Google Cloud

ここまで出来る！
機械学習業務を支援する BigQuery ML の新機能

村上 祐磨

Google Cloud

サーバーレスで
スケーラブルな
データ ウェアハウス
BigQuery

BigQuery は TB 規模から数百 PB 規模のデータを扱うデータ ウェアハウス

フル マネージド
データ ウェアハウス

ペタバイト規模の
クエリに対応

最大 100,000 行 / 秒の
ストリーム*

*クエリごと 10,000 行、
プロジェクトごと 100,000 行

地理空間分析の
ネイティブ対応

半構造化データ型の
ネイティブ対応*

*プレビュー

組み込みの機械学習
BigQuery ML

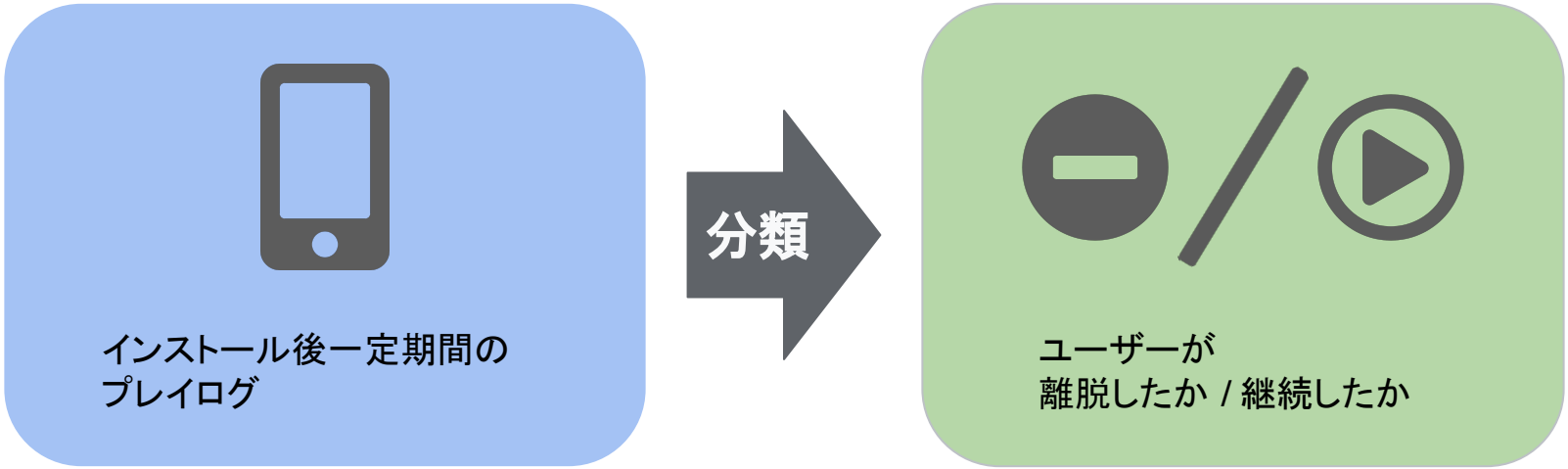
BigQuery 組み込みの機械学習機能 | BigQuery ML

ML の民主化

より多くのユーザーに ML 活用のチャンスを

- SQL で モデルをトレーニング・推論
- ML エンジニア不在のプロジェクトでも
  より手軽に ML の恩恵を得られる

BigQuery ML 活用例:
ゲームアプリの離脱ユーザーを予測(分類)する



分類モデルをトレーニングする

user_pseudo_id	country	cnt_user_ engagement	cnt_spend_ virtual_currency	...	churned
B71CXXX	United States	42	4		0
343BXXX	Japan	9	0		1
33AFXXX	France	12	0		0

入力特徴量

正解ラベル
(離脱の有無)

分類モデルをトレーニングする

user_pseudo_id	country	cnt_user_ engagement	cnt_spend_ virtual_currency	...	churned

```
CREATE OR REPLACE MODEL bqml_ds.churn_xgb
OPTIONS(
 MODEL_TYPE="BOOSTED_TREE_CLASSIFIER",
 LEARN_RATE=0.3,
 MAX_TREE_DEPTH=6,
 EARLY_STOP=True,
 INPUT_LABEL_COLS=["churned"],
 DATA_SPLIT_METHOD="RANDOM"
) AS
SELECT * EXCEPT(user_pseudo_id)
FROM bqml_ds.train_table
```

モデルの名前を指定

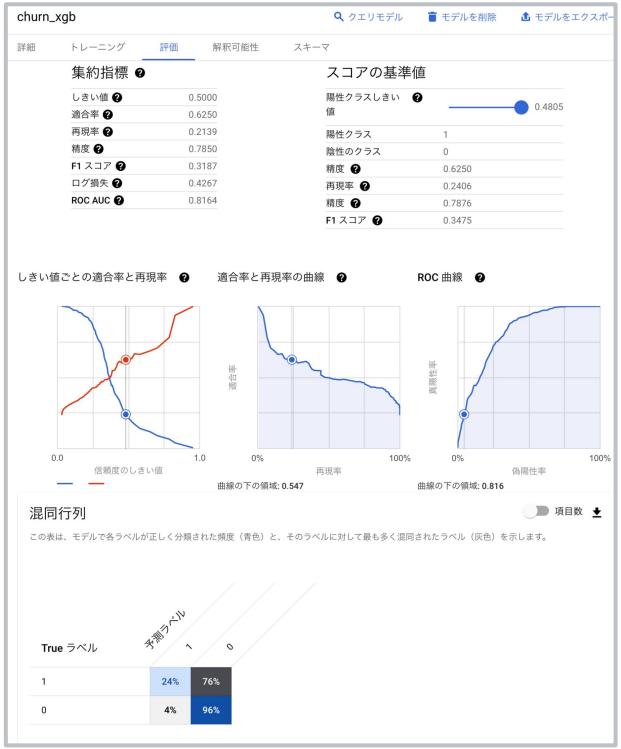
モデルの種類を指定

モデルのハイパー パラメータを指定
(Optional)

正解ラベル列を指定

トレーニングのデータをクエリ

トレーニングしたモデルを確認する



モデルのパフォーマンスを確認できる
これらの指標は SQL でもアクセスできる

Vertex AI Model Registry で
モデルのバージョン管理やデプロイ（後述）もできる
Google Cloud の ML サービスで生成されたモデルを
ここで一括で管理できる

トレーニングしたモデルで推論する - Batch 推論-

user_pseudo_id	country	cnt_user_ engagement	cnt_spend_ virtual_currency	...	churned
G52BXXX	United Kingdom	123	6		?
H35FXXX	South Africa	5	0		?

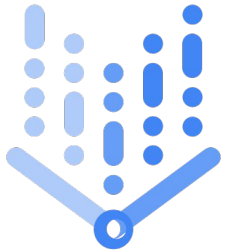
```
SELECT
  predicted_churned
FROM ML.PREDICT(MODEL `bqml_ds.churn_xgb`,
 (
 SELECT * EXCEPT (user_pseudo_id)
 FROM `bqml_ds.ml_features`
))
```

推論結果が入っているカラム
離脱確率を示すカラムも他に存在する
（モデルにより、出力される情報には差がある）

先にトレーニングしたモデルを指定

推論対象となるデータをクエリ

トレーニングしたモデルで推論する - Batch 推論以外の方法 -



Vertex AI Predictionで
モデルを REST API として公開

フルマネージドで、数クリックで設定完了。オートスケール・トラフィック分割機能あり。
スキュー・ドリフトの検出も可能で
MLOps に最適。

モデルをオープンソース形式で
無料でエクスポート

ローカル動作用コンテナ・スクリプトは別途提供。
モデルのデバッグにも便利 。
（AutoML の中身も確認できる）

DWH の ML 機能のメリットは?

1. データ移動の必要なし。大規模モデルもその場でトレーニングできる。
2. データと同じセキュリティ、ガバナンスポリシーを適用できる。
3. SQL で ML 。より多くのユーザーがより手軽にML を利活用できる。

Q. BigQuery ML は「オマケ」か？

A. 否

1. BigQuery ML、機能が限定的なのでは？
2. モデルの性能をより向上させたい、役立つ機能は？
3. モデルの判断根拠を知りたい、役立つ機能は？
4. 作ったモデルを共有したい、役立つ機能は？
5. SQL によるモデル開発と、Python での開発をシームレスに行う方法は？

BigQuery ML 、機能が限定的なのでは？

BigQuery ML supported models and features

Supervised Models

- Logistic regression（分類）
- Linear regression（回帰）
- XGBoost（分類・回帰）
- Random Forest（分類・回帰）
- DNN（分類・回帰）
- Wide-and-Deep（分類・回帰）
- AutoML Tables（分類・回帰）

Unsupervised & Other Models

- k-means++（クラスタリング）
- ARIMA+（時系列）
- PCA（次元削減）
- AutoEncoder（次元削減）
- Matrix Factorization（協調フィルタリング・推薦）

Other Features

- 外れ値検知 (ARIMA+ / k-means / PCA / Autoencoder)
- Explainable AI
- Feature Preprocessing
- and more ...

This is a partial full list of supported models and features. For complete details check out BQML Docs

モデルの性能をより向上させたい、
役立つ機能はある？

ハイパー パラメータ チューニングで性能向上に挑戦

最適なハイパー パラメータを見つけられれば
モデルの性能を向上できる
でも、膨大な組み合わせからどうやって
最適なパラメータを見つければ..？

Google Research 開発の最適化システム
“Vizier” を BigQuery ML から呼び出せる

ハイパー パラメータ チューニングで性能向上に挑戦

```
CREATE OR REPLACE MODEL dbtech22.churn_xgb2_tuned
OPTIONS(
 MODEL_TYPE="BOOSTED_TREE_CLASSIFIER",
 EARLY_STOP=True,
 INPUT_LABEL_COLS=["churned"],
 DATA_SPLIT_METHOD="RANDOM",
 ENABLE_GLOBAL_EXPLAIN=True,
 NUM_TRIALS=32, # 試行回数
 MAX_PARALLEL_TRIALS=4, # 並列数
 HPARAM_TUNING_OBJECTIVES=["roc_auc"], # 最適化目標
 HPARAM_TUNING_ALGORITHM = 'VIZIER_DEFAULT', # 最適化手段
 LEARN_RATE=HPARAM_RANGE(0.01, 0.5), # 探索範囲
 MAX_TREE_DEPTH=HPARAM_CANDIDATES([5,6,7]) # 探索範囲
) AS
SELECT
 * EXCEPT(user_pseudo_id, data_split)
FROM dbtech22.train_table
```

- 試行回数とその並列数
- 最適化目標
- 最適化手段
- 探索範囲

をモデル学習時のクエリに追加

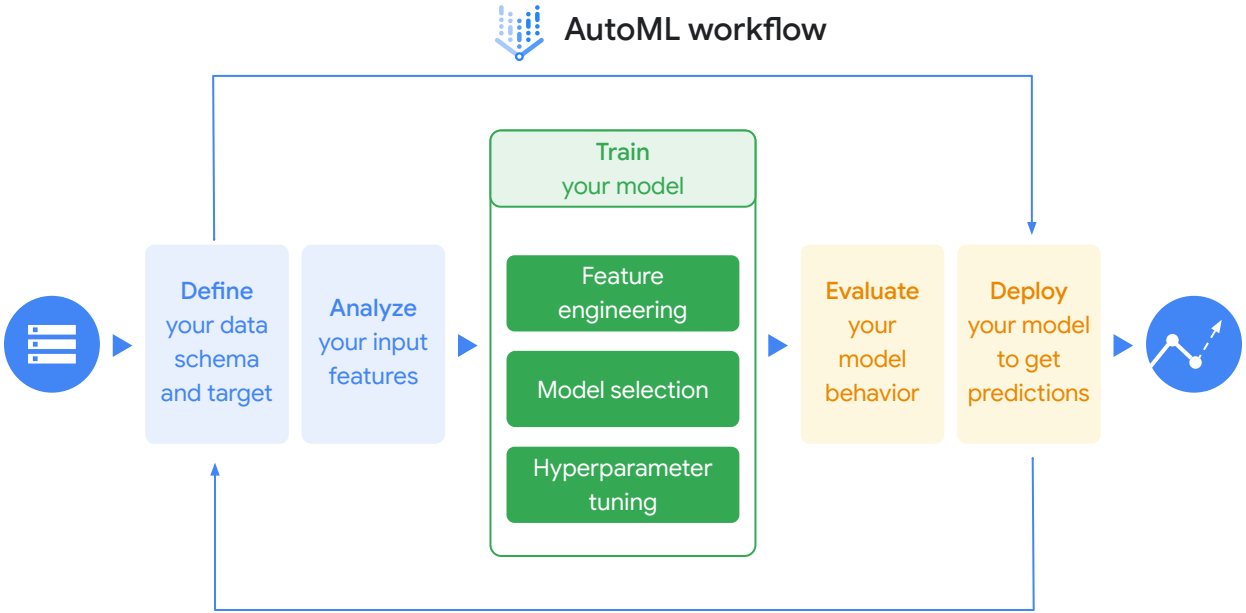
ハイパー パラメータ チューニングで性能向上に挑戦

トライアル ID	学習率	最大ツリー深度	精度	再現率	精度	F1 スコア	ログ損失	ROC AUC	ROC AUC（評価）
1	0.3000	6.0000	0.6182	0.2012	0.7895	0.3036	0.4169	0.8360	0.8033
2	0.1923	7.0000	0.6406	0.2426	0.7962	0.3519	0.4176	0.8305	0.7998
3	0.4153	5.0000	0.6552	0.2249	0.7962	0.3348	0.4113	0.8290	0.8059
4	0.3185	6.0000	0.6111	0.1953	0.7881	0.2960	0.4097	0.8353	0.8078
5	0.3657	7.0000	0.5588	0.2249	0.7827	0.3207	0.4185	0.8212	0.7916
6	0.3456	6.0000	0.6066	0.2189	0.7895	0.3217	0.4154	0.8288	0.8024
7	0.5000	5.0000	0.6393	0.2308	0.7949	0.3391	0.4203	0.8206	0.7971
8	0.0257	6.0000	0.5849	0.1834	0.7841	0.2793	0.5839	0.8001	0.7836
9	0.3267	6.0000	0.6182	0.2012	0.7895	0.3036	0.4203	0.8371	0.7918
10	0.3161	7.0000	0.6029	0.2426	0.7908	0.3460	0.4152	0.8270	0.7929
11	0.4460	7.0000	0.5972	0.2544	0.7908	0.3568	0.4186	0.8113	0.7972
12	0.1399	5.0000	0.6491	0.2189	0.7949	0.3274	0.4317	0.8297	0.8021
13	0.3254	6.0000	0.6508	0.2426	0.7976	0.3534	0.4101	0.8361	0.8020
14	0.4365	6.0000	0.6563	0.2485	0.7989	0.3605	0.4150	0.8190	0.7995
15	0.1004	7.0000	0.5965	0.2012	0.7868	0.3009	0.4416	0.8281	0.8012
16	0.3704	5.0000	0.5957	0.1657	0.7841	0.2593	0.4151	0.8318	0.8035
17	0.2884	5.0000	0.6346	0.1953	0.7908	0.2986	0.4140	0.8361	0.8102
18	0.3692	5.0000	0.6250	0.2071	0.7908	0.3111	0.4191	0.8242	0.8015
19	0.2624	5.0000	0.6429	0.2130	0.7935	0.3200	0.4156	0.8305	0.8119
20	0.3861	5.0000	0.6667	0.1775	0.7922	0.2804	0.4161	0.8271	0.8035
21	0.2435	5.0000	0.6667	0.2249	0.7976	0.3363	0.4131	0.8354	0.8095
22	0.2661	5.0000	0.6250	0.2071	0.7908	0.3111	0.4141	0.8345	0.8097
23	0.2658	5.0000	0.6429	0.2130	0.7935	0.3200	0.4204	0.8252	0.8109
24	0.2657	5.0000	0.6250	0.2071	0.7908	0.3111	0.4201	0.8267	0.8092
25	0.2518	5.0000	0.6491	0.2189	0.7949	0.3274	0.4206	0.8318	0.8073
26（最適）	0.2610	5.0000	0.6415	0.2012	0.7922	0.3063	0.4163	0.8306	0.8127
27	0.2627	5.0000	0.6481	0.2071	0.7935	0.3139	0.4172	0.8258	0.8092
28	0.2632	5.0000	0.6379	0.2189	0.7935	0.3260	0.4161	0.8313	0.8105
29	0.2601	5.0000	0.6364	0.2071	0.7922	0.3125	0.4189	0.8341	0.8060
30	0.2618	5.0000	0.6458	0.1834	0.7908	0.2857	0.4187	0.8348	0.8059
31	0.2625	5.0000	0.6491	0.2189	0.7949	0.3274	0.4165	0.8271	0.8091
32	0.2628	5.0000	0.6481	0.2071	0.7935	0.3139	0.4171	0.8258	0.8092

32 試行のログ

見つかった最適パラメータを使った
モデルが保存される
全試行のパラメータ、モデルの性能は
左図のように参照できる
（SQL でも参照できる）

ハイパー パラメータ チューニングをすべて自動化する方法も -AutoML-



Automatically search through Google's whole model zoo...

Linear, logistic

Feedforward DNN

Wide and Deep NN

Gradient Boosted Decision Tree (GBDT)

DNN + GBDT Hybrid

Adanet ensemble

Neural + Tree Architecture Search

...and more!

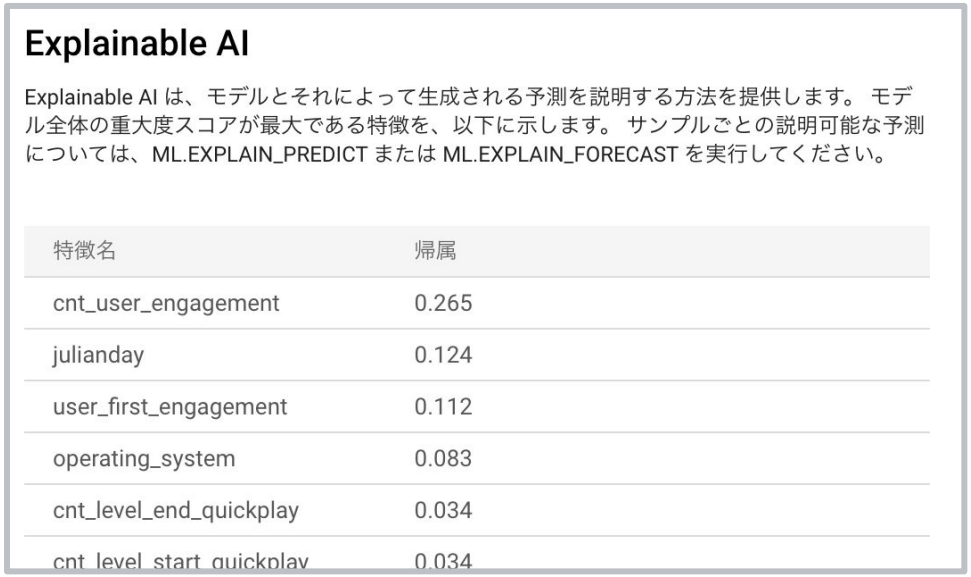
モデルの判断根拠を知りたい、
役立つ機能はある？

BigQuery ML の Explainable AI

Explainable AI（説明可能性）

- **グローバルな説明可能性**
  モデルが何を根拠に判断を
  下す傾向があるか
- **ローカルな説明可能性**
  このサンプル（人・モノ）に対し
  モデルがどう判断したか

Shapley 値、Integrated Gradient、
ジニ係数、p 値などの指標で
モデル解釈を支援



ゲーム離脱予測モデルの
グローバルな説明
(Tree SHAP)

作ったモデルを共有したい、
役立つ機能はある？

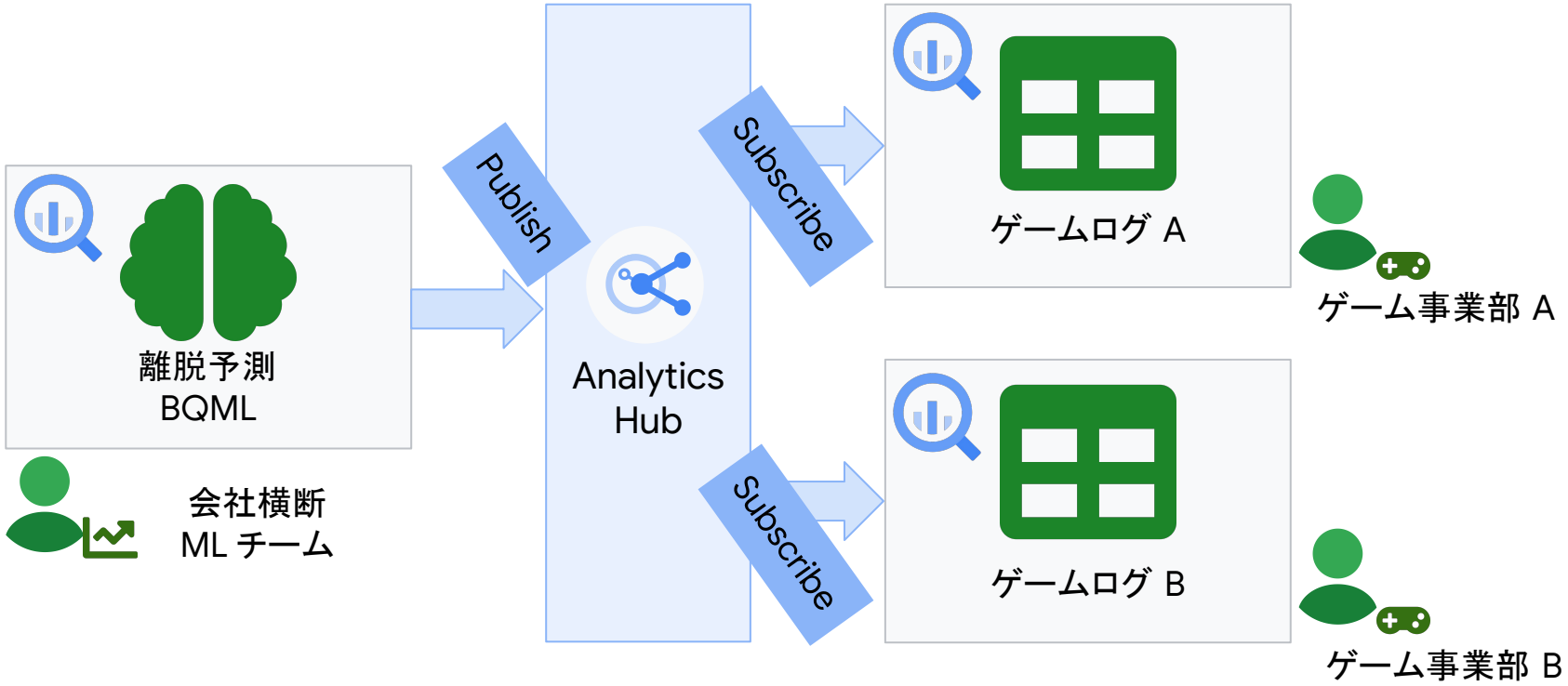
想定シナリオ: ML チームが作ったモデルを展開したい

会社横断
ML チーム

ゲーム事業部 A

ゲーム事業部 B

想定シナリオ: ML チームが作ったモデルを展開したい

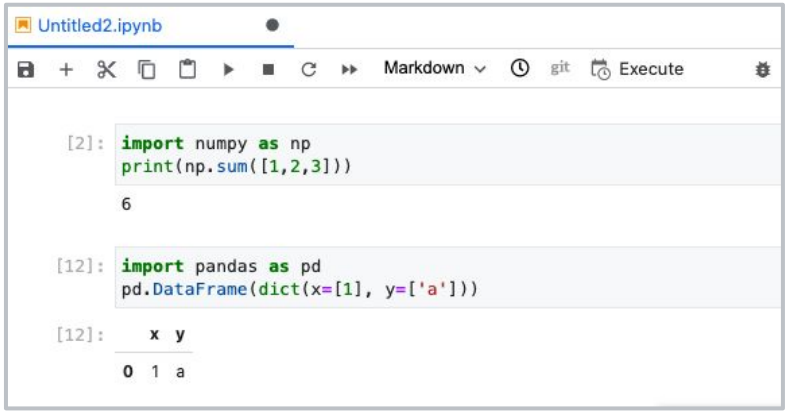


Publish したモデルは、Subscriber の BigQuery ワークスペースに link される。
共有用の追加インフラ管理不要。モデル更新は即時反映。
利用説明の Markdown や テーブルデータもまとめて共有。

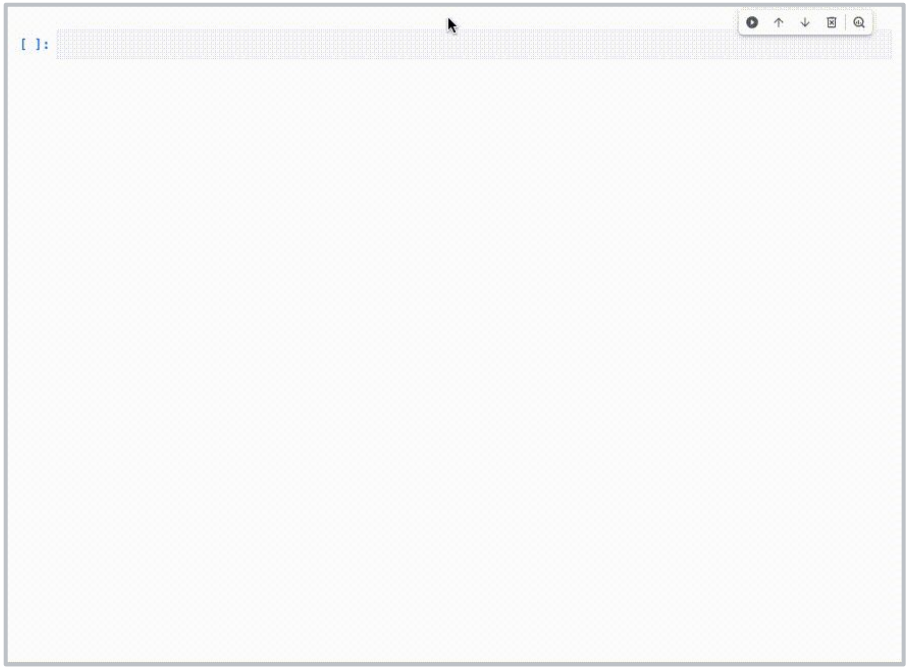
SQL コンソールでのモデル開発、
普段の Python を使ったモデル開発プロセスと
断絶があるんだよな...

by ML エンジニア

Vertex Workbench で SQL (BigQuery ML) と Python 両方まとめて開発



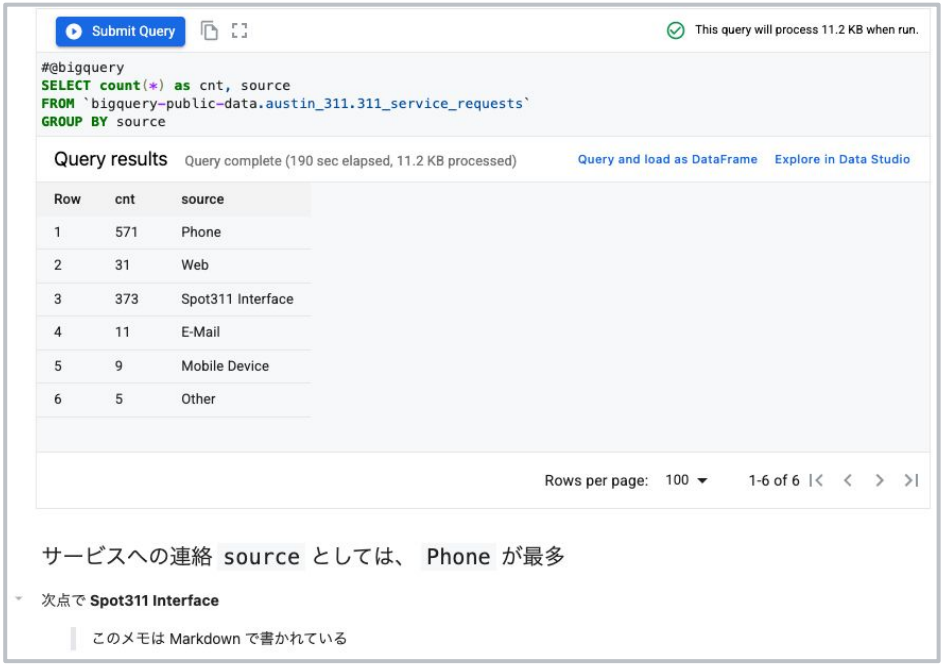
通常の Jupyter Notebook 同様
Python で開発



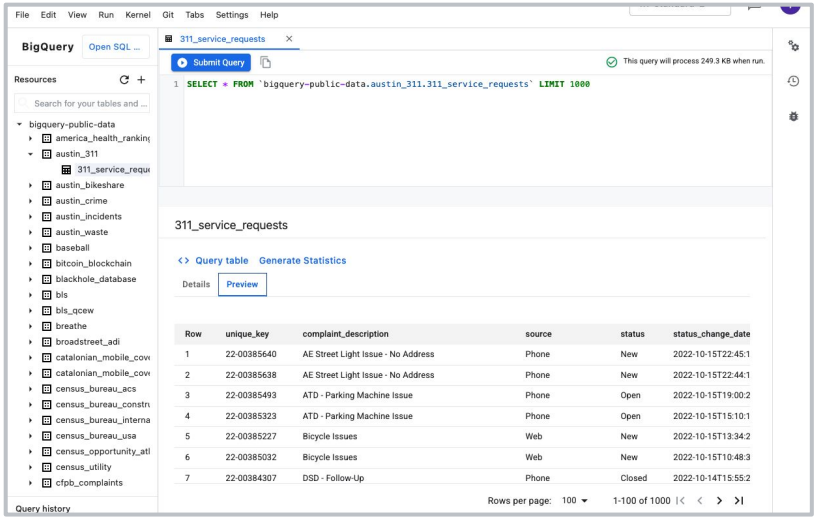
同じ Notebook 内で SQL を実行し、
結果もその場で確認

結果を Python (pandas.DataFrame) で取り込むためのコードをワンクリックで生成

実は SQL データ分析でも役立つ Vertex Workbench



SQL の実行結果とそれに対するメモ（Markdown）
を並べて記述
データはクラウドにおいたまま分析



テーブルのブラウジングや
SQL エディタなど、
本格的な開発環境も提供

1. BigQuery ML、機能が限定的なのでは？
2. モデルの性能をより向上させたい、役立つ機能は？
3. モデルの判断根拠を知りたい、役立つ機能は？
4. 作ったモデルを共有したい、役立つ機能は？
5. SQL によるモデル開発と、Python での開発をシームレスに行う方法は？

Thank you.

Google Cloud