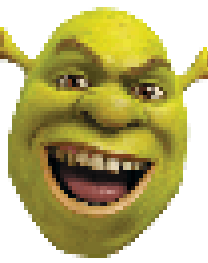




# Shrekommender System

Team 11





01

## Data

---

What data was used and how it was collected.

03

## System

---

The full pipeline and tools used.

02

## Models

---

Overview of used models.

04

## Reflection

---

Future improvements.





**DATA**



# Data Collection Pipeline

We collected raw data from the kafka stream API, then per-processed them into the following 4 files.

## Rating

Contains the interaction matrix between users and movies.

## Users

Meta data about each user, useful in building demographics.

## Movies

Movies meta data such as: genre, director, release year, part of a series, etc.

## Watch-log

How much did the user watch a movie for in intervals of 1 minute.





# Data Challenges, Pipeline, and Fairness Challenges

- Early struggles with limited and bad data
- Containerized ingestion and preprocessing in M3
- Fairness audit on gender
- Bad or sparse data silently degrades model performance

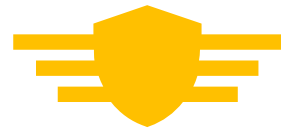
Metric	Score	Threshold	Status
Statistical Parity Difference (SPD)	0.0000	$ \text{SPD}  < 0.1$	PASS
Disparate Impact Ratio (DIR)	1.0000	$\text{DIR} \geq 0.8$	PASS
Diversity Parity	0.4286	$\text{Ratio} \geq 0.8$	FAIL
Personalization Parity	2.83pp	$\text{Diff} < 5\text{pp}$	PASS

Overall Score: 3/4 metrics passed (MOSTLY FAIR)



# Model





Learn low dimensional representation for users/movies by factorizing the interaction matrix



Alternate between fixing one vector and solving for the other by using least square



Great for implicit feedback and is a standard for big recommendation systems



Needs a complementary method for new users such as: quiz, popular movies, etc



# ALS: Alternating Least Squares Model



Learn low dimensional representation for users, movies, and side features. combining coll & cont in one model



Great for sparse data settings where good meta data are available



Optimize a logistic loss and updating all 3 embedding via gradient based methods



Can recommend to new users from their meta data alone



# LightFM Model



# System





# System overview

## Containerization

3 containers for data, ALS, and LightFM. with K8 to manage all of them

## Monitoring

Prometheus + Grafana stack is used to monitor availability and performance of models

---

## Continues training

Retraining and deployment are run automatically on a fixed schedule

## A/B testing

K8 load balancer split traffic between the models

# Provenance

We explicitly version both models and data and propagate this information through the online recommendation API so that every prediction is traceable.



## Model

For model information about the container, model version, training time, data version, etc are kept.



## Data

Statistical info about data, ingestion date, as well as paths for processed files are used for reference

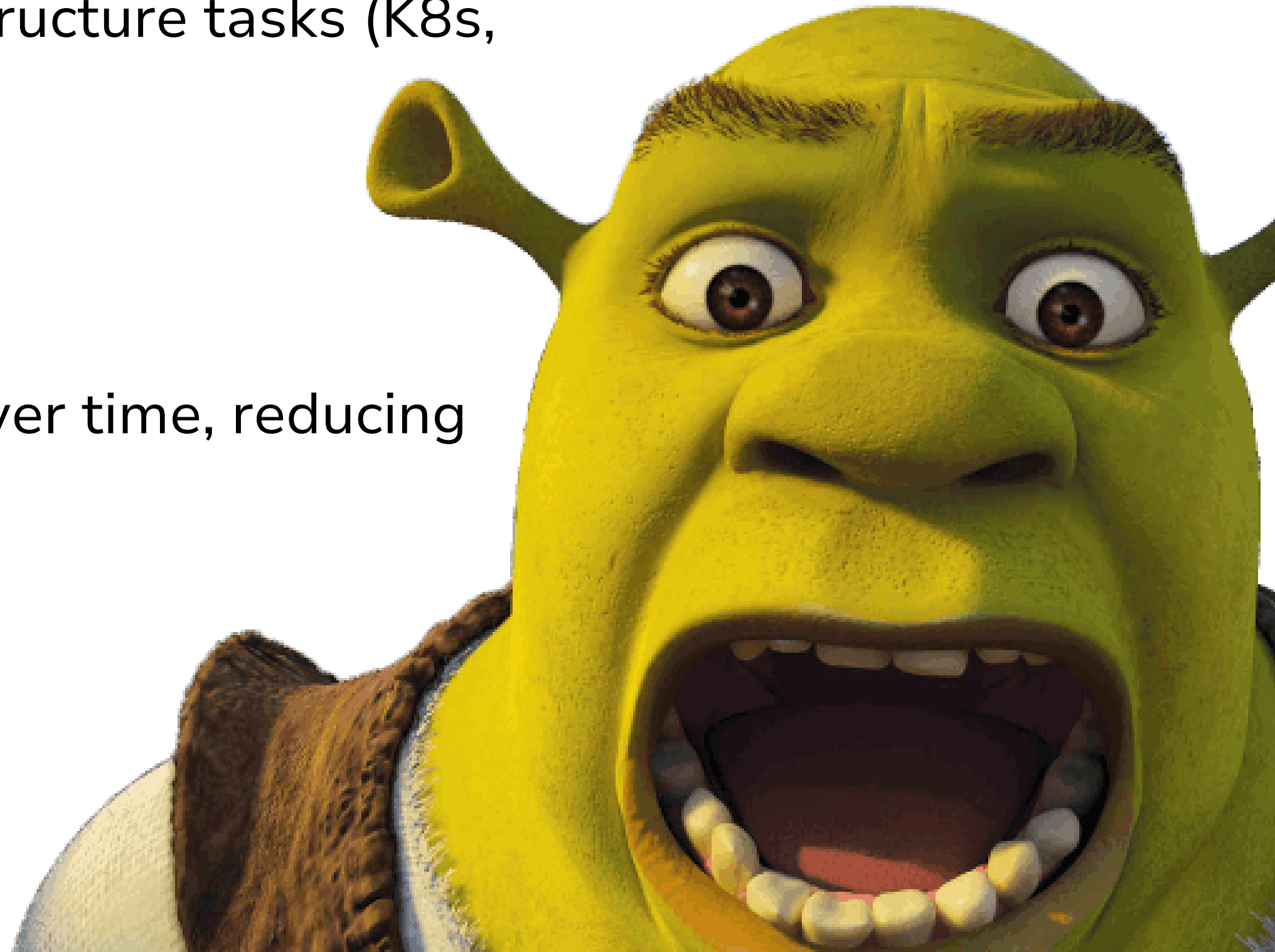


# Reflection



# Key Challenges

- Underestimated complexity of infrastructure tasks (K8s, monitoring, data pipeline setup)
- Small team size, more workload
- PR reviews became less consistent over time, reducing shared system knowledge



- Enforce a stricter PR review process to maintain shared ownership
- Use time-boxing to better manage task complexity
- Increase pairing on complex components to avoid knowledge silos
- Set up basic automation (CI, linting, monitoring) earlier in the project

# Future Improvements



# SHERKEMMONDER SYSTEM



Thank you for your time