Perform a batch prediction. Unlike the online **`models.predict`** (/automl/docs/reference/rest/v1/projects.locations.models/predict#google.cloud.automl.v1.PredictionService.Predict) , batch prediction result won't be immediately available in the response. Instead, a long running operation object is returned. User can poll the operation result via **`operations.get`** (/automl/docs/reference/rest/v1beta1/projects.locations.operations/get#google.longrunning.Operations.GetOperation) method. Once the operation is done, `BatchPredictResult` is returned in the **`response`** (/automl/docs/reference/rest/Shared.Types/ListOperationsResponse#Operation.FIELDS.response) field. Available for following ML scenarios:

- AutoML Vision Classification
- AutoML Vision Object Detection
- AutoML Video Intelligence Classification
- AutoML Video Intelligence Object Tracking * AutoML Natural Language Classification
- AutoML Natural Language Entity Extraction
- AutoML Natural Language Sentiment Analysis
- AutoML Tables

```
POST https://automl.googleapis.com/v1/{name}:batchPredict
```

Parameters

Parameters	
`name`	`string` Name of the model requested to serve the batch prediction. Authorization requires the following Google IAM (https://cloud.google.com/iam) permission on the specified resource **`name`**: • **`automl.models.predict`**

The request body contains data with the following structure:

JSON representation

Fields	
`inputConfig`	`object (`BatchPredictInputConfig (/automl/docs/reference/rest/v1/projects.locations.models/batchPredict#BatchPredictInputConfig)`)` Required. The input configuration for batch prediction.

Fields	
`outputConfig`	`object (BatchPredictOutputConfig` (/automl/docs/reference/rest/v1/projects.locations.models/batchPredict#BatchPredictOutputConfig) `)` Required. The Configuration specifying where output predictions should be written.

`params`	`map (key: string, value: string)`
	Additional domain-specific parameters for the predictions, any string must be up to 25000 characters long.
	`score_threshold`
	(float) A value from 0.0 to 1.0. When the model makes predictions for a text snippet, it will only produce results that have at least this confidence score. The default is 0.5.
	`score_threshold`
	(float) A value from 0.0 to 1.0. When the model makes predictions for an image, it will only produce results that have at least this confidence score. The default is 0.5.
	`score_threshold`
	(float) When Model detects objects on the image, it will only produce bounding boxes which have at least this confidence score. Value in 0 to 1 range, default is 0.5.

Fields

`max_bounding_box_count`

(int64) The maximum number of bounding boxes returned per image. The default is 100, the number of bounding boxes returned might be limited by the server.

`score_threshold`

(float) A value from 0.0 to 1.0. When the model makes predictions for a video, it will only produce results that have at least this confidence score. The default is 0.5.

`segment_classification`

(boolean) Set to true to request segment-level classification. AutoML Video Intelligence returns labels and their confidence scores for the entire segment of the video that user specified in the request configuration. The default is true.

`shot_classification`

(boolean) Set to true to request shot-level classification. AutoML Video Intelligence determines the boundaries for each camera shot in the entire segment of the video that user specified in the request configuration. AutoML Video Intelligence then returns labels and their confidence scores for each detected shot, along with the start and end time of the shot. The default is false.

WARNING: Model evaluation is not done for this classification type, the quality of it depends on training data, but there are no metrics provided to describe that quality.

`1s_interval_classification`

Fields	

	(boolean) Set to true to request classification for a video at one-second intervals. AutoML Video Intelligence returns labels and their confidence scores for each second of the entire segment of the video that user specified in the request configuration. The default is false.
	WARNING: Model evaluation is not done for this classification type, the quality of it depends on training data, but there are no metrics provided to describe that quality.
	`score_threshold`
	(float) When Model detects objects on video frames, it will only produce bounding boxes which have at least this confidence score. Value in 0 to 1 range, default is 0.5.
	`max_bounding_box_count`
	(int64) The maximum number of bounding boxes returned per image. The default is 100, the number of bounding boxes returned might be limited by the server.
	`min_bounding_box_size`
	(float) Only bounding boxes with shortest edge at least that long as a relative value of video frame size are returned. Value in 0 to 1 range. Default is 0.

If successful, the response body contains an instance of `Operation` (/automl/docs/reference/rest/Shared.Types/ListOperationsResponse#Operation).

Requires the following OAuth scope:

- `https://www.googleapis.com/auth/cloud-platform`

For more information, see the Authentication Overview (https://cloud.google.com/docs/authentication/).

Input configuration for models.batchPredict Action.

The format of input depends on the ML problem of the model used for prediction. As input source the `gcsSource` (/automl/docs/reference/rest/v1/projects.locations.datasets/importData#InputConfig.FIELDS.gcs_source) is expected, unless specified otherwise.

The formats are represented in EBNF with commas being literal and with non-terminal symbols defined near the end of this comment. The formats are:

See Preparing your training data (https://cloud.google.com/automl-tables/docs/predict-batch) for more information.

You can use either `gcsSource` (/automl/docs/reference/rest/v1/projects.locations.models/batchPredict#BatchPredictInputConfig.FIELDS.gcs_source)
or [bigquerySource][BatchPredictInputConfig.bigquery_source].

For gcsSource:

CSV file(s), each by itself 10GB or smaller and total size must be 100GB or smaller, where first file must have a header containing column names. If the first row of a subsequent file is the same as the header, then it is also treated as a header. All other rows contain values for the corresponding columns.

The column names must contain the model's

[inputFeatureColumnSpecs'] [google.cloud.automl.v1.TablesModelMetadata.input_feature_column_specs] [displayName-s] [google.cloud.automl.v1.ColumnSpec.display_name] (order doesn't matter). The columns corresponding to the model's input feature column specs must contain values compatible with the column spec's data types. Prediction on all the rows, i.e. the CSV lines, will be attempted.

Sample rows from a CSV file:

For bigquerySource:

The URI of a BigQuery table. The user data size of the BigQuery table must be 100GB or smaller.

The column names must contain the model's

[inputFeatureColumnSpecs'] [google.cloud.automl.v1.TablesModelMetadata.input_feature_column_specs] [displayName-s] [google.cloud.automl.v1.ColumnSpec.display_name] (order doesn't matter). The columns corresponding to the model's input feature column specs must contain values compatible with the column spec's data types. Prediction on all the rows of the table will be attempted.

Input field definitions:

`GCS_FILE_PATH`

> The path to a file on Google Cloud Storage. For example, "gs://folder/video.avi".

`TIME_SEGMENT_START`

(`TIME_OFFSET`) Expresses a beginning, inclusive, of a time segment within an example that has a time dimension (e.g. video).

`TIME_SEGMENT_END`

(`TIME_OFFSET`) Expresses an end, exclusive, of a time segment within n example that has a time dimension (e.g. video).

`TIME_OFFSET`

A number of seconds as measured from the start of an example (e.g. video). Fractions are allowed, up to a microsecond precision. "inf" is allowed, and it means the end of the example.

Errors:

If any of the provided CSV files can't be parsed or if more than certain percent of CSV rows cannot be processed then the operation fails and prediction does not happen. Regardless of overall success or failure the per-row failures, up to a certain count cap, will be listed in Operation.metadata.partial_failures.

JSON representation

Fields	
`gcsSource`	`object (GcsSource` (/automl/docs/reference/rest/v1/GcsSource)`)` Required. The Google Cloud Storage location for the input content.

Output configuration for models.batchPredict Action.

As destination the

`gcsDestination` (/automl/docs/reference/rest/v1/projects.locations.models/batchPredict#BatchPredictOutputConfig.FIELDS.gcs_destination)

must be set unless specified otherwise for a domain. If gcsDestination is set then in the given directory a new directory is created. Its name will be "prediction-

- For Image Classification: In the created directory files `image_classification_1.jsonl`, `image_classification_2.jsonl`,...,`image_classification_N.jsonl` will be created, where N may be 1, and depends on the total number of the successfully predicted images and annotations. A single image will be listed only once with all its annotations, and its annotations will never be split across files. Each .JSONL file will contain, per line, a JSON representation of a proto that wraps image's "ID" : "" followed by a list of zero or more AnnotationPayload protos (called annotations), which have classification detail populated. If prediction for any image failed (partially or completely), then an additional `errors_1.jsonl`, `errors_2.jsonl`,..., `errors_N.jsonl` files will be created (N depends on total number of failed predictions). These files will have a JSON representation of a proto that wraps the same "ID" : "" but here followed by exactly one

[`google.rpc.Status`](https://github.com/googleapis/googleapis/blob/master/google/rpc/status.proto) containing only `code` and `message`fields.

- For Image Object Detection: In the created directory files `image_object_detection_1.jsonl`, `image_object_detection_2.jsonl`,...,`image_object_detection_N.jsonl` will be created, where N may be 1, and depends on the total number of the successfully predicted images and annotations. Each .JSONL file will contain, per line, a JSON representation of a proto that wraps image's "ID" : "" followed by a list of zero or more AnnotationPayload protos (called annotations), which have imageObjectDetection detail populated. A single image will be listed only once with all its annotations, and its annotations will never be split across files. If prediction for any image failed (partially or completely), then additional `errors_1.jsonl`, `errors_2.jsonl`,..., `errors_N.jsonl` files will be created (N depends on total number of failed predictions). These files will have a JSON representation of a proto that wraps the same "ID" : "" but here followed by exactly one [`google.rpc.Status`](https: //github.com/googleapis/googleapis/blob/master/google/rpc/status.proto) containing only `code` and `message`fields. * For Video Classification: In the created directory a videoClassification.csv file, and a .JSON file per each video classification requested in the input (i.e. each line in given CSV(s)), will be created.

GCS_FILE_PATH,TIME_SEGMENT_START,TIME_SEGMENT_END,JSON_FILE_NAME,STATUS where: GCS_FILE_PATH,TIME_SEGMENT_START,TIME_SEGMENT_END = matches 1 to 1 the prediction input lines (i.e. videoClassification.csv has precisely the same number of lines as the prediction input had.) JSON_FILE_NAME = Name of .JSON file in the output directory, which contains prediction responses for the video time segment. STATUS = "OK" if prediction completed successfully, or an error code with message otherwise. If STATUS is not "OK" then the .JSON file for that line may not exist or be empty.

- For Video Object Tracking: In the created directory a videoObjectTracking.csv file will be created, and multiple files video_object_trackinng_1.json, video_object_trackinng_2.json,..., video_object_trackinng_N.json, where N is the number of requests in the input (i.e. the number of lines in given CSV(s)).

GCS_FILE_PATH,TIME_SEGMENT_START,TIME_SEGMENT_END,JSON_FILE_NAME,STATUS where: GCS_FILE_PATH,TIME_SEGMENT_START,TIME_SEGMENT_END = matches 1 to 1 the prediction input lines (i.e. videoObjectTracking.csv has precisely the same number of lines as the prediction input had.) JSON_FILE_NAME = Name of .JSON file in the output directory, which contains prediction responses for the video time segment. STATUS = "OK" if prediction completed successfully, or an error code with message otherwise. If STATUS is not "OK" then the .JSON file for that line may not exist or be empty.

- For Text Classification: In the created directory files `text_classification_1.jsonl`, `text_classification_2.jsonl`,...,`text_classification_N.jsonl` will be created, where N may be 1, and depends on the total number of inputs and annotations found.

[`google.rpc.Status`](https://github.com/googleapis/googleapis/blob/master/google/rpc/status.proto) containing only `code` and `message`.

- For Text Sentiment: In the created directory files `text_sentiment_1.jsonl`, `text_sentiment_2.jsonl`,...,`text_sentiment_N.jsonl` will be created, where N may be 1, and depends on the total number of inputs and annotations found.

[`google.rpc.Status`](https://github.com/googleapis/googleapis/blob/master/google/rpc/status.proto) containing only `code` and `message`.

- For Text Extraction: In the created directory files `text_extraction_1.jsonl`, `text_extraction_2.jsonl`,...,`text_extraction_N.jsonl` will be created, where N may be 1, and depends on the total number of inputs and annotations found. The contents of these .JSONL file(s) depend on whether the input used inline text, or documents. If input was inline, then each .JSONL file will contain, per line, a JSON representation of a proto that wraps given in request text snippet's "id" (if specified), followed by input text snippet, and a list of zero or more AnnotationPayload protos (called annotations), which have textExtraction detail populated. A single text snippet will be listed only once with all its annotations, and its annotations will never be split across files. If input used documents, then each .JSONL file will contain, per line, a JSON representation of a proto that wraps given in request document proto, followed by its OCR-ed representation in the form of a text snippet, finally followed by a list of zero or more AnnotationPayload protos (called annotations), which have textExtraction detail populated and refer, via their indices, to the OCR-ed text snippet. A single document (and its text snippet) will be listed only once with all its annotations, and its annotations will never be split across files. If prediction for any text snippet failed (partially or completely), then additional `errors_1.jsonl`, `errors_2.jsonl`,..., `errors_N.jsonl` files will be created (N depends on total number of failed predictions). These files will have a JSON representation of a proto that wraps either the "id" : "" (in case of inline) or the document proto (in case of document) but here followed by exactly one [`google.rpc.Status`](https: //github.com/googleapis/googleapis/blob/master/google/rpc/status.proto) containing only `code` and `message`.

- For Tables: Output depends on whether

`gcsDestination` or

`bigqueryDestination` is set (either is allowed). Google Cloud Storage case: In the created directory files `tables_1.csv`, `tables_2.csv`,..., `tables_N.csv` will be created, where N may be 1, and depends on the total number of the successfully predicted rows. For all CLASSIFICATION

`predictionType-s`: Each .csv file will contain a header, listing all columns'

`displayName-s` given on input followed by M target column names in the format of

"<`target_column_specs`

`displayName`>__score" where M is the number of distinct target values, i.e. number of distinct values in the target column of the table used to train the model. Subsequent lines will contain the respective values of successfully predicted rows, with the last, i.e. the target, columns having the corresponding prediction `scores`. For REGRESSION and FORECASTING

`predictionType-s`: Each .csv file will contain a header, listing all columns' [displayName-s] [google.cloud.automl.v1p1beta.display_name] given on input followed by the predicted target column with name in the format of

"predicted_<`target_column_specs`

`displayName`>" Subsequent lines will contain the respective values of successfully predicted rows, with the last, i.e. the target, column having the predicted target value. If prediction for any rows failed, then an additional `errors_1.csv`, `errors_2.csv`,..., `errors_N.csv` will be created (N depends on total number of failed rows). These files will have analogous format as `tables_*.csv`, but always with a single target column having

[`google.rpc.Status`](https: //github.com/googleapis/googleapis/blob/master/google/rpc/status.proto) represented as a JSON string, and containing only `code` and `message`. BigQuery case:

`bigqueryDestination` pointing to a BigQuery project must be set. In the given project a new dataset will be created with name `prediction_<model-display-name>_<timestamp-of-`

`prediction-call>` where

`displayName-s` followed by the target column with name in the format of

"predicted_<`target_column_specs`

`displayName`>" The input feature columns will contain the respective values of successfully predicted rows, with the target column having an ARRAY of

`AnnotationPayloads`, represented as STRUCT-s, containing `TablesAnnotation`. The `errors` table contains rows for which the prediction has failed, it has analogous input columns while the target column name is in the format of

"errors_<`target_column_specs`

`displayName`>", and as a value has

[`google.rpc.Status`](https://github.com/googleapis/googleapis/blob/master/google/rpc/status.proto) represented as a STRUCT, and containing only `code` and `message`.

JSON representation

Fields	
`gcsDestination`	`object (`[`GcsDestination`](/automl/docs/reference/rest/v1/GcsDestination)`)` Required. The Google Cloud Storage location of the directory where the output is to be written to.