asana

Insights from building enterprise-grade **ML Experiences** at Asana

Databricks Data + AI Summit 2024

11 June 2024

About us

Sarang Gupta

ML Data Scientist

Zhan Shi

ML Infrastructure Engineer

Agenda

1 Introduction

Asana and the Work Graph ®
AI & ML Themes and Capabilities

2 Project Recommendation System

Key Insights and Learnings

3 ML Infrastructure

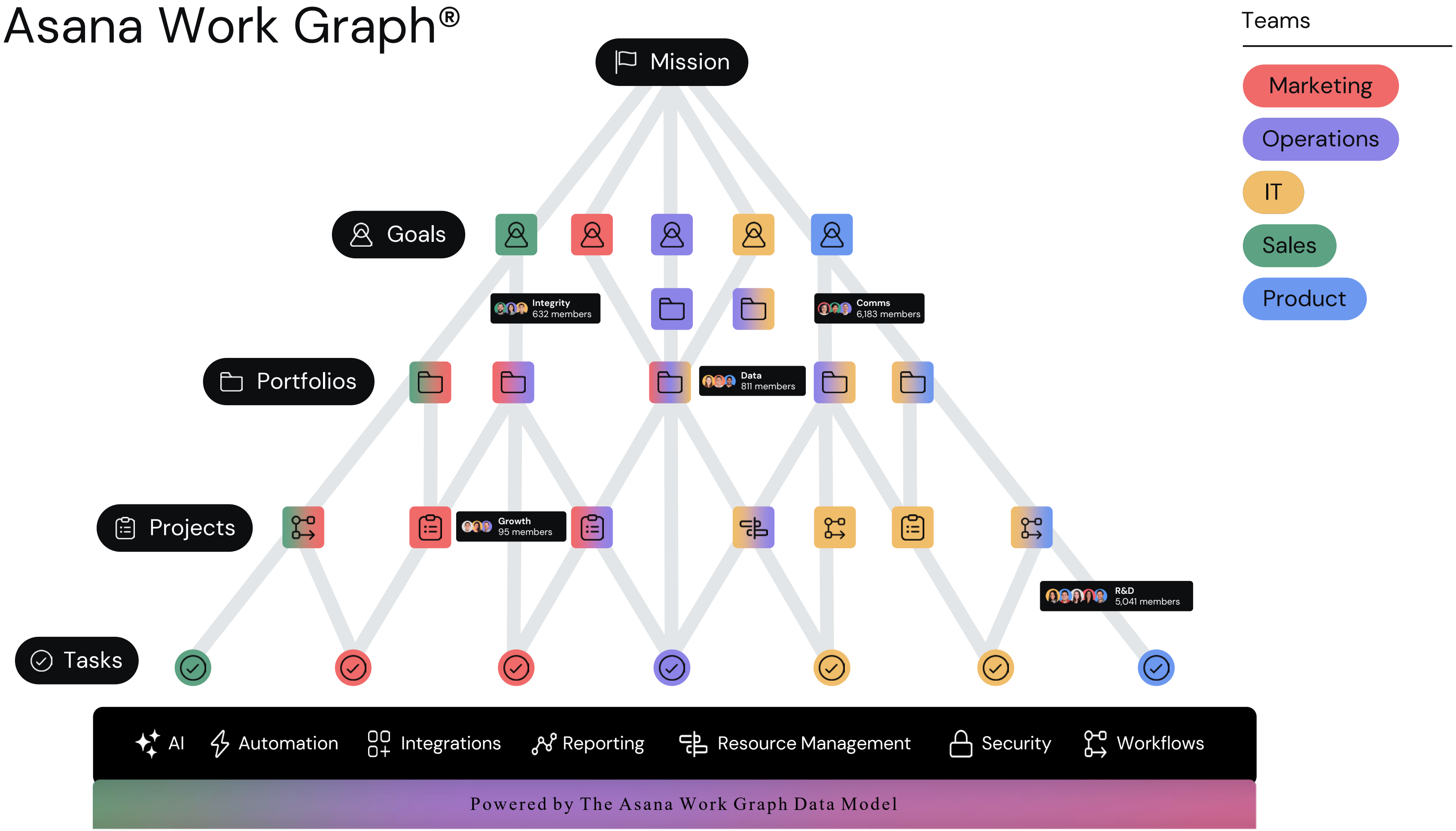
Productionization

asana

Introduction

Asana and The Work Graph®

The Asana Work Graph®



AI/ML at Asana

Capabilities and themes

Pattern recognition	1. Ranking 2. Classification 3. Detecting anomalies	**ML/DL on structured data**
Recommendation and prediction	4. Recommendations 5. Predicting future behavior	**ML/DL on structured data**
Understanding and using natural language	6. Extracting meaning from text 7. Hypothesization 8. Intent unwinding 9. Conversing in text 10. Summarization / translation	**LLMs**
Reasoning, learning and autonomy	11. Reasoning and making decisions 12. Reasoning and acting 13. Learning and durable memory 14. Chaining and composability	**LLMs**

Smart Rankings

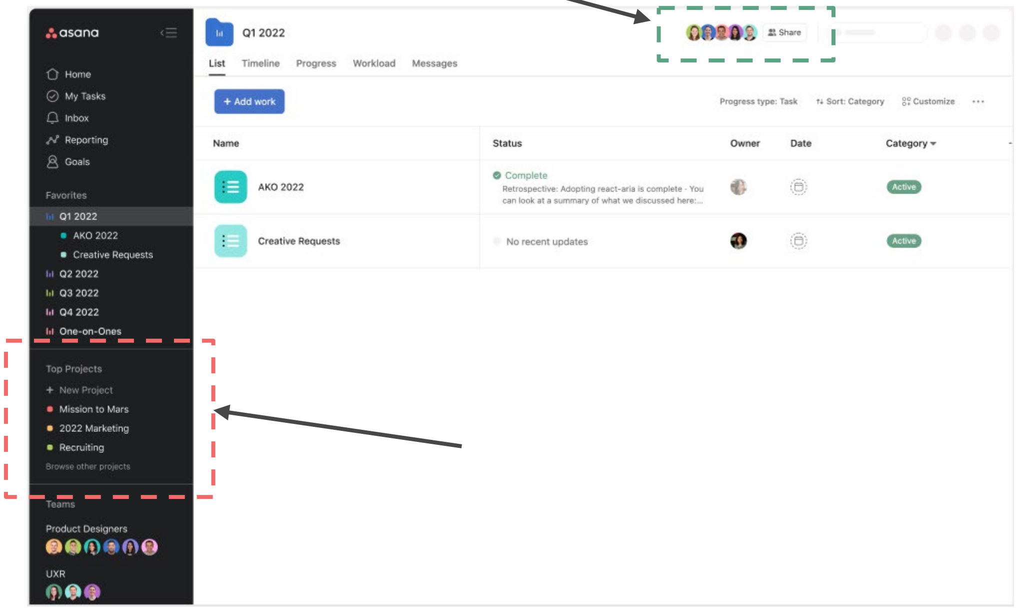
Helps users to navigate to their work easily

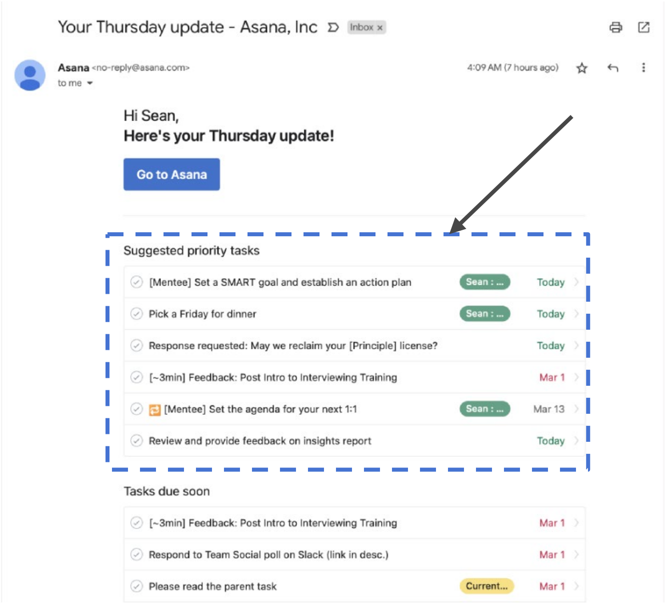
We use smart ranking to help our users to **navigate between their work seamlessly** across Asana

Project Rankings

Facepile Rankings

Priority Tasks in Emails





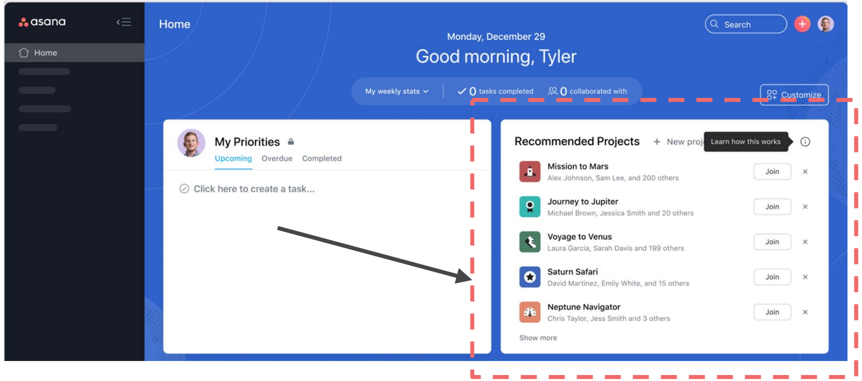
Smart Recommendations

Helps users to discover relevant work happening across their org

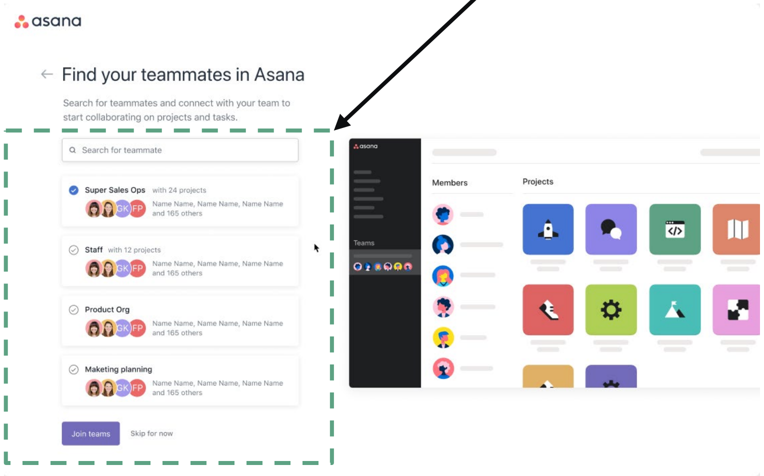
We use smart recommendations to help our users to **discover relevant work** content across Asana

Machine Learning models analyze users’ interactions and their work patterns to ensure is recommendations tailored to maximize their productivity and engagement in Asana

Project Recommendations



Team Recommendations



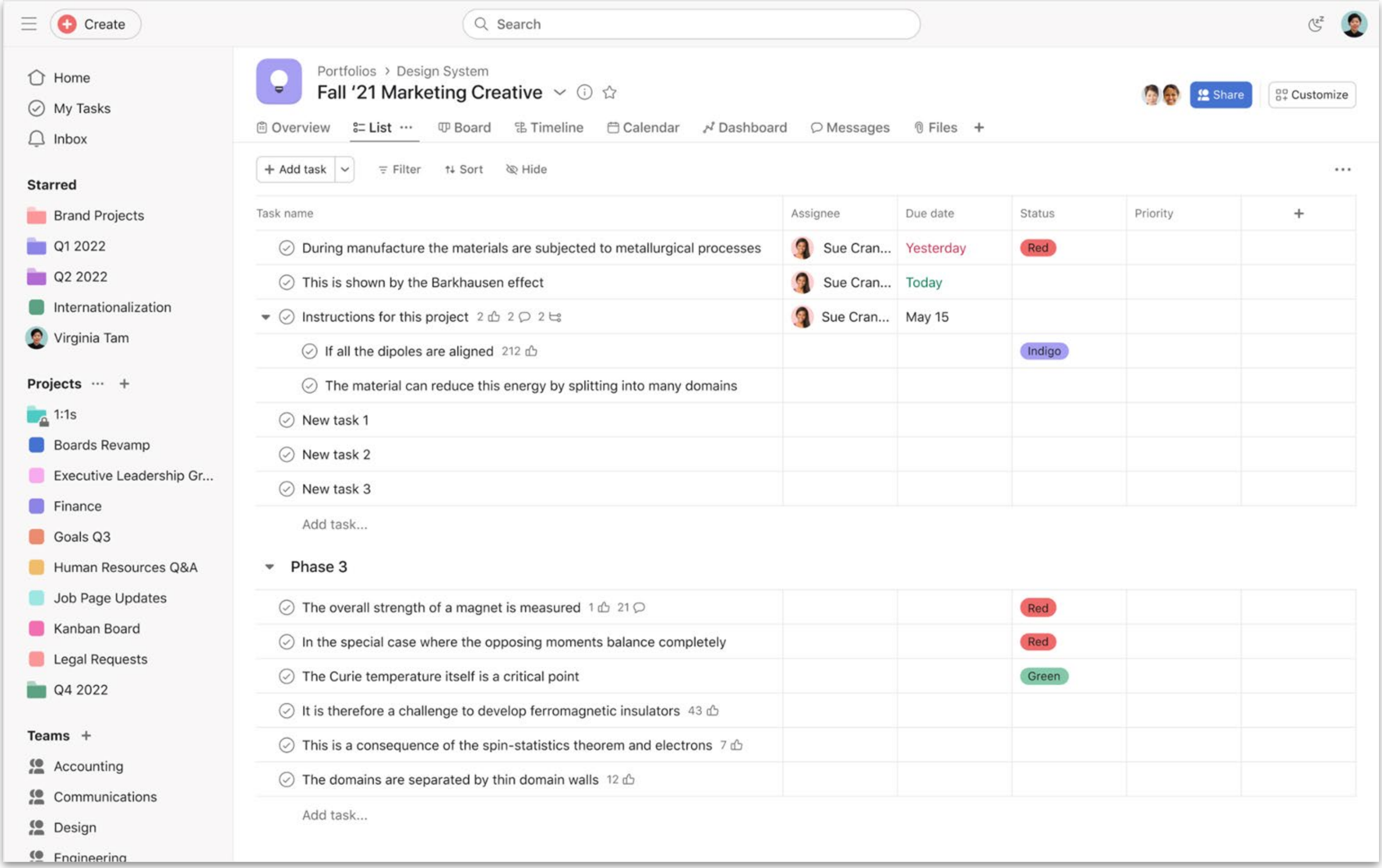
Chapter 2

Recommending Projects

User Interface and Modelling

Projects

User Interface



asana

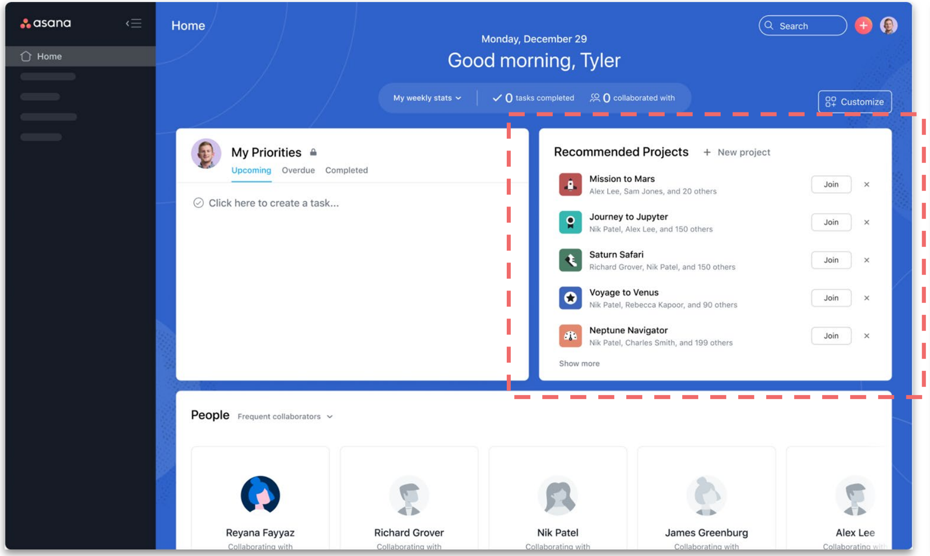
Hypothesis

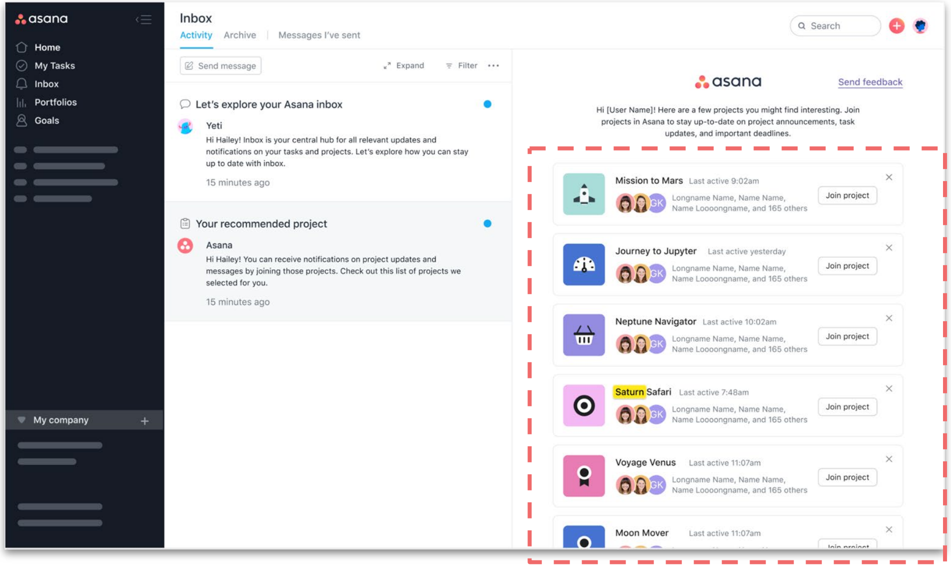
Personalized project recommendations can help users in organizations quickly find relevant work, leading to increased adoption and retention

UI Components for surfacing recommendations

User Interface

We built two new UI components to surface recommendations





Home Widget

Inbox Notification

Modelling Approach

Collaborative Filtering

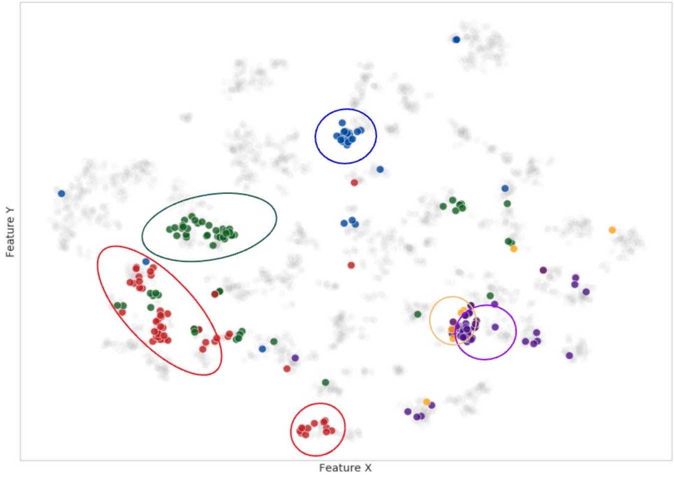
Implicit Collaborative Filtering uses interaction data such as visits, task creations, memberships to infer preferences

N x M User-Project Matrix

	Proj 1	Proj 2	Proj 3	Proj 4	Proj 5	Proj N
User 1	v_{11}	v_{12}	--	v_{14}	--	--
User 2	--	--	v_{23}	v_{24}	--	--
User 3	--	--	--	v_{34}	v_{35}	--
User N	v_{N1}	--	--	v_{N4}	--	--

→ **Collaborative Filtering Model** →

We used CF, but this could be other approaches like two-tower networks



Mission to Mars
Mars Polar Lander Mission
Martian Climate Study
Mars Rover Enhancement Project

Saturn Safari
Saturnian System Survey
Saturn's Rings Educational Initiative
Titan Aerial Vehicle Project

where v_{ij} is some measure of users's engagement with projects

Performance Evaluation

Offline and Online Evaluation

The model was evaluated offline (using past data) and online (A/B tests) against the heuristic and a baseline

Approaches Considered

- **Random Baseline:** Recommend the most popular projects in the domain
- **Heuristics:** Recommend projects that a user's active teammates are a part of
- **Model:** Recommend similar projects to the project that the user was invited to

Offline Evaluation

Using past data (backtesting) to assess effectiveness of the recommendation system without affecting user experience. Specifically used during development phase

Success Metrics:

- **Mean Average Precision @ K (MAP@K):** Indicates how well the Project are ranked
- **Hit Ratio (HR):** Indicates how relevant the projects are

Online Evaluation

A/B testing the system on actual data for more accurate representation of the performance

Success Metric:

- Retention and adoption rates for new and existing users
- Project Memberships, engagement

Secondary Metrics:

- Mean Average Precision @ K (MAP@K)
- Hit Ratio (HR)

Chapter 2

Recommending Projects

Key Insights and Learnings

Insight 1: Establish a baseline

Building a heuristic

We started by building a simple **heuristic** which could be computed and served with our existing infrastructure.

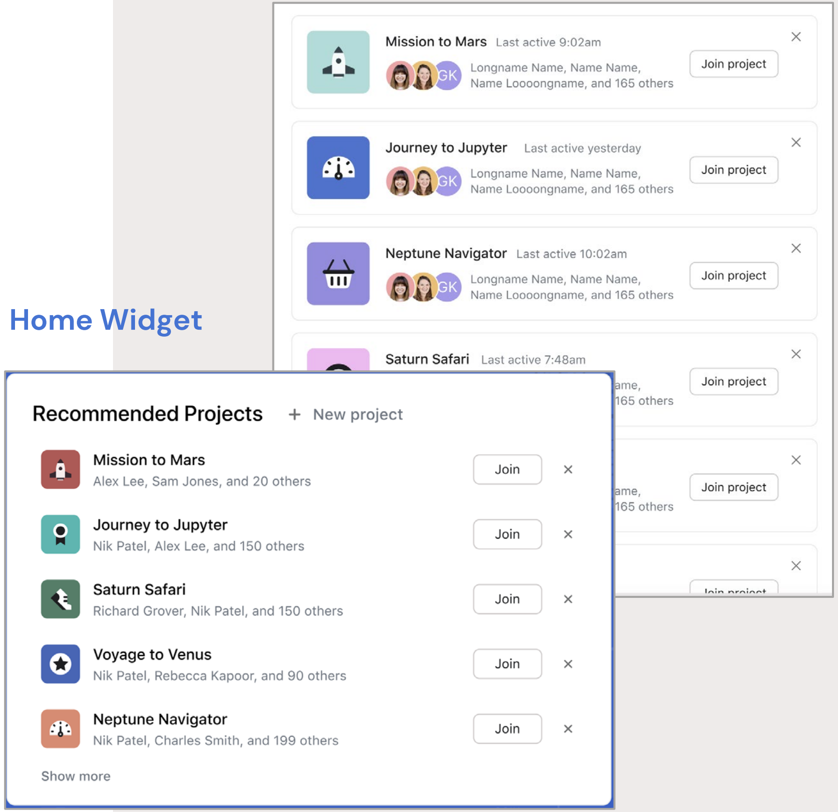
- Surfaced the most active projects in users' organization

Advantages:

- **Set baseline:** Provided a clear benchmark to measure the effectiveness of our ML model.
- **Problem Understanding:** Enhanced our understanding, guiding feature selection and model design
- **Fallback Solution**: Ensured we had a working solution in place even if the ML model needed more development.

Asana Inbox Notification

Home Widget



Insight 2: Consider modelling for different user segments

Segmenting users into new and existing users

Existing users

Impact of recommendations: Existing users are likely already member of all projects that are pertinent to them. Surfacing new projects would increase visibility for other related projects

No cold start: We have information about users' interaction and engagement across projects, tasks and other surfaces in Asana

Data available: Users engagement and interactions with projects (ex: visitation, task creation, project sharing)

Modelling approach: Collaborative-filtering

New users

Impact of recommendations: Much more leveraged for new users as onboarding them to relevant work early in their Asana journey would increase long-term adoption and retention

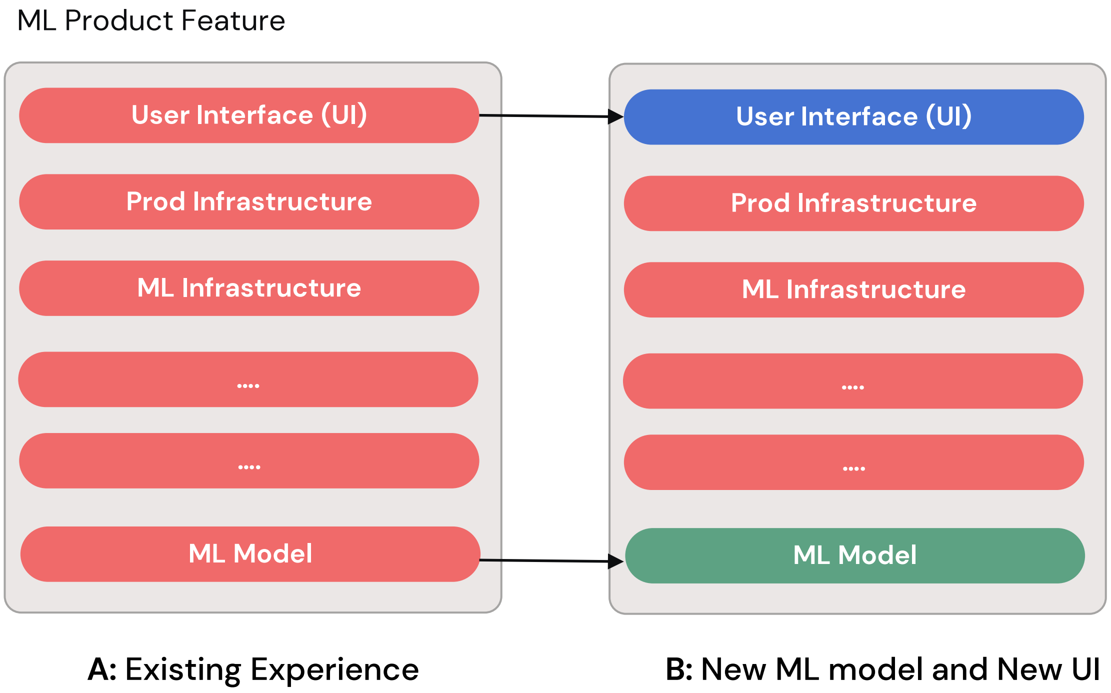
Cold start problem: We have limited to no information about user preferences, teams and relevant work

Data available: Project and teams that the user is invited to

Modelling approach: Similar Projects through CF embeddings

Insight 3: Isolate the impact of different components of the ML system

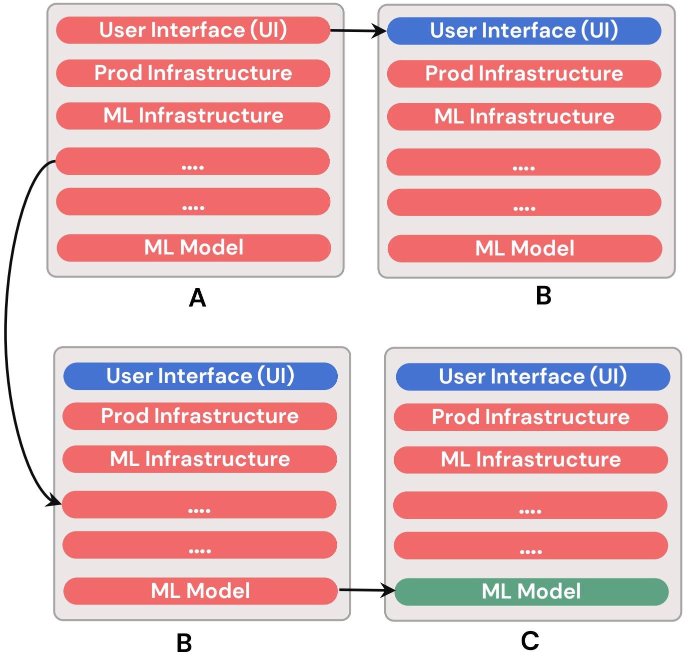
Experiment Design Consideration



- ML Product features have several components that can impact the performance of the feature
- When experimenting with multiple components, it is important to isolate the impact to understand where we should iterate

Insight 3: Isolate the impact of different components of the ML system

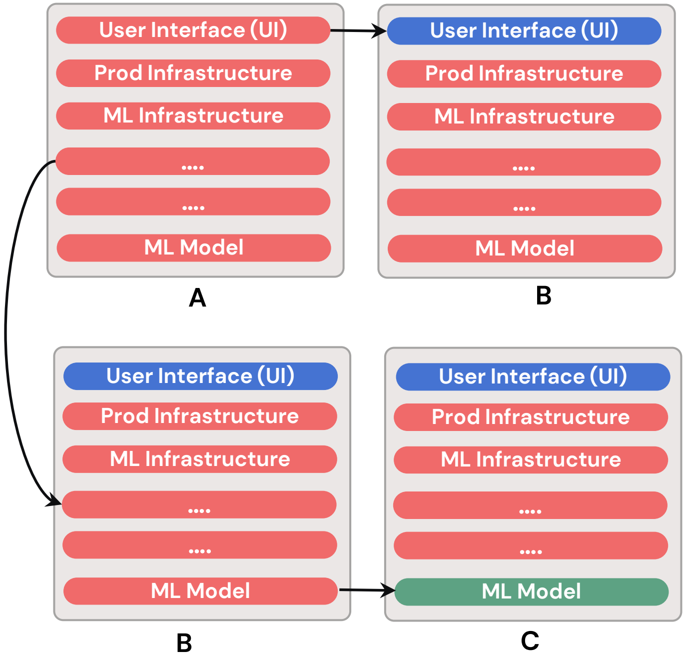
Strategies



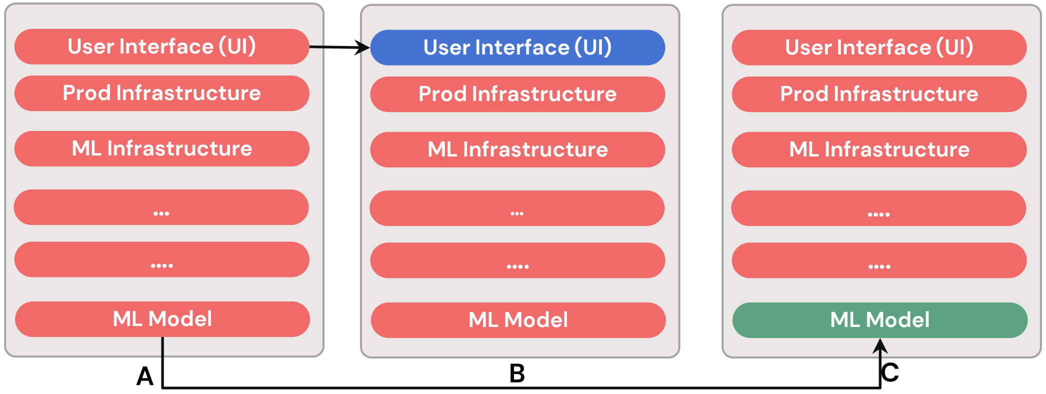
Sequential Testing

Insight 3: Isolate the impact of different components of the ML system

Strategies



Sequential Testing



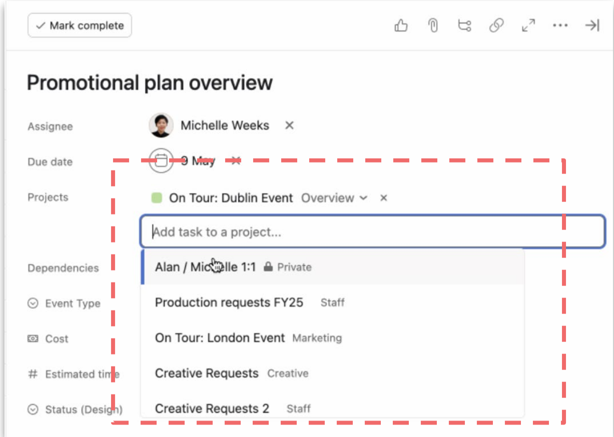
Multivariate Testing

Running a **multivariate** or **sequential** tests can help isolate the impact of changing different components in ML-based product features and help plan for future iterations

Insight 4: Understand the performance implications of serving ML

Impact of latency

Impact on user experience



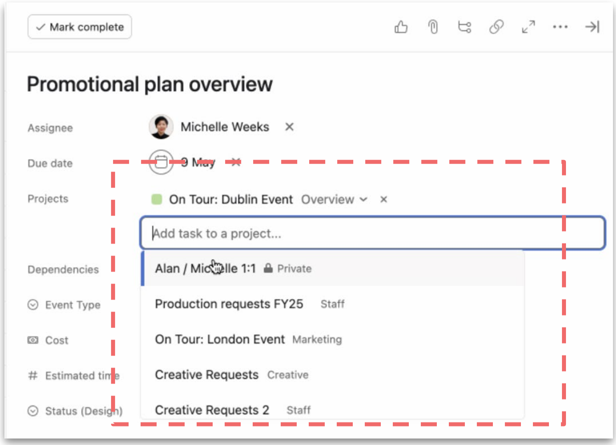
Multi-homing suggestions

- The A/B test was a **loss** on ML ranking due to an impact on load times for ML generated recs

Insight 4: Understand the performance implications of serving ML

Impact of latency

Impact on user experience



Multi-homing suggestions

- The A/B test was a **loss** on ML ranking due to an impact on load times for ML generated recs

Detecting the impact

Control | ML Variant | Dark Variant

- Add a new variant (dark variant) where the system reads from the ML systems but does not alter the user experience.
- Run a multivariate or sequential test, and compare the treatment and the dark variant
- **Key Learning:** Infra Perf play a big role in ML success and something we should monitor while testing features, especially on sensitive surface

Summarizing the key insights

The takeaways

Establish a baseline

Establish simple baseline model or rule-based system
This should be easier to build and implement, and the learning can be used for building the ML model

Consider modelling for different user segments

Analyzing the model's performance across diverse user groups and identifying potential biases or disparities
Get creative with solving the cold-start problem of serving predictions to new users

Isolate the impact different components of the ML systems

Running a multivariate or sequential tests can help isolate the impact of changing different components in ML-based product features and help plan for future iterations

Understand the performance implications of serving ML

Analyze and monitor latency introduced with serving ML, sometime even small changes in latency can have large impacts of user behavior
Considering dark launching the feature first, or running a multivariate test with dark variant

Chapter 3

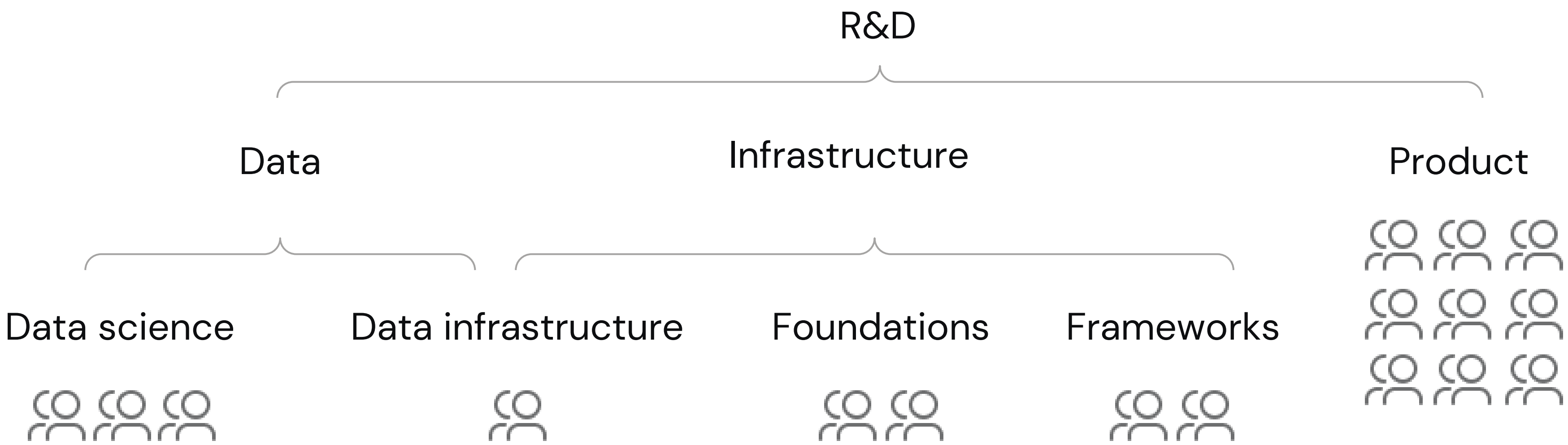
ML Infrastructure

Productionization

Cross-functional collaborations

Requirements & Planning

Conway's Law: "You ship your org chart"

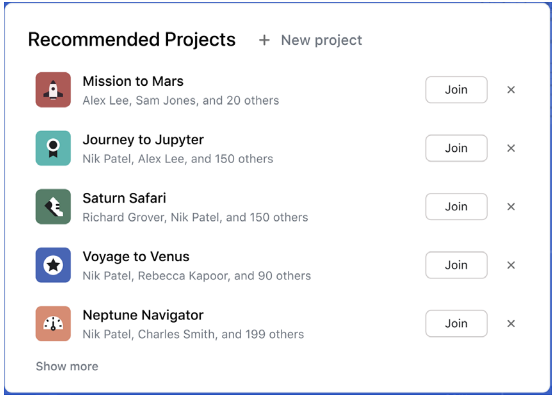


Model re-use

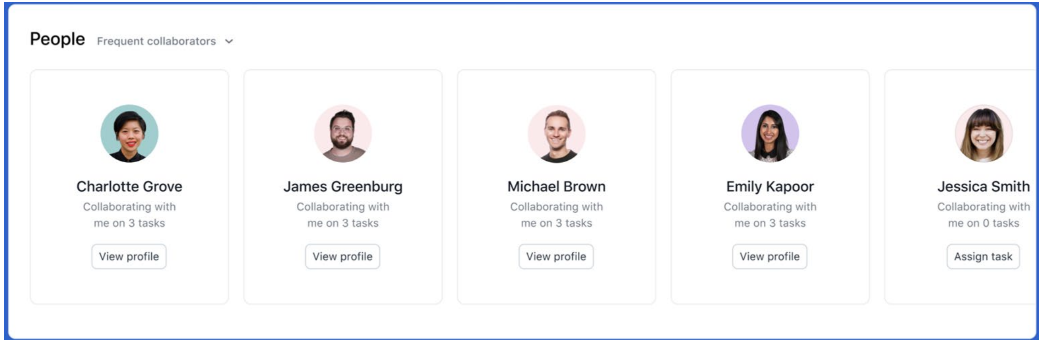
Requirements & Planning

ML datasets can be reused for similar purposes in different surface areas

For example, **Collaborative filtering** generates recommendations for projects, and ‘similar users’ for ranking user facepiles



Project Recommendations



Similar Users

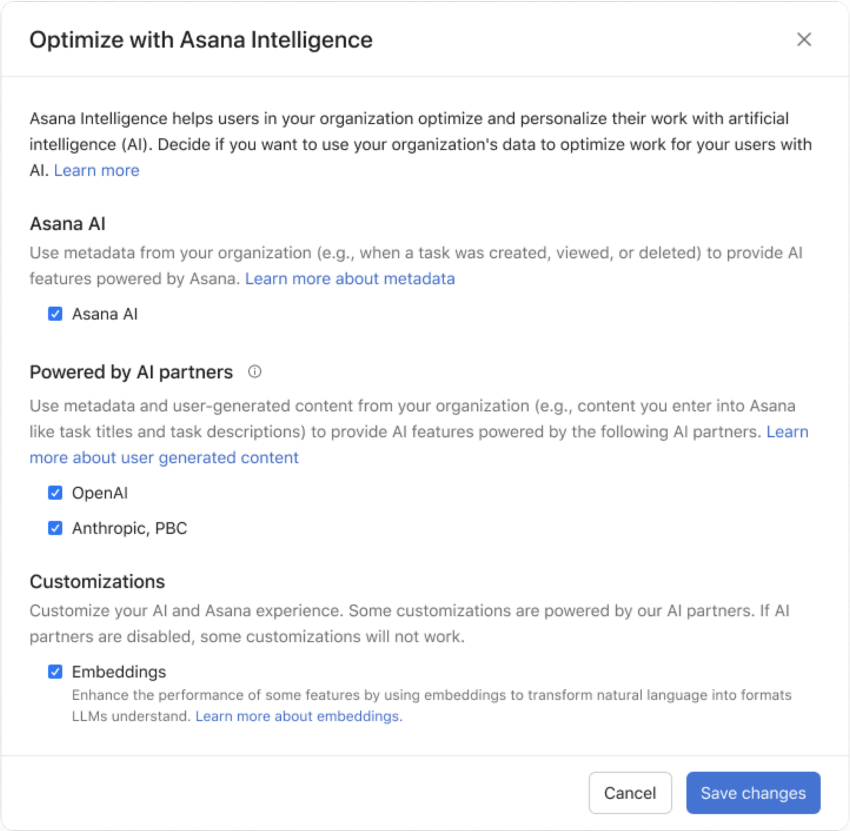
Safeguards for trust and compliance

Requirements & Planning

AI/ML is a **complicated landscape from a legal and regulatory point of view**

Asana has sophisticated launch planning for AI/ML features to **build customer trust, comply with law, and preserve Asana's AI principles**:

- All AI/ML use cases are **reviewed and approved** by a cross-functional group of legal, privacy, security teams prior to implementation
- **Customers are empowered to decide** what scope of AI/ML product features they wish to allow
- AI/ML features are **extensively documented** to maintain transparency



Modes of operation

Requirements & Planning

Batch processing

Run more **resource-intensive methods** on larger datasets

Maintain **consistent user experience** for longer time

Pre-computed results may require **further validation**

Examples:

Project or Team Recommendations
Ranking projects in the sidebar

Real-time / on-demand inference

Time-sensitive context

Operate on **never-before-seen** data

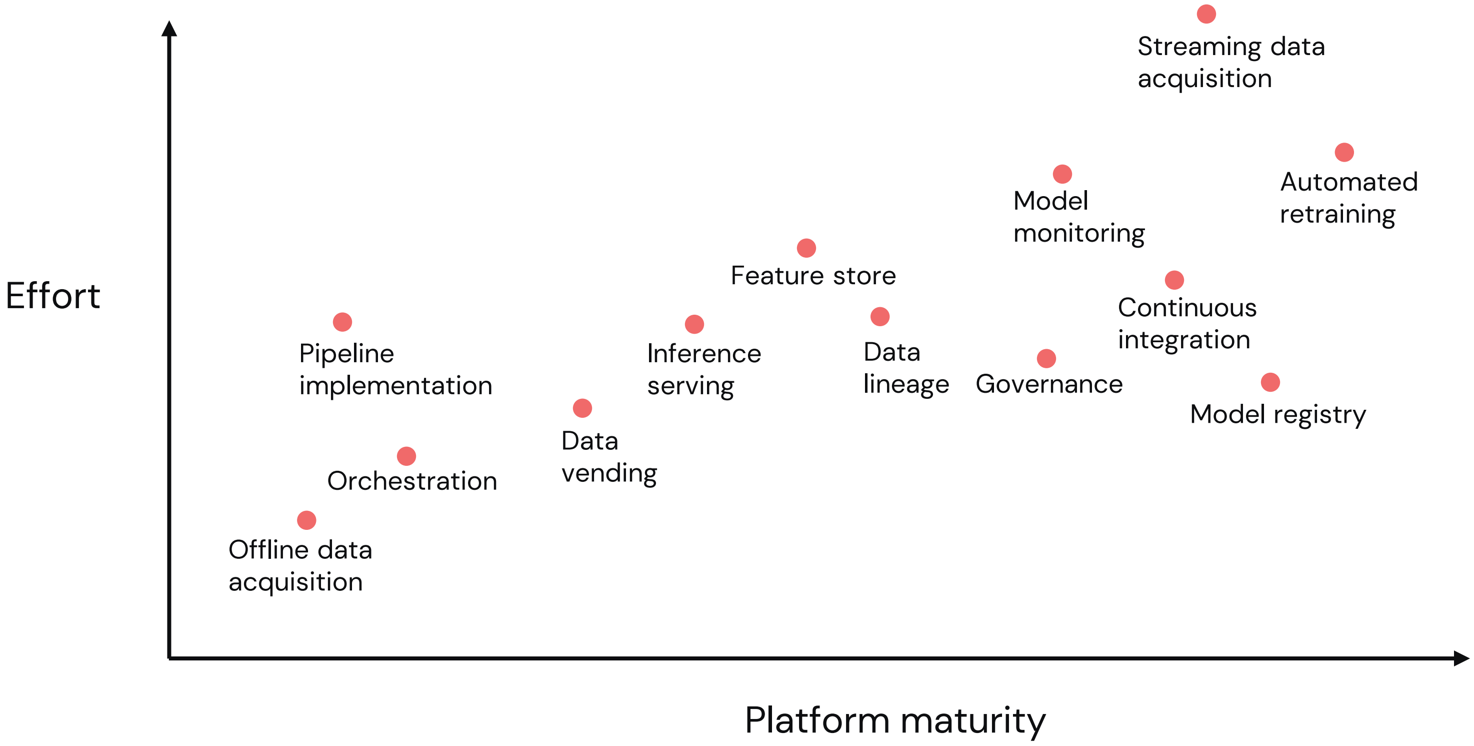
Short feedback loop in response to user activity

Examples:

Ranking notifications in inbox
Prioritizing tasks in "My tasks"

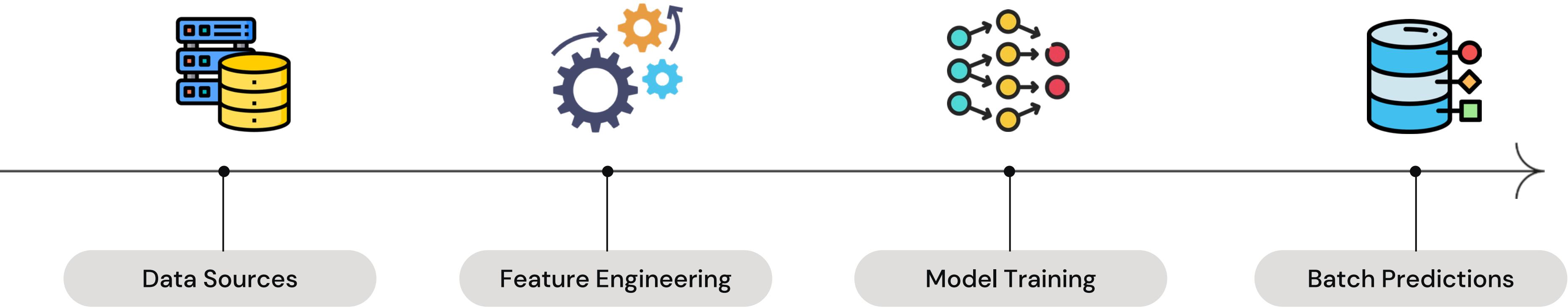
Choosing infrastructure investments

Requirements & Planning



Batch pipeline

Implementation Strategy



NumPy pandas LightGBM PyTorch TensorFlow scikit learn

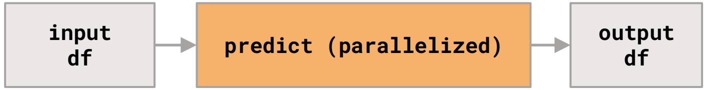
× over 100,000 per day

Scaling batch ML processing

Implementation Strategy

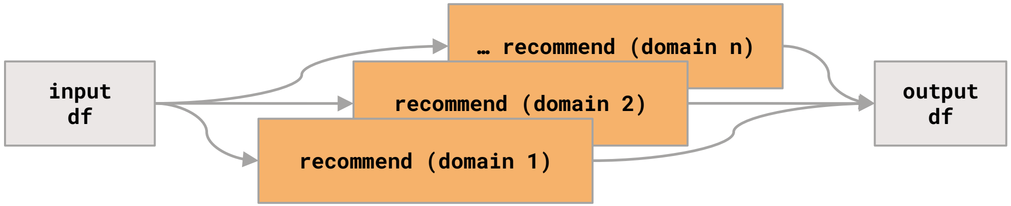
Running inference in parallel

```
def prediction_in_pandas(pandas_df):
   # Predictions generated here
   # ...
   df[“score”] = model.predict_proba(pandas_df)[:, 1]
   return df

df = prediction_set_df.mapInPandas(
   prediction_in_pandas,
   prediction_set_df.withColumn("score", lit(0.0)).schema,
)
```



Training models in parallel

```
def recommendations_in_pandas(group_key, pandas_df):
   # Recommendation models fit here
   # ...
   return output_pandas_df

df = input_data.groupBy("domain_id").applyInPandas(
      recommendations_in_pandas,
      schema=StructType(
          [
              StructField("object_id", LongType(), True),
              StructField("recommended_objects", ArrayType(LongType(), True), True),
              StructField("scores", ArrayType(DoubleType(), True), True),
              StructField("domain_id", LongType(), True),
          ]
      ),
  )
```



Ephemeral data store

Implementation Strategy

Requirements

Deliver batch predictions to product applications

- **High-throughput** bulk data ingestion: Write spike + idle rest of day
- **Low-latency, high-concurrency** random access retrieval
- **Durability and fault tolerance** to minimize on-call burden
- **Easy deletion** of old records
  - Control storage footprint
  - Satisfy compliance requirements

Solution

Distributed key-value store (AWS DynamoDB)

- **Avoid introducing load spikes** onto other production DBs with critical customer data
- **Control costs** with I/O capacity auto-scaling
- **Expire old data by setting a TTL**, which does not incur any explicit write/delete operation
- **Be flexible with data schemas** and easily support future use cases

Serving ML models real-time

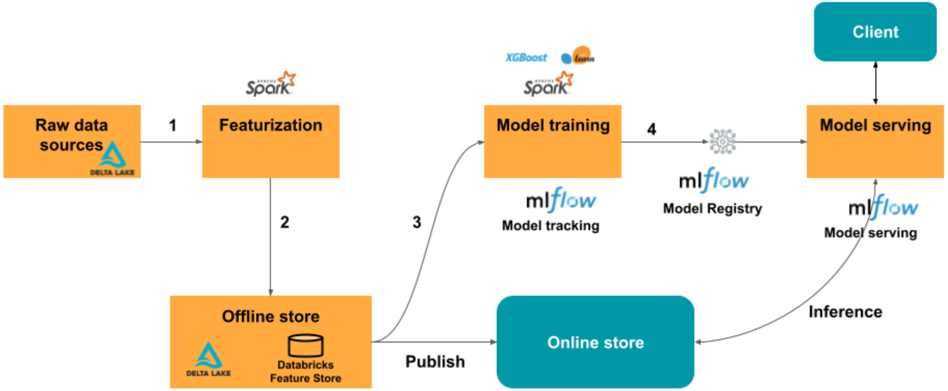
Implementation Strategy

Model Service

- **Train model** in Databricks notebook
- **Deploy model** artifact to SageMaker
  - Multi-model endpoints pick up new artifacts from S3
  - Clients can make requests to new models immediately
- Clients need to assemble feature vectors to supply as input
- This is where online **feature store** will come in handy!

Feature Store

- Easily (but selectively) **publish features from offline store** (used for training and batch inference) **to online store** (used for realtime, on-demand predictions)
- Improve **data discoverability** and **collaboration** for new use cases starting from EDA and prototyping
- **Trace data lineage** and maintain an audit trail for troubleshooting



[Databricks: What is a feature store](#)

Summarizing the key insights

The takeaways

Understand cross-functional requirements beyond just executing ML

Develop consensus with product, infrastructure, security, and compliance stakeholders
Leverage existing internal expertise whenever available

Choose ML serving solution based on data volumes and product needs

Batch-precomputed results are well-suited to key-value stores
On-demand model serving may require setting up endpoints and onboarding feature store

Strike a balance between incremental and iterative platform development

Not every component in a full-featured ML platform will be necessary at every stage
Going wide early will increase supported scope in the short term, going deep early will increase velocity for similar use cases

Pandas on PySpark makes scaling up easy

ML engineers can easily adapt implementations from exploratory notebooks to production jobs
Training hundreds of thousands of models daily becomes a tractable problem

Q&A

asana