

## Modeling POC

Now, we will create a POC of our modeling hypothesis using Google Cloud AutoML service in Vertex AI (GCP's Machine Learning Platform).

### Costs

- Vertex AI: <https://cloud.google.com/vertex-ai/pricing>
- Cloud Storage: <https://cloud.google.com/storage/pricing>

### Setup

- GCP project: <https://cloud.google.com/resource-manager/docs/creating-managing-projects>
- Enable Vertex AI and grant access roles: <https://cloud.google.com/vertex-ai/docs/featurestore/setup> and <https://cloud.google.com/vertex-ai/docs/general/access-control>

### POC

We will do our POC from the Vertex console (<http://console.cloud.google.com/>) as we're short on time. But we can also do this programmatically using the Vertex AI SDK (`google-cloud-aiplatform`). AutoML is a great alternative to start an ML project as it takes care of the data preparation and model testing. The models used are ensemble tree-based and deep-learning models.

The first thing we'll do is create a Dataset (great for version control of our training data).

The screenshot shows the 'Create dataset' page in the Google Cloud Vertex AI console. The 'Dataset name' field is filled with 'online-retail-example'. Below this, the 'Select a data type and objective' section is active, showing 'TABULAR' as the selected data type. Under 'TABULAR', there are two options: 'Regression/classification' (selected with a radio button) and 'Forecasting'. The 'Region' dropdown is set to 'us-central1 (Iowa)'. At the bottom, there are 'CREATE' and 'CANCEL' buttons.

Dataset name \*  
online-retail-example  
Can use up to 128 characters.

Select a data type and objective  
First select the type of data your dataset will contain. Then select an objective, which is the outcome that you want to achieve with the trained model. [Learn more](#)

IMAGE **TABULAR** TEXT VIDEO

☒ Regression/classification  
Predict a target column's value.  
Supports tables with hundreds of columns and millions of rows.

☐ Forecasting  
Predict the likelihood of certain events or demand.

Region  
us-central1 (Iowa)

ADVANCED OPTIONS

CREATE CANCEL

We create it from a local file, but it can be created from a file in a Google Cloud Storage location or a BigQuery table.

online-retail-example

CREATE LABELING TASK

SOURCE ANALYZE

### Add data to your dataset

Before you begin, read the [data guide](#) to learn how to prepare your data. Then choose a data source.

#### Select a data source

- CSV file: Can be uploaded from your computer or on Cloud Storage. [Learn more](#)
- BigQuery: Select a table or view from BigQuery. [Learn more](#)
- ☒ Upload CSV files from your computer
- ☐ Select CSV files from Cloud Storage
- ☐ Select a table or view from BigQuery

#### Upload CSV files from your computer

Add up to 500 CSV files per upload. The files will be stored in a new Cloud Storage bucket ([charges apply](#)). Data from multiple files will be referenced as one dataset.

ModelData.csv 1 file X

[SELECT FILES](#)

#### Select a Cloud Storage path

Choose where your uploaded CSV files will be stored ([charges apply](#))

Cloud Storage path \*

### Summary

You can build two model types with tabular data. The model type is automatically chosen based on the data type of your target column.

- Regression models predict a numeric value. For example, predicting home prices or consumer spending.
- Classification models predict a category from a fixed number of categories. Examples include predicting whether an email is spam or not, or classes a student might be interested in attending.

Next, we create a new training job.

### Train new model

- ☒ Training method
- 2 Model details**
- 3 Training options
- 4 Compute and pricing

START TRAINING CANCEL

☒ Train new model  
Creates a new model group and assigns the trained model as version 1

☐ Train new version  
Trains model as a version of an existing model

Name \*  
classification-online-retail-example

Description  
Model classifies if a customer will make a future purchase.

Target column \*  
FuturePurchase

☐ Export test dataset to BigQuery

#### Data split

☒ Random  
80% of your data is randomly assigned for training, 10% for validation, and 10% for testing

• Training: 80%  
• Validation: 10%  
• Test: 10%

☐ Manual  
You assign each data row for training, validation, and testing. [Learn more](#)

☐ Chronological  
The earliest 80% of your data is assigned to training, the next 10% for validation and the latest 10% for testing. This option requires a Time column in your dataset. [Learn more](#)

Training 80% Validation 10% Testing 10%

Start time End time

Now, what will we choose to optimize? One of the main objectives of this modeling exercise may be sending directed offers or marketing to users whose probability of making a future purchase is low.

If it doesn't matter if we bother a few customers that were already going to buy with an email, but we certainly don't want to miss any customers that will not buy to target them with a discount or recommendation, we should go for maximizing **recall**.

If we don't want to give discounts to customers that were already going to buy (lost revenue), and we don't care to not target some customers that were not going to buy with additional marketing. We should go ahead and maximize **precision**.

As we don't know at the moment, which is more important to the business, we will go for a balance of optimizing the precision recall. Finding a trade-off between both metrics.

Filter Enter property name or value

<input type="checkbox"/>	Column name ↑	Transformation	Missing % (count) ?	Distinct values ?	Correlation w/ target ?	
<input type="checkbox"/>	Description	Text ▾	-	-	-	⊖
<input type="checkbox"/>	FuturePurchase		-	-	-	
	Target					
<input type="checkbox"/>	InvoiceDate	Timestamp ▾	-	-	-	⊖
<input type="checkbox"/>	PurchaseNumber	Automatic ▾	-	-	-	⊖
<input type="checkbox"/>	TotalPaid	Automatic ▾	-	-	-	⊖

Total 5 feature columns are included in the training

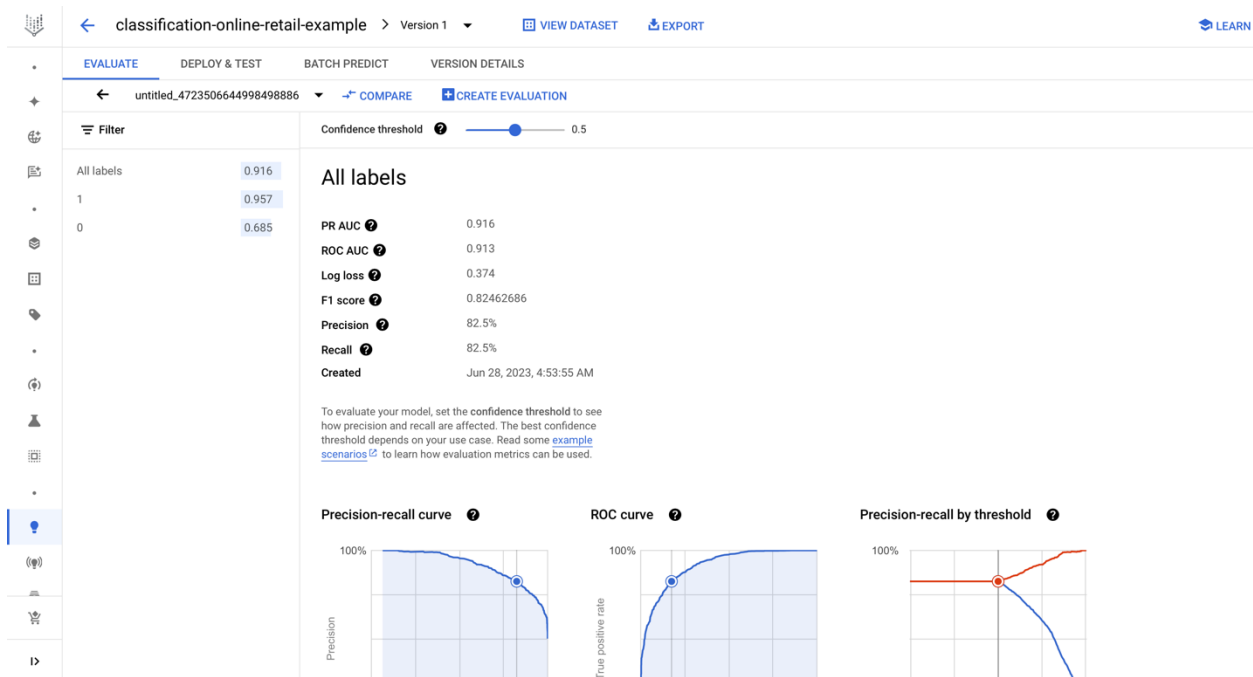
#### Weight column

Select a column to specify how to weight each row of the training data. By default, each row of your training data is weighted equally. ?

#### Optimization objective \*

- ☐ AUC ROC  
Distinguish between classes
- ☐ Log loss  
Keeps prediction probabilities as accurate as possible
- ☒ AUC PRC  
Maximize precision-recall for the less common class
- ☐ Precision at recall  
Maximize precision for the less common class
- ☐ Recall at precision  
Maximize recall for the less common class

After the training job finishes training we can analyze the results of our ML POC as AutoML provides the necessary evaluation metrics.



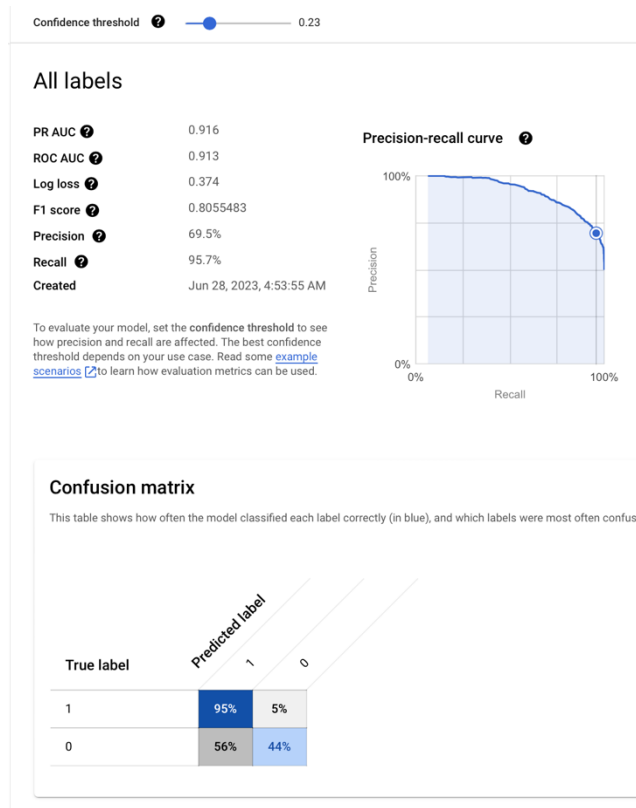
## Confusion matrix

This table shows how often the model classified each label correctly (in blue), and

True label	Predicted label	
	1	0
1	1154	63
0	219	172

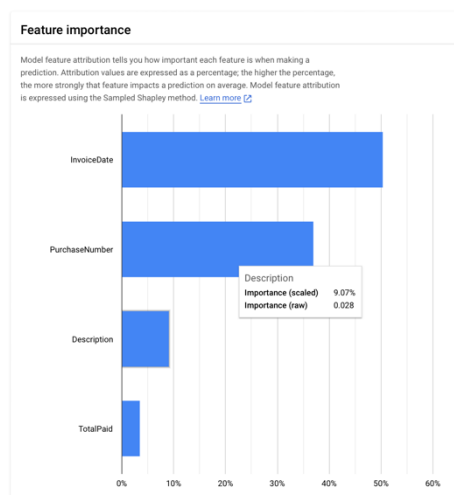
We can see that the model is not doing well at identifying people that will not make a future purchase. The performance is as if, for a customer that doesn't have the intention of buying, we randomly sent an email with a discount or marketing to make them buy. This way we will lose the opportunity to retain customers. On the other hand, the model is good at identifying buying customers, which means we can save money by not sending discounts to customers that were already going to return to make a purchase.

Even if we reduce the confidence threshold we see that the model is missing information to identify customers that will not make a future purchase. This means that is confidently guessing as “will buy in the future” examples that we have categorized as “will not buy”. A thing that could be happening is that we are **failing to correctly create the labels in our data**. With our current way of creating the labels we are assuming that if a customer doesn’t have a next purchase, it will not buy.



This first approach helps identify possible causes of unwanted behavior so that we can iterate to try different ways of creating features and labels to solve our problem.

AutoML providing information on feature importance is also helpful to see which information we’re using to make our predictions, and again, iter in our modeling process. As we can observe, the features that impact the most in our model are the invoice date (with autoML date transformations <https://cloud.google.com/vertex-ai/docs/datasets/data-types-tabular>), and the purchase number (feature importance with Shapley values <https://towardsdatascience.com/the-shapley-value-for-ml-models-f1100bff78d1>).



With this information, we would propose the following change in our model framing:

- **LABEL:** We will handle customers' last purchases quite differently from other purchases. For every other purchase (not the last one): if a customer has  $n$  transactions (distinct invoice numbers), for the first  $n-1$  purchases, we will mark the label "**FuturePurchase**" as 1. To obtain the last purchase's label, we will observe the distribution of the time between purchases, if the time between the last purchase and the current maximum date in our data is longer than a defined threshold, we will mark the label "**FuturePurchase**" as 0. If not, we will not use the purchase in our modeling data as we can't be sure if the customer will purchase again or not.
- **FEATURES:** in terms of the features, we could provide more information of the customer's purchase history by adding an additional "TimeSinceLastPurchase" variable. We could also experiment with data feature engineering (as finding a better way to aggregate the product description data for an invoice -> right now we concatenate texts and then embed with AutoML, we could embed the text first and try different aggregation methods).

Once we have features and a model framing, we find useful we can start experimenting with custom-trained models and hyperparameter tuning.