

IT 資格 道 場

Google Cloud Professional Machine Learning Engineer 学習用 テキスト

予備知識ゼロから合格ラインへ —— 読んで理解する1冊目

無料サンプル(第1章まで)

Google Cloud 公式 Professional Machine Learning Engineer Certification exam
guide(as of June 1, 2026・英語版) 準拠
2026年10月版

ご利用にあたって

- このテキストは「IT資格道場」(www.shikaku-doujou.com)の購入者限定テキストのうち、学習用テキストの第1章までを収録した**無料サンプル**です。どなたでもご覧いただけます。
- 無料サンプルも著作物です。転載・改変しての再配布・二次販売はできません。
- AI モデルの学習データとしてのご利用(学習目的の入力・データセット化を含む)はできません。
- 製品版(学習用テキスト・頻出度別ノート・暗記用ノートの3点)は、ご購入者を識別できる透かし入りのPDFとして提供されます。
- 本PDFはスマートフォン・タブレットでの閲覧に合わせたA5版面・文字サイズで組版しています。

目次

この教材の使い方

第1章 Architecting low-code AI solutions [1.1/1.2]

ビジネス課題から BigQuery ML と Agent Platform AutoML を選ぶ [1.1]

BigQuery ML の分類・回帰モデル [1.1]

BigQuery ML の時系列予測 [1.1]

BigQuery ML のクラスタリングと行列分解・異常検知 [1.1]

BigQuery ML の特徴量エンジニアリング [1.1]

BigQuery ML の特徴量選択と特徴量の重要度 [1.1]

BigQuery ML の評価 [1.1]

BigQuery ML の予測の生成 [1.1]

Agent Platform AutoML の表形式データのトレーニング [1.1]

Agent Platform AutoML の画像・テキスト・動画のデータセットとトレーニング [1.1]

BigQuery を使用した Gemini モデルのファインチューニング [1.1]

BigQuery の生成 AI 関数とリモートモデル [1.1]

Model Garden のモデルの評価と選択 [1.2]

Gemini のモデルの選び分け [1.2]

Document AI API [1.2]

Vision API・Video Intelligence API・Speech-to-Text などの事前トレーニング済み API [1.2]

Translation API [1.2]

用途別のソリューションの構築とモデルのチューニング [1.2]

Model Garden のサービスとしてのモデル (MaaS) [1.2]

プロンプト設計・グラウンディング・RAG [1.2]

Gemini アプリの費用の最適化 [1.2]

Gemini アプリのレイテンシと可用性 [1.2]

サンプルはここまでです

SAMPLE

このテキストの位置づけ: 3 点セットの本編。これだけで問題が解けるようになることが目標で、サービスの選び分け・設定値・制限値を合格ラインの深さで扱う。

進め方: ① 学習用を読む → ② 頻出度別で解き方を掴む → ③ 問題演習で腕試し → ④ 頻出度別に戻って再確認(試験直前もここ)。暗記用は空き時間や用語の再確認に使う。

この教材の使い方

Professional Machine Learning Engineer は、Google Cloud で ML の仕組みを設計し、作り、本番で動かし続ける力を問う認定です。

問われるのは次の 3 つ。

- 要件を読んで、どのサービスや構成を選ぶか
- 選んだ部品の設定値・制限値・既定の動き
- 動かなくなったときの切り分けと、見張り方

用語の暗記だけでは足りません。中心は、条件を 3 つほど重ねた場面から最も適切な 1 つを選ぶ問い。各章の節は「何を選ぶか」から入り、場面で判断する段落で終わる構成です。

準拠する版:2026 年 6 月 1 日時点の試験ガイド

このテキストの準拠先は、Google Cloud の公式サイトにある **Certification exam guide as of June 1, 2026**。

- 2026 年の改訂で、Vertex AI は **Gemini Enterprise Agent Platform**(試験ガイドでは Agent Platform と略す)に改称

- Cloud Composer は **Managed Service for Apache Airflow** に、Dataproc は **Managed Service for Apache Spark** に改称
- 旧名の資料や、API のフィールド名・モジュール名(`aiplatform` など)に残る旧名は、読み替えが必要

試験の基本情報

項目	内容
試験名	Professional Machine Learning Engineer
出題範囲の版	試験ガイド as of June 1, 2026
問題数	50～60 問
試験時間	2 時間
出題形式	択一(multiple choice)と複数選択(multiple select)。実技・シミュレーションの記載はなし
合格ライン	公表なし
受験言語	英語・日本語
推奨される経験	業界での経験 3 年以上(うち Google Cloud での設計・管理 1 年以上)

受験料・受験方法・認定の有効期間は、申し込み前に Google Cloud の公式サイトで最新情報を確認してください。

出題範囲と対応する章

分野	内容	配点(公式の約値)	対応する章
1	ローコードの AI ソリューションの設計	約 13%	第 1 章
2	チーム内・チーム間でのデータとモデルの管理	約 16%	第 2 章
3	プロトタイプ of ML モデルへのスケーリング	約 21%	第 3 章
4	モデルのサービングとスケーリング	約 20%	第 4 章
5	ML パイプラインの自動化とオーケストレーション	約 18%	第 5 章
6	AI ソリューションのモニタリング	約 13%	第 6 章

章の並びは、試験ガイドの並びと同じ。節見出しの末尾にある [1.1] のような番号は、試験ガイドの細目の番号です。公式が公表しているのは分野ごとの配点だけで、細目ごとの出題数は公表されていません。

サービス名・機能名は英語のまま覚えてください。 本番の設問文と選択肢に出るのは英語の表記。日本語訳だけを覚えても、画面では出会いません。

第1章 Architecting low-code AI solutions

[1.1/1.2]

試験ガイドの配点は約 13%。細目は 1.1・1.2。

この章の中心は「どの道具を選ぶか」です。出題の軸は振り分けで、データが BigQuery にあれば BigQuery ML を選びます。画像や表形式なら Agent Platform の AutoML、文章や画像の理解や生成なら Gemini と専用 API です。製品名は 2026 年の改訂で変わりました。Vertex AI は Gemini Enterprise Agent Platform (略して Agent Platform) になりました。Cloud Composer は Managed Service for Apache Airflow になりました。旧名の資料は読み替えてください。[^c1-1]

ビジネス課題から BigQuery ML と Agent Platform AutoML を選ぶ [1.1]

BigQuery ML は、SQL の `CREATE MODEL` 文でモデルを作る機能です。データを BigQuery の外へ出さずに学習から予測まで終わります。Agent Platform AutoML は、データセットを作り、目的を選び、予算を決めるとモデルを自動で探索してくれるマネージドの機能です。コードは書きません。

選び分けの軸は 3 つ。データがどこにあるか、データの種類は何か、使う人が SQL を書くかです。

状況	選ぶ道具
データが BigQuery にあり、分析担当が SQL で完結させたい	BigQuery ML

状況	選ぶ道具
表形式で、SQL を書かずに高い精度を自動探索に任せたい	Agent Platform AutoML (表形式)
画像・表形式のデータを分類したい	Agent Platform AutoML
表形式で手法を自分で選びたい(木、DNN、線形など)	BigQuery ML の各 MODEL_TYPE
BigQuery のデータで自動探索したい	BigQuery ML の AUTOML_CLASSIFIER / AUTOML_REGRESSOR

最後の行は引っかけになりやすい所です。BigQuery ML の中にも AutoML がありますが、Agent Platform AutoML とは別物なので、混同してはいけません。前者は SQL から呼び、後者はデータセットとトレーニング ジョブで動きます。

課題の型と手法の対応は次のとおりです。

課題	BigQuery ML の手法
分類(解約するか、不正か)	ロジスティック回帰、ブースト ツリー、ランダム フォレスト、DNN
回帰(売上額、価格)	線形回帰、ブースト ツリー、ランダム フォレスト、DNN
時系列予測(来月の需要)	ARIMA_PLUS 、 ARIMA_PLUS_XREG
クラスタリング(顧客の分け方)	KMEANS
推薦(誰に何を勧めるか)	MATRIX_FACTORIZATION
異常検知	ARIMA_PLUS 、 KMEANS 、 PCA 、 AUTOENCODER

場面で考えます。小売企業が BigQuery に 3 年分の購買履歴を持ち、データアナリストは SQL しか書けません。来月の店舗別需要を予測したいなら、データを移さずに `ARIMA_PLUS` で作れます。一方、画像が写真ファイルで Cloud Storage にあり、画像分類のモデルを学習させたいなら、BigQuery ML の組み込みのモデルでは作れません。選ぶのは AutoML の画像分類です。

別の場면을 3 つ続けます。1 つ目は、BigQuery の表で解約予測を作り、営業部門に「なぜこの顧客が高リスクか」を説明する必要があるケースです。ブースト ツリーを BigQuery ML で作ります。顧客ごとの寄与は `ML.EXPLAIN_PREDICT`、全体で効いている列は `ML.GLOBAL_EXPLAIN` で見せれば、1 つの SQL 環境で完結。

2 つ目は、手元の表が 400 行しかないケース。Agent Platform AutoML の表形式は 1,000 行以上が必須なので、使えません。BigQuery ML の線形やブースト ツリーなど、手法を選んで作る側に回るしかありません。

3 つ目は、数 TB・1,000 列近い表で、特徴量の作り方や探索の範囲まで管理したいケースです。通常の AutoML では制御しきれないため、選ぶのは Tabular Workflows のエンドツーエンド AutoML。

BigQuery ML の分類・回帰モデル [1.1]

基本の形はコードのとおり。ラベル列は `INPUT_LABEL_COLS` で指定し、省略すると `label` という名前の列を探します。

```
CREATE OR REPLACE MODEL `proj.ds.churn`  
OPTIONS (  
  model_type = 'BOOSTED_TREE_CLASSIFIER',  
  input_label_cols = ['churned'],  
  data_split_method = 'RANDOM',  
  data_split_eval_fraction = 0.2  
) AS  
SELECT * FROM `proj.ds.customers`;
```

教師あり学習の `MODEL_TYPE` の値を表にします。[^c1-2]

系統	値(分類／回帰)
線形	<code>LOGISTIC_REG</code> (分類)／ <code>LINEAR_REG</code> (回帰)
ブースト ツリー	<code>BOOSTED_TREE_CLASSIFIER</code> ／ <code>BOOSTED_TREE_REGRESSOR</code>
ランダム フォレスト	<code>RANDOM_FOREST_CLASSIFIER</code> ／ <code>RANDOM_FOREST_REGRESSOR</code>
DNN	<code>DNN_CLASSIFIER</code> ／ <code>DNN_REGRESSOR</code>
ワイド&ディープ	<code>DNN_LINEAR_COMBINED_CLASSIFIER</code> ／ <code>DNN_LINEAR_COMBINED_REGRESSOR</code>
自動探索	<code>AUTOML_CLASSIFIER</code> ／ <code>AUTOML_REGRESSOR</code>

教師なしは `KMEANS` ・ `MATRIX_FACTORIZATION` ・ `PCA` ・ `AUTOENCODER` 、時系列は `ARIMA_PLUS` ・ `ARIMA_PLUS_XREG` です。ほかに `TENSORFLOW` ・ `TENSORFLOW_LITE` ・

ONNX・XGBOOST という値があります。外部で作ったモデルの取り込み用です。
CONTRIBUTION_ANALYSIS も MODEL_TYPE の値として並びます。

データの分割

DATA_SPLIT_METHOD は 5 種類。既定は AUTO_SPLIT です。[^c1-2]

値	動き
AUTO_SPLIT	件数で自動。500 行未満は全行を学習に使う。チューニングなしで 500~50,000 行は評価 20%・学習 80%。50,000 行超は評価に 10,000 行を使い、残りを学習に使う。チューニングありで 500 行超は評価 10%・テスト 10%・学習 80%
RANDOM	ランダムに分ける。FARM_FINGERPRINT による決定的な分割で、同じデータなら同じ結果になる
CUSTOM	列で指定する。チューニングなしなら BOOL 列(TRUE と NULL が評価、FALSE が学習)。チューニングありなら STRING 列 (TRAIN・EVAL・TEST)
SEQ	指定列 (NUMERIC・BIGNUMERIC・STRING・TIMESTAMP) を昇順に並べ、末尾の n 行を評価に使う
NO_SPLIT	全入力を学習に使う (評価データなし)

計算例を見ます。チューニングなしで 2,000 行なら学習 1,600・評価 400、60,000 行なら学習 50,000・評価 10,000 です。評価の割合は DATA_SPLIT_EVAL_FRACTION (既定 0.2) で決まります。テストの割合は DATA_SPLIT_TEST_FRACTION (既定 0) です。どちらも効くのは RANDOM か SEQ の

ときだけ。チューニングで評価 0.2・テスト 0.08 を指定した場合、学習は $1-0.2-0.08=72\%$ 。

時間で並んだデータを `RANDOM` で分けると、未来のデータが学習に混ざります。時系列の性質を持つ表形式データでは、`SEQ`（時刻列を指定）か `CUSTOM` を選ぶのが筋。

場面で確かめます。直近 3 か月を評価用にして、それより前を学習に使いたいなら、日付(TIMESTAMP 型)の列を指定した `SEQ`。顧客 ID の単位で学習と評価を分けたい(同じ顧客が両方に入ると評価が甘くなる)なら、顧客単位で自分が振り分けた列を `CUSTOM` に渡します。ハイパーパラメータ チューニングを併用するときの `CUSTOM` は、`BOOL` でなく `STRING` 列(`TRAIN`・`EVAL`・`TEST`)です。`NO_SPLIT` は全行を学習に使うので、評価データが残りません。評価が要るときは選べません。

BigQuery ML の AutoML

`AUTOML_CLASSIFIER` と `AUTOML_REGRESSOR` の主な設定は次の表のとおり。[^{c1-3}]

設定	内容
<code>BUDGET_HOURS</code>	学習予算。1.0～72.0 の <code>FLOAT64</code> 、既定は 1.0。学習後のモデル圧縮に学習時間の最大 50% かかることがあり、その時間は予算に含まれない
<code>OPTIMIZATION_OBJECTIVE</code> (回帰)	<code>MINIMIZE_RMSE</code> (既定)・ <code>MINIMIZE_MAE</code> ・ <code>MINIMIZE_RMSLE</code>

設定	内容
同(二値分類)	<code>MAXIMIZE_AU_ROC</code> (既定)・ <code>MINIMIZE_LOG_LOSS</code> ・ <code>MAXIMIZE_AU_PRC</code> ・ <code>MAXIMIZE_PRECISION_AT_RECALL</code> ・ <code>MAXIMIZE_RECALL_AT_PRECISION</code>
同(多クラス分類)	<code>MINIMIZE_LOG_LOSS</code>
<code>DATA_SPLIT_COL</code>	省略するとランダム分割。指定するなら TIMESTAMP (時系列の分割) か STRING (<code>TRAIN</code> ・ <code>VALIDATE</code> ・ <code>TEST</code> ・ <code>UNASSIGNED</code>)

適合率と再現率のどちらかを固定したいときは、目的に文字列と数値の組を渡します。数値の範囲は 0 以上 1 以下。

```
OPTIMIZATION_OBJECTIVE = STRUCT('MAXIMIZE_PRECISION_AT_RECALL', 0.3)
```

これは再現率 0.3 を満たしたうえで適合率を最大にする指定です。不正検知で「見逃しを一定以下に抑えたまま誤検知を減らしたい」と言われたら、この形が答えになる典型例。

BigQuery ML の時系列予測 [1.1]

`ARIMA_PLUS` は、トレンド・季節性・休日・ステップ変化・急増急減に分解して予測する時系列モデルです。気温や広告費のような外部の説明変数 (外部リグレッサー) を加えたいなら、`ARIMA_PLUS_XREG` を選びます。[^c1-4]

```
CREATE OR REPLACE MODEL `proj.ds.demand`
OPTIONS (
  model_type = 'ARIMA_PLUS',
  time_series_timestamp_col = 'order_date',
  time_series_data_col = 'qty',
  time_series_id_col = 'sku',
  horizon = 90,
  data_frequency = 'DAILY',
  holiday_region = 'JP'
) AS
SELECT order_date, qty, sku FROM `proj.ds.orders`;

SELECT * FROM ML.FORECAST(MODEL `proj.ds.demand`,
  STRUCT(30 AS horizon, 0.9 AS confidence_level));
```

必須の指定は 2 つ。1 つ目は時刻列 `TIME_SERIES_TIMESTAMP_COL` で、型は `TIMESTAMP・DATE・DATETIME` です。2 つ目は値の列 `TIME_SERIES_DATA_COL` で、型は `INT64・NUMERIC・BIGNUMERIC・FLOAT64` です。`TIME_SERIES_ID_COL` に商品コードなどを渡せば、1 回のクエリで多数の時系列を別々に学習して予測できます。

設定値の数字は紛らわしいので、表で見比べてください。

場所	項目	既定	上限・値
<code>CREATE MODEL</code>	<code>HORIZON</code>	1,000	10,000

場所	項目	既定	上限・値
CREATE MODEL	DATA_FREQUENCY	AUTO_FREQUENCY	PER_MINUTE・ HOURLY・DAILY・ WEEKLY・ MONTHLY・ QUARTERLY・ YEARLY
CREATE MODEL	HOLIDAY_REGION	休日効果を使わない	'JP'・'US'・'GLOBAL' など
ML.FORECAST	HORIZON	3	CREATE MODEL の 値、未指定なら 1000
ML.FORECAST	CONFIDENCE_LEVEL	0.95	0 以上 1 未満

予測は、CREATE MODEL の実行時にすでに終わっているもの。ML.FORECAST は値を取り出して予測区間を計算するだけ。そのため ML.FORECAST の HORIZON は、多数の時系列から必要な範囲だけを絞る用途が主になります。[1-5] 最も細かい粒度は 1 分で、秒単位の時系列は扱えません。日本の祝日を反映したいなら HOLIDAY_REGION に 'JP' を渡してください。指定しないと休日効果は入りません。指定しても、休日効果が使われるのは日次か週次で、1 年より長い系列だけです。

BigQuery ML のクラスタリングと行列分解・異常検知 [1.1]

KMEANS は、正解ラベルなしで似たデータをグループに分けます。顧客を購買パターンで分けて、施策を変えたい場面に合います。主な設定は 4 つです。

- NUM_CLUSTERS : グループ数

- `KMEANS_INIT_METHOD` : `RANDOM` • `KMEANS++` • `CUSTOM`
- `DISTANCE_TYPE` : `EUCLIDEAN` • `COSINE`
- `STANDARDIZE_FEATURES` : 標準化するか

単位の違う列が混ざるなら標準化を使います。[^c1-31] 予測では、`ML.PREDICT` が各行の所属クラスタ (`CENTROID_ID`) を返します。

`MATRIX_FACTORIZATION` は、ユーザーと商品の評価表から協調フィルタリングで推薦を作ります。`FEEDBACK_TYPE` が `'EXPLICIT'` なら星の数のような明示的な評価で、`'IMPLICIT'` なら閲覧や購入のような暗黙の行動を表します。推薦の取り出しは `ML.RECOMMEND` の仕事。[^c1-32]

異常検知は `ML.DETECT_ANOMALIES` で行います。`ARIMA_PLUS` なら時系列の外れ値、`KMEANS` • `PCA` • `AUTOENCODER` なら各行の外れ度が判定されます。ラベルが要らないので、不正のサンプルが少ない場面に向く方式です。厳しさは `CONTAMINATION` (異常の想定割合) や `ANOMALY_PROB_THRESHOLD` (異常と見なす確率のしきい値) で変えます。[^c1-33]

BigQuery ML の特徴量エンジニアリング [1.1]

前処理には 2 つの方式があります。学習時に自動でかかる自動前処理と、`TRANSFORM` 句や前処理関数で自分で書く手動前処理です。`ML.FEATURE_INFO` を使うと、入力列の統計情報を見られます。[^c1-8]

`TRANSFORM` 句の価値は、予測時に同じ変換が自動で再現されることです。

```
CREATE OR REPLACE MODEL `proj.ds.m`
TRANSFORM (
  ML.FEATURE_CROSS(STRUCT(city, device)) AS cross_f,
  ML.QUANTILE_BUCKETIZE(age, 5) OVER() AS age_bucket,
  churned
)
OPTIONS (model_type = 'LOGISTIC_REG', input_label_cols = ['churned'])
AS
SELECT city, device, age, churned FROM `proj.ds.customers`;

SELECT * FROM ML.PREDICT(MODEL `proj.ds.m`,
  (SELECT city, device, age FROM `proj.ds.new_customers`));
```

予測側は元の列(city、device、age)だけを渡せば、分位点の境界値まで含めて学習時の変換が復元されます。変換を SQL の別クエリに書いて予測時に書き忘れる事故を防げます。[^c1-1]

TRANSFORM には決まりが3つあります。

- 句がある場合、学習に使われるのは句の出力列だけで、**query_statement**の結果にあっても句に現れない列は無視されます。
- 句の入力列は **STRUCT・ARRAY** 以外の単純な型か、その **ARRAY** に限られます。
- **label**・**data_split_col**・**kmeans_init_col**・**instance_weight_col** は列名のまま(または *****)で参照します。別名は付けられません。

ML.EVALUATE の入力に指定できる列も、**TRANSFORM** 句に含まれる列だけです。

TRANSFORM を使う利点は、学習時と予測時で前処理がずれる事故を防げる点にあります。分位点の境界値のように、学習データ全体から計算する統計は、予測側で同じ値を再現しなければなりません。**TRANSFORM** に書いておけば、その統計もモデルに保存されます。予測の入力に必要なのは元の列だけで、変換済みの列を用意する必要はありません。

自動前処理と手動前処理の選び分けも出ます。標準的な変換で足りるなら自動に任せます。特徴量の交差やバケット化のように、意図して作りたい特徴量があるなら **TRANSFORM** を使います。

BigQuery ML の特徴量選択と特徴量の重要度 [1.1]

どの列が効いているかを見る道具はモデルで違います。[¹c1-7]

モデル	説明の方法	関数
ブースト ツリー・ランダム フォレスト	ジニ係数ベースの重要度 (全体)	<code>ML.FEATURE_IMPORTANCE</code>
ブースト ツリー・ランダム フォレスト	Tree SHAP または近似の寄与 (行ごと／全体)	<code>ML.EXPLAIN_PREDICT</code> ／ <code>ML.GLOBAL_EXPLAIN</code>
線形・ロジスティック回帰	Shapley 値、標準誤差と p 値	<code>ML.EXPLAIN_PREDICT</code> ／ <code>ML.ADVANCED_WEIGHTS</code>
DNN・ワイド&ディープ	統合勾配 (文書によっては記載が無い)	<code>ML.EXPLAIN_PREDICT</code> ／ <code>ML.GLOBAL_EXPLAIN</code>
AutoML	サンプリングされた Shapley	<code>ML.GLOBAL_EXPLAIN</code>
<code>ARIMA_PLUS</code>	時系列の分解	<code>ML.EXPLAIN_FORECAST</code>

ローカルは 1 行ごとの寄与で、グローバルは評価データ全体の平均的な絶対値の寄与です。寄与が小さい列を落として再学習すれば、学習も予測も軽くなりますが、精度の確認は欠かせません。偏りの確認や、モデルが期待どおり動くかの検証にも使えます。

BigQuery ML の評価 [1.1]

`ML.EVALUATE` にモデルを渡すと指標が返ります。テーブルもクエリも指定しなければ、学習中に作られた評価指標しか返りません。[^c1-34]

種類	返る主な指標
分類	<code>precision</code> • <code>recall</code> • <code>accuracy</code> • <code>f1_score</code> • <code>log_loss</code> • <code>roc_auc</code>
回帰	<code>mean_absolute_error</code> • <code>mean_squared_error</code> • <code>mean_squared_log_error</code> • <code>median_absolute_error</code> • <code>r2_score</code> • <code>explained_variance</code>
時系列	<code>mean_absolute_percentage_error</code> • <code>symmetric_mean_absolute_percentage_error</code> • <code>mean_absolute_scaled_error</code> など

二値分類では、`ML.EVALUATE` の `THRESHOLD` (既定 0.5) で境界を変えて評価できます。`ML.CONFUSION_MATRIX` が返すのは混同行列で、行が実際のラベル (`expected_label`)、列が予測したラベルごとの件数です (多クラスにも使える)。`ML.ROC_CURVE` はしきい値ごとの再現率と偽陽性率を返すので、しきい値を決める材料になります。

見逃しを減らしたいなら、しきい値を下げて再現率を上げます。誤検知を減らしたいなら、上げて適合率を取ります。陽性が 1% のように偏ったデータでは、`accuracy` が高く出ても役に立ちません。見るのは `recall`・`precision`・PR 曲線の面積です。

BigQuery ML の予測の生成 [1.1]

予測の関数は `ML.PREDICT` です。出力列は `predicted_<ラベル列名>` で、分類ではさらに `predicted_<ラベル列名>_probs` に各クラスの確率が入ります。[[c1-6](#)]

```
CREATE OR REPLACE TABLE `proj.ds.scores` AS
SELECT * FROM ML.PREDICT(MODEL `proj.ds.churn`,
  (SELECT * FROM `proj.ds.new_customers`),
  STRUCT(0.7 AS threshold));
```

`THRESHOLD` (二値分類のみ、既定 0.5) を上げると、陽性判定が厳しくなります。結果を `CREATE TABLE AS SELECT` で表に書き出せば、夜間のバッチ スコアリングの出来上がり。`KEEP_ORIGINAL_COLUMNS` の既定は `FALSE` で、適用されるのは PCA だけです。

`ML.EXPLAIN_PREDICT` は、`ML.PREDICT` に特徴量の寄与を足した関数。`TOP_K_FEATURES` の既定は 5 で、寄与の絶対値が大きい上位 5 つが返ります。寄与が返るのは予測されたラベルに対してだけ。二値分類の `THRESHOLD` の既定は 0.5 です。

Agent Platform AutoML の表形式データのトレーニング

[1.1]

表形式の AutoML を動かすときに決めるのは 4 つ。ターゲット列、予測タイプ (分類・回帰・予測)、最適化の目標、トレーニング予算です。[^c1-10]

データの下限は 1,000 行です。1 つのクラスに数行しかないと言質が下がるため、できれば全クラス 100 行以上を用意してください。[^c1-11]

データ分割の既定はランダムで、学習 80%・検証 10%・テスト 10% です。手動 (分割列の値で指定) と時系列 (時間列のタイムスタンプで分割) も選べます。時間に沿った予測では、ランダムのままだと学習に未来のデータが混ざり、評価が甘く出るのを避けられません。時系列の分割が正解です。

最適化の目標は予測のタイプで決まります。

種類	目標	向く課題
二値分類 (既定)	AUC ROC (<code>maximize-auc-roc</code>)	クラスを区別する
分類	AUC PR (<code>maximize-auc-prc</code>)	まれなクラスの結果を最適化
分類	再現率での適合率 (<code>maximize-precision-at-recall</code>)	再現率を固定して適合率を上げる
分類	適合率での再現率 (<code>maximize-recall-at-precision</code>)	適合率を固定して再現率を上げる
多クラス分類	ログ損失 (<code>minimize-log-loss</code>)	確率を正確に保つ (多クラスで使える目標はこれだけ)

種類	目標	向く課題
回帰(既定)	RMSE (minimize-rmse)	極端な値を正確に捉える
回帰	MAE (minimize-mae)	極端な値を外れ値として影響を抑える
回帰	RMSLE (minimize-rmsle)	相対的な大きさに誤差を見る。値が非常に大きくなりうるとき

トレーニング予算の単位はミリノード時間(1,000 ミリノード時間が 1 ノード時間)。たとえば 3 ノード時間なら 3,000 と指定します。推奨の予算は行数別。

行数	推奨の予算
10 万行未満	1〜3 時間
10 万〜100 万行	1〜6 時間
100 万〜1,000 万行	1〜12 時間
1,000 万行超	3〜24 時間

データが数 TB で最大 1,000 列に及ぶなら、エンドツーエンド AutoML (Tabular Workflows) の出番です。Agent Platform Pipelines 上のマネージドなパイプラインで、前処理や探索を細かく制御できます。制御できる段は、データ分割・特徴量エンジニアリング・アーキテクチャ検索・学習・アンサンブル・蒸留の 6 つ。[^c1-14]

Agent Platform AutoML の画像・テキスト・動画のデータセットとトレーニング [1.1]

動画とテキストの AutoML は提供を終えました(動画は 2025 年 7 月 31 日、テキストの分類・エンティティ抽出・感情分析は 2025 年 6 月 15 日に停止)。動画やテキストを自分のデータに合わせて分類したいときは、Gemini のチューニングを使います。いま使える画像の AutoML は、まずデータセットを作り、Cloud Storage のファイルと正解ラベルを JSON Lines か CSV のインポートファイルで取り込みます。次にコンソールか API で、トレーニング メソッドに AutoML を選んで学習します。Cloud Storage の画像を使うには、バケットの Storage Object Viewer ロールを Agent Platform サービス エージェントに付ける必要があります。[^c1-12]

画像分類のデータの目安です。

項目	値
ラベルごとの画像数	約 1,000 枚を推奨、最小は 10
ラベル間の偏り	最多のラベルが最少のラベルの 100 倍以内だと最もよく動く。極端に少ないラベルは削る
学習画像 1 枚の最大サイズ	30 MB
予測に送る画像の最大サイズ	1.5 MB (base64 でエンコードして送る)
学習画像の形式	JPEG・GIF・PNG・BMP・ICO
予算 (Cloud モデル)	8,000～800,000 ミリノード時間、既定 192,000 (8 ノードで 1 日の経過時間に相当)

学習データは、予測するデータにできる限り近いものにします。監視カメラのぼやけた画像を判定したいなら、ぼやけた画像で学習します。人が 1〜2 秒見てラベルを付けられないものは、モデルにも付けられません。どれにも当たらない画像には `None_of_the_above` のラベルを付けて含めると、定義外の画像に無理にラベルを付ける誤りが減ります。[^c1-13]

分割は、分割列を使うか、`fractionSplit`（学習・検証・テストの比率で、合計が 1）で指定します。AutoML の画像モデルは、現実の物体の写真に合わせて最適化されています。

BigQuery を使用した Gemini モデルのファインチューニング [1.1]

BigQuery ML では、Agent Platform の Gemini をリモートモデルとして参照し、教師ありチューニングをかけられます。追加の学習データは表で渡します。[^c1-9]

```
CREATE OR REPLACE MODEL `proj.ds.tuned`  
REMOTE WITH CONNECTION `proj.us.conn`  
OPTIONS (  
  endpoint = 'gemini-2.5-flash-lite',  
  max_iterations = 300,  
  evaluation_task = 'CLASSIFICATION',  
  prompt_col = 'prompt',  
  input_label_cols = ['label']  
) AS  
SELECT prompt, label FROM `proj.ds.train_examples`;
```

完成したモデルは `ML.EVALUATE` で評価し、生成 AI 関数で使います。設定を表に並べます。

設定	内容
<code>ENDPOINT</code>	教師ありチューニングに対応する Agent Platform の Gemini を指定する (ページの例は <code>gemini-3.5-flash-lite</code>)。対応モデルの一覧は文書によって違うので、作る前に Agent Platform の教師ありチューニングのページで確かめる
<code>MAX_ITERATIONS</code>	Gemini はステップでなくエポックで学習するため、既定は入力行数=1 エポック。100 行で 2 エポックなら 200。101 を指定すると 2 エポックに切り上げ
<code>EVALUATION_TASK</code>	<code>TEXT_GENERATION</code> ・ <code>CLASSIFICATION</code> ・ <code>SUMMARIZATION</code> ・ <code>QUESTION_ANSWERING</code> ・ <code>UNSPECIFIED</code> (既定)
<code>PROMPT_COL</code>	プロンプトの列。既定名は <code>prompt</code>
<code>INPUT_LABEL_COLS</code>	要素は 1 つだけ
<code>DATA_SPLIT_EVAL_FRACTION</code>	既定 0.2

権限は、リモートモデルの接続のサービス アカウントに付けます。教師ありチューニングを使うなら、付けるのは Vertex AI Service Agent ロール (`roles/aiplatform.serviceAgent`)。チューニングを使わないリモートモデルなら、Agent Platform ユーザー ロールを付けます。取り違えると権限不足で止まるので、気をつけてください。

場面で考えます。顧客の問い合わせ文を 12 種類のカテゴリに分類する Gemini を、BigQuery 内の 5,000 件のラベル付きデータで調整したい、という依頼。まず、プロンプト設計で足りるかを試します。足りなければ、prompt 列とラベル列の表を作ります。リモートモデルは CREATE MODEL ... REMOTE WITH CONNECTION で作ります。EVALUATION_TASK には CLASSIFICATION を指定し、教師ありチューニングにかけます。接続のサービス アカウントに必要なのは、Vertex AI Service Agent ロール。完成後の流れは、ML.EVALUATE で評価し、AI.GENERATE_TEXT で呼ぶこと。

チューニングの前に、プロンプト設計で足りないかを確認めます。有効な条件は、ラベル付きデータが 100 件以上あり、複雑で固有のタスクで、プロンプトを工夫しても品質が足りないこと。

BigQuery の生成 AI 関数とリモートモデル [1.1]

生成 AI の関数は、SQL の中で Gemini を呼び出す道具で、大きく 2 系統あります。[^c1-15]

系統	中身	特徴
汎用 AI 関数	AI.GENERATE (最も柔軟)、AI.GENERATE_BOOL • AI.GENERATE_DOUBLE • AI.GENERATE_INT、AI.GENERATE_TABLE、AI.GENERATE_TEXT、AI.EMBED • AI.GENERATE_EMBEDDING • AI.SIMILARITY	モデル、プロンプト、パラメータを自分で選ぶ。出力にステータスや完全なレスポンス(安全性評価や引用を含む)が付く

系統	中身	特徴
マネージド AI 関数	AI.IF（自然言語の条件で絞る）、AI.SCORE（品質や感情の採点）、 AI.CLASSIFY（カテゴリ分類）、AI.AGG（自然言語の指示で集約・要約）	構文が簡単で、BigQuery がモデルを自動で選び、費用と品質を最適化する

AI.GENERATE に出力のスキーマを渡すと、通話の文字起こしから名前、電話番号、依頼を取り出すような構造化出力ができます。旧来の ML.GENERATE_TEXT は、リモートモデルを CREATE MODEL ... REMOTE WITH CONNECTION で作ってから呼ぶ形でした。この関数も引き続き使えますが、新しいクエリには同じ働きの AI.GENERATE_TEXT が勧められています。選び方は単純で、細かく制御したいなら汎用、簡単に済ませたいならマネージドです。

Model Garden のモデルの評価と選択 [1.2]

Model Garden は、Agent Platform で使える AI モデルの一覧と入口で、カテゴリは 3 つ。[[^]c1-16]

カテゴリ	内容
基盤モデル	事前学習済みのマルチタスク大規模モデル。チューニングやカスタマイズができる
ファインチューニング可能なモデル	独自のノートブックやパイプラインで調整できるオープンモデル
タスク固有のソリューション	構築済みで、多くはすぐ使え、データでカスタマイズもできる

モデルを探すには、[フィルタ] ペインで 4 軸から絞り込み、モデルカードで詳細を見ます。軸は、タスク、モデル コレクション (Google・パートナー・ユーザー管理)、プロバイダ、特徴です。モデル コレクションは、誰が管理しているモデルかで絞る軸で、ユーザー管理は自分たち (自分の組織) が管理しているモデルを指します。

オープンモデルの課金は、使い方ごとに別。チューニングは、カスタムトレーニングと同じレートで、使ったコンピューティング リソースに課金されます。デプロイは、エンドポイントのコンピューティング リソースが課金対象です。Colab Enterprise を使えば、その料金も別にかかります。

統制の道具は 2 つ。Model Garden の組織ポリシーを組織・フォルダ・プロジェクトで設定すれば、承認済みのモデルだけを許可して他を拒否できます。もう一つはセキュリティ スキャン。Hugging Face のサードパーティ モデルは、マルウェア、pickle ファイル、Keras Lambda レイヤ、シークレットの有無を調べられます。

安全でないと判定されたモデルは、Model Garden でのデプロイがブロックされます。組織ポリシーの許可は使えるモデルを絞る仕組みで、スキャンの判定とは別に働きます。許可したモデルでも、安全でないと判定されればデプロイはできません。

場面で考えます。情報システム部門が「承認済みのモデルだけを全社で使わせたい」と求めるなら、Model Garden の組織ポリシーを組織かフォルダのレベルで設定します。開発者ごとに口頭で注意するのでは統制になりません。外部の Hugging Face のモデルを試したい開発者がいる場合は、スキャンの仕組みも押さえておく必要があります。安全でないとフラグされたモデルは、ブロックされます。

モデルの選定は、要件を軸に絞り込む作業。画像から文字を読むだけなら、基盤モデルを探すより、タスク固有のソリューションや専用 API のほうが早く、費用も小さくなります。独自データで調整する前提なら、ファインチューニング可能なモデルを探します。

Gemini のモデルの選び分け [1.2]

Gemini は用途で階層に分かれます。Pro と Argon の系列は最も高い品質を狙い、Flash は品質と速度と費用の均衡を取ります。Flash-Lite は低コスト・低レイテンシ向け。ほかに、音声のやり取り用の Live と、画像生成用の Image 系もあります。[^c1-18]

要件	選ぶ階層
複雑な推論、長い文書の分析、品質が最優先	Pro / Argon 系
汎用のチャット、要約、抽出で、応答速度と費用も大事	Flash
分類や短文抽出のような大量の単純処理、最小の費用	Flash-Lite
音声での会話	Live

モデル名と世代は更新が速く、2.5 世代も一覧に残っています。名前の暗記より、階層の役割で選べるのが大切。

場面で考えます。コールセンターの通話ログ 100 万件を、毎晩「クレームか問い合わせか」に 2 分類するなら、単純で大量の処理なので、合うのは Flash-Lite。契約書 300 ページを読ませて条項間の矛盾を指摘させるなら、長い文脈での推論が要るので Pro 系です。社内向けチャットで、品質と応答速度の両方が要るなら Flash が出発点になり、品質が足りなければ上位へ、費用が高

ければ下位へ動かします。上位に固定せず、タスクごとに評価して決める姿勢が正解。

Document AI API [1.2]

Document AI は、書類からテキストや項目を取り出す API で、プロセッサを作って文書を投げて使います。公式の区分はデジタル化 (OCR)・抽出・分類の 3 つです。下の表は、学習が要るかどうかで汎用・特化型・カスタムに分けたもので、公式の区分ではありません。[^c1-19]

システム	プロセッサ (タイプ名)	使いどころ
汎用	Enterprise Document OCR (OCR_PROCESSOR)、Form Parser (FORM_PARSER_PROCESSOR)、Layout Parser (LAYOUT_PARSER_PROCESSOR)	OCR は 200 以上の言語と品質分析。Form Parser はキーと値、チェックボックス、表。Layout Parser は生成 AI や検索向けにテキスト、表、リストを取り出す
特化型	INVOICE_PROCESSOR (請求書)、EXPENSE_PROCESSOR (経費)、FORM_W2_PROCESSOR、BANK_STATEMENT_PROCESSOR、PAYSTUB_PROCESSOR、US_DRIVER_LICENSE_PROCESSOR、ID_PROOFING_PROCESSOR など	書類の種類ごとに事前学習済み。学習なしで項目が取れる

システム	プロセッサ(タイプ名)	使いどころ
カスタム	Custom Extractor (CUSTOM_EXTRACTION_PROCESSOR)、Custom Classifier (CUSTOM_CLASSIFICATION_PROCESSOR)、Custom Splitter (CUSTOM_SPLITTING_PROCESSOR)	独自の書類。抽出、分類、束ねられた複数文書の分割

選ぶ順番は、特化型に該当する書類ならそれを使い、なければ Form Parser、それでも足りなければカスタムです。請求書なら INVOICE_PROCESSOR が最短でしょう。独自様式の発注書は Custom Extractor になります。Custom Classifier は事前学習済みで、ゼロショットの分類にも対応します。

Vision API・Video Intelligence API・Speech-to-Text などの事前トレーニング済み API [1.2]

自社データでの学習が要らず、API を呼ぶだけで使えるのが事前トレーニング済み API です。Vision API の機能は 11 種類あります。

- テキスト検出、ドキュメント テキスト検出
- ランドマーク検出、ロゴ検出、ラベル検出
- 画像プロパティ(ドミナント カラー)、オブジェクトのローカライズ
- クロップヒント検出、ウェブ検出、セーフサーチ、顔検出

[^c1-20]

文字の読み取りは 2 種類あります。

機能	向く画像
TEXT_DETECTION	大きな画像の中のまばらなテキスト(看板、標識)
DOCUMENT_TEXT_DETECTION	PDF／TIFF や、高密度のテキスト、手書き

両方をリクエストした場合は DOCUMENT_TEXT_DETECTION が優先されます。スキャンした書類を項目に構造化したいなら、Vision API でなく Document AI の出番です。セーフサーチの評価項目は adult・spoof・medical・violence・racy の5つ。各項目を UNKNOWN から VERY_LIKELY までの6段階で判定。クロップヒントでは、1 画像に最大 16 の画像比率(幅:高さ)を指定できます。

動画の中身(シーンやラベル)の解析は、これまで Video Intelligence API の担当でした。ただし Video Intelligence API は 2026 年 9 月 14 日に非推奨(サポート終了)となり、2027 年 9 月 14 日に停止します。公式は Gemini のモデルへの移行を勧めているので、新しく作るなら動画は Gemini に渡して分析させます。音声の文字起こしは Speech-to-Text の担当で、API は入力の種類で決まります。画像は Vision、動画は Gemini (Video Intelligence は停止予定)、音声は Speech-to-Text、書類は Document AI、文章の言語は Translation と覚えてください。[^c1-20v]

Translation API [1.2]

Cloud Translation の版は 2 つ。Basic は、標準のニューラル機械翻訳(NMT)をそのまま使うプラグアンドプレイ型。Advanced は、NMT に加えて Translation LLM (TLLM) に対応しています。用語集、ドキュメント翻訳、バッチ翻訳、適応型翻訳が使えるのも Advanced の側です。[^c1-21]

用語集（グロッサリー）は、社名や製品名の訳を固定する機能です。適応型翻訳は、少数の翻訳ペアの例を渡して文体や用語に寄せる機能で、モデルの学習は要りません。

要件	選ぶもの
業界用語の訳を揃えたい	用語集か適応型翻訳
独自の MLOps を回したい	Model Garden のファインチューニング
試作したい	Vertex AI Studio
基本の翻訳だけ	Basic

用途別のソリューションの構築とモデルのチューニング [1.2]

生成の対象で、使うモデルが変わります。

対象	モデル	メモ
文章・コード・マルチモーダル理解	Gemini	チューニングできる
画像生成	Gemini の Image 系	旧 Imagen のエンドポイント (imagegeneration@002 ~ @006 、 imagen-3.0- * 、 imagen-4.0-*) は非 推奨で、移行先は gemini- 2.5-flash-image
動画生成	Veo、Gemini Omni Flash	テキストから動画、最初のフ レームから、最初と最後のフ レームから、参照から、動画 の拡張、動画の編集

Imagen の旧エンドポイントを使い続けているコードは、移行の対象です。
[^c1-23] 動画の機能として挙がっているのは、表の 6 つ。[^c1-22]

Gemini のチューニングには、教師ありファインチューニングが使えます。考え方は 2 つあります。パラメータ効率チューニング (アダプタ チューニング) は、パラメータの小さな一部だけを更新するので低コストです。フル ファインチューニングは全パラメータを更新するため高品質ですが、コストが大きくなります。[^c1-25]

教師ありチューニングの上限と制約です。[^c1-26]

項目	値 (Gemini 3.5 Flash/Flash-Lite)
1 例あたりの入出カトークン合計	131,072
検証サンプル	最大 5,000 (1,000 超なら学習数の 30%)
JSONL ファイル	最大 1 GB
学習データの件数	テキストのみ 1,000 万件、マルチモーダル 30 万件
アダプタサイズ	1・2・4・8・16
リージョン	チューニングとチューニング済みモデルの提供は us と eu のマルチリージョン エンドポイントのみ

チューニングは SLA の対象外で、SLO から除外されます。学習データは 1 行 1 サンプルの JSONL にして Cloud Storage に置きます。最初は 100 サンプルが推奨。量より質。検証データセットの用意も強く勧められています。思考に対応するモデルでは、思考予算を 0 (または思考レベル MINIMAL) にすると、性能が上がりコストも下がります。

運用の落とし穴が 1 つあります。Gemini 3.5 Flash・Flash-Lite と 3.1 Flash-Lite では、教師ありチューニングとチューニング済みモデルの提供は、**us** と **eu** のマルチリージョン エンドポイントだけです。これらのモデルを、アジアの単一リージョンでチューニングして提供する前提の設計は成立しません。データの所在地要件がある案件では、この制約が選定を左右します。

Model Garden のサービスとしてのモデル(MaaS) [1.2]

Model as a Service (MaaS) は、キュレートされたオープンモデルやパートナー モデルを、サーバーレスのマネージド API として呼ぶ方式です。リクエストは Agent Platform のエンドポイントに送るので、インフラの管理は要りません。利用前に、オープンモデルへのユーザー アクセス権を付与しておく必要があります。[^c1-17]

自分でデプロイする方式と比べます。MaaS は従量課金で、立ち上げが速く、運用の手間がありません。自前デプロイは、専用のコンピューティングを持つ代わりに、チューニング済みの独自モデルを置けます。まず MaaS で試し、独自モデルや専用の性能が要る段階で自前へ移るのが基本線です。

プロンプト設計・グラウンディング・RAG [1.2]

モデルの出力を改善する手は、プロンプト設計から試します。公式のチューニングの概要が勧める順は、まずプロンプトで最適な形を探し、必要ならファインチューニングへ進む形。原因がモデルの知らない情報なら、グラウンディングや RAG で根拠を渡します。[^c1-25]

グラウンディングは、モデルの出力を検証可能な情報源に結び付けて、ハルシネーションを減らす仕組みです。応答を根拠にリンクできるので、監査もしやすくなります。Agent Platform がサポートする方法を表にします。[^c1-24]

方法	つなぐ先
Google 検索	世界の最新情報
Google マップ	地理空間のデータ
Agent Search	ウェブサイトやドキュメント セット (RAG)
RAG Engine	構成可能なマネージド RAG サービス。自社データ
Elasticsearch	既存の Elasticsearch インデックス
Search API	任意の検索 API
エンタープライズ向けウェブ グラウンディング	規制の厳しい業界向けのウェブ インデックス
パラレルウェブ検索	LLM 向けに最適化されたウェブ インデックス

RAG (検索拡張生成) は、質問に関係する文書を検索して、プロンプトに添えてから生成させる方式です。社内規程のように、モデルの学習に入っていない最新の自社情報に答えさせたいときに使います。再学習は要らず、知識の更新はデータの入れ替えだけで済みます。ファインチューニングは、文体や形式、専門タスクの振る舞いを変えるのに向き、事実の更新には向きません。

プロンプトに入れる情報 (指示、例、参照文書、会話の履歴) を設計する作業は、コンテキスト エンジニアリングと呼ばれます。毎回同じ長い前置きを使うなら、次の節のコンテキスト キャッシュに載せる設計です。

Gemini アプリの費用の最適化 [1.2]

費用を下げる手は 3 本立て。入力トークンの再利用、処理の非同期化、モデルの格下げです。

コンテキスト キャッシュ

同じ長い入力(大きな文書や共通の指示)を何度も送るときに、再利用できる仕組みで、2 種類あります。[^c1-27]

項目	暗黙的キャッシュ	明示的キャッシュ
作り方	既定で有効、自動	API で手動作成
割引	キャッシュヒットで標準の入力トークンより 90% 安い	参照する入力トークンは Gemini 2.5 以降で 90%、Gemini 2.0 で 75% 安い
保存料金	なし	保存期間に応じて課金
期間	作成後 24 時間以内に期限切れ	既定の TTL は 60 分、最小 1 分、最大の上限なし

キャッシュの作成に使った入力トークンは、標準料金で請求されます。最小トークン数は次の表の値。

モデル	最小トークン数
Gemini 3 (例外あり)、Gemini 4	4,096
Gemini 2 ファミリー	2,048
Gemini 3.0 Flash プレビュー、3.1 Pro プレビュー、3.7 Flash、3.8 Flash	6,144 (暗黙的キャッシュのみ)

キャッシュできるのは blob とテキストで、最大 10 MB です。キャッシュされたトークン数は、応答メタデータの `cachedContentTokenCount` で確かめられます。

バッチ推論

リアルタイム性が要らない大量の処理は、バッチ推論に回す設計が基本。
[^c1-28]

項目	値
割引	リアルタイムより 50% 安い
1 ジョブのリクエスト数	最大 200,000
Cloud Storage 入力ファイル	最大 1 GB
キュー待ち	最大 72 時間
実行後の完了	ほとんどが 24 時間以内。24 時間で未完 の分はキャンセルされ、完了した分だけ課 金

バッチとキャッシュの割引は、Gemini 3 以降なら併用でき、キャッシュされた入力トークンは標準の 95% 引きです。標準が 100 なら、バッチだけで 50、暗黙的キャッシュだけで 10、Gemini 3 以降の併用で 5 になります。Gemini 2.5 は併用できません。90% のキャッシュ割引が、バッチ割引より優先されるからです。

場面で考えます。毎晩 10 万件の問い合わせに同じ 30 ページの手順書を付けて分類し、翌朝までに結果があればよい処理があるとしています。応答速度が要らないのでバッチ推論が使える、共通の手順書はキャッシュできます。この 2 つの組み合わせが正解。分類のような単純な処理なので、Flash-Lite への格下げも候補になります。

Gemini アプリのレイテンシと可用性 [1.2]

使用オプションの一覧(保証が弱い順)。

使用オプション	契約	性格
Standard PayGo	事前コミットなしの従量課金	共有容量。保証なし
Priority PayGo	従量課金	Standard より一貫した性能
プロビジョンド スループット	固定料金の定期サブスクリプション	モデルのスループットを予約。専用の保証された容量

プロビジョンド スループットが向くのは、次のような場面です。

- チャットボットやエージェントのような、本番のリアルタイム アプリ
- 重要なワークロードで、常に高いスループットが要るとき
- 利用者に、一貫して予測しやすい体験を出したいとき
- 月額や週単位の固定料金で費用を管理したいとき

契約期間は複数あり、モデルと利用できるロケーションを指定して予約します。^[^c1-29]

Standard PayGo には使用量ティアがあります。対象サービスの 30 日間のローリングの費用が増えるほど、組織の優先スループットのしきい値 (1 分あたりのトークン数、TPM) が上がる仕組みです。^[^c1-30]

費用 (30 日間)	Pro / Argon 系の TPM	Flash / Flash-Lite の TPM
Tier 1 (\$10～\$250)	500,000	2,000,000
Tier 2 (\$250～\$2,000)	1,000,000	4,000,000
Tier 3 (\$2,000～\$50,000)	2,000,000	10,000,000
Tier 4 (\$50,000 超)	10,000,000	50,000,000

しきい値までのリクエストは、超過分のバーストより優先されます。ただし厳密な上限ではなく、共有容量に空きがあれば、超えても処理されます。混んだ時間帯は、超過分から先に調整され、遅延の増加や HTTP 429 エラーが起きがち。1 分の平均がしきい値内でも、2 秒目の急なスパイクで調整されることがあります。送るときは、1 分の中でできるだけ均等にならしてください。

エンドポイントの種類でも、余裕は変わります。しきい値が適用されるのは、global エンドポイント宛てのリクエストだけ。

エンドポイント	容量の余裕	向く場面
global	最大。世界中の容量から最も空いた場所へ動的にルーティング	データ所在地の要件がない
マルチリージョン(us・eu)	地域内で容量をプール。global の容量は共有しない	処理を地域に限定したい
単一リージョン(us-central1 など)	最小	厳格な所在地要件

データ所在地の要件がなく、429 に悩まされているなら、まず global エンドポイントへ切り替えます。保証が必要で費用を固定したいなら、プロビジョンドスループット。事前コミットなしで安定させたいなら、Priority PayGo が候補になります。

総合の場面で締めます。社内文書の Q&A アプリが、朝の始業時間帯に 429 エラーを返し、同じ 20 ページの規程を毎回プロンプトに付けているとします。打つ手は 3 つ。1 つ目は、データ所在地の要件がなければ global エンドポイントを使うこと。2 つ目は、毎回付ける規程をコンテキスト キャッシュに載せること(最小トークン数を超えていれば暗黙的キャッシュが自動で効き、確実に

使うなら明示的キャッシュを作る)。3つ目は、常に安定した容量が要るなら、プロビジョンド スループットを検討すること。

逆に、リアルタイム性が要らない処理にプロビジョンド スループットを買うのは過剰です。バッチ推論のほうが安く済みます。国内の厳格なデータ所在地要件がある場合は、global が使えないので、マルチリージョンか単一リージョンのエンドポイントを選び、余裕が小さい分を平準化した送信で補います。

■ サンプルはここまでです

続き(第2章～巻末)は、Professional Machine Learning Engineer 問題集のご購入で、頻出度別ノート・暗記用ノートと合わせて3点すべてダウンロードできます。

Google Cloud Professional Machine Learning Engineer 学習用テキスト(無料サンプル)

版

2026年10月版(Google Cloud 公式 Professional Machine Learning Engineer Certification exam guide (as of June 1, 2026・英語版) 準拠)

発行

IT資格道場(<https://www.shikaku-doujou.com>)

お問い合わせ

info@shikaku-doujou.com

Google Cloud、Vertex AI、BigQuery および Google の製品名・サービス名は、Google LLC の商標です。記載されているその他の会社名・製品名・サービス名は、各社の商標または登録商標です。

本教材は公式シラバスの内容を基に作成したもので、Google LLC との提携・承認関係はありません。

本文・図表・設問はすべてオリジナルの著作物です。無断での転載・複製・再配布・二次販売はできません。お気づきの点は info@shikaku-doujou.com までご連絡ください。