ROKT

Machine Learning in Production for Recommender Systems

2024 | Benjamin Akres

ROKT

About Rokt

My Journey

2017-2020	2021	2022	2023	2024
Studied Advanced Science and Arts at UNSW, majoring in Maths and Philosophy. Became interested in pursuing a career ML at this time.	Completed honours in Applied Mathematics with a focus on Data Mining and a thesis on clustering nodes in graphs.	Joined Rokt as a Graduate ML engineer. Worked on data centric and model centric improvements to CTR prediction models.	Progressed out of a graduate role. Worked on more complex projects such as unifying several specialised models into a Multi-Task model.	Working as a Senior ML Engineer on larger projects, such as expanding which parts of the transaction we can optimise and improving the auction logic.

ROKT

Rokt connects global ecommerce in the moment that matters most.

As the global leader in ecommerce technology Rokt acts as a trusted intermediary, helping companies seize the full potential of every transaction moment to grow revenue and acquire new customers at scale—from cart to confirmation page

2B+
Transactions powered by Rokt annually

$200M+
Investment in R&D to date

130%
YoY Platform Growth

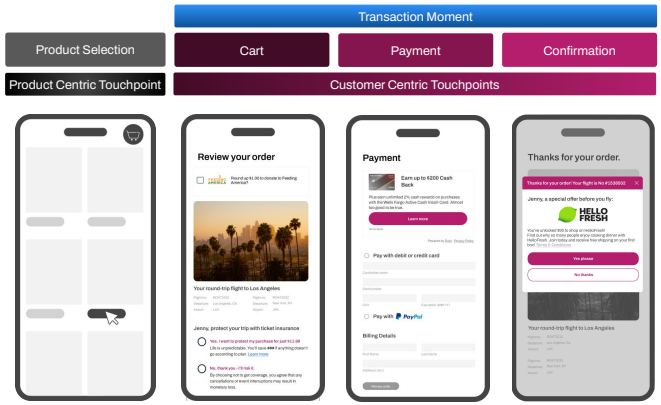
15
Global Markets

ROKT

Unlock the moments that matter most through relevance in ecommerce, from cart to confirmation

Leverage machine learning and customer relevance in the transaction moment to improve customer experiences and increase profit.

Demo



Upsell relevant products and services

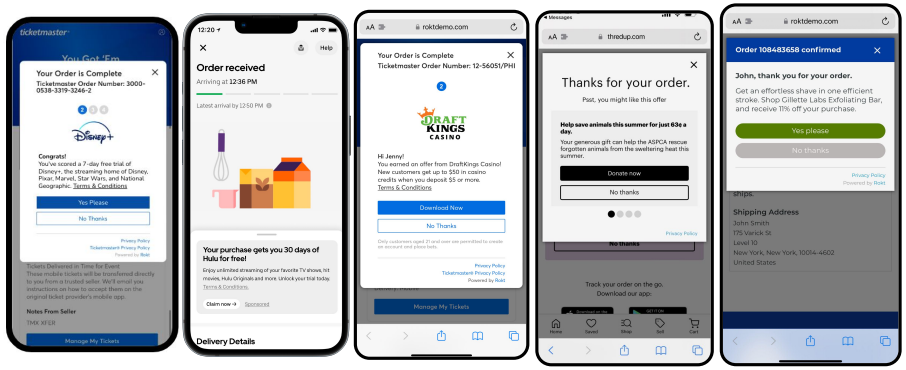
Transform a cost center into a profit center

Engage your customers with relevant offers

ROKT

Machine Learning at Rokt

Example Offers



ROKT

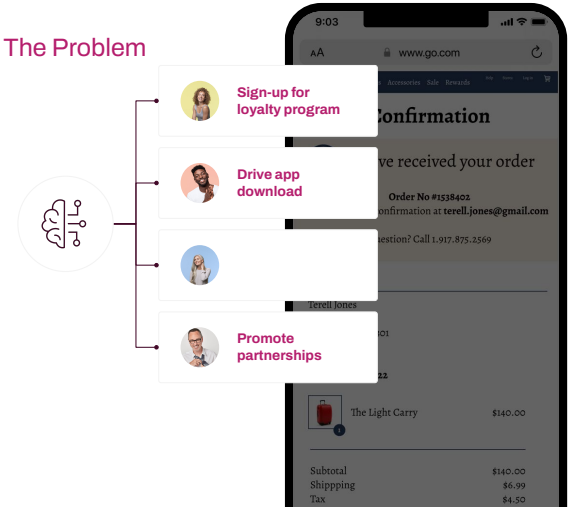
Generate incremental profit

Machine Learning at Rokt - The Problem

Decide the most relevant experience to show a user (e.g. Upsell, Payment offer, 3rd party Ad, etc.), as well as how it is shown.

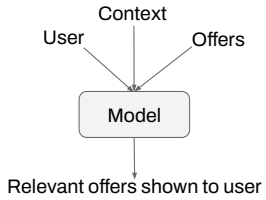
Models must:

- Perform at scale, processing millions of transactions per day with low latency.
- Be robust to changes in a dynamic environment.
- Deliver consistently on business objectives (e.g. revenue).

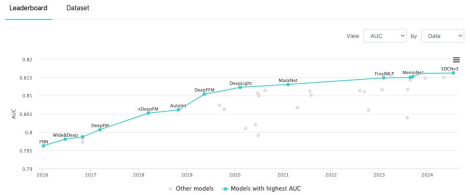


Machine Learning at Rokt - Ideas

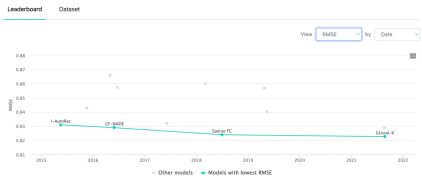
- Deep Learning is state of the art for recommendation and CTR prediction tasks.
- Active area of research with new ideas and model architectures being explored.
- Similar problem encountered in many large tech companies (Google, Facebook, Netflix, etc.).



Criteo CTR Prediction



Recommendation Systems on MovieLens 1M

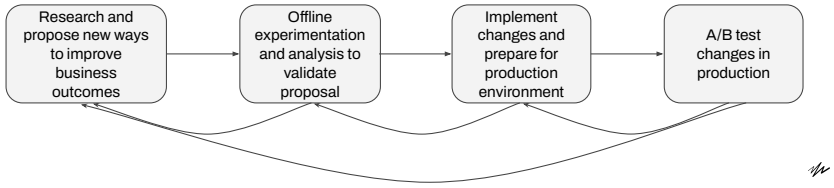


ROKT

Ad Recommendation System Design

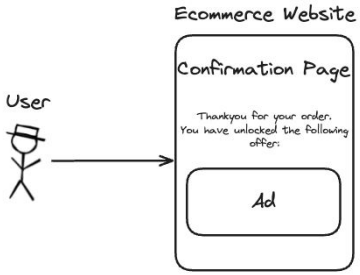
Machine Learning at Rokt - What Does an ML Engineer Do?

- EDAs (exploratory data analysis), offline experiments and online experiments adding new feature(s) to the model(s).
- Research new architectures and algorithms to improve models.
- Perform deep dive investigations and analyses on how our models are performing and interacting with other systems throughout the business.
- Improve the way our model's predictions are utilized by other systems.
- Propose and implement methods for optimizing different aspects of the product to improve business metrics.
- Ensure production ML systems are scalable, performant, reliable, low latency, robust, etc.



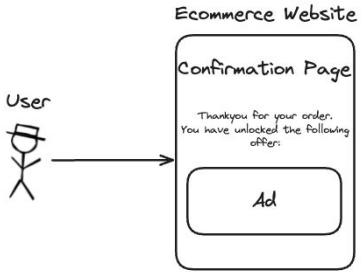
ROKT

Problem Statement and Requirement Gathering



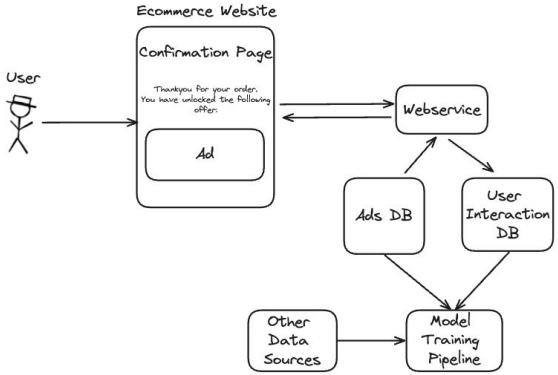
- Problem:
  - A user has just completed a transaction on an ecommerce website.
  - We can show them offers on the confirmation page to unlock additional revenue.
  - How can we decide which offer(s) to show?
- Requirement Gathering:
  - Latency requirements
  - Expected throughput, volume of traffic, spikes in traffic
  - Number of offers available to select between
  - Data availability:
    - User data
    - Advertiser data
    - Partner data
  - Success criteria:
    - Business metrics
    - Model metrics
    - User experience

Understanding the Business Problem

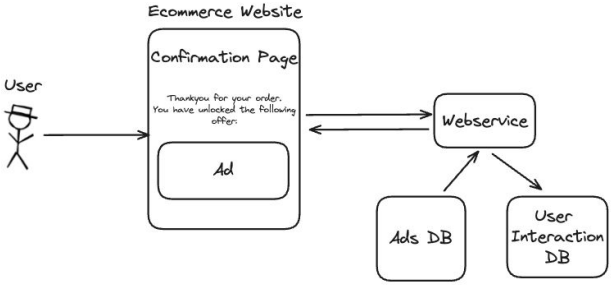


- Ecommerce Website Business objective: Maximise profit
  - We control the choice of Ad
  - Expected incremental revenue = P * CTR where P is the price the advertiser will pay per click and CTR is the click through rate for the Ad
  - Other factors to consider:
    - Impact on user repeat rate - maximising profit this impression should not come at the expense of long term profit
    - Infrastructure costs - if the cost of deciding which Ad to show is too high it will result in lower profit
- Advertiser Business objective: Maximise return on ad spend (ROAS)
  - We do not know the lifetime value of customers, but we can use the advertisers desired CPA (cost per acquisition) as a proxy
  - Possible objectives include scale and efficiency

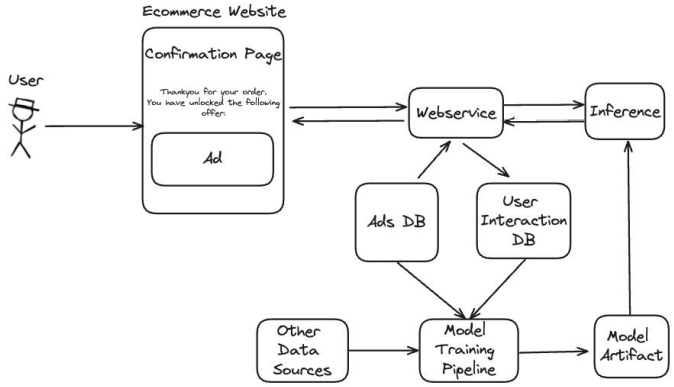
Model Training



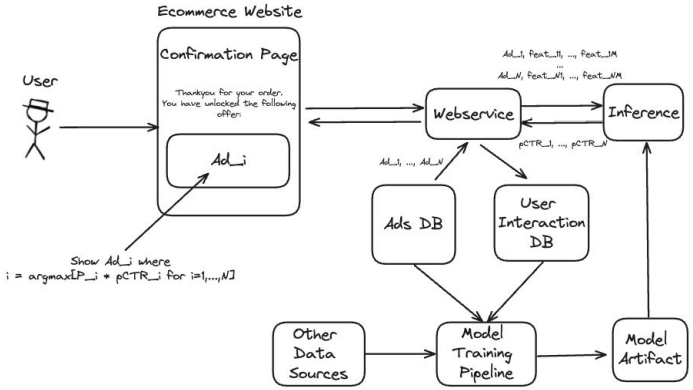
Data Collection



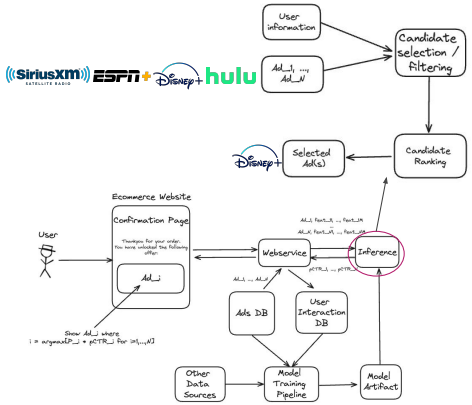
Realtime Serving



Solving the Business Problem

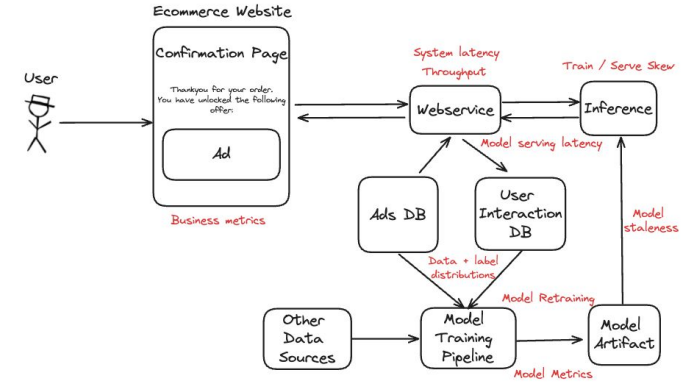


Two Stage Recommender Systems



- If we have too many candidates it may be expensive or slow to rank them all - in these cases we may want to cut down the number of candidates
- Tradeoff between latency and accuracy - predicting for all candidates may be more accurate but this comes with a cost
- Filtering methods:
  - Simple low latency prediction model
  - Content filtering (e.g. show offers similar to items in the users cart)
  - Collaborative filtering (e.g. show offers that similar users have interacted with, or show offers similar to offers this user has interacted with in the past)
  - Approximate nearest neighbour lookup for user and/or offer embeddings
  - Business logic / targeting

Monitoring



Modelling Approaches - Deep Cross Networks (DCN)

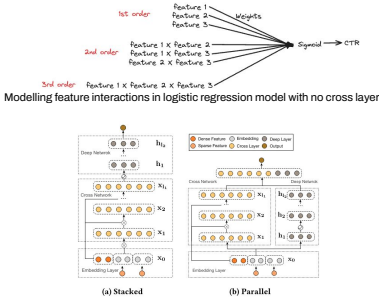


Figure 1: Visualization of DCN-V2. ⊙ represents the cross operation in Eq. (1), i.e., $x_{l+1} = x_0 \odot (W_l x_l + b_l) + x_l$.

Source: https://arxiv.org/pdf/2008.13535

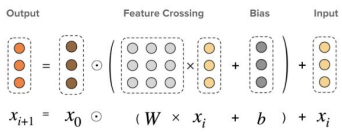


Figure 2: Visualization of a cross layer.

- Feature interactions are important in recommender systems
  - For example, offer A performs better than B on average, but performs worse than B on Android devices
- You can add cross terms to your model explicitly through feature engineering, but identifying the right crosses can be time consuming
- Cross layers allow you to explicitly model feature interaction terms
- Stacking cross layers models higher order feature interactions
- A toy example using a DCN architecture can be found here: https://www.tensorflow.org/recommenders/examples/dcn
- DCN based architectures are state of the art on CTR prediction benchmarks

Modelling Approaches - Multi-Task Learning with Multi-Gate Mixture of Experts (MMOE)

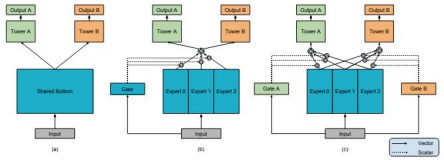
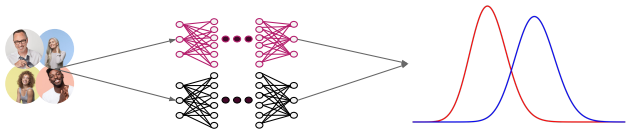


Figure 1: (a) Shared-Bottom model. (b) One-gate MoE model. (c) Multi-gate MoE model.

Source: https://dl.acm.org/doi/pdf/10.1145/3219819.3220007

- In some contexts we may require multiple predictions to be utilized by our recommender system (e.g. Click Rate, Conversion Rate, Cart Abandonment, user time on site, etc.).
- We can have different models for each of these tasks; however, if there are similarities between these tasks then using a multi-task model can lead to better results.
- In multi-task learning there can be tradeoffs between task specific relationships and inter-task relationships.
- "Each gating network can learn to 'select' a subset of experts to use conditioned on the input example. This is desirable for a flexible parameter sharing in the multi-task learning situation."
- Additional considerations:
  - Weighting the loss functions from each task
  - Defining primary task(s) and auxiliary task(s)
  - Ensuring loss for each task converges at same time.

A/B Testing



- **All major model changes are A/B tested to ensure they improve business metrics.**
- **New model releases are done in stages to minimize disruptions.**
  1. (Optional) Allocate traffic to the model in shadow deployment.
  2. Allocate <5% of traffic to confirm the model behaves as expected.
  3. Allocate 50% of traffic to measure business metrics and real-time performance.
  4. Allocate 100% of traffic and celebrate!
- **Challenges:**
  - How to split traffic (e.g. user based, session based, partner based, etc.)?
  - How long to run experiment for (ie. tradeoff between rapid iteration and confidence in results)?
  - How to measure statistical significance of results (e.g. different types of t-tests, Bayesian vs frequentist approach)?
  - How to analyse results (ie. changes over time, confounding variables, handling of outliers, Simpsons paradoxes, orthogonal experiments, etc.)?

ROKT

Modelling Approaches - Entire Space Multi-Task Model (ESMM)

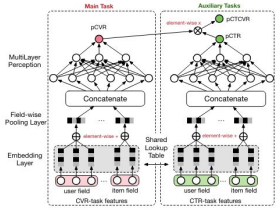


Figure 2: Architecture overview of ESMM for CVR modeling. In ESMM, two auxiliary tasks of CTR and CTCVR are introduced which: i) help to model CVR over entire input space, ii) provide feature representation transfer learning. ESMM mainly consists of two sub-networks: CVR network illustrated in the left part of this figure and CTR network in the right part. Embedding parameters of CTR and CVR network are shared. CTCVR takes the product of outputs from CTR and CVR network as the output.

Source: https://dl.acm.org/doi/pdf/10.1145/3219819.3220007

$$\underbrace{p(y=1,z=1|\boldsymbol{x})}_{pCTCVR} = \underbrace{p(y=1|\boldsymbol{x})}_{pCTR} \times \underbrace{p(z=1|y=1,\boldsymbol{x})}_{pCVR}$$

$$L(\theta_{cvr},\theta_{ctr}) = \sum_{i=1}^{N} l(y_i, f(\boldsymbol{x}_i;\theta_{ctr})) + \sum_{i=1}^{N} l(y_i \& z_i, f(\boldsymbol{x}_i;\theta_{ctr}) \times f(\boldsymbol{x}_i;\theta_{cvr}))$$

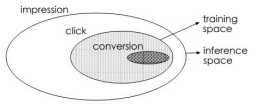
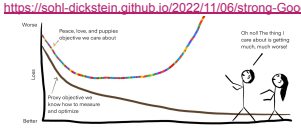


Figure 1: Illustration of sample selection bias problem in conventional CVR modeling. Training space is composed of samples with clicked impressions. It is only part of the inference space which is composed of all impressions.

- In recommender systems there are known relationships between the quantities we would like to predict - we can use this information when designing our model architecture
- Modelling CVR can be hard due to sparse labels and selection bias in the training set
- ESMM helps these problems by sharing feature embeddings with the click prediction task (helping with label sparsity) and allowing us to train on samples from the whole space of impressions (helping with selection bias)

Source: https://sohl-dickstein.github.io/2022/11/06/strong-Goodhart.html

Offline vs Online Performance

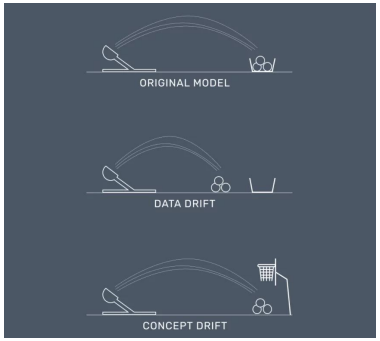


- **Train/Serve skew**
  - Problem: The model may produce different outputs on training and serving data.
  - Example: A feature is missing during serving, resulting in inaccurate predictions.
- **Alignment with business objectives**
  - Problem: Metrics used to measure model performance during training and evaluation do not always reflect performance on business metrics.
  - Example: A video recommendation model that is used to show videos with the highest click rate may unintentionally promote clickbait, hurting long term engagement.
  - A model that performs well on model metrics may not perform well on business metrics!
- **Solutions**
  - Ensure there are no discrepancies between data sources or feature engineering in online and offline environments (or ensure there is a single source of truth).
    - Observability
    - Skew detectors
  - Understand the business problem well enough to ensure the model is optimising for the right thing, and is being used in the right way.
  - A/B testing to ensure online model performance outperforms a benchmark or baseline.

ROKT

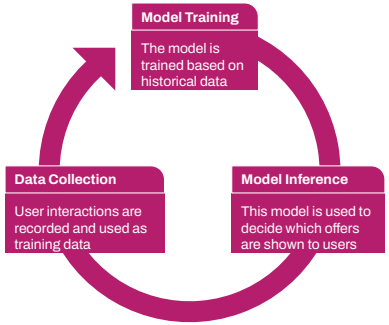
Model Drift

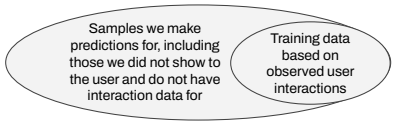
- **Concept drift**
  - Problem: The input-output mapping that a model has learned may change over time.
  - Example: A model which accurately predicted housing prices 10 years ago will perform poorly today as the relationship between features and price has changed (due to a general increase in prices).
- **Data drift**
  - Problem: The distribution of input data can change over time.
  - Example: A model trained to recommend clothes based on data collected during the summer may perform poorly during winter as it has not trained on winter data.
- **Solutions**
  - Observability
  - Scheduled retraining or online learning



Source: https://kdimensions.com/gallery-mlops/

Exploration vs Exploitation





Recommender systems create feedback loops; the choice of recommendation determines the data available for training.

Multi-Armed Bandit Problem

- If we always show what we think is the best offer (exploiting), then we will not have the opportunity to learn about potentially better offers (exploring).
- Need to strike the right balance between exploration and exploitation.
- Epsilon-Greedy is an example of a simple strategy to handle this (explore a small percent of the time, exploit otherwise).

Model Interpretability

Logistic regression KNN Shallow Neural Networks

Linear regression Decision trees Ensembles of trees Deep learning

More interpretable ← → More complex

More complex models can be less interpretable.

An advantage of tree based models is that they are easy to interpret.

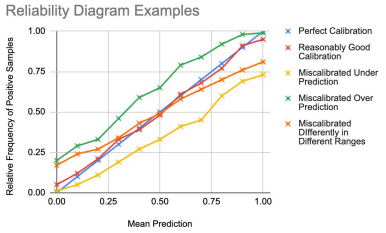
- It is easy to understand how decisions are made for an individual decision tree.
- When considering ensembles of trees we can gauge feature importance from factors like the information gain, or number of splits on a feature.

Deep learning models are hard to interpret.

- There are model agnostic methods for obtaining feature importance, such as feature permutation and SHAP, but these are often computationally expensive.
- Ongoing area of research.

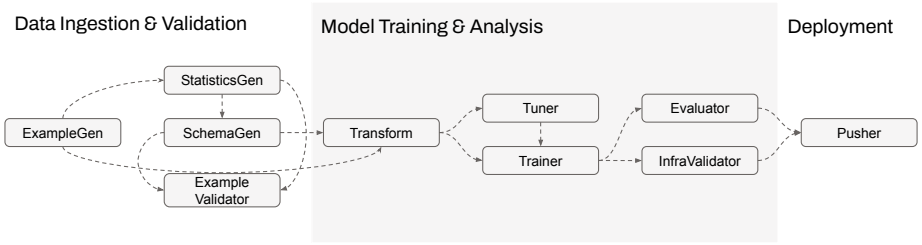
Model Calibration

- In some contexts it is not sufficient to have models which correctly identify the best offer to show; we may require our model predictions to be well calibrated probabilities.
- A classifier is considered well calibrated if a prediction of *x* indicates that the probability of the sample having a positive label is actually *x*.
- Some classification metrics (like AUC) ignore model calibration.
- Deep learning models can be overconfident in their predictions, particularly on unbalanced datasets (upsampling or sample weighting can exacerbate this).



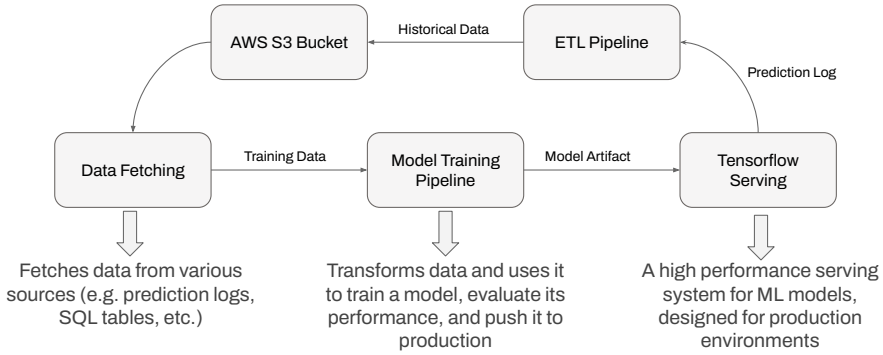
ML Infrastructure

Tensorflow Extended (TFX) Pipeline



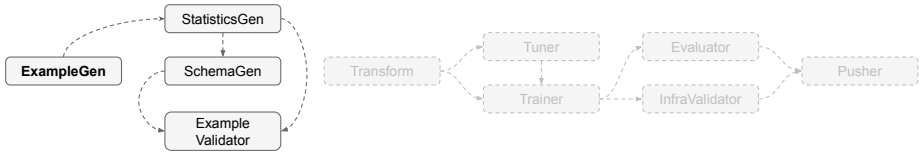
ROKT

Infrastructure Overview



ROKT

Tensorflow Extended (TFX) Pipeline - Data Ingestion and Validation



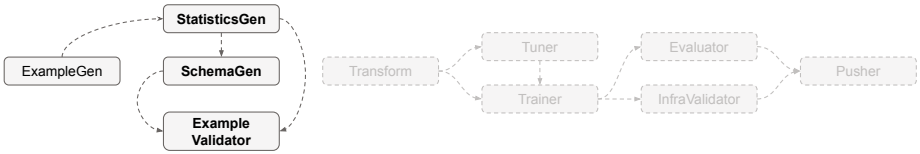
ExampleGen

- Converts various forms of data into tf.Record instances.
- Splits the dataset into training and evaluation sets.

- Considerations:
  - What sort of evaluation split would we like? Random, datetime, other?
  - Do we need an evaluation set in production? Helpful for monitoring performance, but results in less training data.

ROKT

Tensorflow Extended (TFX) Pipeline - Data Ingestion and Validation

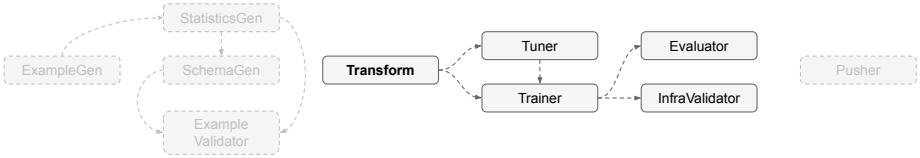


StatisticsGen and SchemaGen
- Generate statistics and pre-transform schema for the dataset.
- Used for monitoring, and as input to transform step.

ExampleValidator
- Detects and reports anomalies in the data (e.g. is any data missing, is the schema valid, are there features not present in the schema, etc.).

ROKT

Tensorflow Extended (TFX) Pipeline - Model Training and Analysis

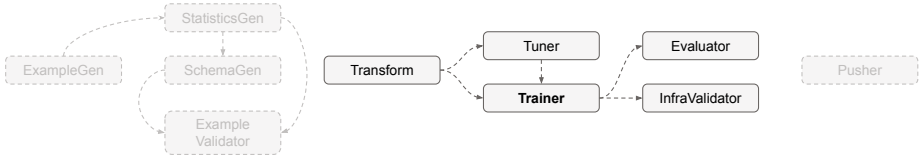


Transform
- Performs feature engineering on the dataset.
- The transformation is packaged together into the model artifact to ensure the same transform is applied during training and serving.

- Considerations:
  - How to encode features of different types?
  - How to handle null or missing values?

ROKT

Tensorflow Extended (TFX) Pipeline - Model Training and Analysis

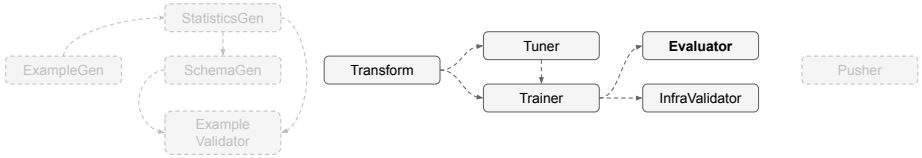


Trainer
- Responsible for the actual model training - this is where you define your model architecture, optimizer, etc.
- Enables data scientists to move their modeling result directly to production.

- Considerations:
  - What model architecture to use? What optimizer to use? What loss function to use?
  - How to prevent overfitting (regularisation, early stopping, etc.)?

ROKT

Tensorflow Extended (TFX) Pipeline - Model Training and Analysis

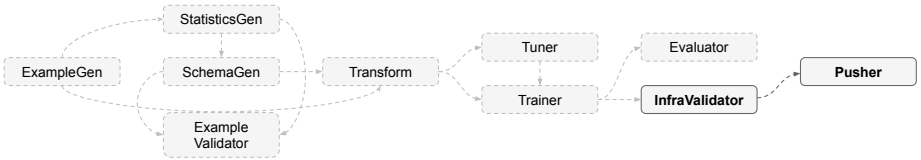


Evaluator
- Evaluates model performance using pre-defined metrics.
- Can evaluate on different slices of the data to help detect bias and ensure performance can generalize.

- Considerations:
  - What evaluation metrics are appropriate? Do these correlate with business outcomes?
  - Are there any important slices of data that need to be considered?

ROKT

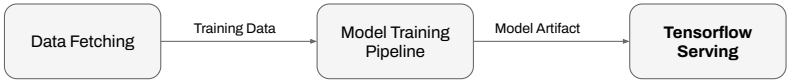
Tensorflow Extended (TFX) Pipeline - Deployment



InfraValidator
- Checks if we can feasibly run the model on the serving infrastructure.
- Tests if model can be successfully loaded and queried before pushing to production.

Pusher
- Checks that we get the expected outputs from Evaluator and InfraValidator.
- Tags a version to the model and pushes it to the serving infrastructure (TFS).

ML Serving Infrastructure



Tensorflow Serving
- Scalable and high-performance serving for ML models.
- Can handle high throughput and low latency requirements.
- Designed for production environments.
- Monitors new model pushes and gradually allocates traffic to new models in order to ensure a controlled and smooth transition between model versions.

Infrastructure Summary

In summary
- Single source of truth for all logic involved in mapping data to predictions.
- Observability on every stage of the training pipeline from inputs to outputs.
  - Monitor input data and prediction statistics.
  - Detect any anomalies.
  - Automatic checks to prevent bad models from being pushed.
- Robust real time serving infrastructure

Rokt is hiring!

Great perks:
- Competitive package with stock options.
- Lots of perks (level-up allowance, health and wellness budget, daily lunches, etc.).
- Annual travel for Global Kick Off event.
- Opportunity to work with a talented team of ML Engineers and Data Scientists.
- See jobs at https://apply.workable.com/rokt.

Thank you.

ROKT