Welcome to the module **Custom Model Building with AutoML**. AutoML is a service on Google Cloud that allows you to build powerful machine learning models with minimal effort and minimal machine learning expertise. They're ideal when you have a need to get a machine learning model off the ground as quickly as possible.

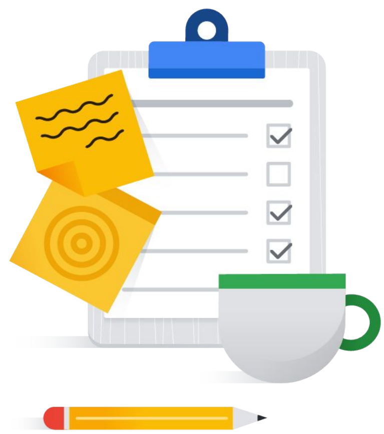
Custom Model Building with AutoML

01	Why AutoML?
02	AutoML Vision
03	AutoML Natural Language
04	AutoML Tables

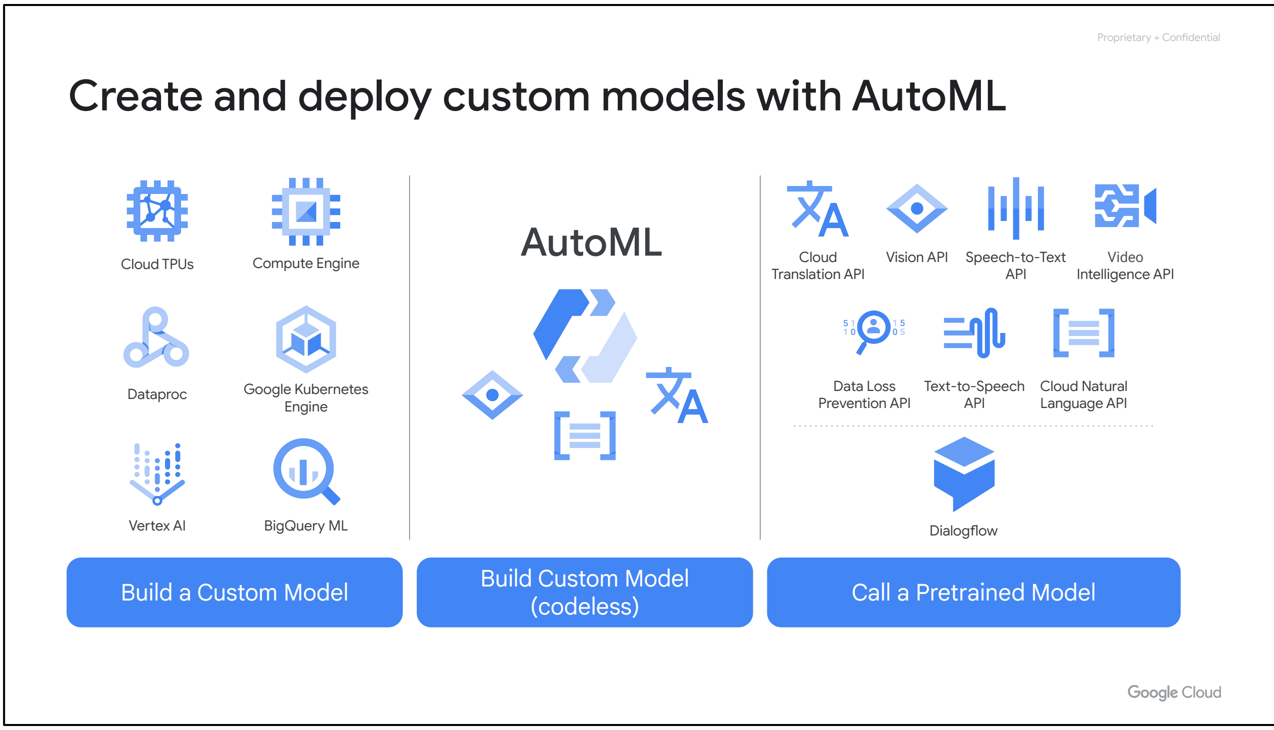
In this module, we're going to dive into some of the AutoML products. AutoML Vision is for image data, Natural Language is for text-based data, and Tables is for tabular data.

Custom Model Building with AutoML

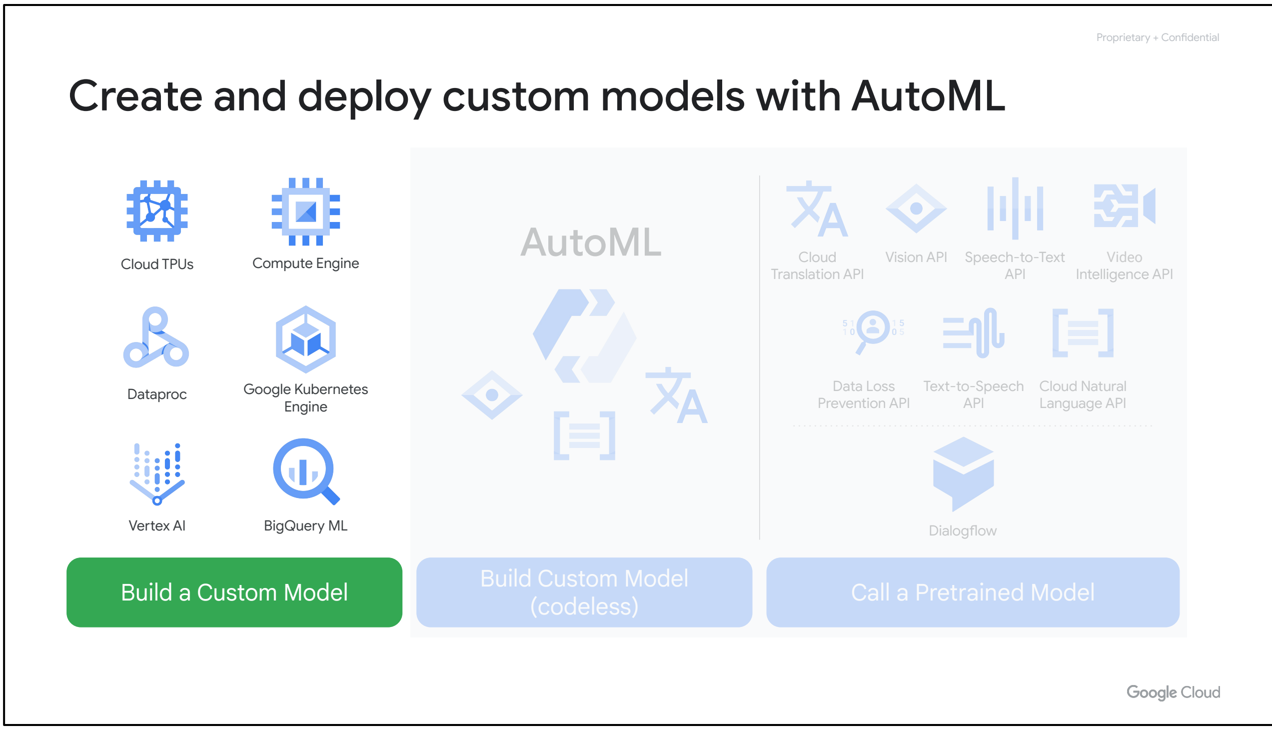
01	Why AutoML?
02	AutoML Vision
03	AutoML Natural Language
04	AutoML Tables



First, let's start off by making the case for why you may want to use AutoML.

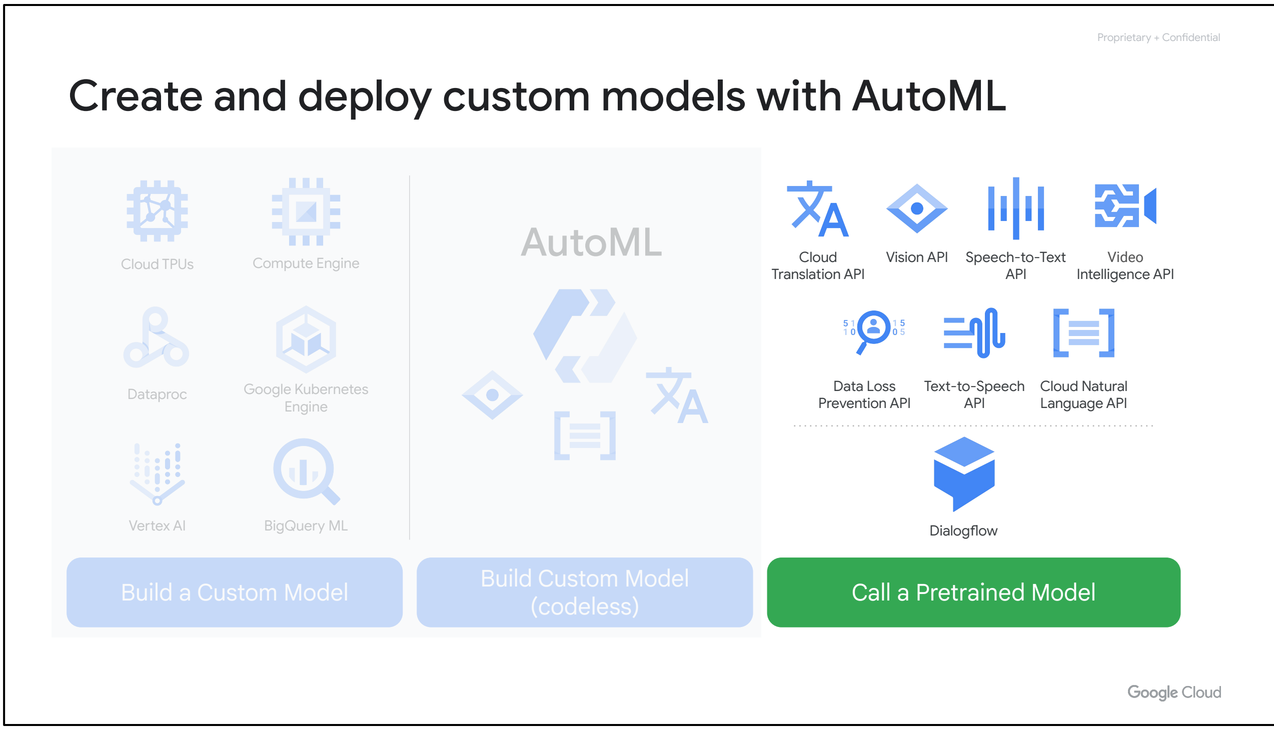


Where does AutoML sit in the suite of Google Cloud products that can be used for machine learning?

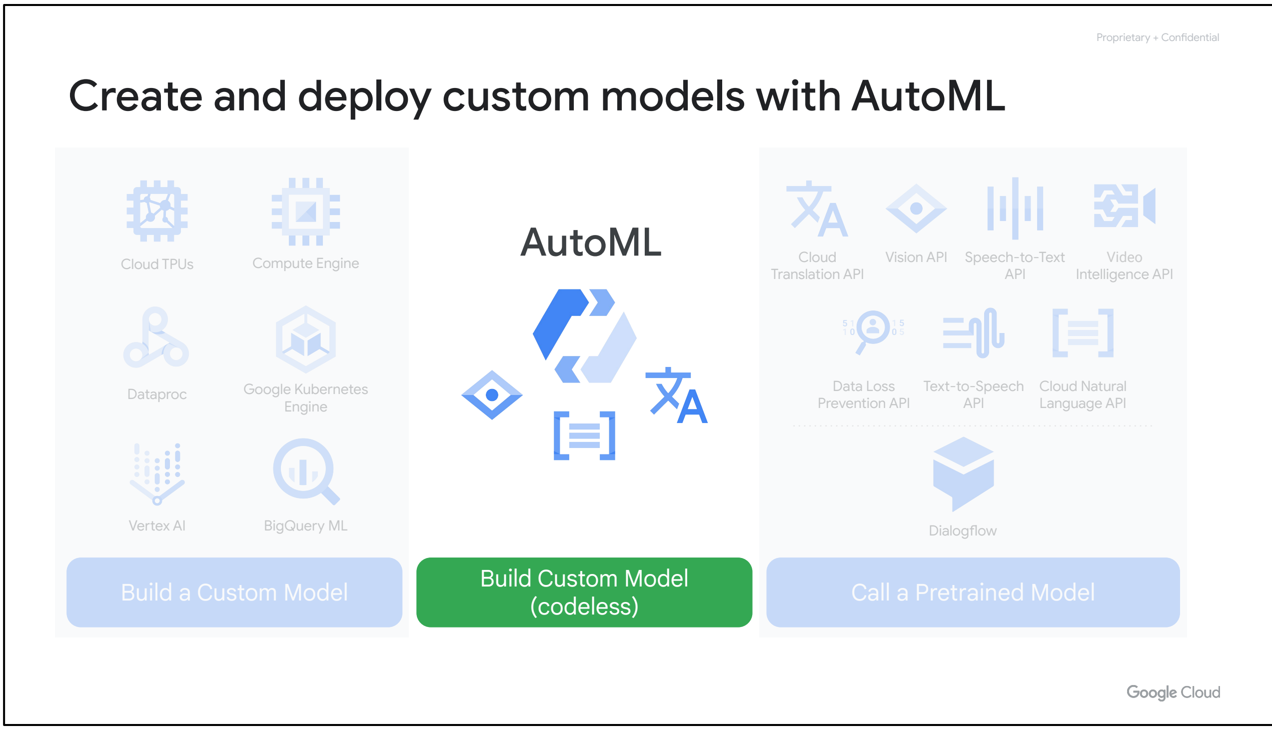


On one hand, with products such as Vertex AI and BigQuery ML, you can build very customized machine learning models.

However, to use these products you'll need someone with machine learning expertise and coding experience. You'll be responsible for training the machine learning model yourself, which can take a lot of time.

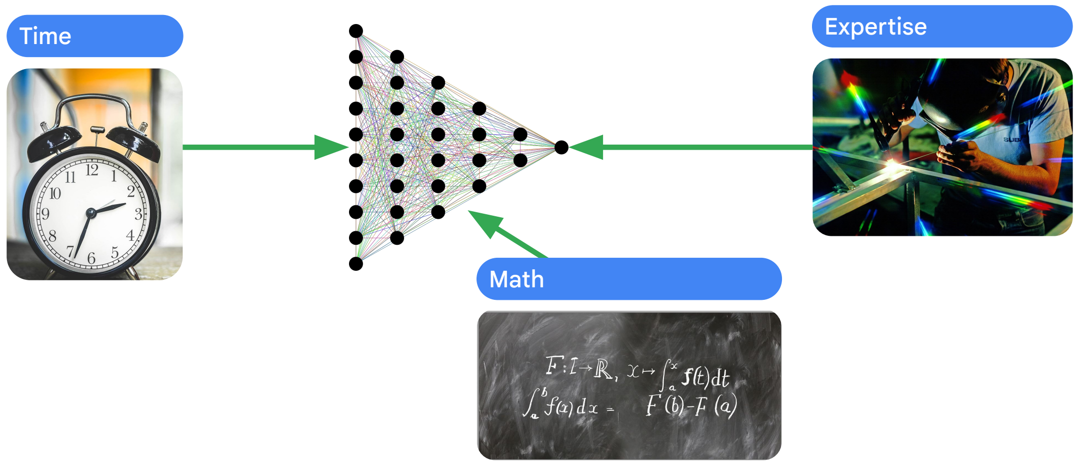


On the other hand, on Google Cloud you can call pre-trained models using services like the Vision API and the Speech-to-Text API. No model training is required for these services. You simply feed the API your data and it returns predictions. The downside of using pre-trained models is that they only yield good predictions when your data is relatively common-place, as in social media images or customer reviews.

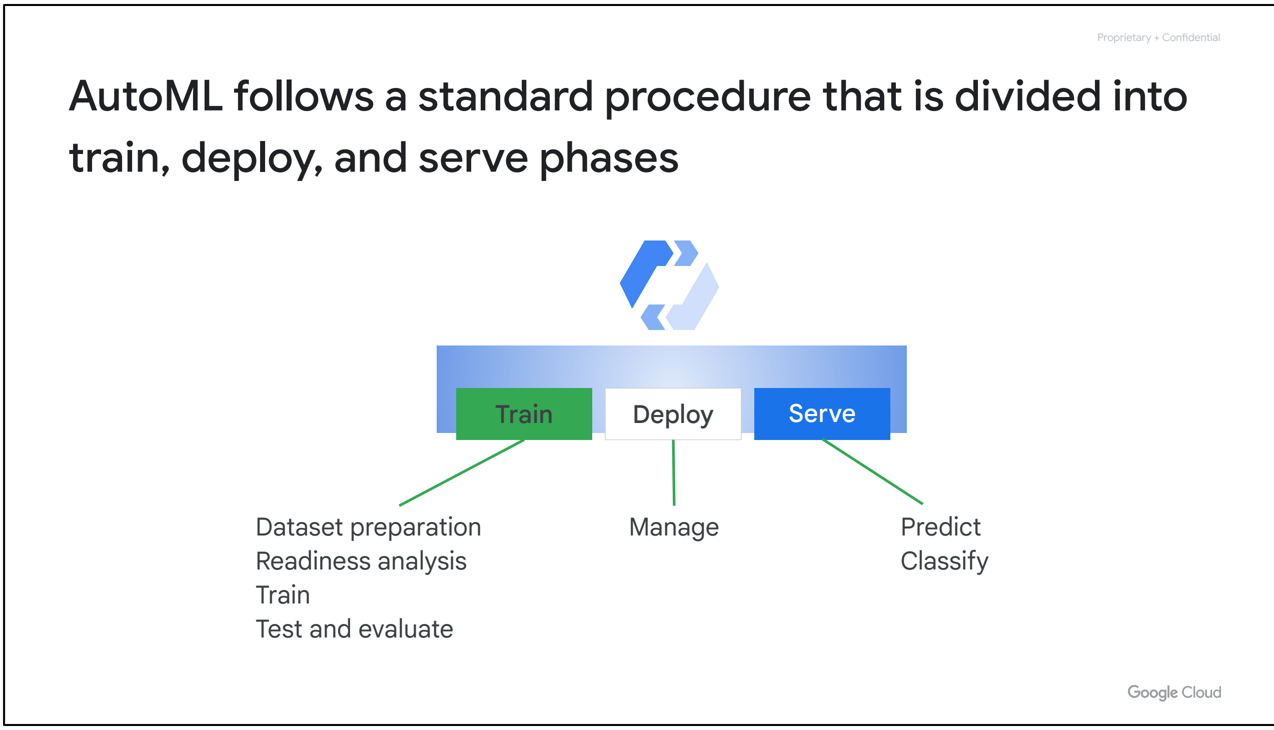


AutoML sits somewhere in between these two. A model is trained specific to your data but you don't need any code to train it.

Training high-quality, custom ML models requires a lot of effort and expertise



If you’re going to train a machine learning model from scratch you need machine learning and coding expertise. Anecdotally, building machine learning models follows the Pareto principle where you can launch a functional machine learning model relatively quickly. However, most time and effort will go into debugging and making the machine learning model performant.



AutoML follows a standard procedure that is divided into three phases, which are train, deploy, and serve.

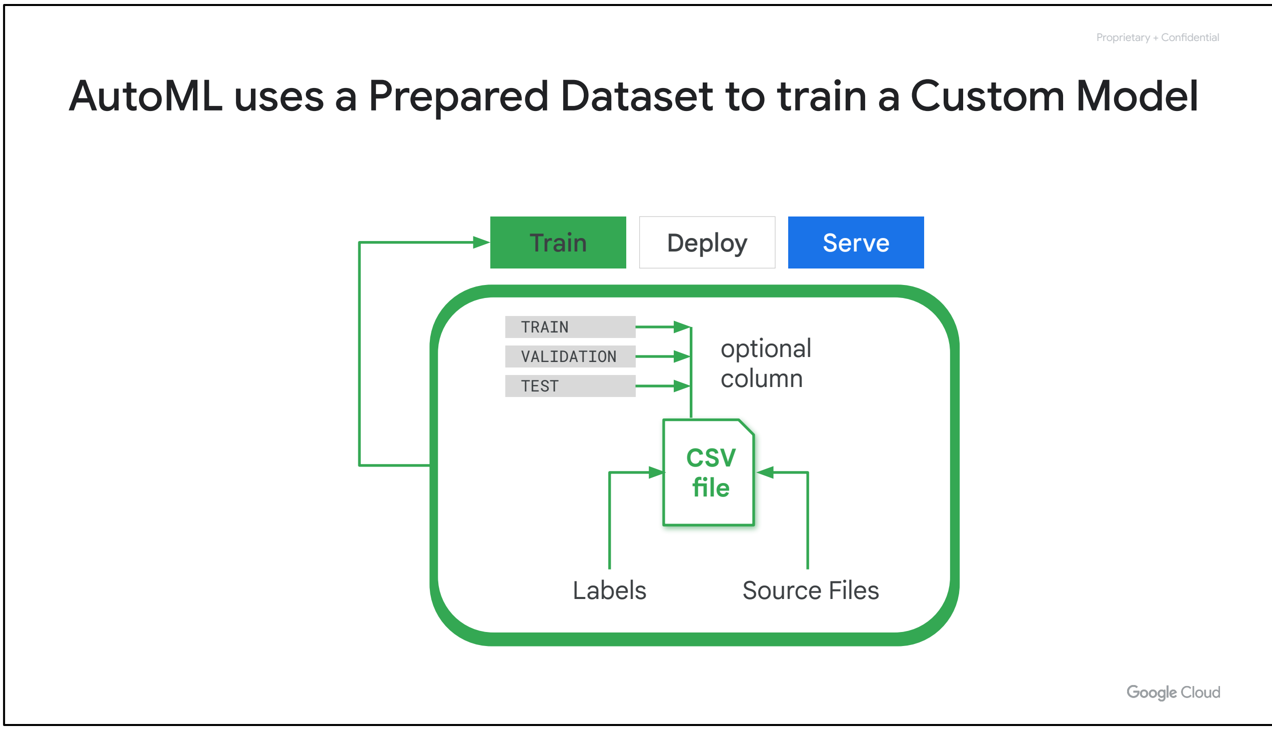
Train
The training phase has several steps:

- First you have to prepare a dataset that will be used in the supervised training process.
- Next, you need to analyze the dataset to make sure it has qualities that will enable it to be effective. And you may need to correct the dataset.
- After the dataset is prepared and validated, you use it to train the model.
- And finally, the model is used with test data to evaluate whether it is going to be effective in predicting and classifying new cases.
- If the model doesn't work well at this point, you may have to go back and modify the dataset and try again.

Deploy
The second phase is to deploy the model and manage it. That means getting rid of old or unused models.

Serve
The third phase is hosting the model on a service where it can be used to predict and classify.

In traditional machine learning, the deploy and serve phases are complicated and involve moving the model from a model-building system like TensorFlow to a model hosting system like Vertex AI. However, AutoML handles most of the complexity of these activities for you, making these activities easy.



AutoML uses a Prepared Dataset to train a Custom Model. You can make small Prepared Datasets for experimentation directly in the Web UI but it is more common to assemble the information in a CSV (comma separated value) file. The CSV file must be UTF-8 encoded and located in the same Cloud Storage bucket with the source files. You can also create and manage Prepared Datasets programmatically in Python, Java, or Node.js

The first column in the CSV file is optional. It assigns the data in each row into one of the three groups, TRAIN, VALIDATION, or TEST. If you leave out this column, the rows will automatically be assigned with 80% going to TRAIN, and 10% each to VALIDATION and TEST.

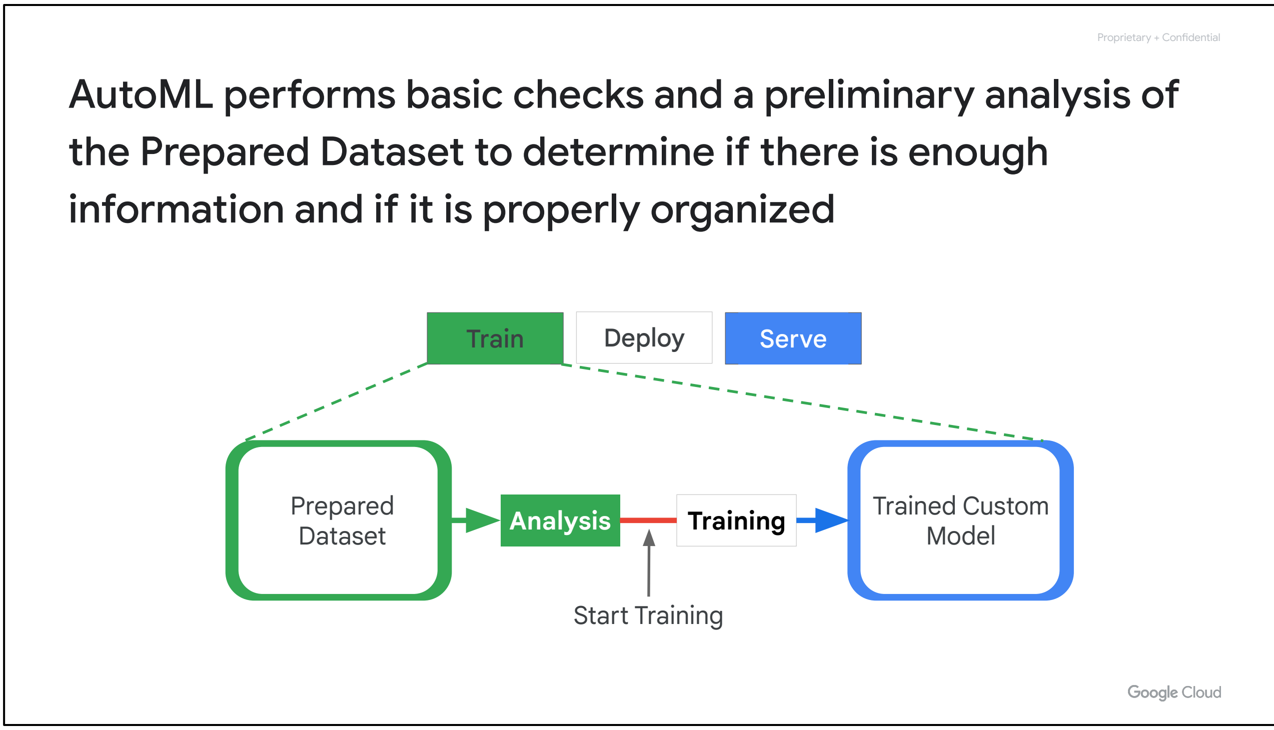
The next column in the CSV file identifies source files that are hosted in Cloud Storage. These are paths beginning with gs://

The source file format depends on the kind of model you are training but can also be compressed ZIP files.

Subsequent columns specify labels. The labels are alphanumeric and can contain underscores, but not special characters. The CSV file should not contain duplicate lines and may not contain blank lines or unicode characters.

Currently, the CSV file and all the Source Files must be in a Cloud Storage bucket in the project where AutoML runs.

Prepared Datasets do not expire. You may accumulate many Prepared Datasets in a project. You can list and delete those you don't need.



AutoML performs basic checks and a preliminary analysis of the Prepared Dataset to determine if there is enough information and if it is properly organized. If the Prepared Dataset is not ready, you will need to add more rows or more labels to the CSV file. When it is ready, you can start training.

Training can take from ten minutes to several hours depending on the kind of model. You can check the status while it is running. Import and training tasks can be canceled.

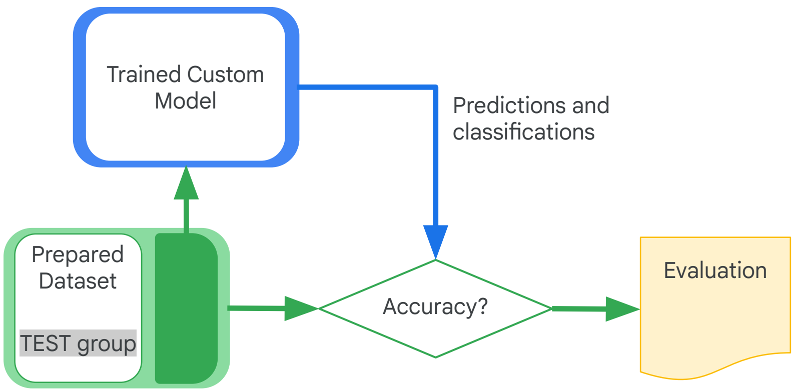
The TRAIN group of data is used to train the Custom Model. The source files have already been associated with the correct labels in the Prepared Dataset, so AutoML uses a supervised learning method to train the Custom Model. Part of the process uses the VALIDATION group data to verify how well the model works at classifying and predicting.

Supervised learning works on correctable errors. AutoML constructs an algorithm that guesses the labels for source data. When the guess is right, it strengthens the algorithm. When the guess is wrong, the error is used to correct the algorithm. And this is how learning occurs. One full run through all the TRAIN group data is called an epoch. Total error is tracked and minimized through multiple epochs to create the best model possible from the training data provided.

The result is a trained Custom Model.

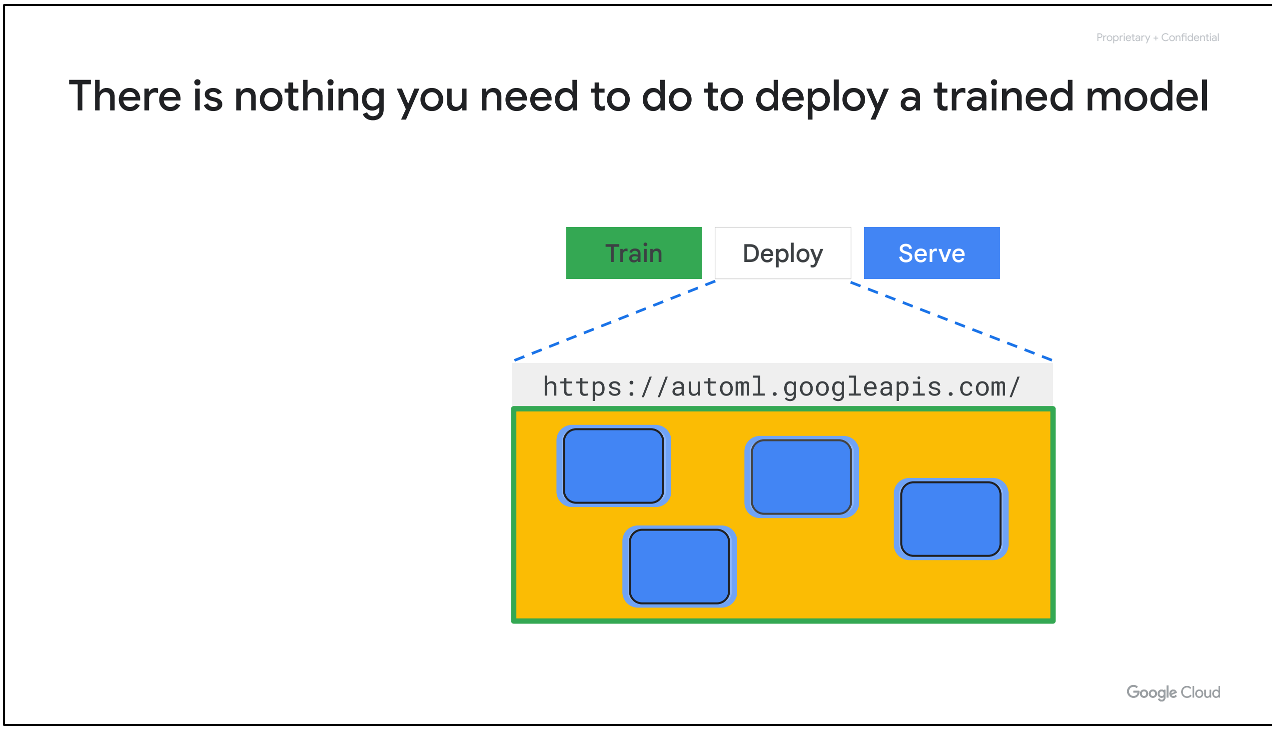
The custom model works well with the training data. But is it good at categorizing new instances of data it has not seen before?

Data from the TEST group is used to evaluate the Custom Model and to remove bias from the evaluation



Data from the TEST group is used to evaluate the Custom Model and to remove bias from the evaluation. The predictions and classifications are compared with the labels in the Prepared Dataset.

The evaluation report provides indicators that are specific to the kind of model and help understand how effective the model is at predicting and classifying.



There is nothing you need to do to activate a model. However, if it has been some time since you used a model, the system may need to "warm up" for a few minutes before the model becomes active.

Once it exists, if you have the project credentials and model-name you can access and use the Custom Model.

Each time you train with a Prepared Dataset it creates a new Custom Model.

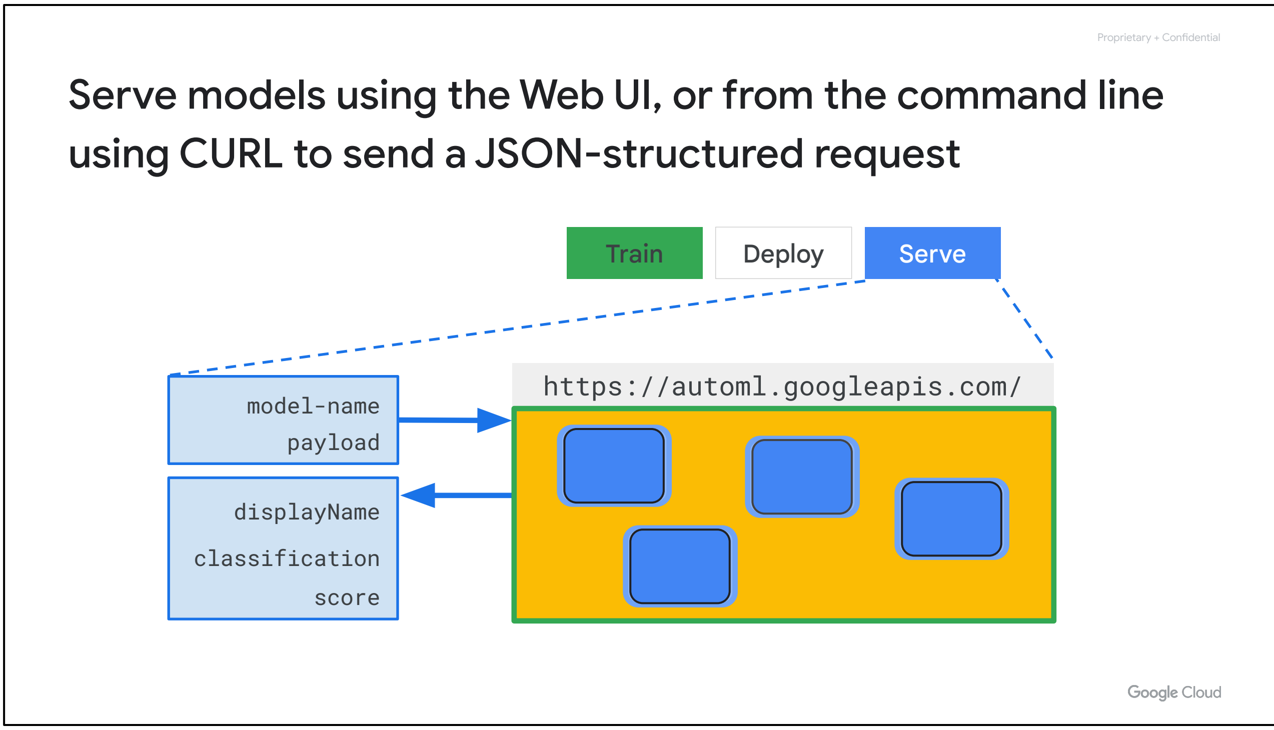
You can list and delete unneeded models.

Custom Models are temporary. They are eventually deleted. And they cannot be exported or saved externally.

Models that ***are not used*** for prediction are automatically deleted after a period.

And models that ***are*** used are eventually deleted. So you will need to train a new Custom Model periodically to continue predicting and classifying.

How long models remain before they are deleted depends on the model type.

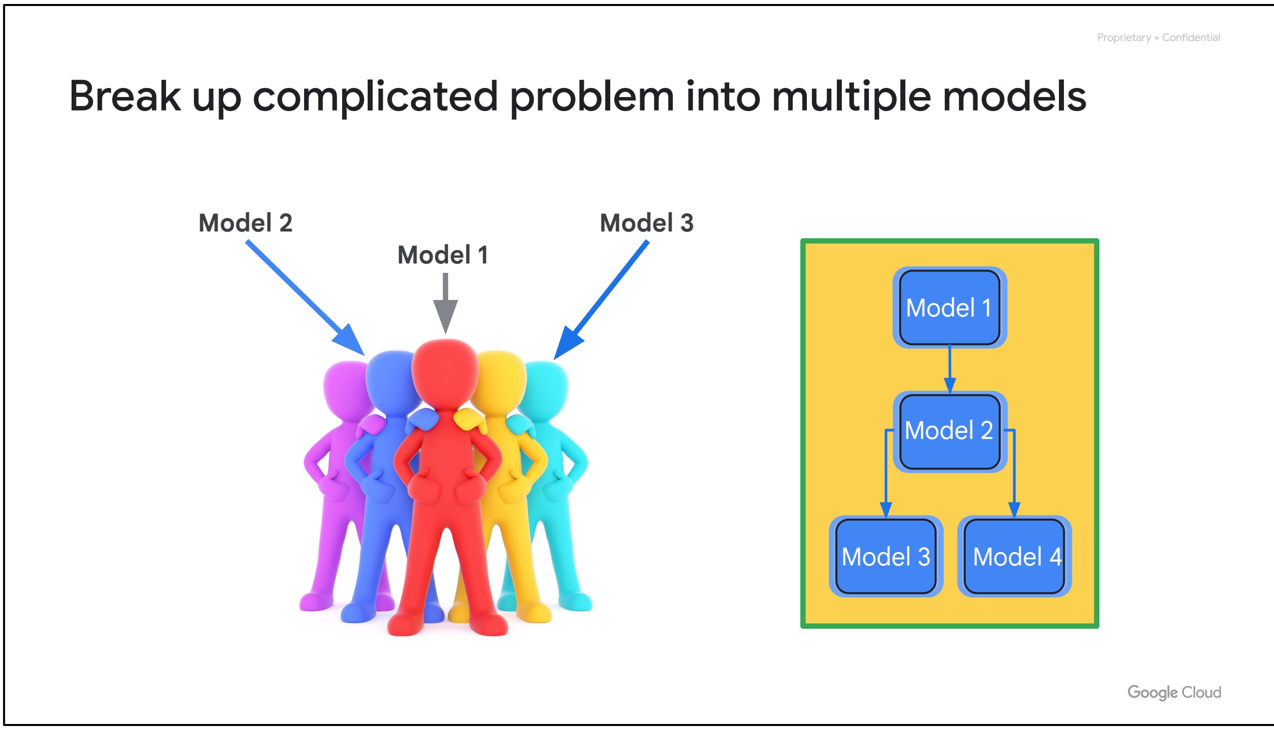


The primary classification interface is at the URI shown. You can make a classification using the Web UI, or from the command line using CURL to send a JSON-structured request. There are also client libraries for Python, Java, and Node.js.

After you have set up authentication to use the REST API, you can send a request with the model-name and the payload, which is the data you want classified.

The service returns JSON containing multiple fields called displayName. These are the labels that matched. Then it contains the keyword classification, followed by a score. The score is a confidence value, where 1.0 is absolute confidence, and lower fractional numbers represent lower confidence in the correctness of the classification.

Quotas apply for both model creation and service requests.



AutoML lowers the effort required to create a model when compared to traditional Machine Learning. With traditional ML, models were hard to create so there was a tendency to try to make the dataset and the model inclusive.

With AutoML you can create smaller, more specialized Custom Models and use them programmatically. So you don't have to squeeze everything into one model. You can break apart a classification into multiple steps. And you can use the results of one classification to make choices about what kind of classification to perform next.

Here’s an example:

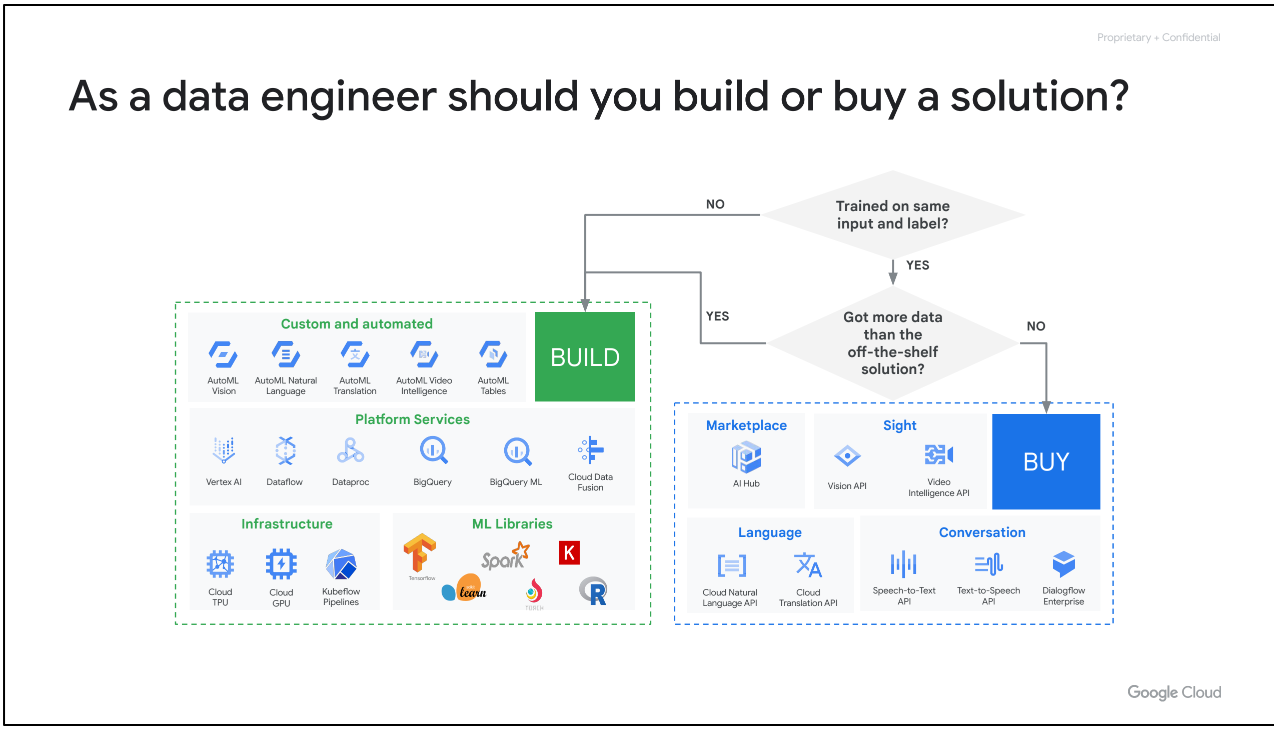
A company that sells clothing has a service office that receives emails from customers. The first job might be to distinguish emails containing feedback about products from emails requesting information about the company. Model 1 could be used to classify feedback email.

The second job might be to distinguish whether the email is describing pants, shirts, shoes, shirts, or hats. And this might be the job of Model 2.

Model 3 might be used only for emails talking about shirts, to see if the style of the shirt is mentioned. And Model 4 might be used only for emails about shoes, to see if the shoe style is mentioned.

You can see from this example that a collection of models might be able to accomplish magic in your application by focusing the scope and purpose of the models.

You can also programmatically combine your custom model with a standard model, such as the Cloud Natural Language API.

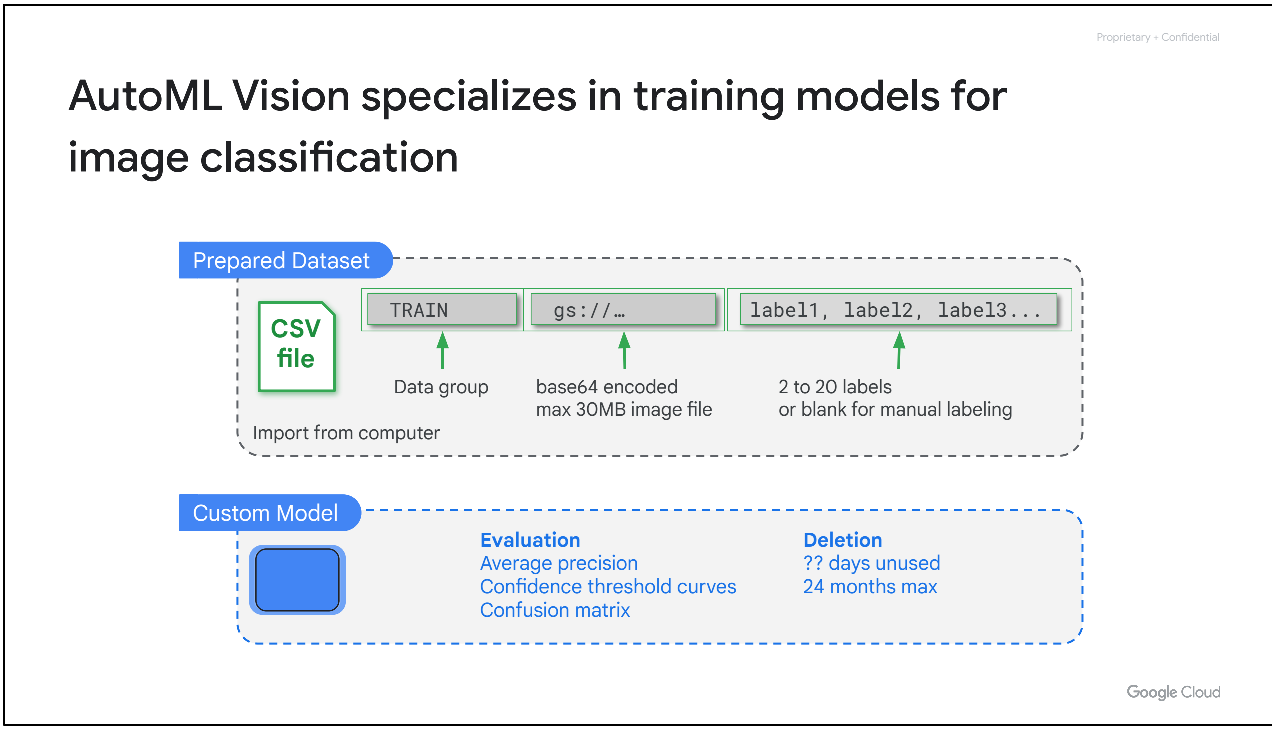


This concludes the discussion of AutoML.

The recommended application strategy is to first, use the pre-built artificial intelligence services. Next, you can use AutoML to produce Custom Models which can be used with the pre-built services or on their own. Remember that you can divide a problem into specialized parts and use multiple Custom Models together. Finally, if you discover you need more advanced features, you can use the Machine Learning and Artificial Intelligence services to create new models.

Custom Model Building with AutoML

Now, we'll describe AutoML Vision. This is an AutoML product for image data.



AutoML Vision specializes in training models for image classification.

You can load the CSV file and the image files from Cloud Storage or you can upload them from your local computer using **Import**.

Training supports several file formats including JPEG and PNG. The images can be up to 30 megabytes in size. Images have to be converted to base64 encoding which stores the image as a text file. So the prepared file will be a TXT file or a compressed ZIP file.

The Service only recognizes JPEG, PNG, and GIF files up to 1.5 megabytes.

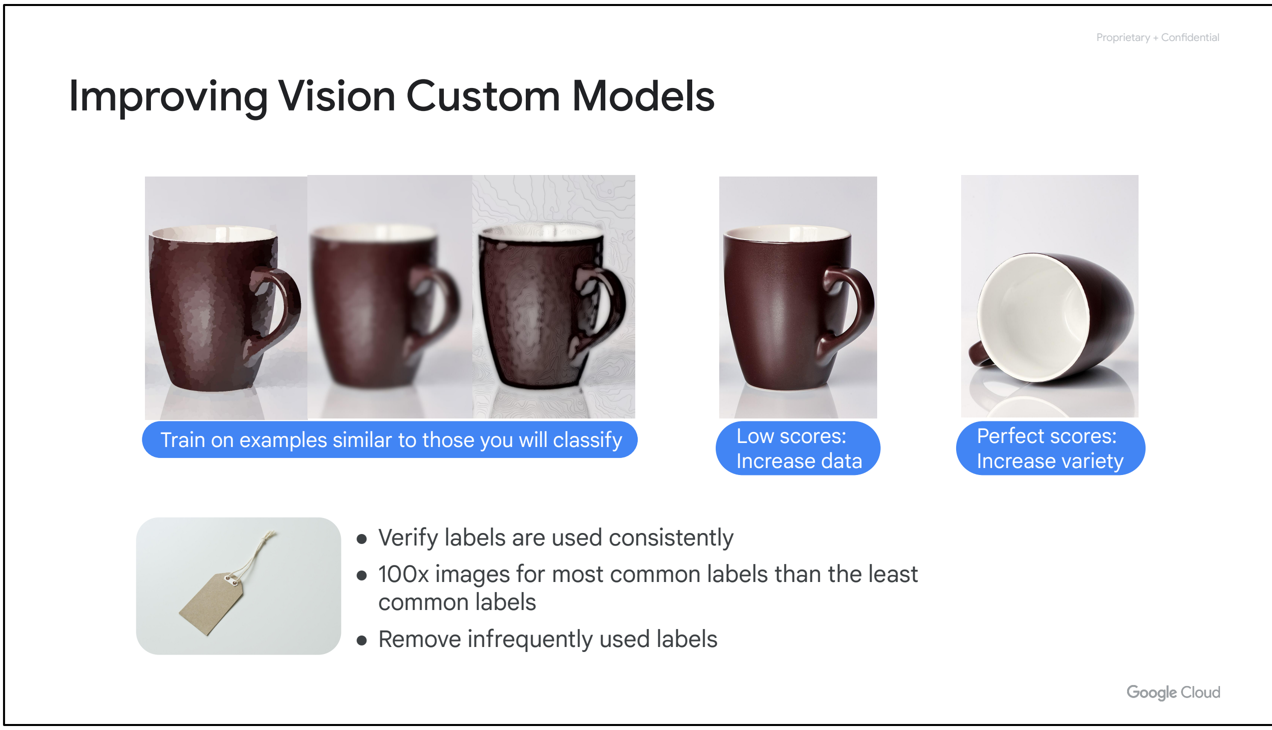
The trained model will work best when there are at most 100-times more images for the most common label than for the least common label. Consequently, for model performance, it is recommended that you remove very low frequency labels.

You can label the images in the Web UI or, in some cases, you can use the human labeling service offered by Google if you have more than 100 unlabeled images.

To gauge your trained model’s readiness, AutoML Vision creates the confusion matrix for up to 10 labels. If you have more than 10 labels, the matrix includes the 10 labels with the most incorrect predictions.

Use this data to evaluate your model's readiness.

Training file formats include JPEG, PNG, WEBP, GIF, BMP, TIFF, ICO up to 30 MB.
Service requests support JPEG, PNG, or GIF files up to 1.5MB.



The quality of Vision Custom Models has a lot to do with the choice of training data. Train on images with similar properties to those you intend to classify. For example, images of similar resolution, lighting, focus, and level of detail.

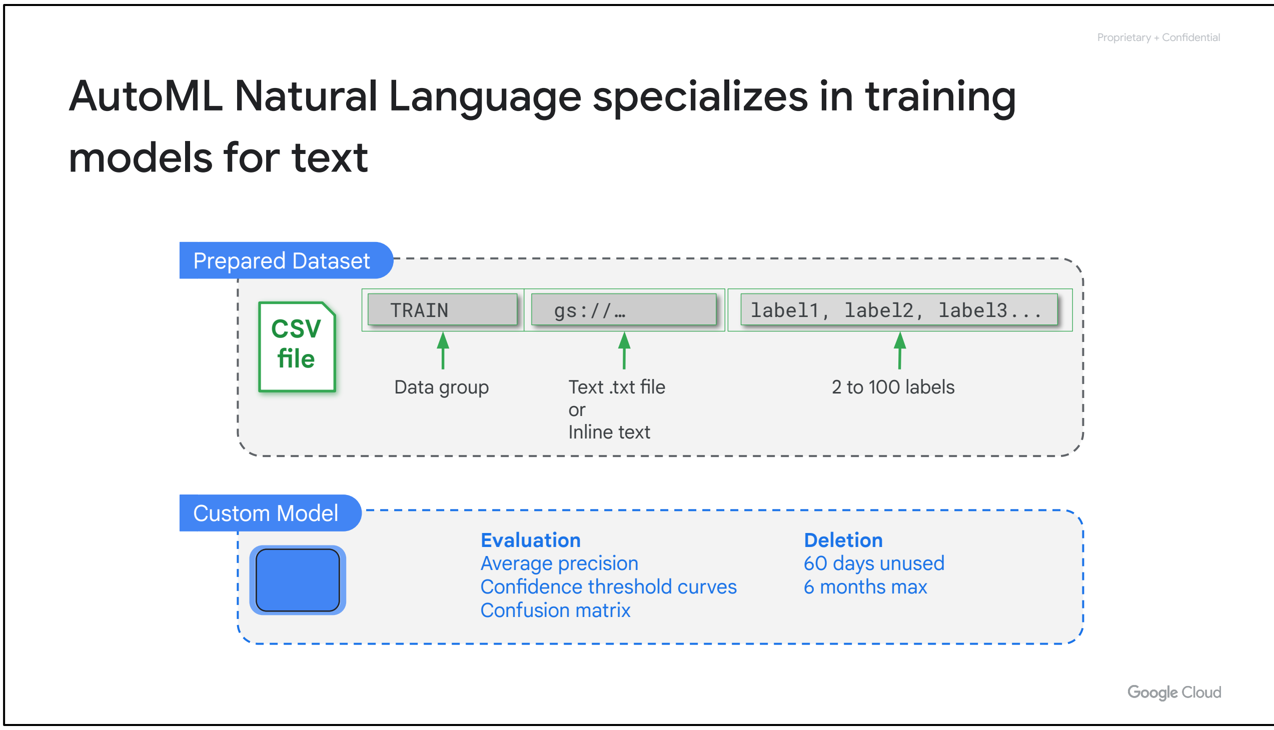
High confusion, low average precision scores, or low precision and recall scores can indicate that your model needs additional training data or has inconsistent labels.

Perfect is the enemy of good. Perfect or very high average precision scores could indicate that something is wrong in the model. The data is too easy and not varied enough. It could mean the model might not work well beyond the test data. In this case, increase the variety of images in the Prepared Dataset.

Custom Model Building with AutoML

Google Cloud

Let’s move on to AutoML Natural Language.



AutoML Natural Language specializes in training models for text data. For example, if you have a set of newspaper articles you can use the AutoML Natural Language service to classify if a given article is, for example, about sports or politics.

- The text to be recognized can be inline text in the cell of the CSV file.
- More commonly the text is contained in documents which are .txt files or compressed in zip format.
- The path to the Cloud Storage location of the document appears in the CSV file.
- Currently, the documents must be standard text and does not support unicode.
- The documents can be as small as one sentence or up to a maximum of 128 kilobytes.
- You can have from 2 to 100 labels.

The Custom Model is evaluated on average precision. That is a value from .5 to 1.0. Its formal name is the "Area under the Precision/Recall curve". A higher number indicates more accurate classification and prediction. The evaluation report also supplies confidence threshold curves, which is a way of characterizing false positive classification against true positives. For models that apply one label per document, the evaluation includes a confusion matrix. You can read more about the evaluations and how to interpret them in the online documentation.

You can also view evaluations for each label.

If a Natural Language Custom Model is not used for 60 days, it will be deleted. If a Natural Language Custom Model is being used, it will be deleted after 6 months. To preserve a model you'll be required to retrain it.

The training and serving methods inside AutoML are frequently improved and updated. These changes are not guaranteed to be backwardly compatible. They may render a Custom Model incompatible with the current service. So you should plan to periodically re-generate the Custom Model to keep using it.

[https://cloud.google.com/natural-language/automl/docs/evaluate]

Proprietary + Confidential

Improving Natural Language custom models

Add more documents

Increase document variety

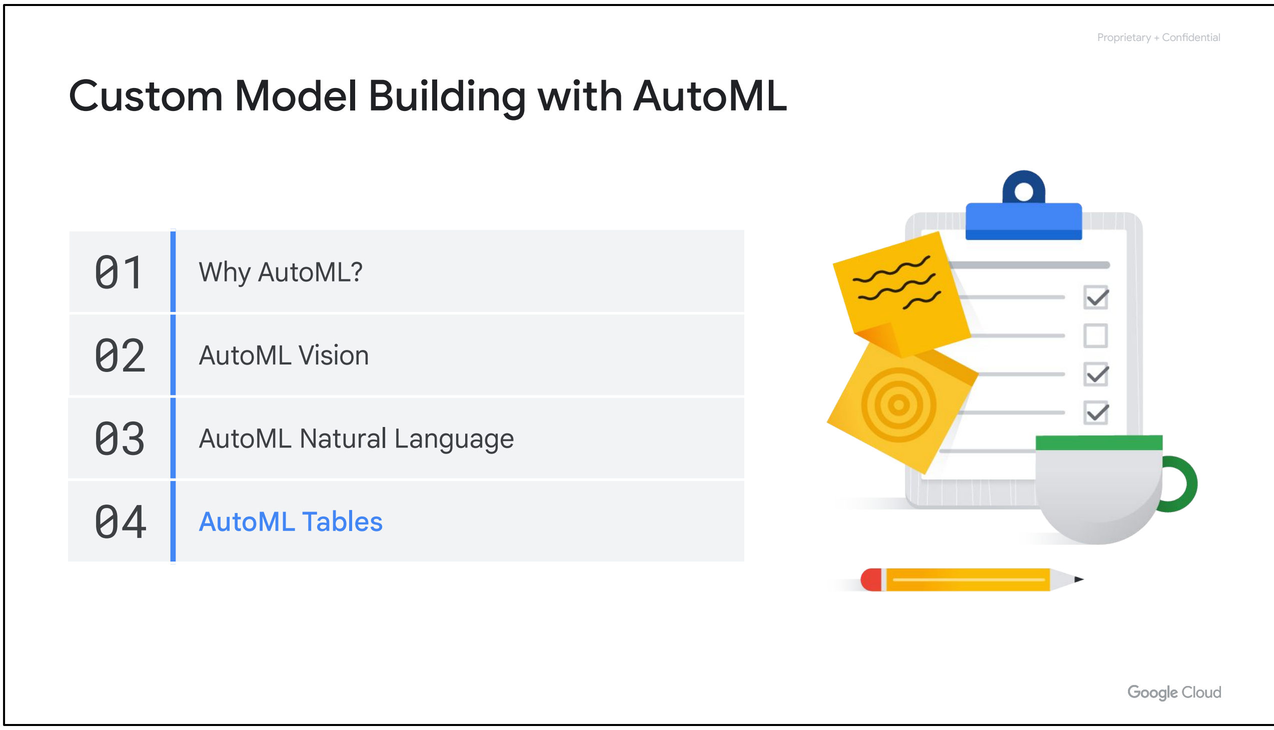
Reduce the number of labels

Google Cloud

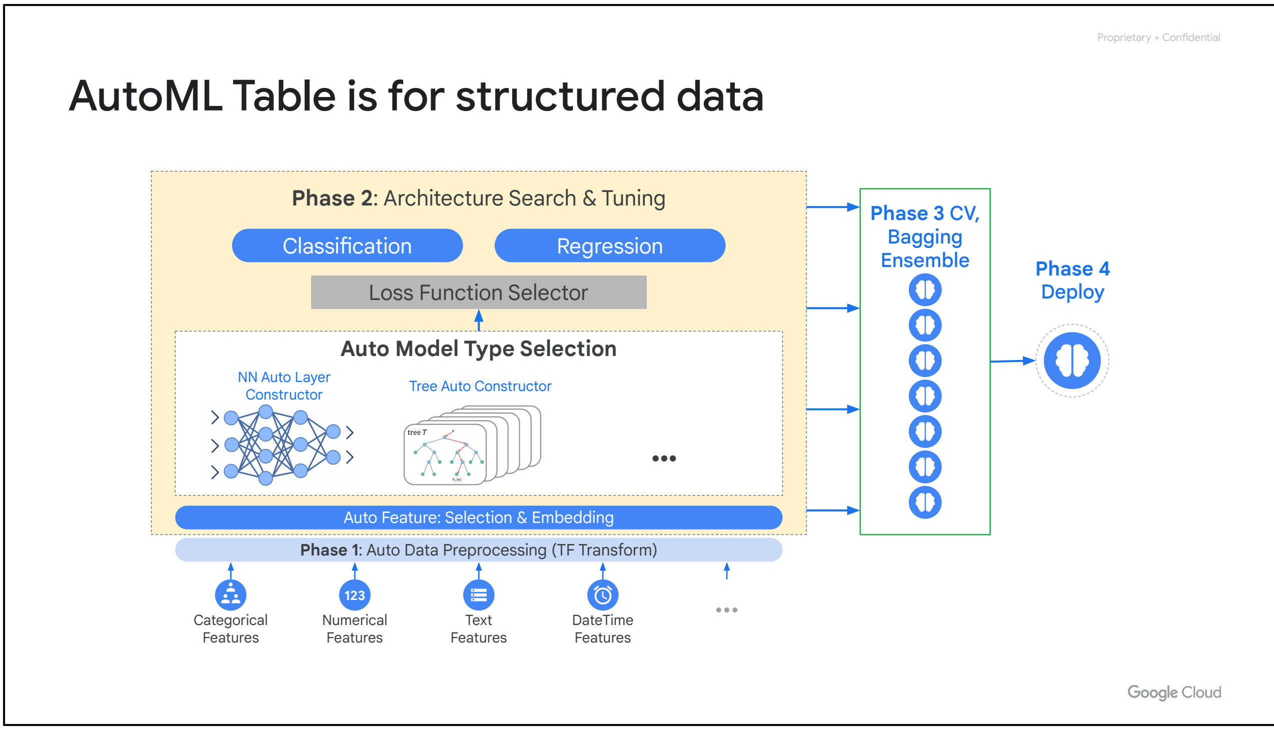
High confusion and low average precision scores indicate that a Prepared Dataset needs additional entries or that the labels are being used inconsistently.

You may be able to improve low quality evaluations for particular labels with one of these methods:

- Add more documents associated with those labels. In other words, there might just not be enough training data to get a good result.
- You also may need to increase the variety of documents by adding longer or shorter examples, documents with different writing styles or word choice, or by different authors.
- Finally, for labels that are not useful or have low quality, you may want to remove them altogether to increase the accuracy of the remaining labels.



Finally, let’s cover AutoML Tables for tabular data. Tabular data is what you might find in a spreadsheet, for example.



While AutoML Vision and Natural Language are for unstructured data, AutoML Table is for structured data. The development of AutoML Table was a collaboration between Google Cloud and the Google Brain team. While the technical details of the project haven’t been released to the public, the team basically took the architecture for the search capability used in image classification and translation problems and found a way to apply it to tabular data.

Example: Mercari price suggestion challenge

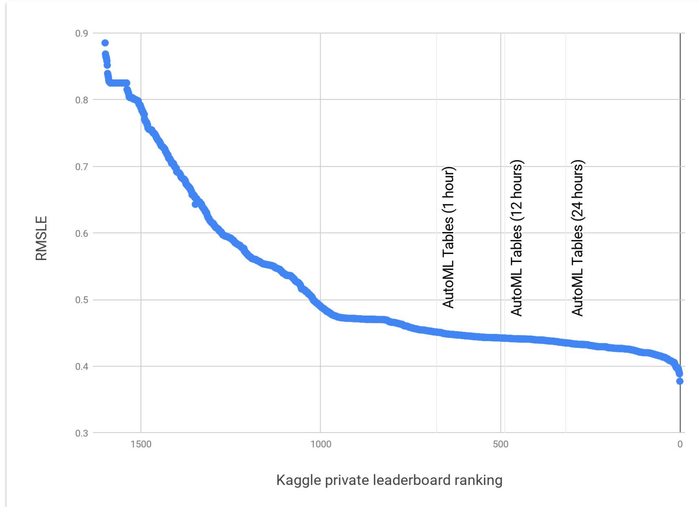
Goal: Automatically suggest product prices to online sellers

Training data

ID	Name	Item Condition	Categories	Brand name	Shipping	Item description	Price
0	MLB Cincinnati Reds T Shirt Size XL	3	Men, Tops, T-shirts		1	No description yet.	$10
1	Razer BlackWidow Chroma Keyboard	3	Electronics, Computers & Tablets, Components & Parts	Razer	0	This keyboard is in great condition and works like it came out of the box. All of the ports are tested and work perfectly. The lights are customizable via the Razer Synapse app on your PC.	$52
2	AVA-VIV Blouse	1	Women, Tops & Blouses, Blouse	Target	1	Adorable top with a hint of lace and a key hole in the back! The pale pink is a 1X, and I also have a 3X available in white!	$10
3	Leather Horse Statues	1	Home, Home Décor, Home Décor Accents		1	New with tags. Leather horses. Retail for [rm] each. Stand about a foot high. They are being sold as a pair. Any questions please ask. Free shipping. Just got out of storage.	$35

Let's describe a dataset where AutoML Tables performs really well; Mercari's price suggestion challenge. Mercari is Japan's biggest community-powered shopping app and marketplace. Mercari created a price suggestion challenge for predicting the price of a product offered on their marketplace, so that they could give price suggestions to their sellers. Participants were given some 1.5 million rows of rich data with plenty of noise. The challenge lasted for 3 months, and culminated in a $100,000 prize. Over 2000 data scientists competed for the prize.

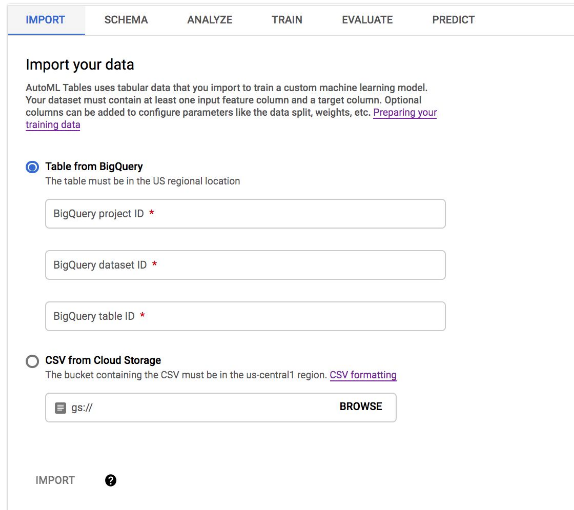
AutoML Tables produced some of the best results on the challenge



This plot shows the performance of AutoML Tables on the Mercari Challenge for several different training times. You can see that after 24 hours of training, AutoML Tables pretty much puts you on the leaderboard. Even after only one hour of training you get to the plateau of leaders, which is extremely impressive performance on a million-plus row dataset with significant complexity. Compared to the $100,000 prize for this challenge, one hour of training is just $19.

Since the search process for AutoML tables is random you might get slightly different results if you tried to reproduce this performance.

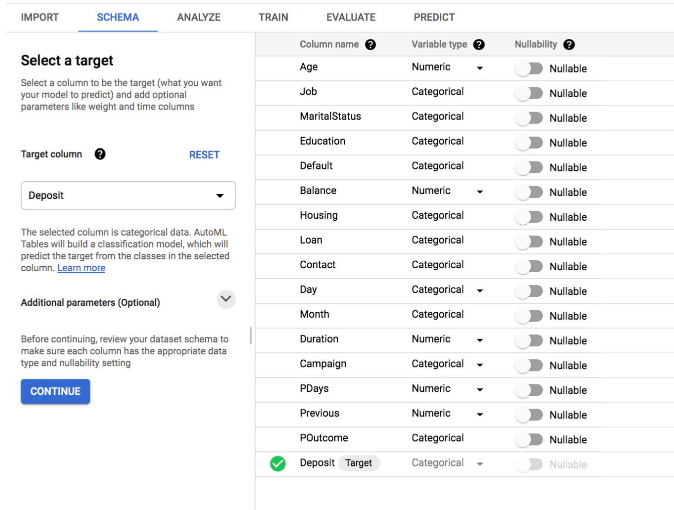
The easiest way to import data into AutoML Tables is through BigQuery



Google Cloud

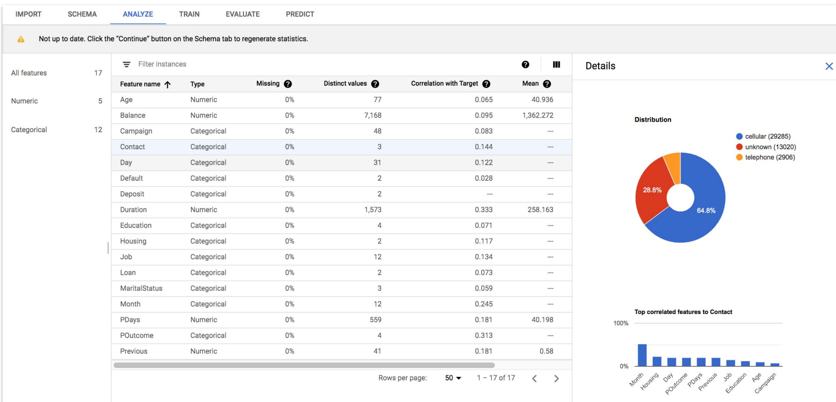
The easiest way to import your data into AutoML Tables is through BigQuery. You can also import data using CSV files stored locally or on Cloud Storage. One of the advantages of importing data through BigQuery is its support for arrays and structs. Regardless, for both import sources your data must have between 1,000 and 100 million rows, between 2 and 1,000 columns and be 100 gigabytes or less in size.

Start by setting the features/label that will be used for training



Once your data is imported, the next step is to select the features you want to use and to specify the column you're trying to predict.

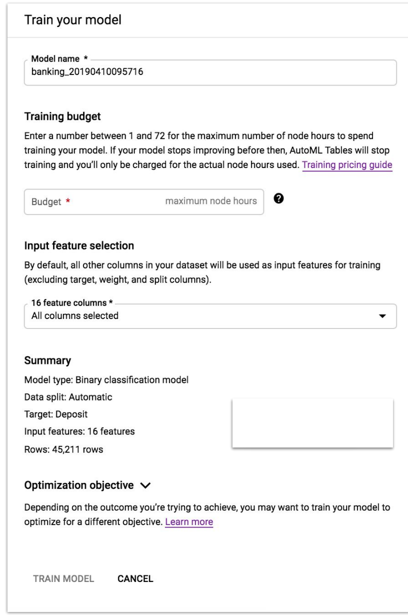
Next, do some data validation to ensure you're not passing junk into your model



Carry out some experiments in BigQuery ML to set some base metrics for model performance

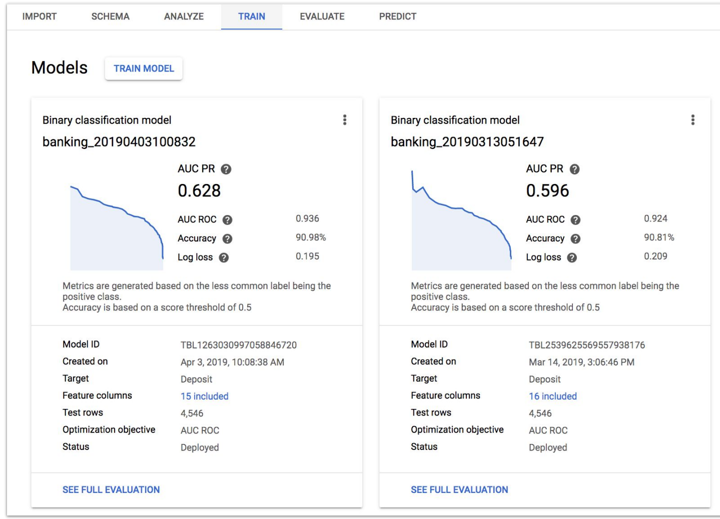
In the next step of building an AutoML Table model, you go through a data validation phase. The purpose of this step is to ensure you're not passing bad data to your model. This includes checking for columns that have too many null values, outlier columns that are skewing the distribution of a column, and columns that are not correlated to the target you're trying to predict.

You can allocate a budget when training the model



As you saw in the slide on AutoML Table's performance on the Mercari challenge, you can train a model for a variable amount of time. You can set a training budget in node-hours to cap costs. By default, AutoML tables will stop training if the model isn't seeing significant performance gains anymore.

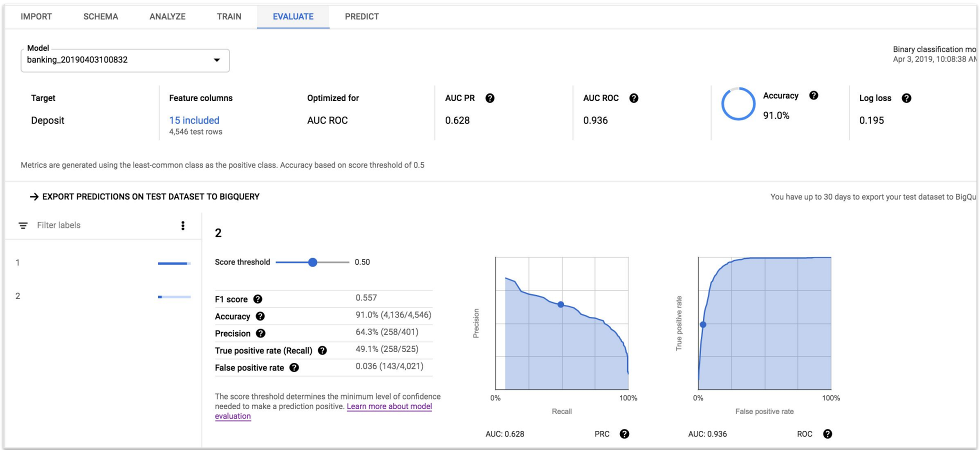
Inspect the training metrics across multiple models



Google Cloud

Once your model is trained, you should look at the training metrics. Be wary of models that are too good to be true. In this case you likely have a data issue you'll need to resolve. For classification, the report includes metrics such as area under the curve for the precision-recall curve, accuracy, and the F1 score. Also a confusion matrix is output along with feature importances. These two sets of metrics are particularly useful in diagnosing low-performing models. For regression models, the root mean squared error, mean absolute percentage error, and feature importances are returned among other metrics. Check the AutoML Table documentation for a full list of metrics that get generated after model training.

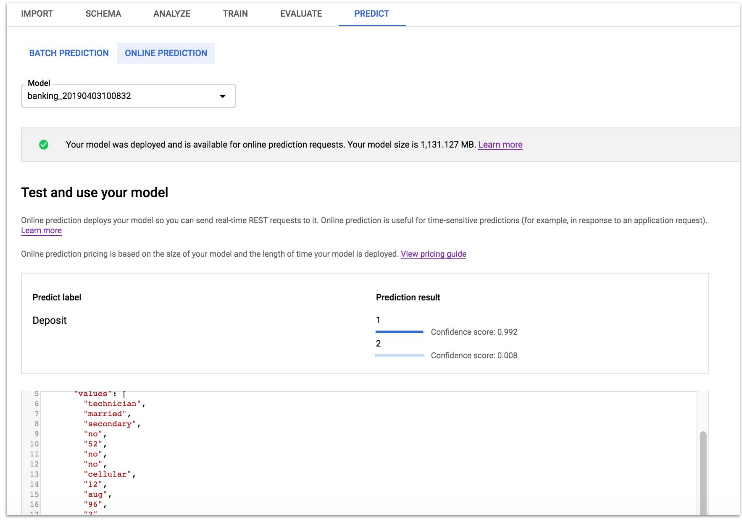
Check how model performs against test data to gauge how well it will generalize in the wild



It's arguably more important to look at the performance metrics generated on the test set to get a feel for how well your model will generalize. The same metrics generated for the training data are available for the test data. For classification models it may be useful to set the score threshold to a value other than the default of 0.5. Increase the score threshold to make your classifier output a positive label with more confidence.

Proprietary + Confidential

Integrate your trained model into your applications



Google Cloud

Once you’re happy with your model performance you can go ahead and deploy it. You have the option of making batch or online predictions. For online predictions you can make calls using a curl command, or by one of the Java, node.js or Python APIs. The same APIs are available for batch predictions. You can make batch predictions on either BigQuery tables or CSV files. However, the BigQuery data source tables must be no larger than 100 gigabytes. For CSV files, each data source file can be no larger than 10 gigabytes and if you include multiple files, the sum of all files cannot exceed 100 gigabytes.

Proprietary + Confidential

How to choose between BigQuery ML, AutoML and a custom model

Model type	BigQuery ML	AutoML	Custom deep learning model
How	SQL in BigQuery for ML on structured data	AutoML uses neural architecture search and best-of-class model architectures for the specific problem	Keras with a TensorFlow backend, trained on Cloud ML Engine
Best if you are a	Data analyst who can wrangle data with SQL	Developer who can create the dataset in the required format	ML Engineer who knows Python and knows deep learning, NLP techniques
How long it takes an experienced practitioner	About an hour	About a day	A week to a month
Most of this time is spent in	Writing SQL	Waiting for job to finish	Coding Python and experimentation with ML
Cloud computing costs	Low	Medium	Medium to high depending on size of data, number of experiments, etc.
Accuracy	Moderate to high, mostly depending on the size of your dataset	High	Low if you don't know what you are doing; extremely high if you employ appropriate architectures and have a large-enough dataset

Google Cloud

So to close out this module, let’s return to the question of when should you use BigQuery ML or AutoML versus building a custom model?

The short answer is, it depends on how much time you have to build the model and what resources you have available. This table may provide some guidance. Given the low barrier of building a model in either BigQuery ML or AutoML, give either a try first. If the resulting model is not sufficient, only then should you throw more resources at the problem you are seeking to solve.

Proprietary + Confidential

Summary

- AutoML can be used to create powerful ML models without any coding.
- Use AutoML Vision when you have image data.
- Use AutoML Natural Language when you have text data.
- Use AutoML Tables when you have structured data.

Google Cloud

To summarize, AutoML can build really powerful machine learning models with no coding. The models will be customized to your data. Use AutoML Vision for image data, AutoML Natural Language for text data and AutoML Tables for structured or tabular data.