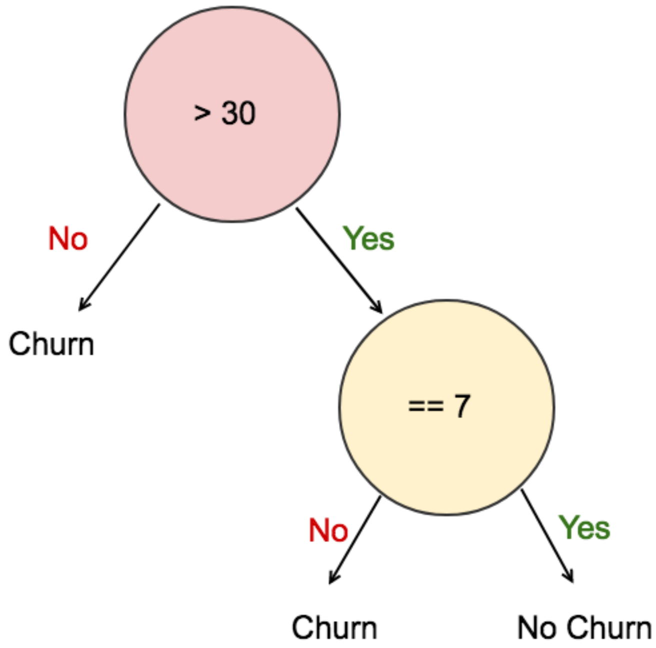
Build a model that gracefully degrades

- If we don't have certain data, we could aggregate the possible outcomes beneath that node (assuming tree model)



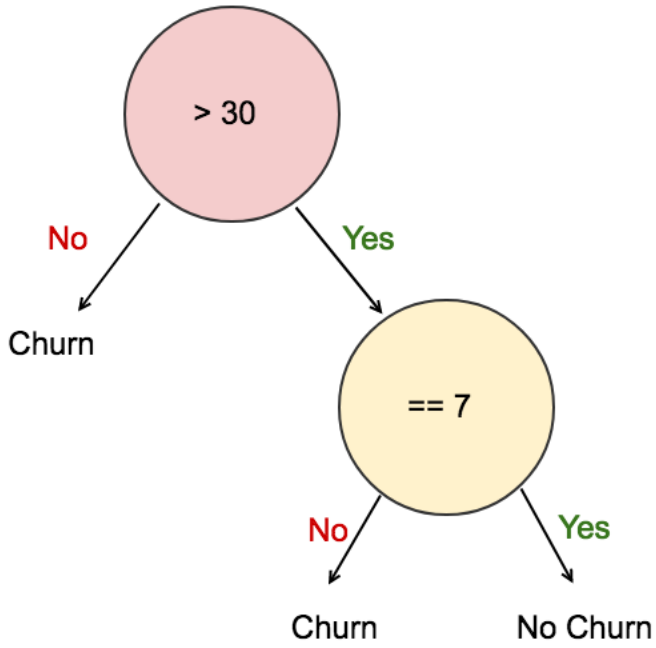
Build a model that gracefully degrades

- Average / Most Common label?

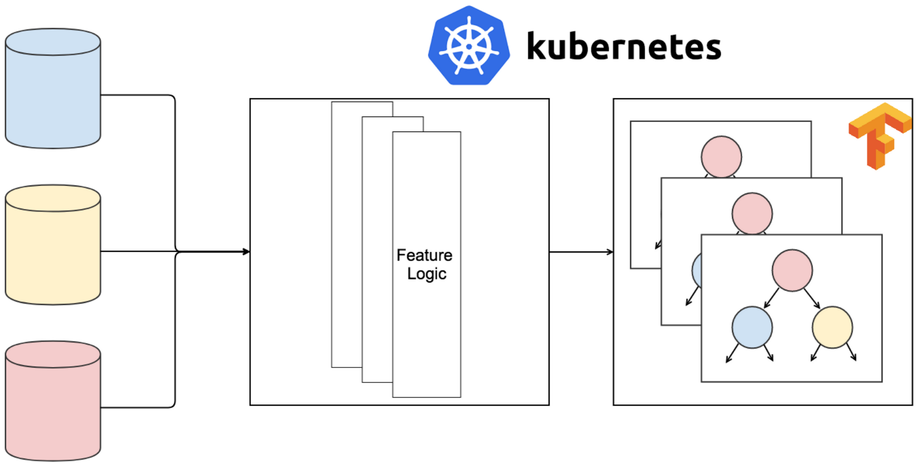


Problems?

- Model dependent custom code
- Expensive to build and maintain
- Will likely still degrade the model performance



Deployment

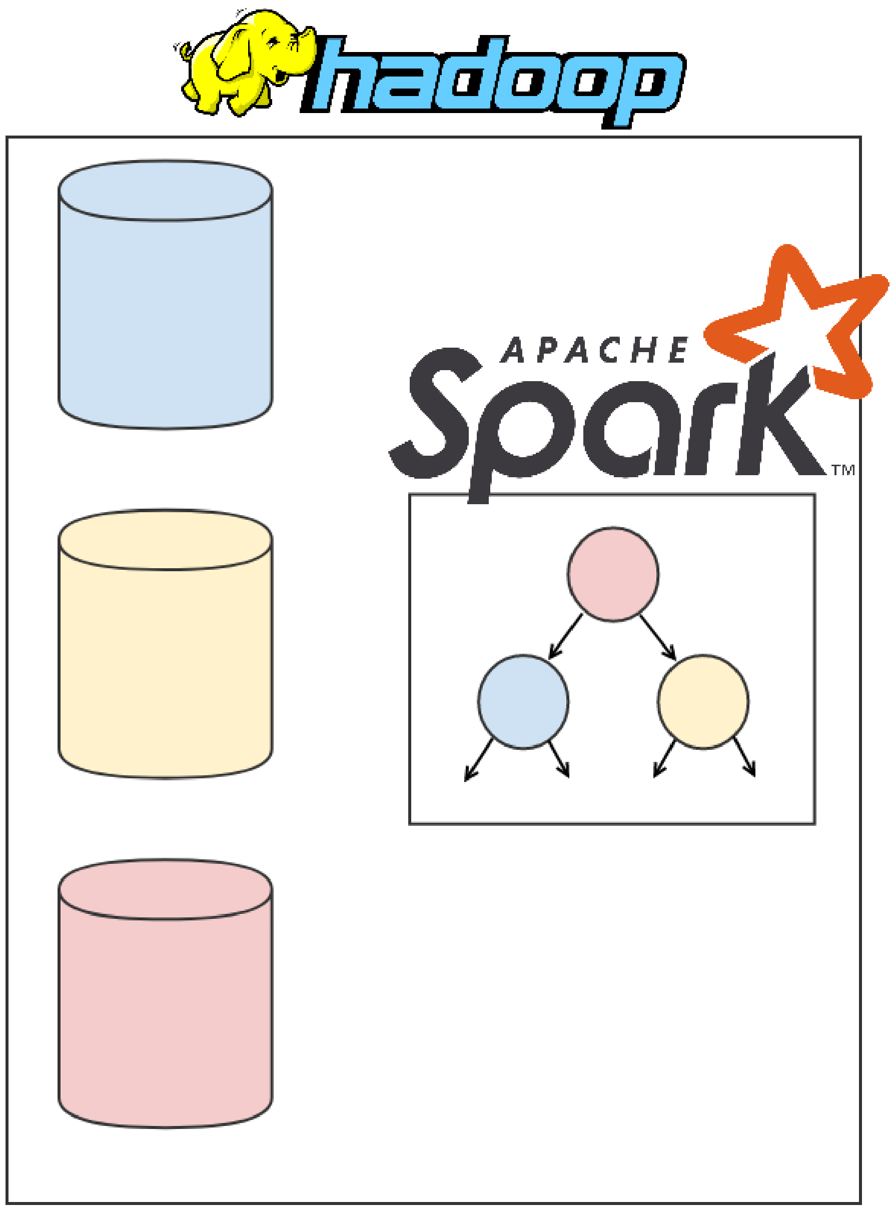


Small Data Deployment

- Containerise models + feature logic
- Send your data to your models
- Single machine scoring
- Resembles standard deployment models

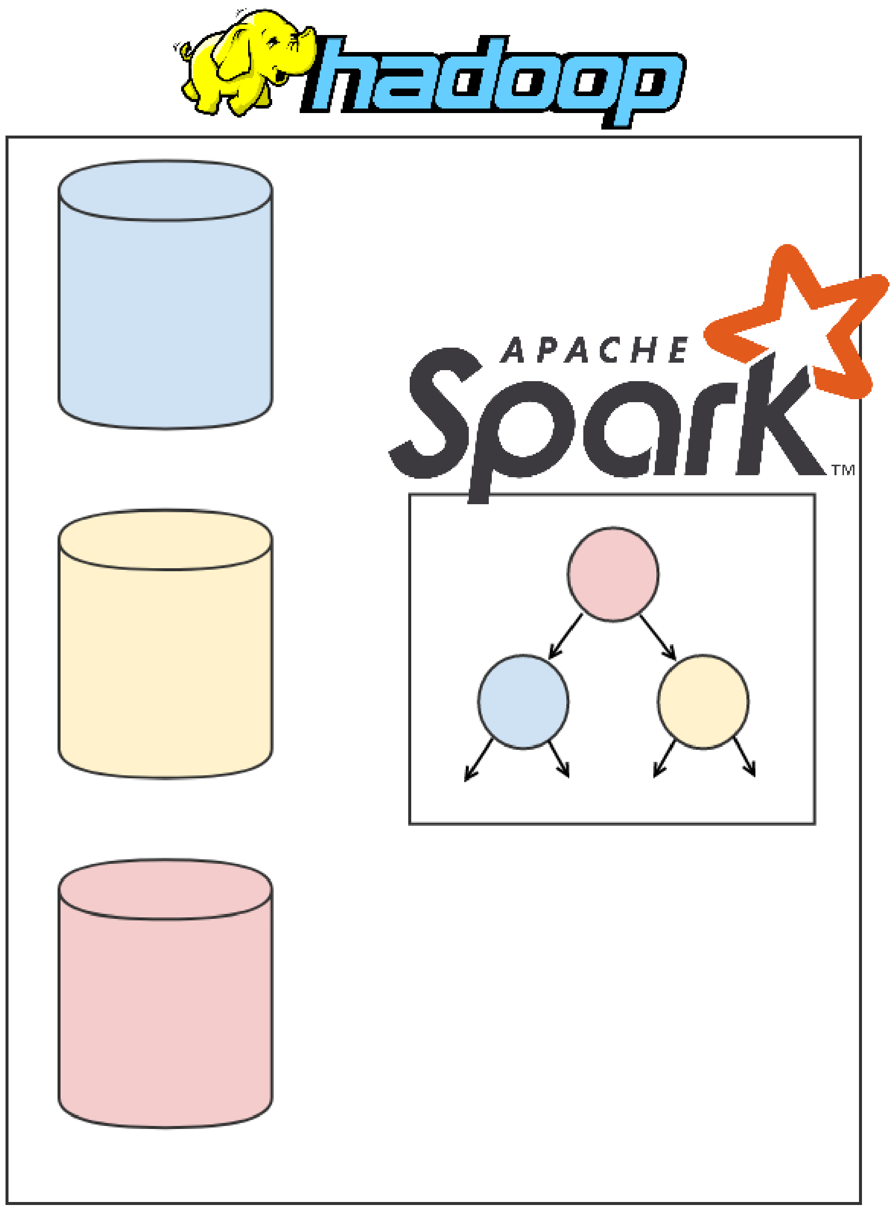
Problems?

- Doesn't scale to larger datasets without significant engineering overhead
- Huge amounts of data shuffled over the network = slower/more expensive scoring process



Big Data Deployment

- Distributed processing framework performs scoring (e.g. Spark)
- Send your models to your data



Big Data Deployment Options

1. Deploy by copying model files to HDFS/S3?
2. Deploy by embedding model in JAR file and using Spark Job Server?

Problems - Option 1?

- Copying files makes deployment lifecycle management harder
- Have to rebuild things that Kubernetes etc give us for free
  - Rollout deployments
  - Canary Deployments
  - A/B testing
  - Rollback

Problems - Option 2?

- Tightly couples scoring code to models
- We typically want to decouple our scoring code from our models so that they can evolve at different rates

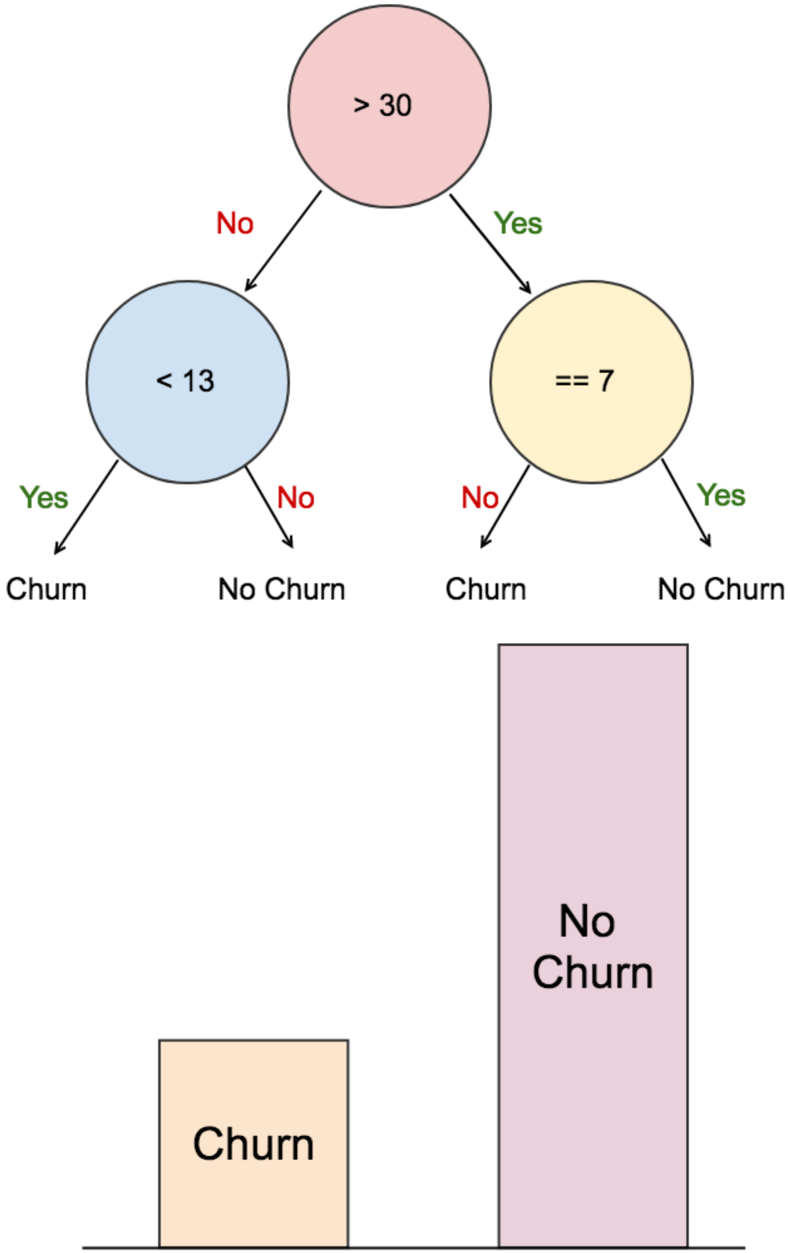
Future Solutions?

- Spark on Kubernetes?
- Manage data locality and application deployment through the same framework?

Metrics

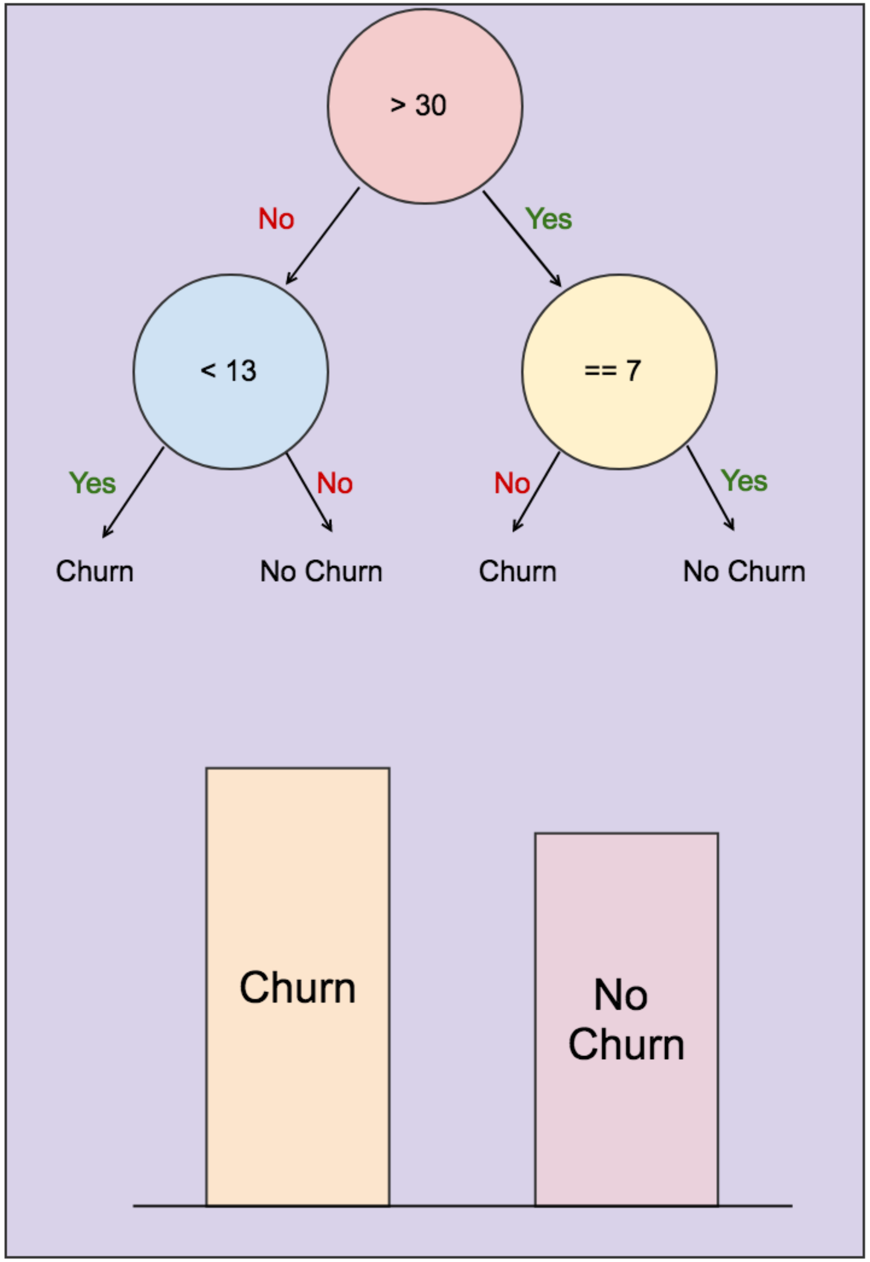
A few ML system metrics

- Data distribution
- Effectiveness in market



Data Distribution

Your training data will have some distribution of labels



Data Distribution

- In production, your data distribution may be significantly different
- This can happen over time as these systems tend to be dynamic

Possible causes

- Changes to the domain you're modelling
- Seasonality or external effects
- Changes to the customers themselves or the way the customers are using your service
- Problems with the data collection pipelines (corrupted data feeds etc)

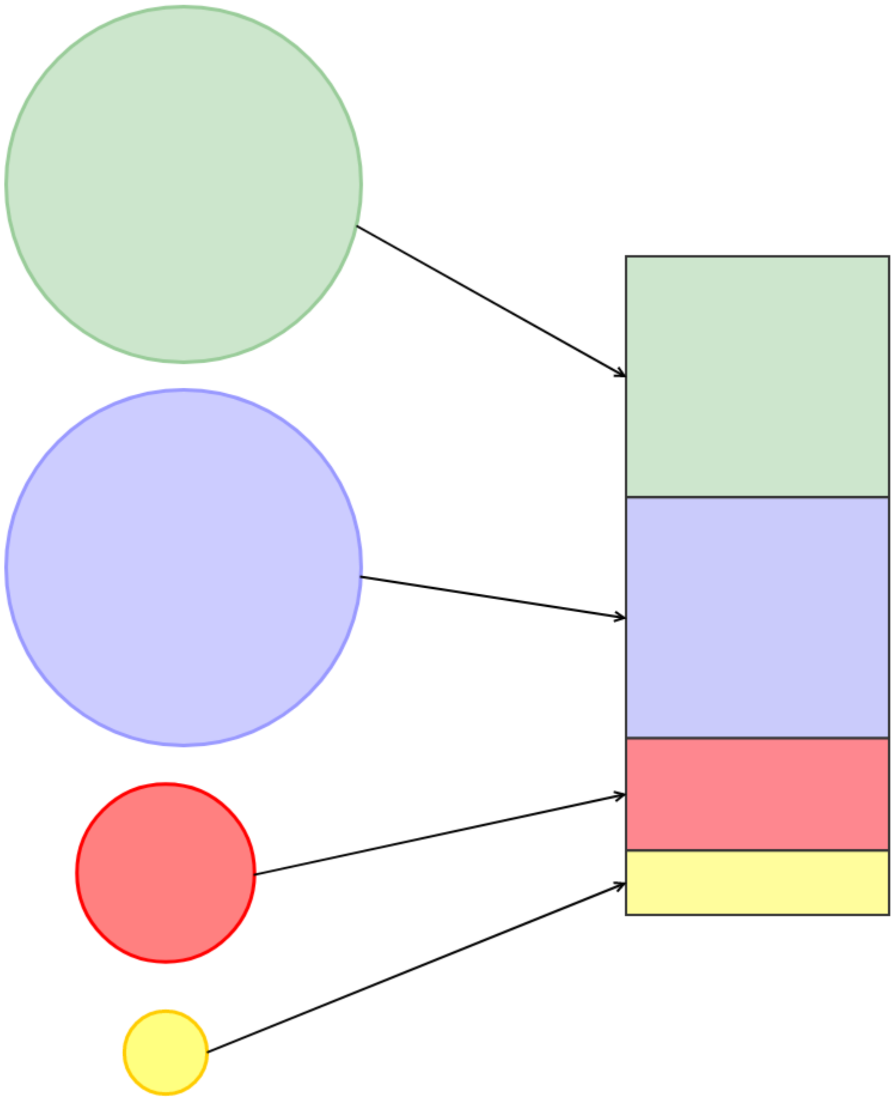
Effectiveness in market

- Production is the first real test
- Need to capture metrics to measure the effect of the model for its intended purpose
- Paves the road towards
  - Effective A/B testing
  - Incremental model improvement
  - Measurability of ROI

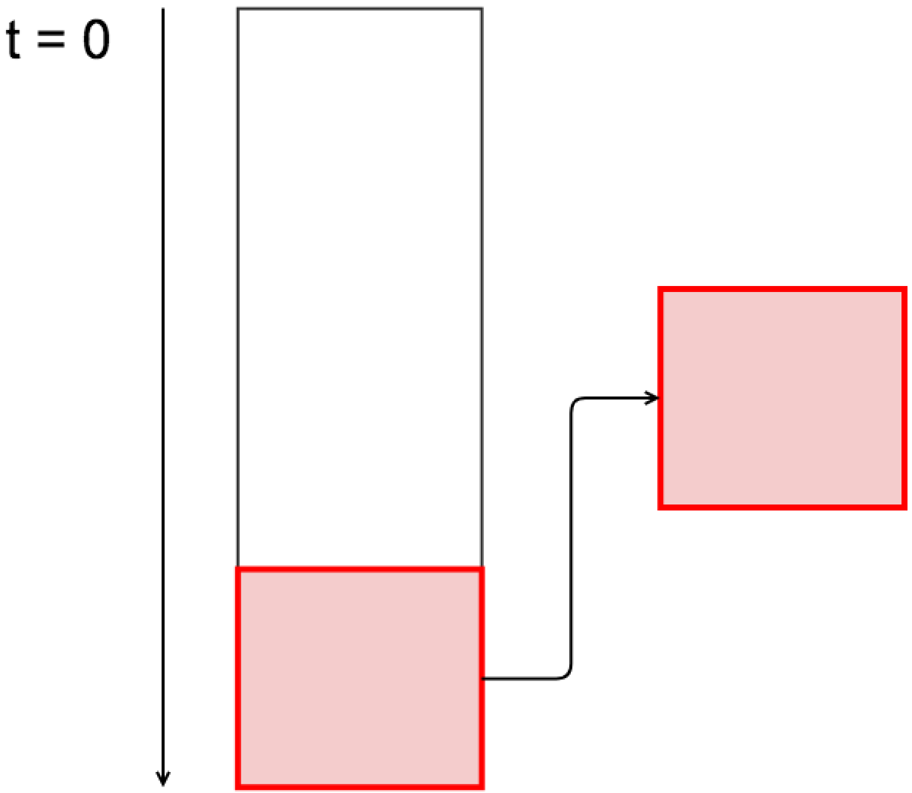
Big Data Iteration Speed

Training Models on Big Data is slow

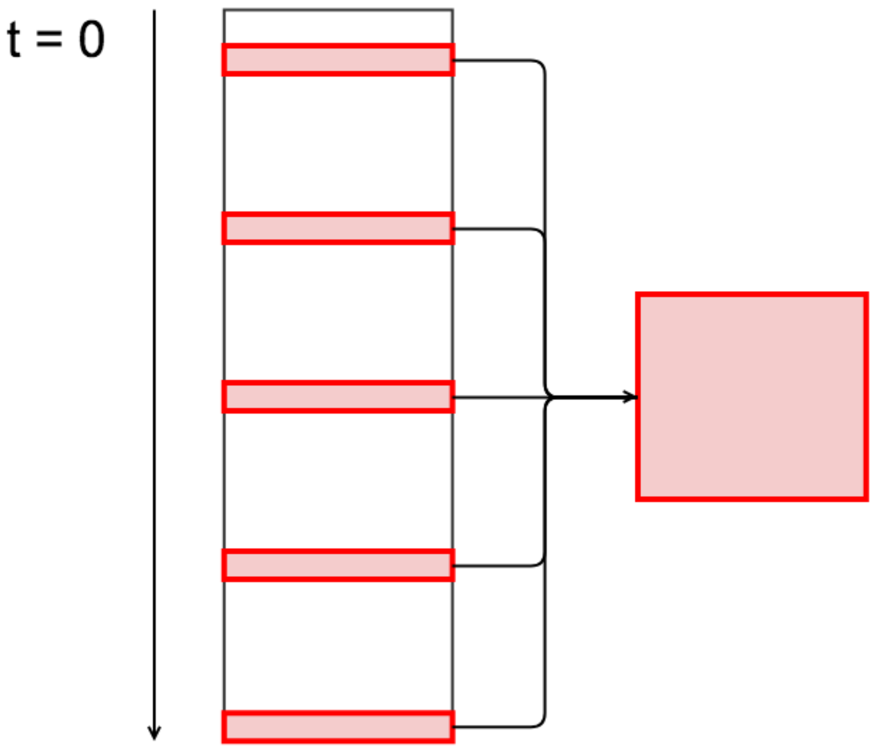
- Not all algorithms scale linearly as data/model complexity increases
- Hit computation/memory bottlenecks
- Number of hypothesis we can test is reduced
- Generating new features can become prohibitively expensive



Stratified Sampling



Sampling History



Customer Subset Sampling

Know where to spend your time

- Bad performance on training data = Bias Problem
  - Train longer
  - More complex model
  - Improve features
- Good performance on training data and bad performance on test set = Variance Problem
  - Get more data for training
  - Regularisation

Choice of Framework / Technology

- Modelling in R/Python and rewriting in production in Scala/Spark is an expensive process
- Choose a tech stack that allows engineers and data scientists to work together and productionise things quickly. Leads to faster feedback loops

What we've covered

- Data issues can be be a central issue to ML systems are require a lot of up front design thought
- There are several modes of deployment, each with their own tradeoffs for different scenarios
- Production is not the end of the process for ML models. Metrics are a fundamental part of enabling improvement and growth.
- Ways to improve iteration speed on ML projects

Thank you

Questions?