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

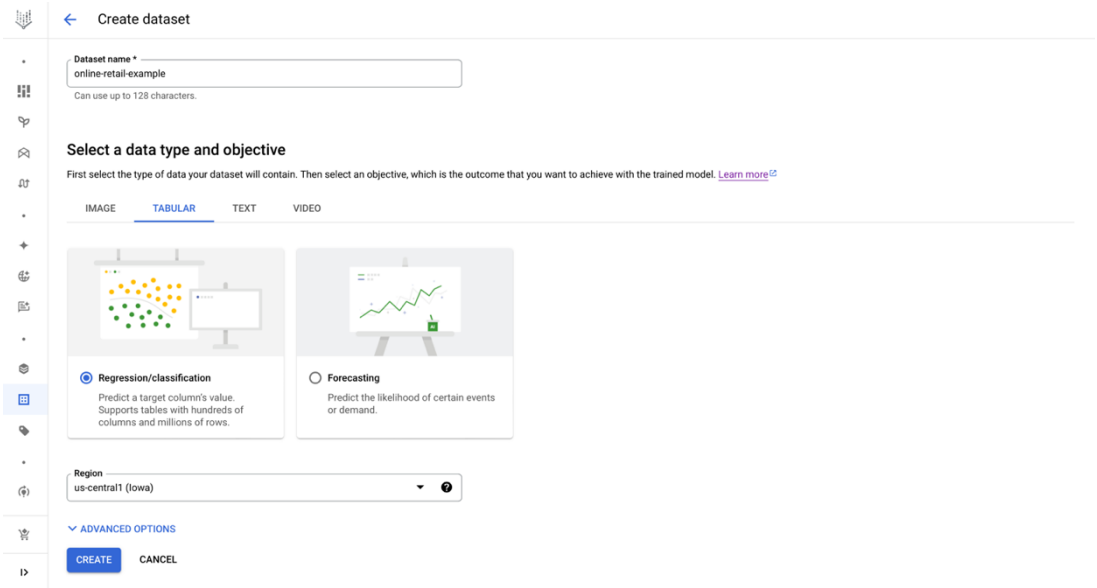
Setup

- 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

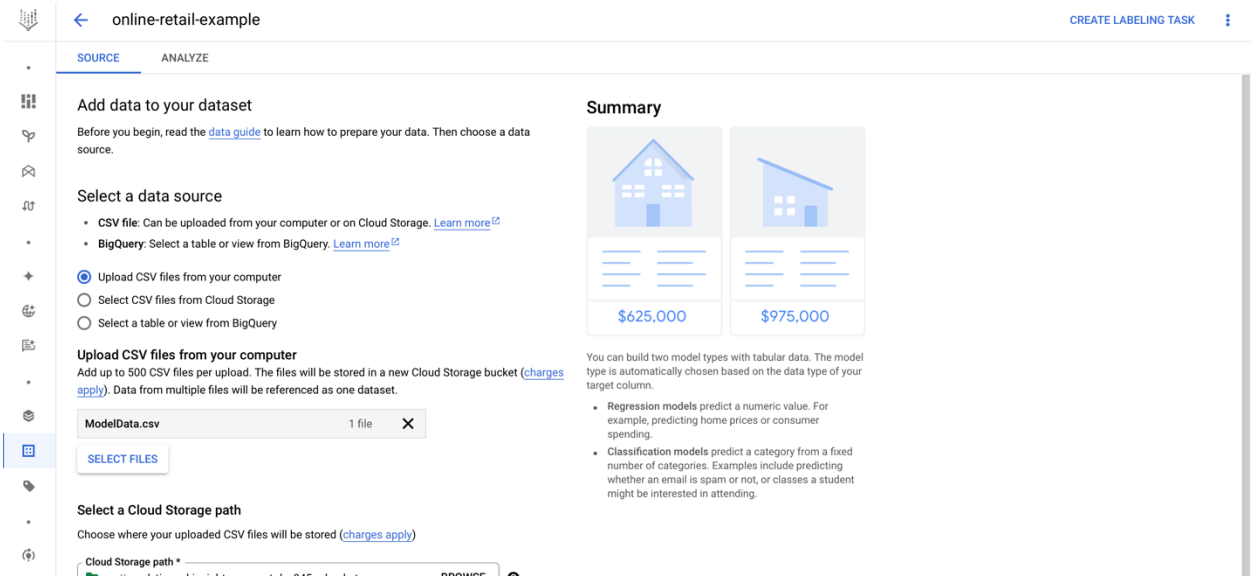
POC

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.

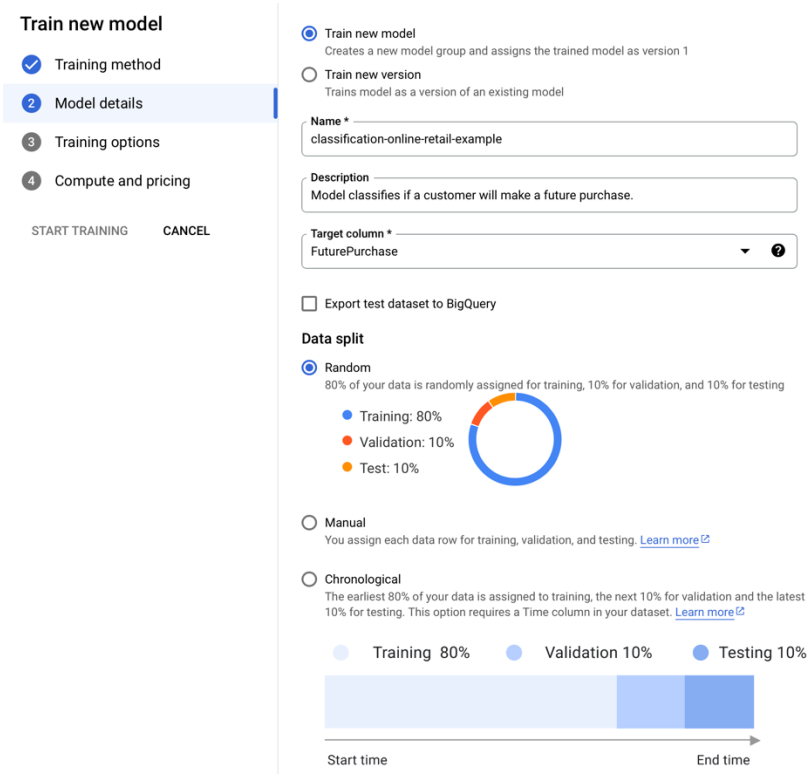
The first thing we'll do is create a Dataset (great for version control of our training data).



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.



Next, we create a new training job.

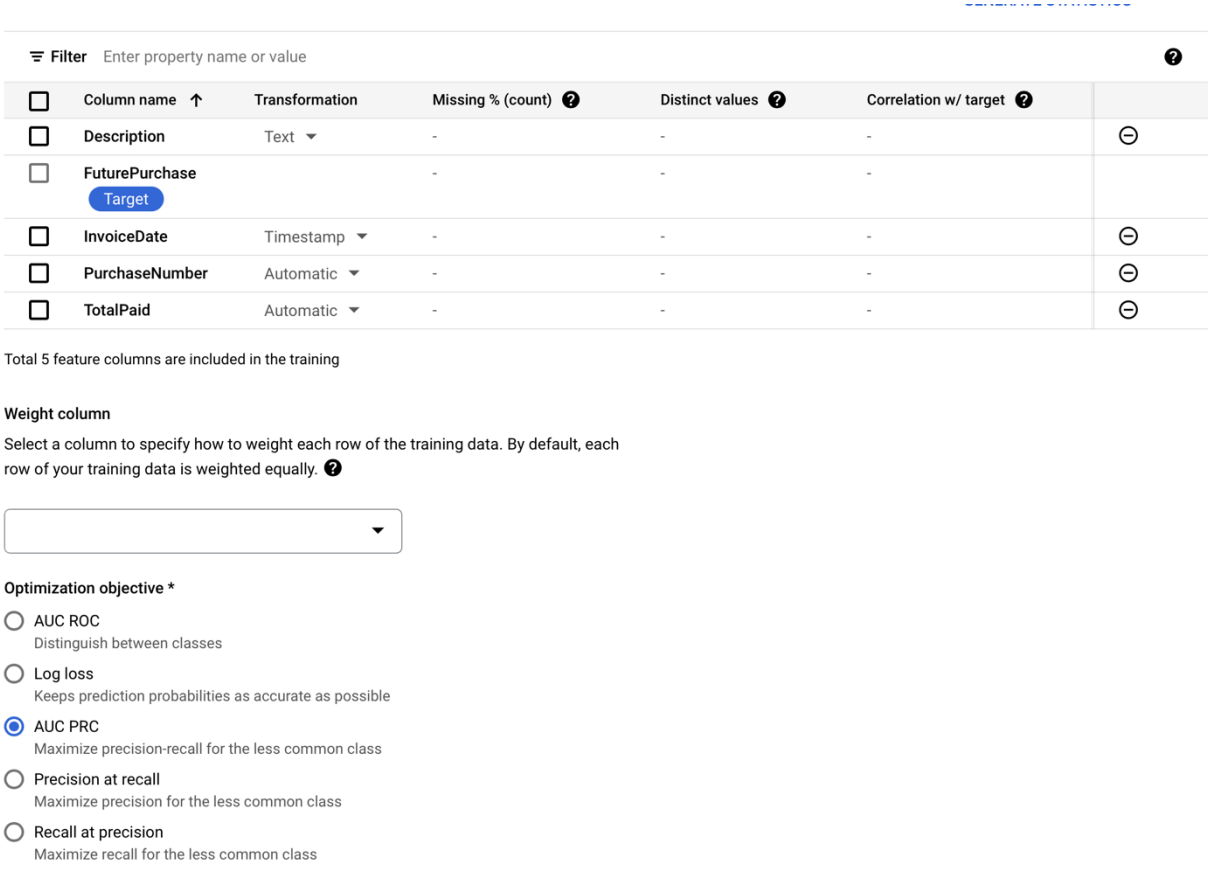


Now, what will we choose to optimize? One of the main objectives of this modeling exercise may be sending directed offers or marketing to users whose probability of making a future purchase is low.

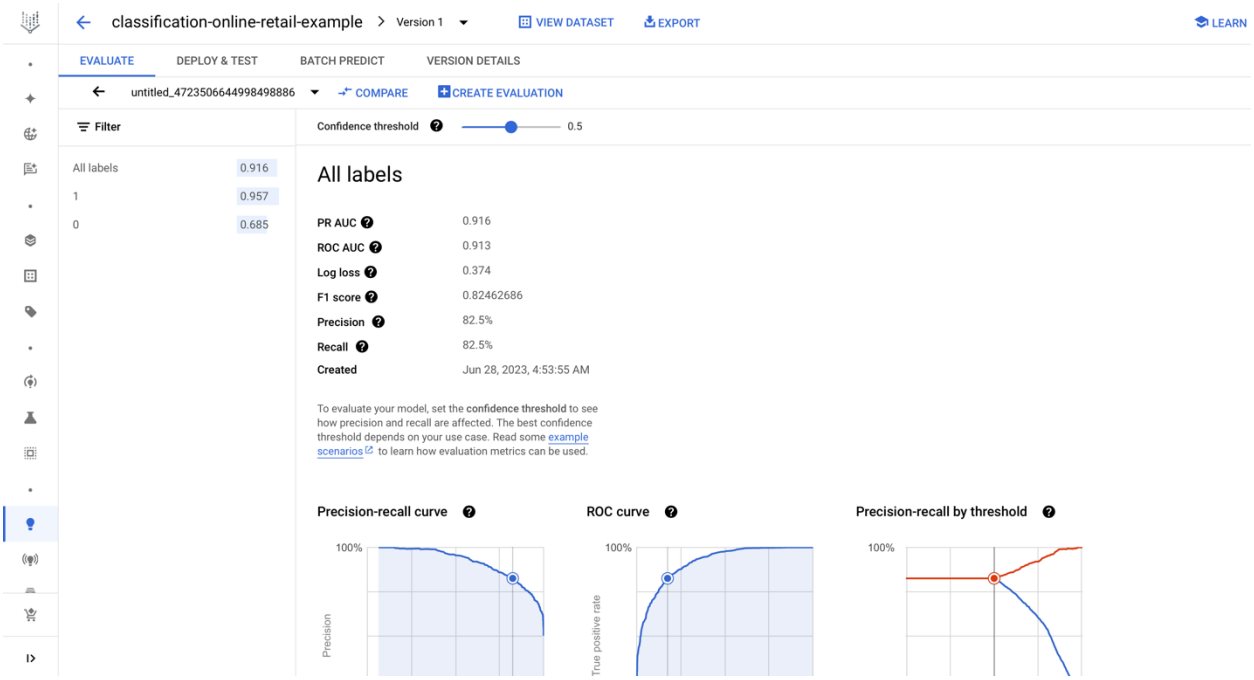
If it doesn't matter if we bother a few customers that were already going to buy with an email, but we certainly don't want to miss any customers that will not buy to target them with a discount or recommendation, we should go for maximizing **recall**.

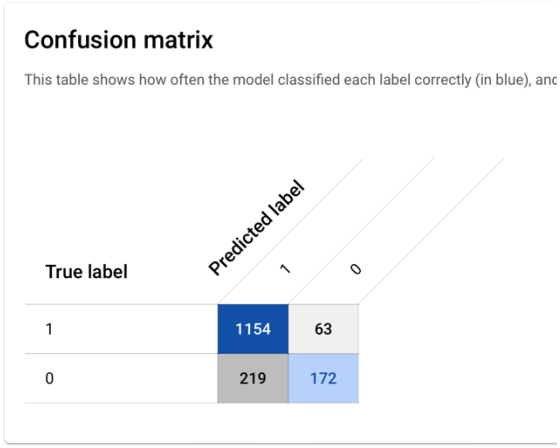
If we don't want to give discounts to customers that were already going to buy (lost revenue), and we don't care to not target some customers that were not going to buy with additional marketing. We should go ahead and maximize **precision**.

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.



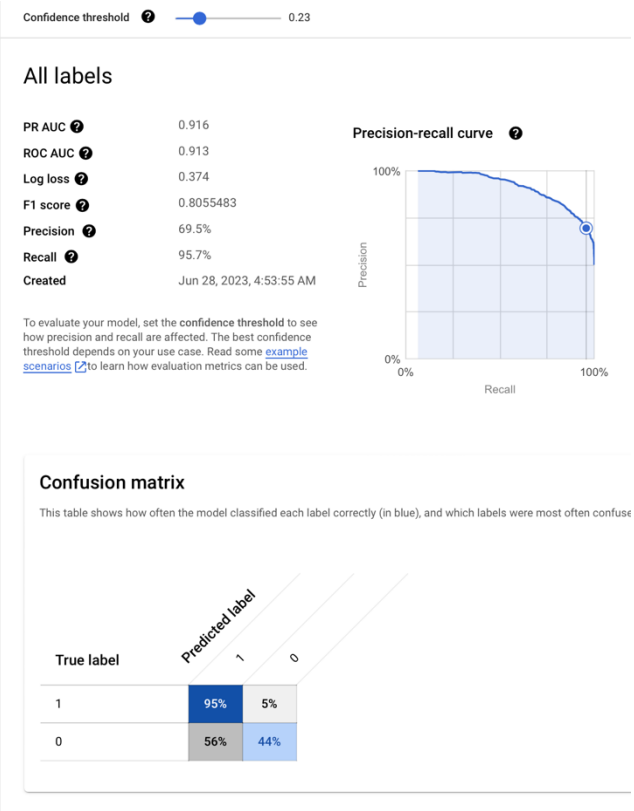
After the training job finishes training we can analyze the results of our ML POC as AutoML provides the necessary evaluation metrics.





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

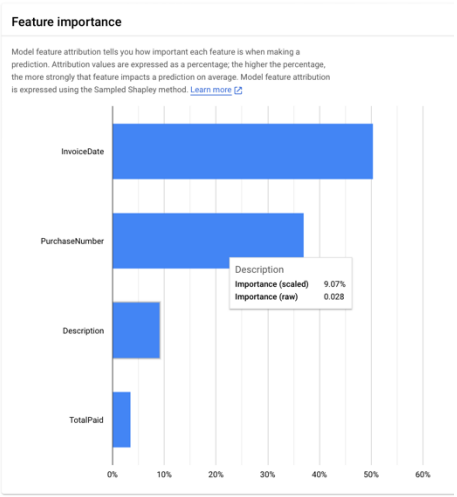
as “will buy in the future” examples that we have categorized as “will not buy”. A thing that could be happening is that we are **failing to correctly create the labels in our data**. With our current way of creating the labels we are assuming that if a customer doesn’t have a next



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.