



ZOZO おすすめアイテム推薦の 改善に向けた分析と挑戦

宮本 知弥 築山 将央
株式会社 ZOZO

桃井 啓行
Google Cloud

スピーカー自己紹介



桃井 啓行

デジタル ネイティブ エンタープライズ担当カスタマー エンジニア。Google Cloud ソリューションの紹介やアーキテクチャ設計、PoC 支援などを通してお客様のクラウドシステム構築を技術面よりサポートしています。



01 | Recommendations AI 概要

02 | ZOZO おすすめアイテム推薦について

03 | 分析とAB テストの取り組み

04 | ZOZO おすすめアイテム推薦システムアーキテクチャ

05 | ZOZO 内製推薦ロジック開発における Google Cloud の活用



Recommendations AI 概要

アイテム推薦システムを導入する上でのチャレンジはなんでしょう？



顧客行動の理解

- 新規ユーザー
- ユーザー行動の変化
- フィードバックの欠損



オムニチャネル コンテキストの理解

- ページ遷移
- デバイスの変更
- ブランドの変更

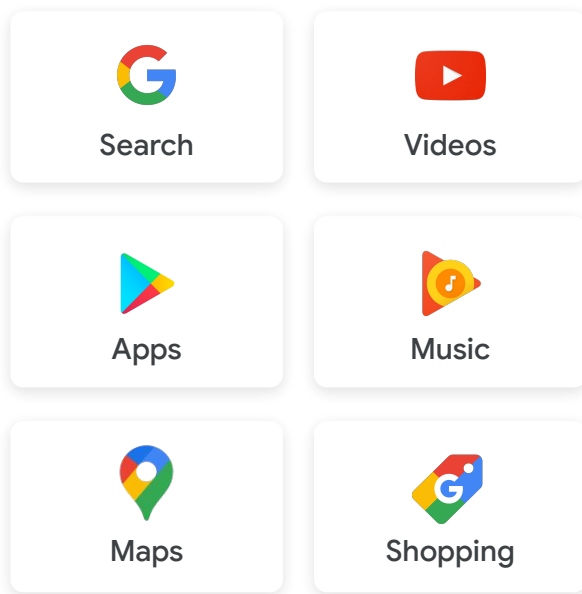
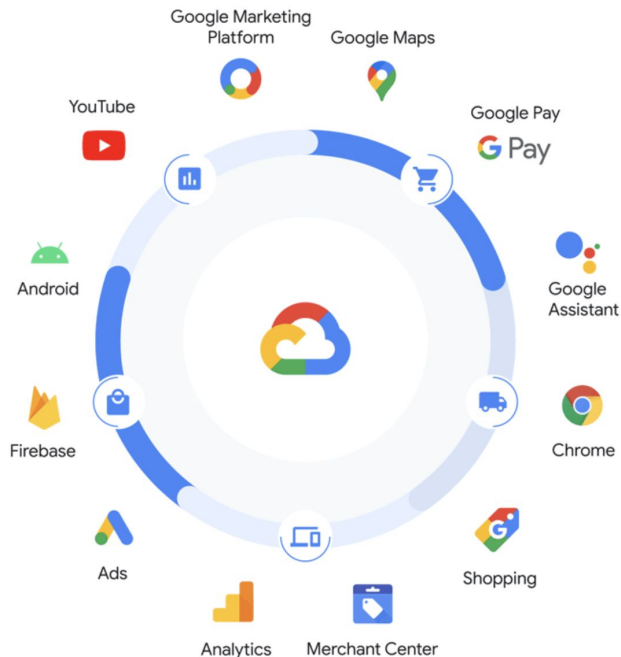


商品データの理解

- 新商品
- ラベルの欠損
- 構造化されていない
メタデータ

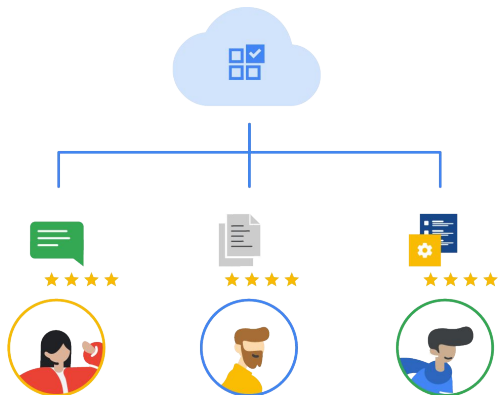
Google Cloud を使う理由は?







Google Cloud は、世界中の何十億もの人々が使用するレコメンデーションと検索の専門知識を提供しています。



Recommendations AI でユーザの行動、コンテキスト、商品ニュアンスを理解して顧客エンゲージメントを促進

Recommendations AI で
高度にパーソナライズされた製品の
推奨事項を大規模に提供



-  インテリジェントで最先端のパーソナライズ ML
-  クリック率、コンバージョン率、収益などの劇的な改善
-  フルマネージドでグローバル規模なサービス
-  容易なデータ統合
-  世界中の顧客にあらゆるタッチポイントを提供
-  データの所有権はユーザにあり、Google が利用することはありません

カスタマージャーニーと購買者の意図を理解した推薦

既存の推薦システム

人気商品の推薦

同じカテゴリ内で人気の商品を推薦



- ✓ 人気商品
- ✗ クロスセル
- ✗ 他のユーザの購買意向
- ✗ パーソナライズ

商品ベースの推薦

他のユーザが興味を示した商品を推薦

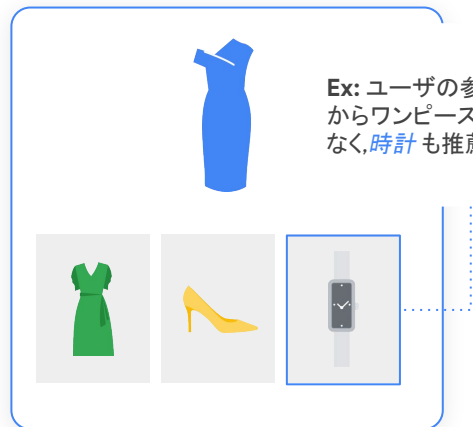


- ✓ 人気商品
- ✓ クロスセル
- ✓ 他のユーザの購買意向
- ✗ パーソナライズ

Recommendations AI

パーソナライズされた推薦

顧客の行動ベースでの推薦



- ✓ 人気商品
- ✓ クロスセル
- ✓ 他のユーザの購買意向
- ✓ パーソナライズ

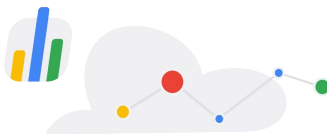
Recommendations AI をはじめる 3 つのステップ

1 データ収集



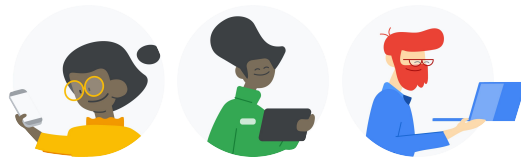
既存ツールを使用してユーザーイベントとカタログデータをコードレスで統合可能

2 ビジネスニーズに合わせてすばやくカスタマイズ



データの専門知識やインフラストラクチャがなくても、モデルトレーニングを数か月から数分に短縮

3 カスタマージャーニーのどこにでも配信



さまざまな顧客のタッチポイント、デバイスタイプ、および場所にわたって、パーソナライズされた推奨事項を活用可能

毎日再トレーニングされるモデルで常に推薦商品を改善 & 最適化



ZOZO おすすめアイテム推薦について

スピーカー自己紹介



宮本知弥

株式会社 ZOZO
技術本部
ML・データ部
推薦基盤ブロック

2020 年 4 月に新卒入社。

学生時代はロボットラーニングの研究を通じて、ガウス過程やベイズ推論などの確率モデルを専攻してきた。

現在は推薦システム導入のプロジェクト マネジメントをはじめ、統計分析や AB テストを用いたレコメンド機能の改善に取り組んでいる。

ファッションのビッグデータ分析・活用に興味がある。

スピーカー自己紹介



築山将央

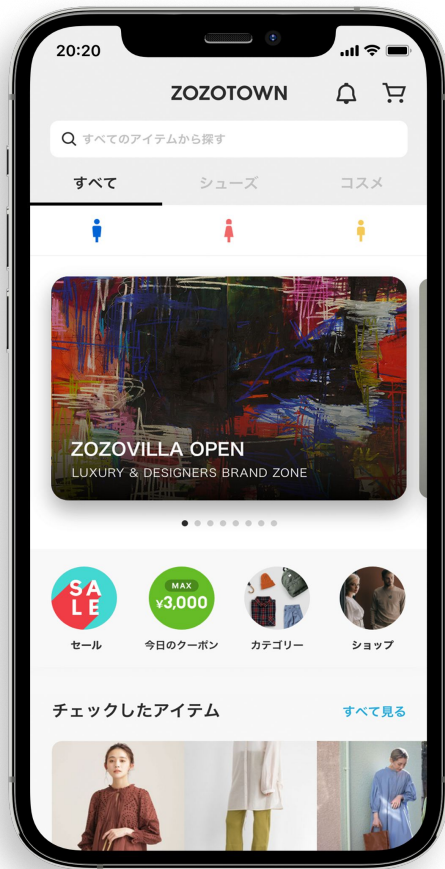
株式会社 ZOZO
技術本部
ML・データ部
MLOps ブロック

2021 年 1 月に中途入社。

学生時代はコンピュータビジョンの研究に従事し、前職では MLOps エンジニアとしてオートモーティブ領域の機械学習システムの開発運用を行っていた。

現在は ZOZO にてレコメンドを始めとする複数の機械学習システムの開発運用に携わっている。

クラウドネイティブアーキテクチャと自動化に興味がある。



ZOZOTOWN

<https://zozo.jp/>

- ファッション通販サイト
- 1,500 以上のショップ、8,400 以上のブランドの取り扱い
- 常時 83 万点以上の商品アイテム数と毎日平均2,900 点以上の新着商品を掲載(2021 年 12 月末時点)
- ブランド古着のファッションゾーン「ZOZOUSUED」やコスメ専門モール「ZOZOCOSME」、靴の専門モール「ZOZOSHOES」、ラグジュアリー& デザイナーズゾーン「ZOZOVILLA」を展開
- 即日配送サービス
- ギフトラッピングサービス
- ツケ払いなど





あなたは右の商品を見ました。
一緒に推薦されて嬉しい商品は
次のうちどれですか？



一緒に推薦されて嬉しい商品は次のうちどれですか？

閲覧商品



- ホワイト
- Tシャツ

推薦商品

01



- ブラック
- Tシャツ

02



- ボーダー
- Tシャツ

03



- ホワイト
- シャツ

04



- ブラック
- デニム

一緒に推薦されて嬉しい商品は次のうちどれですか？

閲覧商品



- ホワイト
- Tシャツ

推薦商品

01



- ブラック
- Tシャツ

02



- ボーダー
- Tシャツ

03



- ホワイト
- シャツ

04



- ブラック
- デニム

- ユーザーによって趣味・嗜好が異なる
- 正解はユーザーによって異なる

推薦ロジックを改善していくためには

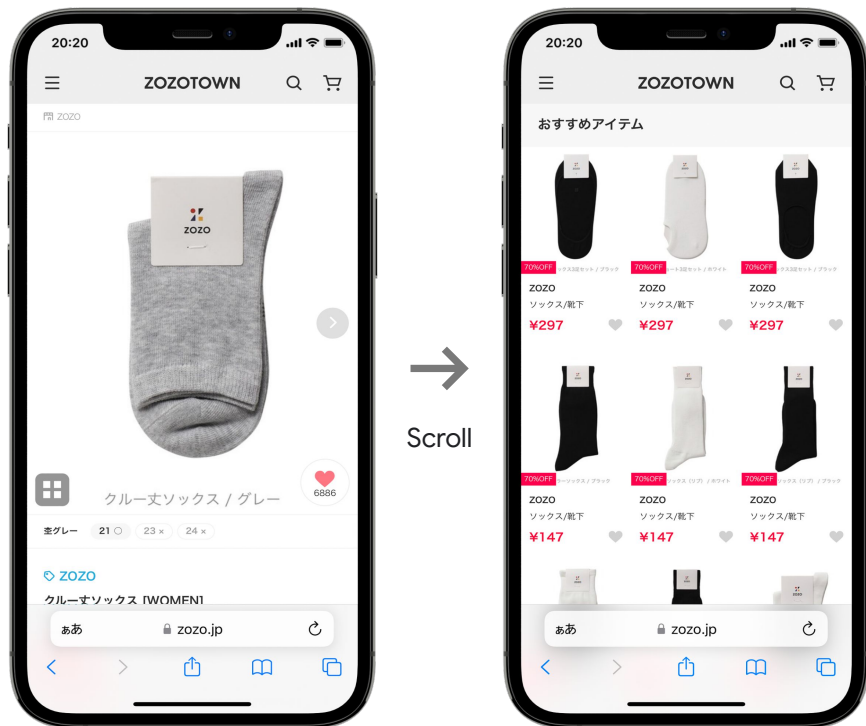
- 01 | ユーザーの嗜好によって、最適な推薦アイテムは異なる
→レコメンド・パーソナライズの強化
- 02 | ユーザーに聞いてみないと、推薦アイテムが本当に最適だったかわからない
→ABテストの実施

ZOZO でも上記の 2 点に関して、積極的に取り組んでいる

おすすめアイテム推薦

ZOZOTOWN 内の商品ページを Scroll したときに表示される「おすすめアイテム」枠のアイテム推薦を指す。厳密には、広告などの固定枠のアイテムも含むが、こちらは推薦対象外。

Recommendations AI 導入以前の推薦ロジックを「旧ロジック」と呼ぶ。この「旧ロジック」はアイテム情報や閲覧情報をもとにしたルールベースのロジックであり、ZOZO 内で長年の間使用されてきた。



旧ロジックの問題点

- 01 | 推薦アイテムが変動しづらく、単純に同じカテゴリや同じブランドのアイテムが並び続ける
- 02 | 推薦アイテムがパーソナライズされていないため全ユーザー同じアイテムが推薦される
- 03 | コールド アイテム・コールド ユーザーに対して推薦できない

おすすめアイテム推薦の改善に向けて

Recommendations AI の導入

Pros

- フルマネージド
- ML エンジニアが居なくても運用可能

Cons

- ブラックボックスな項目が多い
- モデルの種類が限定される

内製推薦ロジックの導入

Pros

- 柔軟度が高い
- 多種多様なモデルの検証が可能

Cons

- ML エンジニアが必要
- モデル開発に一定以上の工数が必要
- 学習・モデルデプロイの仕組みが必要

旧ロジックに対して、Recommendations AI・内製推薦ロジックの **AB テストを実施し、推薦ロジックのリプレイス**を目指す



分析と AB テストの取り組み

AB テスト概要

目的

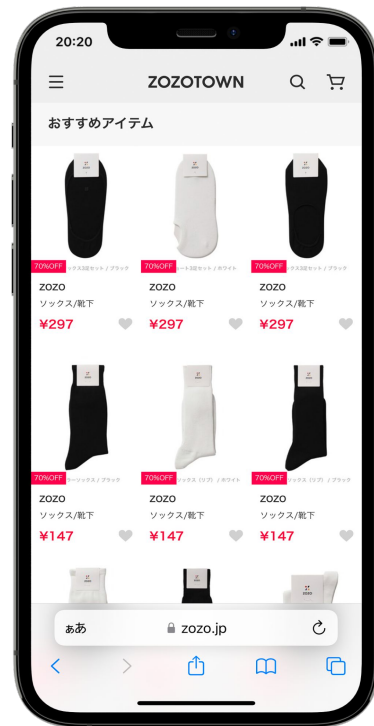
- KGI である GMV 最大化に向けて、推薦モデルをリプレイスする

期間

- 3 - 4 週間 (AB テストによる)

対象者

- 「おすすめアイテム」を viewable impression した全ユーザー
- 下記の条件にマッチするユーザーは除く
 - 社員
 - 注文金額が極端に多いユーザー (上位 0.01 %)
 - アクセス数が極端に多いユーザー (上位 0.01 %)
 - etc...



KPI 指標

KPI 一覧

モデル間のテスト対象者数の分散を考慮するために
「[テスト対象者あたり](#)」で算出している。

サイト全体とレコメンド経由を比較するために
「[レコメンド経由](#)」も算出している。

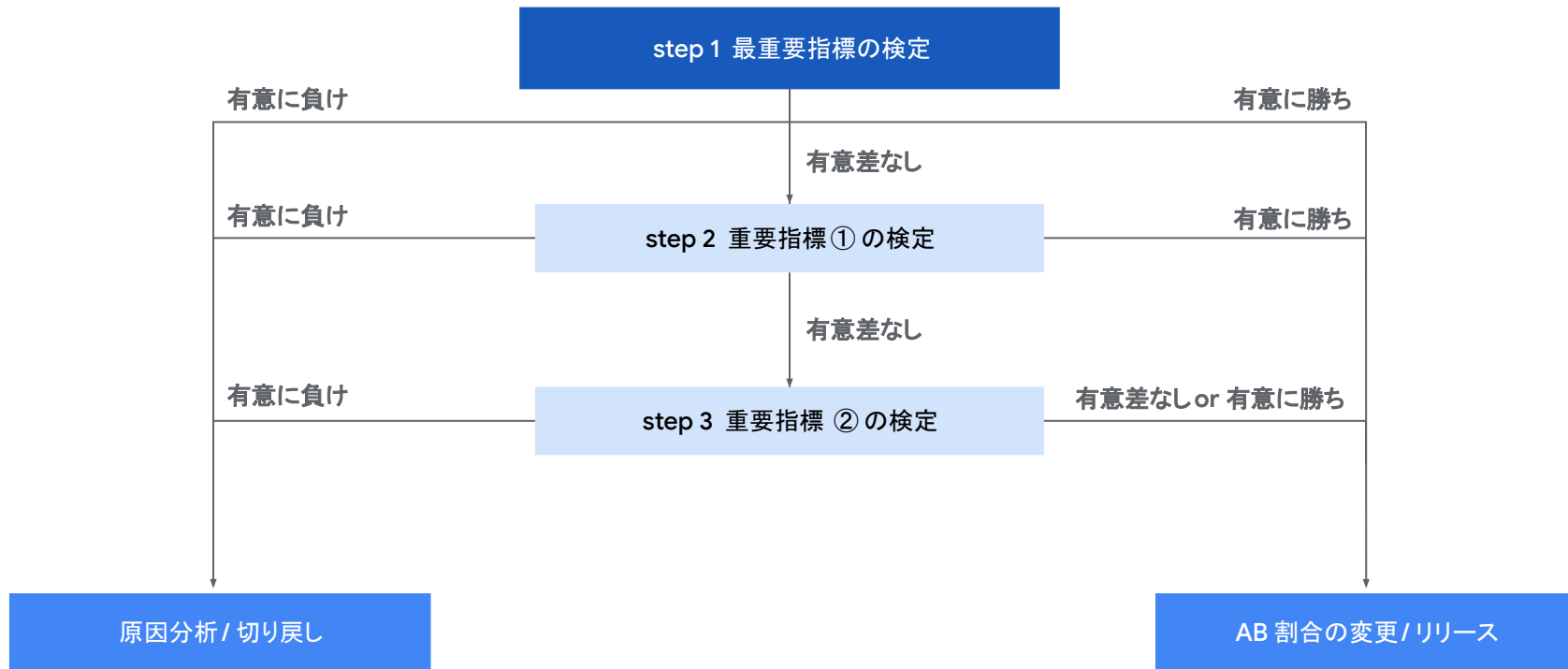
★★★: 最重要指標

★★ : 重要指標

★ : サブ指標

優先度	No	項目
-	1	テスト対象者数
★★★	2	テスト対象者あたり注文金額
★★	3	テスト対象者あたり注文数
★	4	テスト対象者あたり注文商品点数
★	5	テスト対象者あたり注文商品単価
★★	6	テスト対象者あたり商品閲覧数
★	7	テスト対象者あたりユニーク商品閲覧数
★	8	テスト対象者あたりカート投入数
★	9	テスト対象者あたりユニーク商品カート投入数
★	10	テスト対象者あたりレコメンド経由注文金額
★	11	テスト対象者あたりレコメンド経由注文商品点数
★	12	テスト対象者あたりレコメンド経由注文商品単価
★	13	テスト対象者あたりレコメンド経由商品閲覧数

リリース意思決定フロー



AB テスト変遷

これまでに Recommendations AI と内製推薦ロジックの AB テストを合計 5 回実施



AB テスト変遷

これまでに Recommendations AI と内製推薦ロジックの AB テストを合計 5 回実施



第 4 回 AB テスト Recommendations AI vs 旧ロジック

※ 第 1 - 3 回 AB テストの詳細は割愛

Recommendations AI モデルについて

- 2 種類の最新モデルを使用
- パーソナライズあり / なし
 - あり
 - キャッシュが無効
 - 損益分岐点が比較的高い
 - なし
 - キャッシュが有効
 - 損益分岐点が比較的低い

※損益分岐点: (営業利益) - (リクエスト課金による費用)

10% テスト → Recs-AI 1 が勝利

呼称	内容	パーソナライズ	割合
Control	旧ロジック	-	10%
Recs-AI 1	CVR モデル 1	あり	10%
Recs-AI 2	CVR モデル 2	あり	10%
Recs-AI 3	CVR モデル 1	なし	10%
Recs-AI 4	CVR モデル 2	なし	10%

50% テスト → Recs-AI 1 が勝利

呼称	内容	パーソナライズ	割合
Control	旧ロジック	-	50%
Recs-AI 1	CVR モデル 1	あり	50%

CVR モデル 1・パーソナライズありが 10% と 50% とともに有意に勝利

これまでの AB テストを振り返って

レコメンド経由注文金額が増加しても、全体の注文金額が増加するとは限らない

- 検索面やリタゲ系モジュールなどの他の面の売上を奪っている
- 推薦に利用するトラフィックが少ない
- CVR・CTR 最適化はあくまで局所的な最適化である

注文金額で有意差を出して勝つことは難しい

- 「テスト期間が短い」や「テスト対象者が少ない」などが原因で 検出力が足りていない
- Recommendations AI のパーソナライズありモデルはキャッシュが使用できないため
キャッシュありと比較して損益分岐点が高い
- AB テストの限られた期間内で 1 ユーザーが購入できる金額には限界がある

前半まとめ

ZOZO ではおすすめアイテム推薦において「旧ロジック」のリプレイスを進めている

- Recommendations AI や内製推薦ロジックによるレコメンド・パーソナライズ強化
- AB テスト実施・統計分析に基づく意思決定

今後の展開

- さらなるレコメンド・パーソナライズ強化に向けて内製推薦ロジックの展開
- テスト対象者の絞り込み条件・テスト実施期間などの最適な AB テスト設計の改善
- 長期的なレコメンド効果の検証
- iOS アプリ / Android アプリへの展開(現在はスマートフォン Web ブラウザのみ)



ZOZOTOWN

おすすめアイテム推薦 システムアーキテクチャ

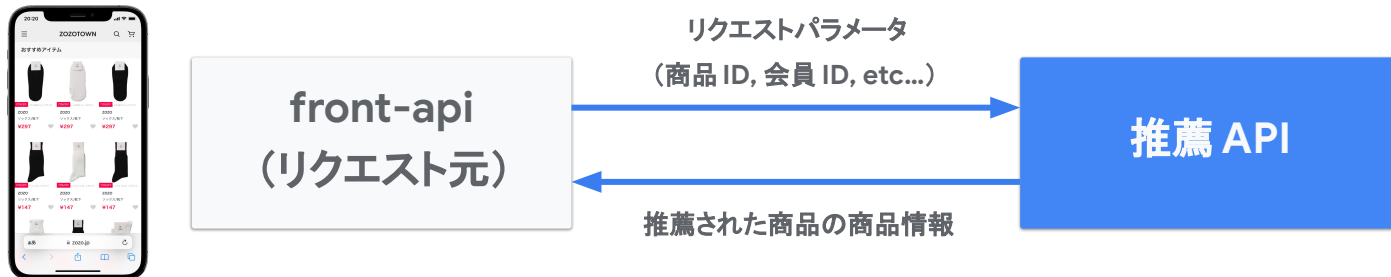
ZOZO おすすめアイテム推薦の仕様

商品詳細ページにアクセスされた際に推薦 API へのリクエストが発火する

リクエストパラメータ

- 閲覧された商品の情報(商品 ID、色 ID など)
- 閲覧したユーザの情報 ID(会員 ID、UserAgent など)
- 取得したいおすすめ商品の件数
- etc...

推薦 API は、**推薦された商品(30 件程度)の商品情報**を返却する



ZOZO おすすめアイテム推薦のソフトウェアスタック

推薦 API

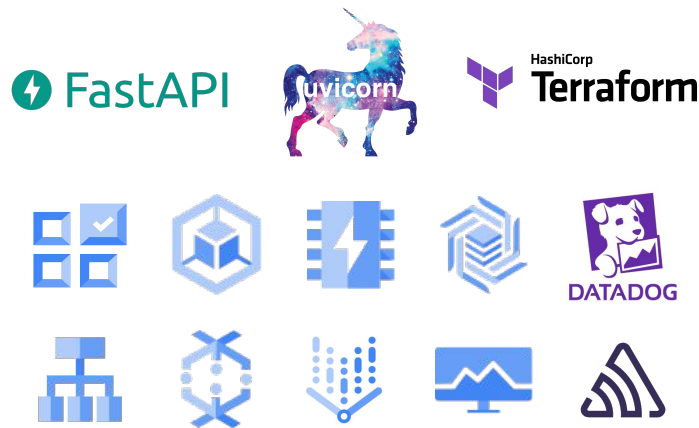
- Java / SpringBoot

内製ロジック用推薦 API(後述)

- Python / FastAPI / uvicorn / gunicorn

インフラ

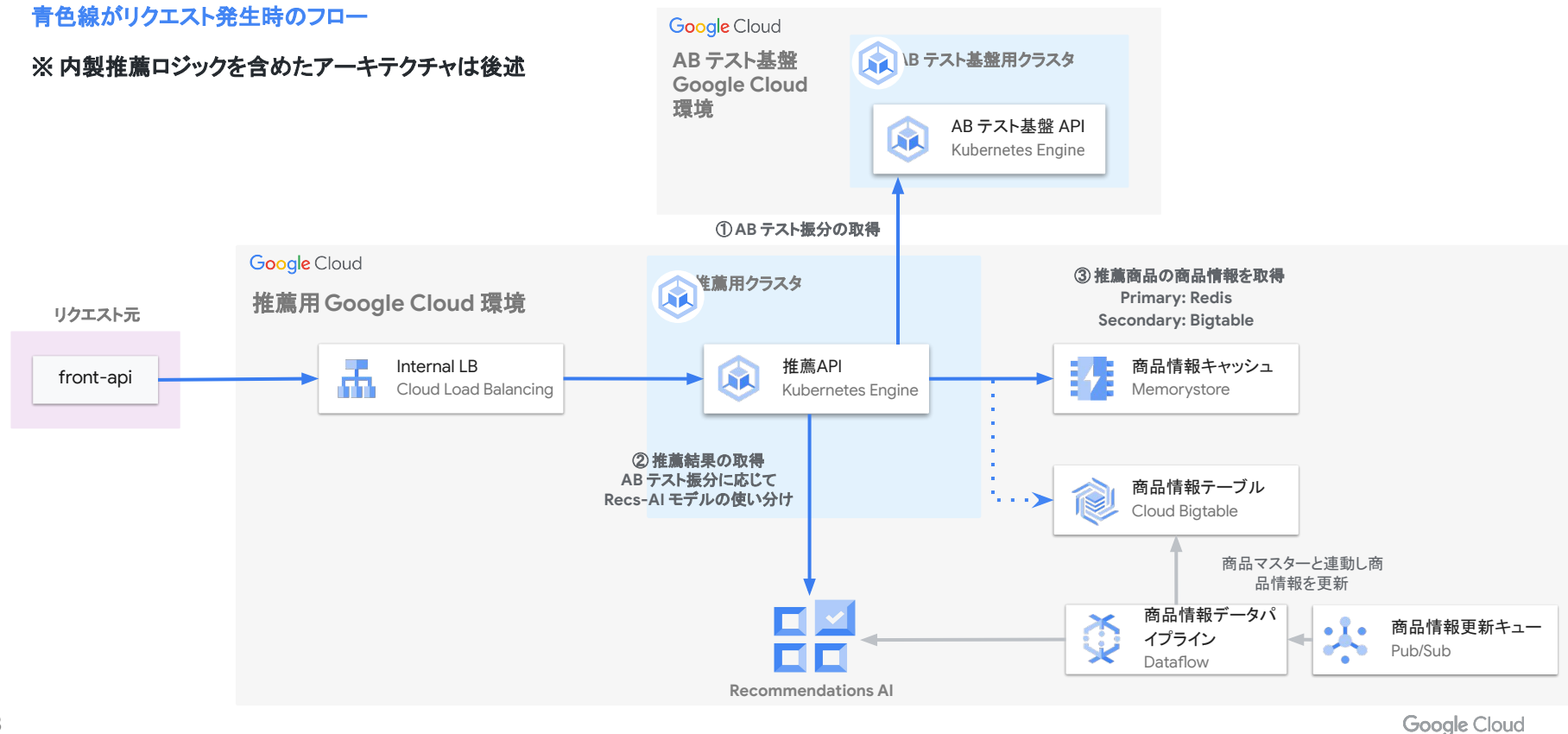
- **API:** Recommendations AI / GKE / Memorystore(Redis) / Bigtable / Cloud Load Balancing
- **データパイプライン:** Dataflow
- **内製推薦ロジックの学習パイプライン:** Vertex Pipelines / Dataflow / BigQuery
- **監視:** Datadog / Sentry / Cloud Monitoring / Pagerduty
- **構成管理:** Terraform



ZOZO おすすめアイテム推薦の全体アーキテクチャ

青色線がリクエスト発生時のフロー

※ 内製推薦ロジックを含めたアーキテクチャは後述



Recommendations AI の運用

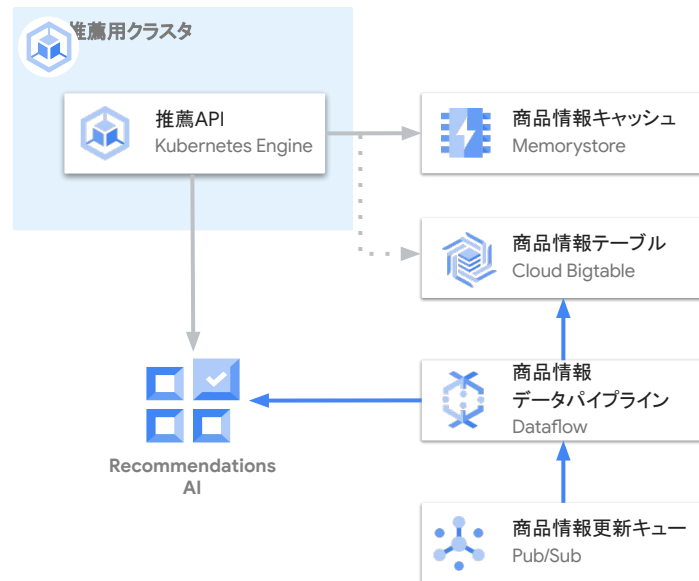
カタログ情報(商品情報)の登録・更新とユーザイベントの投入が必要

カタログ情報の投入

- 商品情報マスター DB の更新と連動して更新される
- 更新を Pub/Sub でサブスクライブし、Dataflow で投入
- 推薦 API で使用する Bigtable にも同時に商品情報を投入

ユーザイベントのリアルタイム投入

- イベントの例
 - 商品ページ表示、カート追加、お気に入り追加、商品購入
- Google Tag Manager を使用してリアルタイムに投入



負荷試験における工夫

負荷試験における Recommendations AI のコストは無視できないため、対策を取る

- 例: 予測 1,000 件あたりの料金を \$0.2 と仮定し、3,000 req/s の負荷を 5 分間掛けると \$180 掛かる

Dry run モードを有効にすることでダミー推薦が可能

- 推薦される商品がデタラメになり、料金は掛からない
- ただしレイテンシも抑えられてしまうため、これだけで試験を完結できない

最小限の回数のみ Dry run を無効にして負荷を掛け、レイテンシの差分を見積もる

- 参考
 - Dry run 有効 → 99%tile 50ms 程度
 - Dry run 無効 → 99%tile 250ms 程度
- レイテンシの差分を考慮しつつ、SLO を満たせるか試験を実施する

推薦結果のキャッシュは有効か？

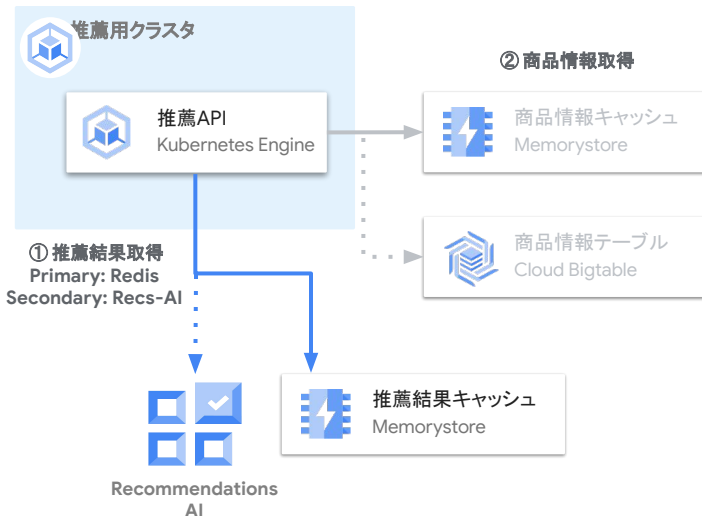
Recommendations AI の料金を抑えるため、推薦結果キャッシュを用いるパターンで AB テスト実施

キャッシュ仕様

- 閲覧商品毎の推薦結果をキャッシュし、有効期限は 24 時間
- 全ユーザに対して共通のキャッシュを用いる
 - すなわち **パーソナライズ**されなくなる

結論: キャッシュによるコスト削減分を GMV 増加分が上回った

- つまりキャッシュは不要だった
- 要因 1: ユーザごとのパーソナライズがされない影響
- 要因 2: 24 時間固定の推薦結果
 - モデル更新・商品情報更新が行われても反映されない
 - 有効期限の短縮はコストとトレードオフ





ZOZO 内製推薦ロジック開発における Google Cloud の活用

Recommendations AI と内製推薦ロジックの比較(再掲)

Recommendations AI の導入

Pros

- フルマネージド
- ML エンジニアが居なくても運用可能

Cons

- ブラックボックスな項目が多い
- モデルの種類が限定される

内製推薦ロジックの導入

Pros

- 柔軟度が高い
- 多種多様なモデルの検証が可能

Cons

- ML エンジニアが必要
- モデル開発に一定以上の工数が必要
- 学習・モデルデプロイの仕組みが必要

Recommendations AI で推薦リプレイスを達成し、長期的には **内製ロジックへ置き換え** を目指す

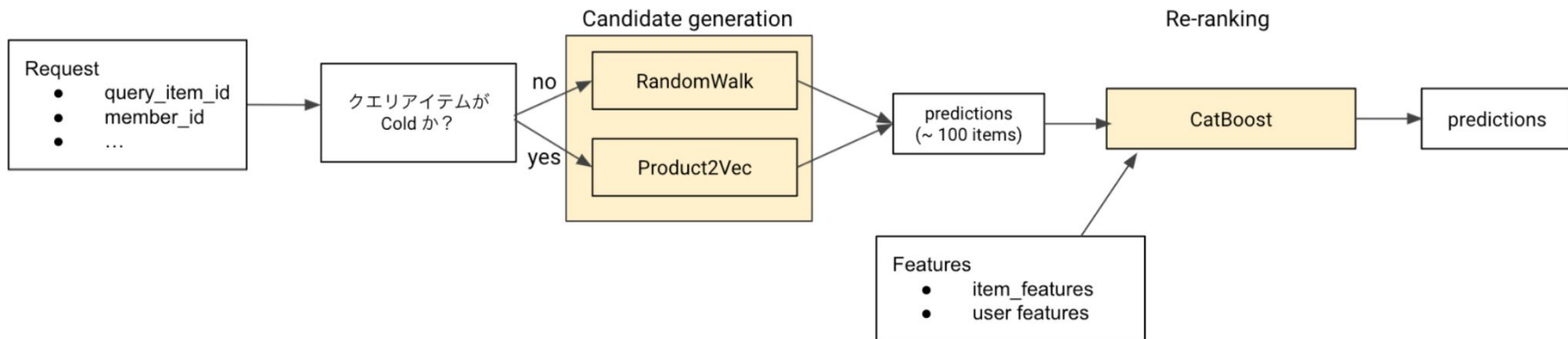
内製推薦ロジックの概要

推薦候補の選出とリランキングの 2 ステージで構成される推薦ロジック

ステージ 1 では RandomWalk または Product2Vec で 100 件程度の候補を選出

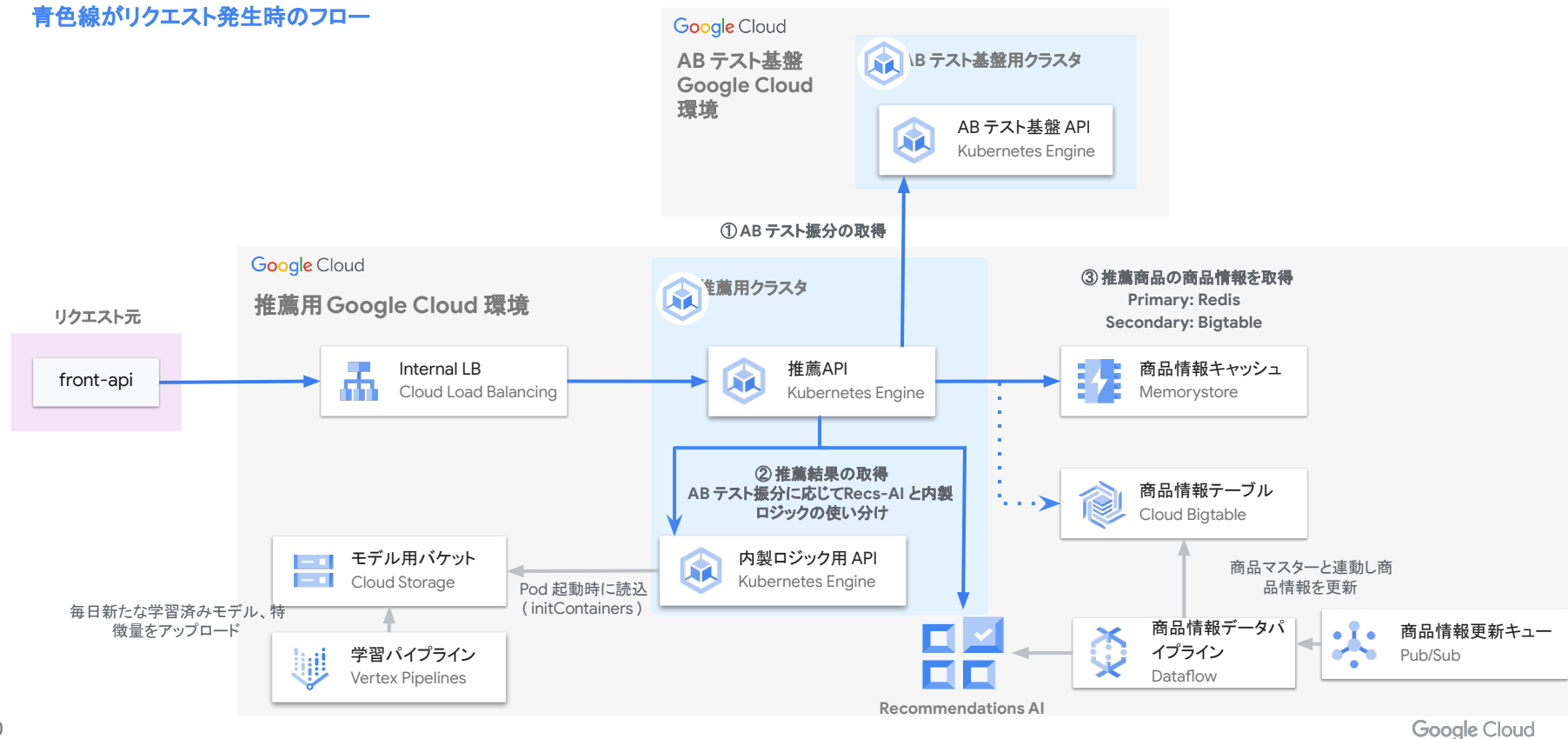
- コールドアイテムに対しては Product2Vec を使う

ステージ 2 では CatBoost でリランキングし、リクエストされた件数まで絞る



内製推薦ロジックを用いたアーキテクチャ

青色線がリクエスト発生時のフロー



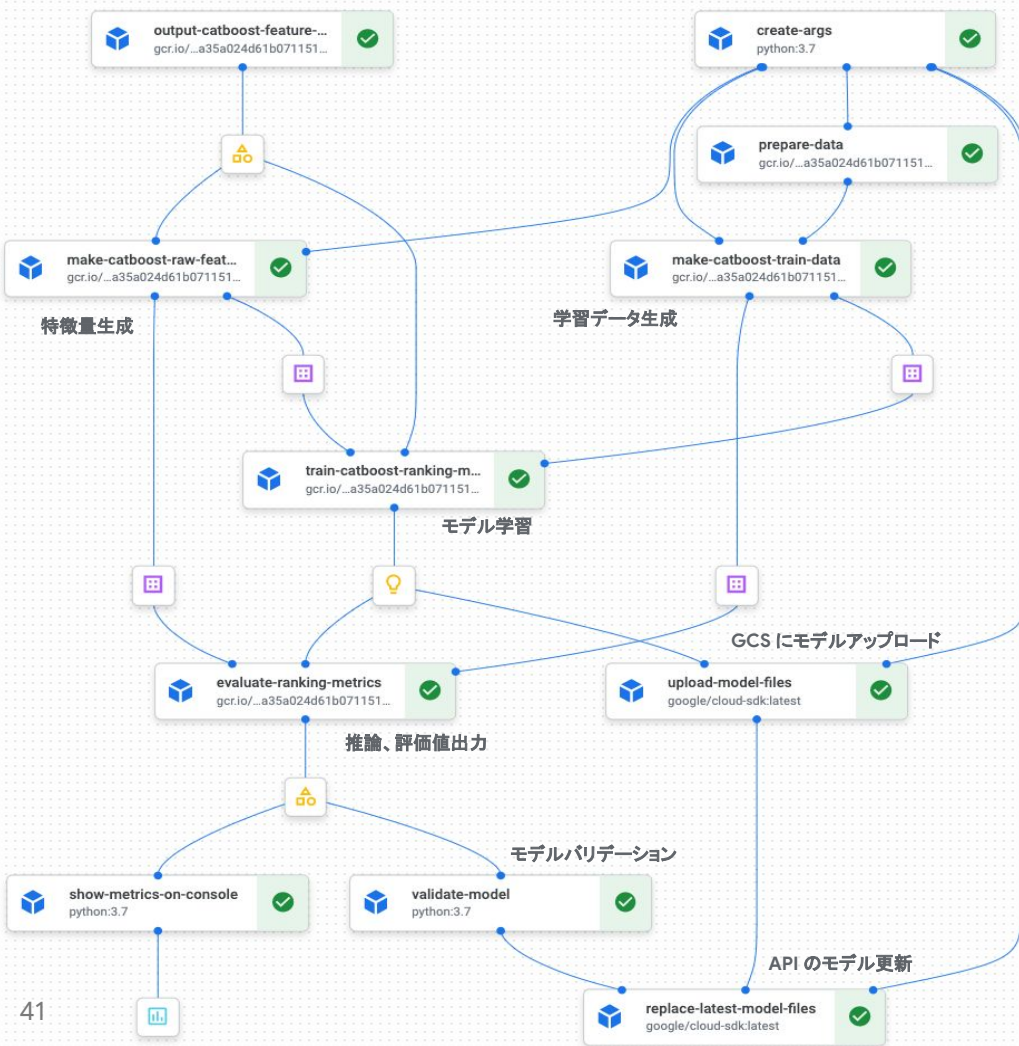
Vertex Pipelines を用いた 学習パイプライン

画像は CatBoost の学習パイプライン

- パイプラインはモデル毎に存在する
(RW, P2V, CatBoost の 3 つ)
- 学習と更新は全モデル毎日行われる

パイプラインの流れ

- 特徴量、学習データ準備 on BigQuery
- モデル学習
- 評価のための推論 on Dataflow
- モデルのバリデーション(後述)
- API で利用するモデルの更新(後述)



モデルデプロイの自動化とバリデーション

毎日のモデル更新は自動化したいが、精度の悪いモデルはデプロイしたくない

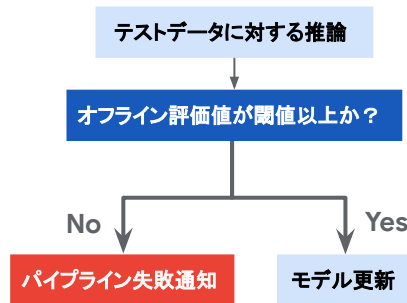
デプロイ自動化

- Pod 起動時にモデルを GCS からロードしている
- パイプラインでモデルアップロード後、k8s Deployment を Rollout して API 上のモデルを更新

Model Validation

- やりたいこと: 壊れたモデルをデプロイするのを防ぐ
 - 壊れたモデル = 特徴量データが異常、学習に失敗しているなど
- やらないこと: 前日のモデルと精度を比較して勝ち負けを判定
 - そもそも前日のモデルとは学習に使われるデータの日付が違う
 - 商品も毎日移り変わるので、オフラインでの単純比較はできない
- 結局 → モデル毎に学習後の評価ステップを追加。オフライン評価値が閾値を下回ったら

パイプ
ラインを失敗させる



パイプ

パフォーマンス問題への対処

当初 n1-standard-4 VM で **1 Pod あたり 20 req/s** しか捌けず、パフォーマンスが課題となった

ボトルネック

- RandomWalk, Product2Vec, CatBoost の 3 モデルの中で RandomWalk が圧倒的に遅い(他は誤差)
- 「閲覧数の多い商品」に対する推論が特に遅く、悪いと 1 秒以上掛かっていることが判明

RandomWalk 推論結果キャッシュの導入

- RandomWalk の推論はパーソナライズされていない
 - パーソナライズはステージ 2 の CatBoost の役割
- そのため Recommendations AI と比較してキャッシュによる精度悪化はほぼ無い
- パイプラインで閲覧数上位 10 万件の商品に対する推論結果をキャッシュし、API 起動時に読み込む
 - 結果 → **1 Pod で 70 req/s** 程度捌けるようになった

後半まとめ

ZOZO ではおすすめアイテム推薦において Recommendations AI を活用している

- 開発工数を抑えたい or ML エンジニア不在の場合に最適
- 負荷試験の際は Dry run on/off を使い分けるなど工夫が必要
- 推薦キャッシュ導入によるコスト削減分 < パーソナライズ有効時の GMV 増加分

長期的には内製推薦ロジックに置き換えるべく、さらなる改善を進めている

- 複数の ML モデルを用いたリランキングロジックを開発
- Vertex Pipelines を活用し毎日の学習・バリデーション・デプロイを自動化
- Recommendations AI vs. 内製推薦ロジック AB テストも実施中

Thank you.

