10 Things That Can Go Wrong with ML Projects

(and what you can do about it)

Karl Weinmeister {Developer Advocacy Manager}
@kweinmeister

Machine learning practitioners are solving important problems every day. They're also experiencing a new set of challenges. Let's discuss the best practices and tools to help.

What can go wrong?

Building a Model

1. You aren't solving the right problem
2. Jumping right into development without a prototype
3. The training process is slowing your team down

Model accuracy

4. You have an imbalanced dataset
5. Model accuracy is not good enough

Transparency and Fairness

6. The model doesn't serve all of your users well
7. It's unclear how your model works

MLOps

8. You accidentally push a bad model into production
9. Your model accuracy is drifting downward
10. Model inference isn't scaling well in production

Building a Model

1. You aren't solving the right problem

Some organizations are **transforming** how they operate and serve their users with machine learning.

Others **struggle to get value** out of their machine learning projects.

What **goal** is your machine learning model trying to achieve?

How do you answer whether your model is "good" or "bad"? Do you know what your **baseline** is?

Focus on a long-term mission with maximum impact

Insights from the
[Product Management for AI video](#):

- Stay focused on your goals, but be **flexible on the tactics** to get there
- **Set milestones**, but be prepared to pivot
- Your **users** should be involved at each stage of the project

Ensure that your problem is a good fit for machine learning

Predictive Analytics

Fraud detection
Preventive maintenance
Click-through-rate
Demand forecasting
Will customer buy?

Unstructured data

Annotate videos
Identify eye disease
Triage emails

Automation

Schedule maintenance
Reject transactions
Count retail footfall
Scan medical forms
Triage customer emails
Act on user reviews

Personalization

Customer segmentation
Customer targeting
Product recommendation

2. Jumping into development without a prototype

A machine learning project is an **iterative process**.

You should start with a **simple model** and continue to **refine it** until you've reached your goal.

A quick prototype can **tell you a lot** about hidden requirements, implementation challenges, scope, etc.

Create models directly from BigQuery with BQML

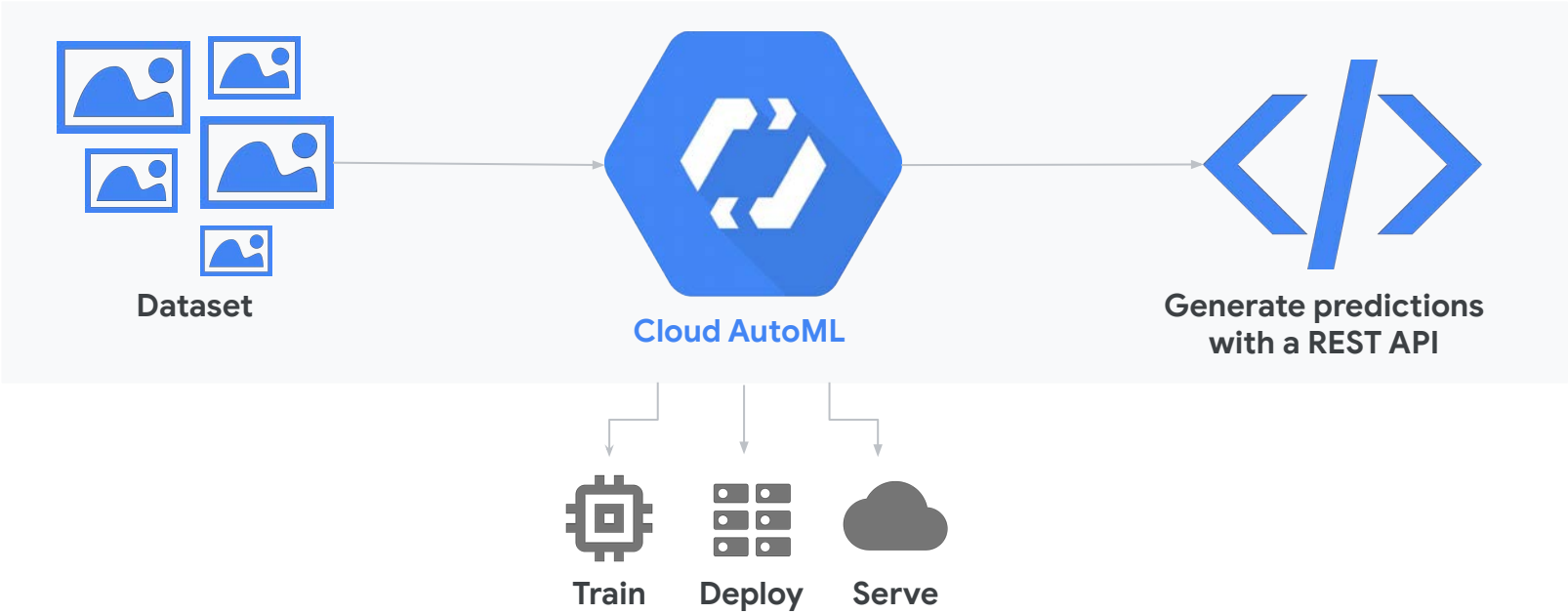
Execute ML initiatives without moving data from BigQuery

Iterate on models in SQL in BigQuery to increase development speed

Automate common ML tasks

Built-in infra management, security & compliance, supporting a seamless transition to production model development

Automate the model building and deployment process with Cloud AutoML



3. Model training can take a long time

Model training is the process of learning from input data to construct a model.

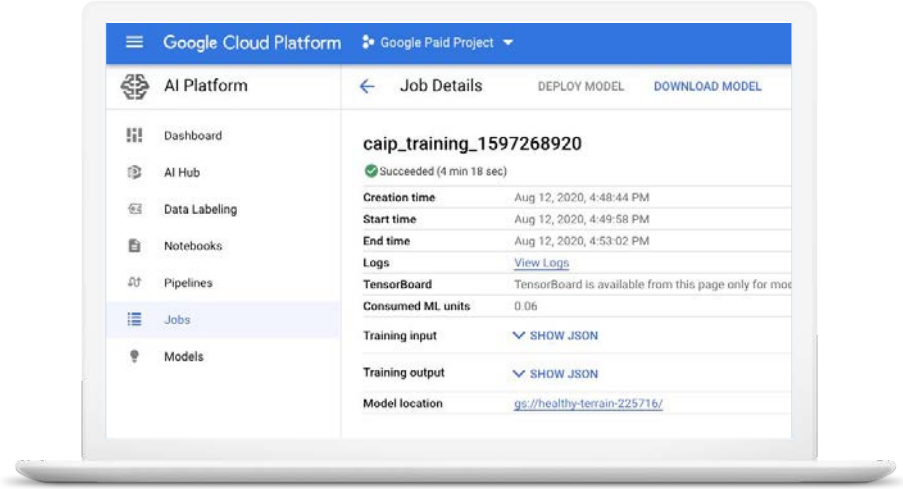
For many real-world datasets, **training can take hours, days, or even longer**.

After your model is trained, you can then evaluate its accuracy to see if you've made an improvement.

When your team is trying to rapidly iterate on new ideas and techniques, **training time can slow down projects and get in the way of innovation**.

Serverless Training with Vertex AI

- Train models without managing infrastructure
- Supports TensorFlow 2.x and other popular data science and machine learning frameworks. You can even run your own Docker container.
- Leverage distributed training on the latest GPUs and TPUs to finish jobs faster
- Improve your model quality with automated hyperparameter tuning using parallel trials



Google Cloud TPU

Cloud TPUs enable businesses and researchers to **train and run** cutting-edge & large-size ML models at faster speed and scale.

With Cloud TPUs, you can train more powerful and accurate models than ever before.

Cloud TPUs can reduce the cost of machine learning work and speed up time to market for new AI applications.

Model Accuracy

4. You have an imbalanced dataset

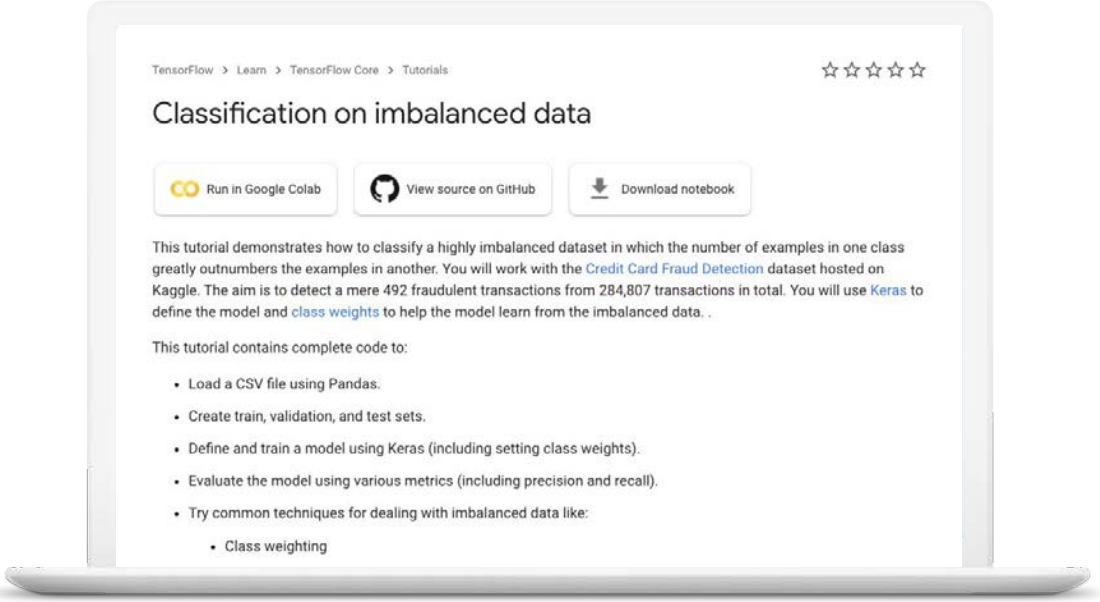
Some ML tasks, such as fraud detection, use data with many more labels of one class than another.

If 99% of data is benign, and 1% is fraudulent, it's easy to get 99% accuracy in your model. Just predict that every transaction is benign!

How can you balance accuracy across each class?

Techniques to deal with imbalanced data:

- Weighting each class
- Oversampling and undersampling
- Generating synthetic data



TensorFlow > Learn > TensorFlow Core > Tutorials

Classification on imbalanced data

Run in Google Colab | View source on GitHub | Download notebook

This tutorial demonstrates how to classify a highly imbalanced dataset in which the number of examples in one class greatly outnumbers the examples in another. You will work with the Credit Card Fraud Detection dataset hosted on Kaggle. The aim is to detect a mere 492 fraudulent transactions from 284,807 transactions in total. You will use Keras to define the model and class weights to help the model learn from the imbalanced data. .

This tutorial contains complete code to:

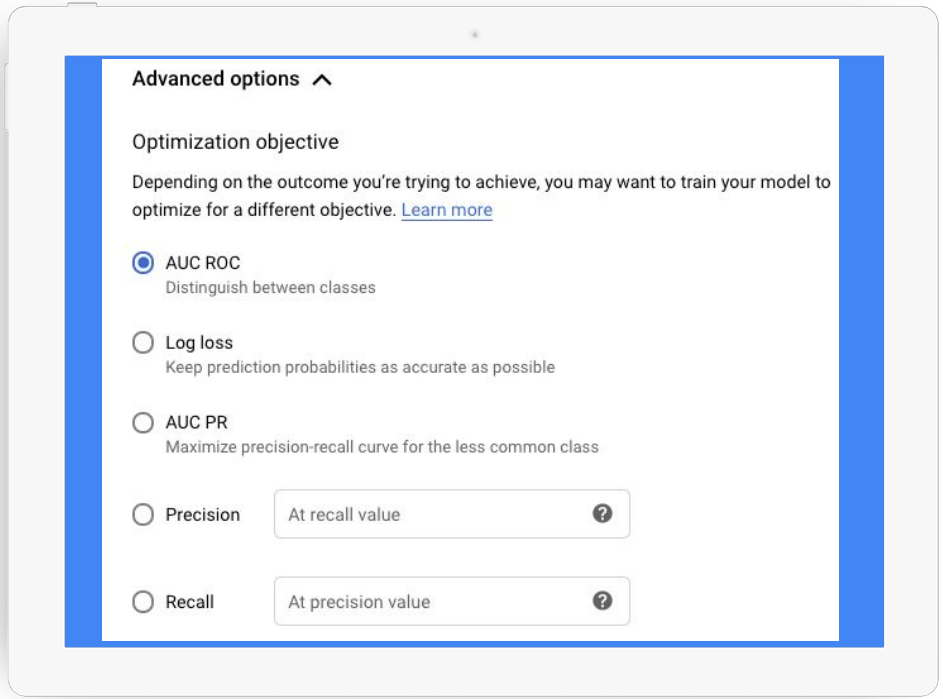
- Load a CSV file using Pandas.
- Create train, validation, and test sets.
- Define and train a model using Keras (including setting class weights).
- Evaluate the model using various metrics (including precision and recall).
- Try common techniques for dealing with imbalanced data like:
  - Class weighting

tensorflow.org/tutorials/structured_data/imbalanced_data

AutoML - Optimization Objective

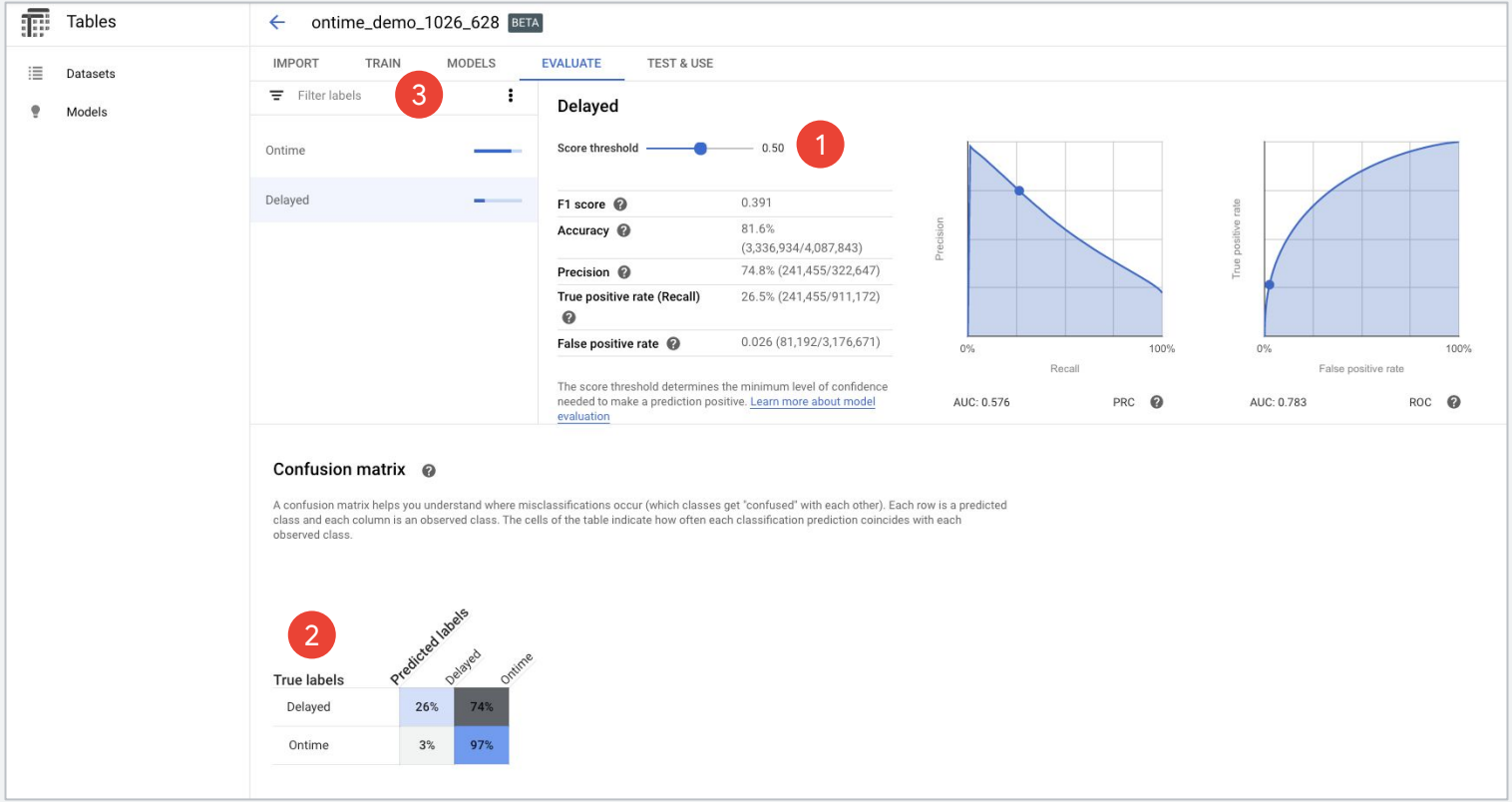
When training a model, select an appropriate optimization objective for your project goals:

- **AUC PR** (maximize precision-recall curve for less common class)
- **AUC ROC** (distinguish between classes)
- **Log loss** (maximize accuracy across entire dataset)



AutoML - Model Evaluation

1. Review accuracy metrics at different thresholds
2. Review confusion matrix to understand where misclassifications occur
3. Consider updating Optimization Objective and re-training if desired



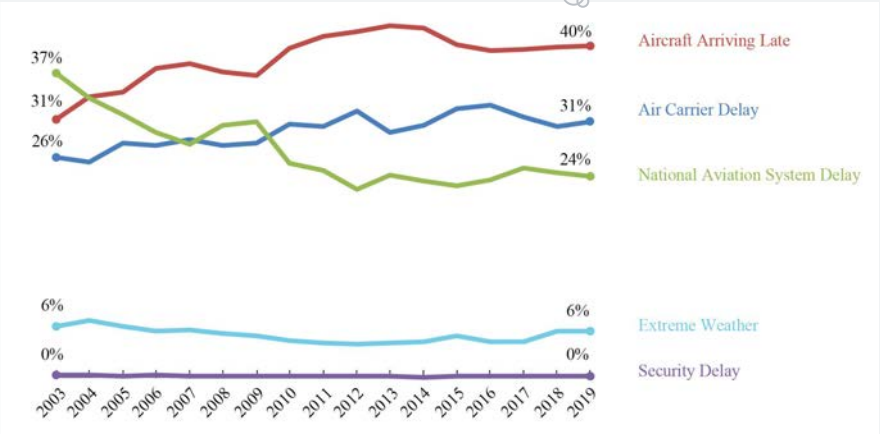
5. Model accuracy is not good enough

Sometimes, you might feel stuck.

You can't improve the model accuracy any more.

Is there anything that can be done?

Does your model include features for each root cause?



[Understanding the Reporting of Causes of Flight Delays and Cancellations]

Solutions that may help

- Improve domain expertise for the problem you're trying to solve
- Include more, varied training data
- Feature engineering
- Consider removing features that may be causing overfitting; or start with a smaller model and incrementally add features
- Try different model architectures, hyper-parameter tuning, and ensembles
- Try AutoML to see what's possible with your input data

Transparency and Fairness

6. Your model doesn't serve all of your users well

"Fairness is the process of understanding bias introduced by your data, and ensuring your model provides equitable predictions across all demographic groups. Rather than thinking of fairness as a separate initiative, it's important to apply fairness analysis throughout your entire ML process."

- Sara Robinson, [Building ML models for everyone: understanding fairness in machine learning]

Responsible AI: Questions for each phase of the project

Define problem

What problem(s) will the model solve?

Who's the intended user?

What are the risks associated with the use-case?

What will 'success' look like?

Collect & prepare data

How was the training data collected, sampled, labeled?

Is it representative of the real-world?

Is the training data skewed?

Is the data privacy-protecting?

Train model

How was the model trained?

Who trained it? When?

How was the model debugged/improved?

What are the models' limitations?

Evaluate

How was the model tested?

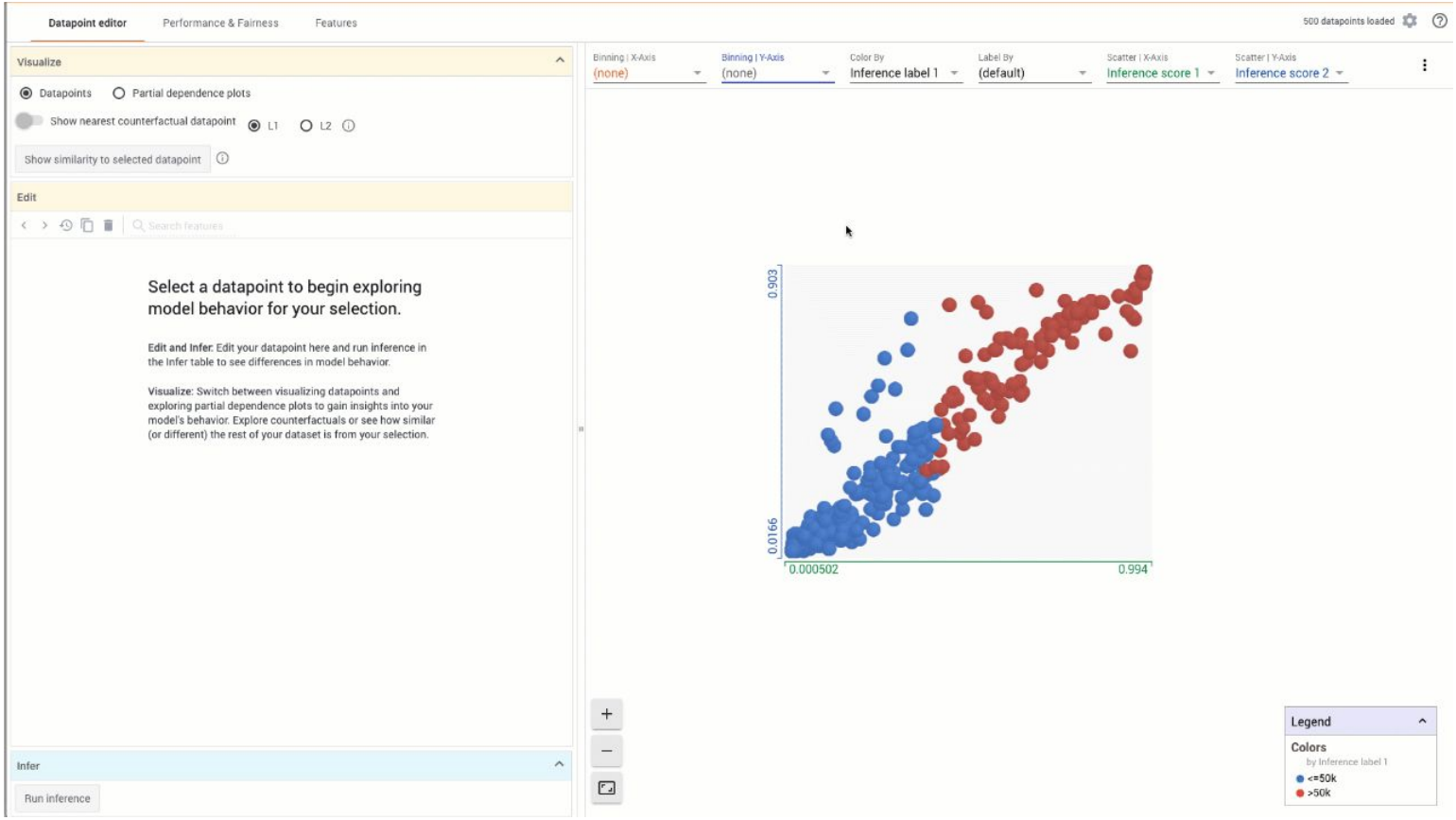
What test datasets were used?

Deploy & monitor

Is the model behaving as expected?

Why did the model fail in this case?

What-If Tool: Visually probe the behavior of trained ML models



Google Cloud

TensorFlow Model Analysis: Sliced ML model metrics

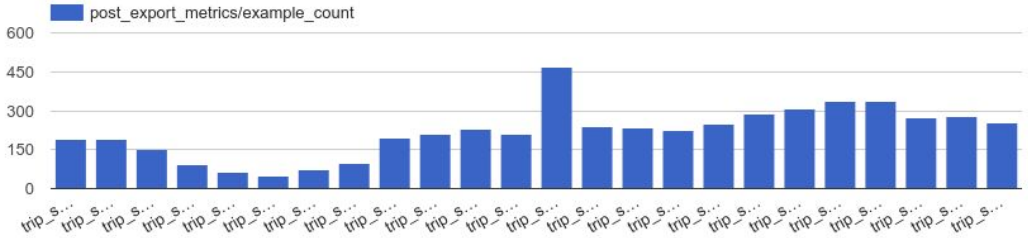
```
In [13]: # Show data sliced along feature column trip_start_hour.
         tfma.view.render_slicing_metrics(
             tfma_result_1, slicing_column='trip_start_hour')
```

Visualization: Slices Overview

Examples (Weighted) Threshold: 0

Show: post_export_metrics/example_count

Sort by: Slice



feature	accuracy	accuracy_baseline	auc	auc_precision_recall	average_los
trip_start_hour:8	0.97938	0.97938	0.66513	0.99010	0.1111
trip_start_hour:9	0.98113	0.98113	0.69231	0.99140	0.0892
trip_start_hour:10	0.95197	0.95197	0.77377	0.98236	0.1541
trip_start_hour:1	0.94180	0.94180	0.78422	0.98231	0.1901

7. It's unclear how your model works

ML models are often distributed without a clear understanding of they function.

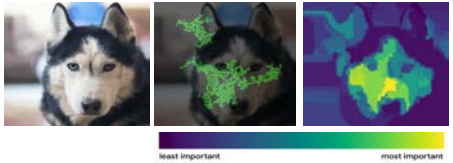
How will you be able to explain how your model makes predictions?

Under what conditions does the model perform best and most consistently?

Does it have blind spots? If so, where?

Explainable AI on Google Cloud: Support for multiple data types

Images



Text

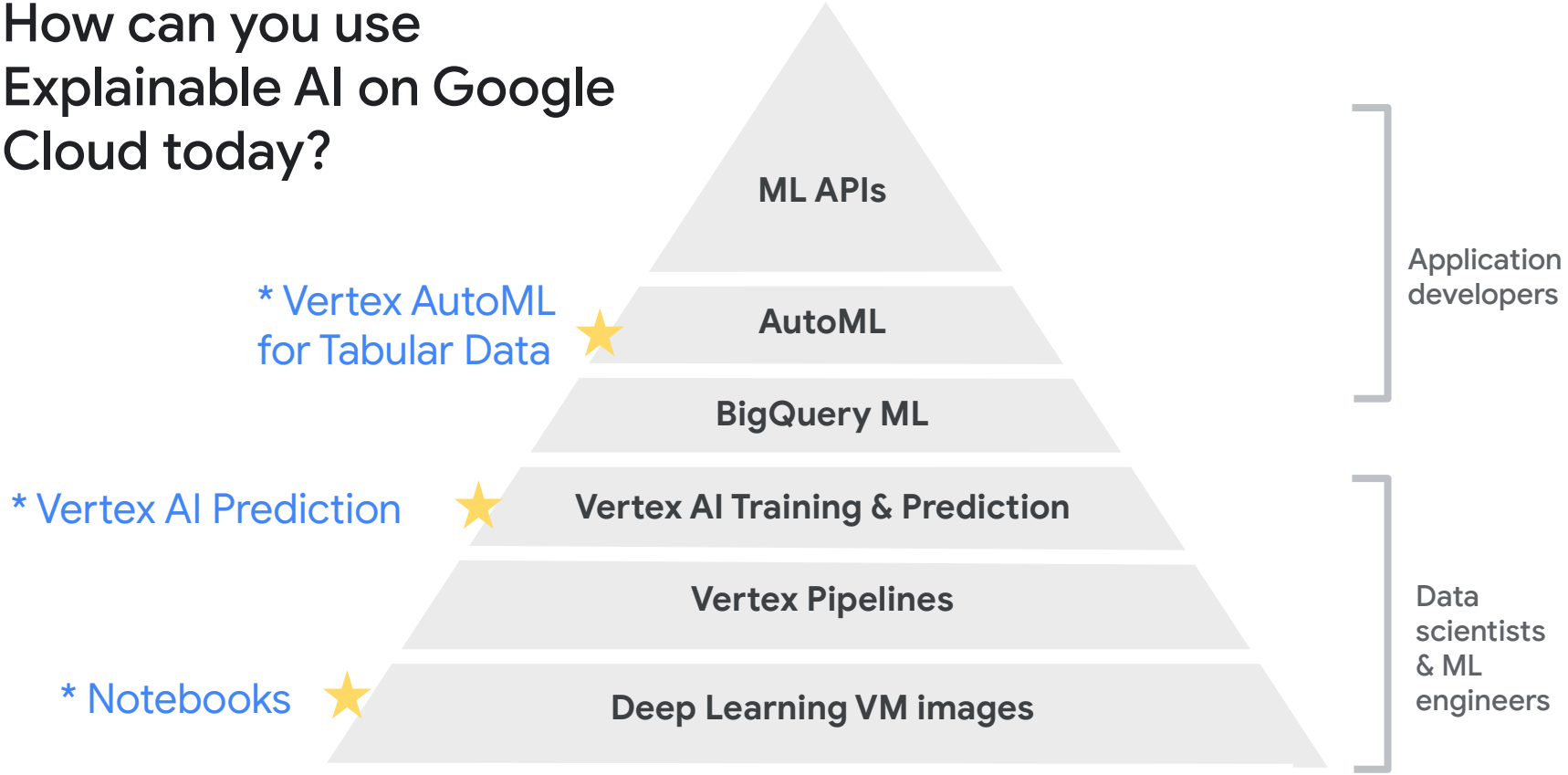
How could you not love cake?!

Sentiment score: 0.9

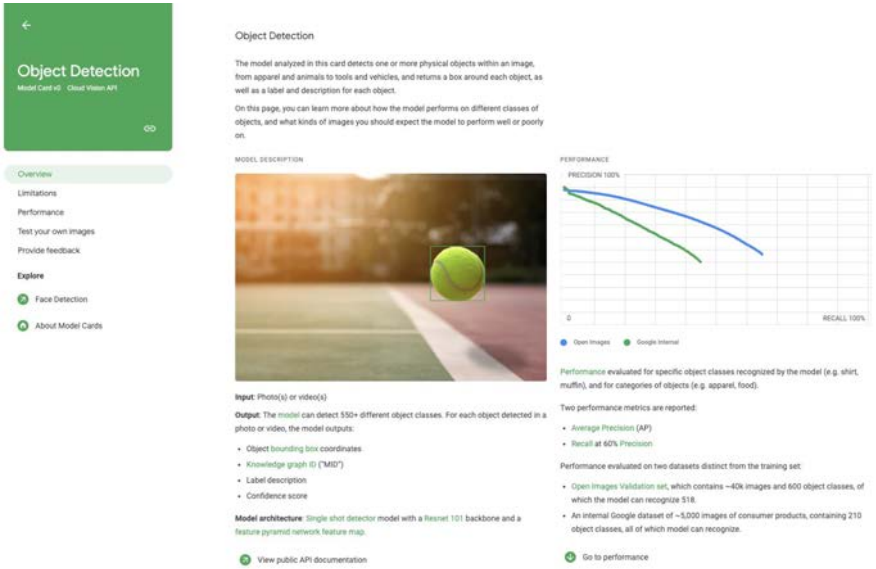
Tabular

Name	Feature value	Attribution value
distance	1395.51	-2.44478
start_hr	18	-1.29039
max_temp	20.7239	0.690506
temp	16.168	0.12629
dew_point	7.83396	0.0110318
prcp	0.03	-0.00134132

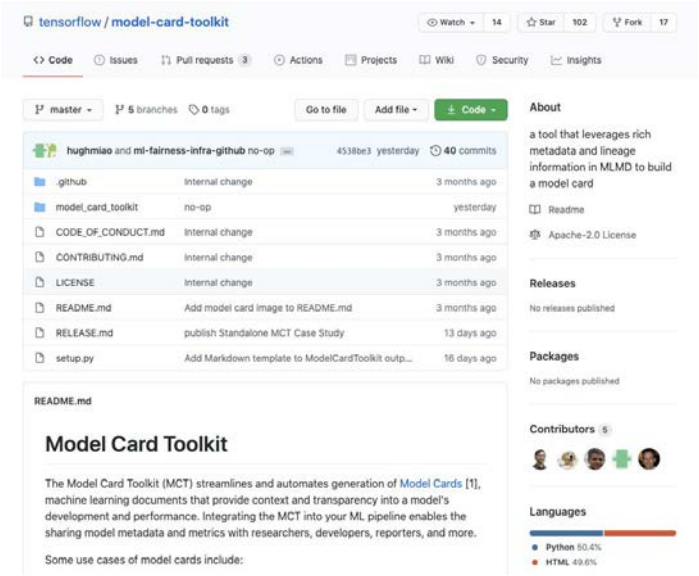
How can you use Explainable AI on Google Cloud today?



Document your Model with **Model Cards**



modelcards.withgoogle.com



Model Card Toolkit

MLOps

8. You could accidentally push a bad model into production

Your data science team is constantly running experiments to find a better model.

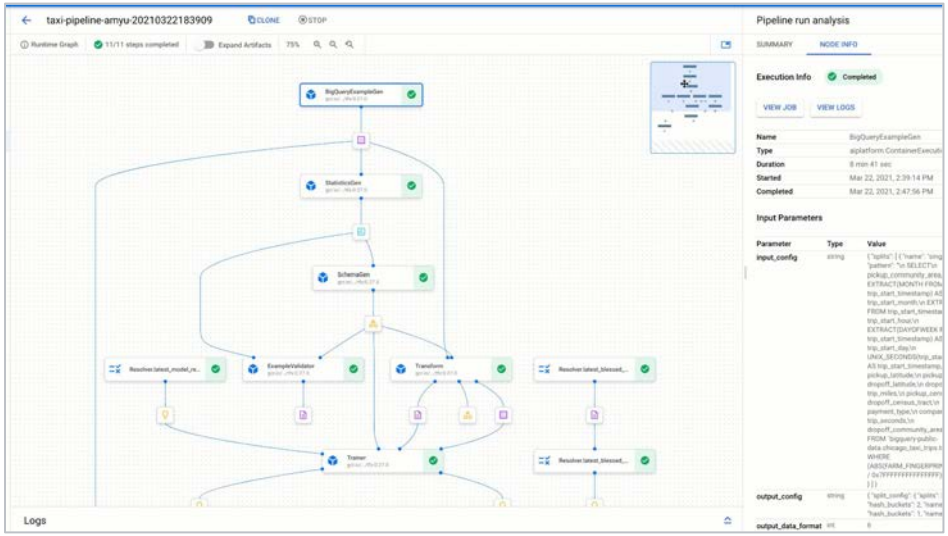
What if you accidentally deploy a model that wasn't tested properly, and your users are impacted?

Create a reproducible workflow with **Vertex Pipelines**

- An ML pipeline allows you to run the same steps in the same environment every time with a trigger or schedule

- Include steps for:
  a. Data validation
  b. Model evaluation
  c. Conditional deployment

- Track pipeline runs and generated artifacts



9. Your model accuracy is drifting downward

Over time, the performance of most models will decay.

Conditions in the outside world change. This means that the data distributions for the features the model were trained on will also change.

How do you detect this data drift and manage it?

Detect and manage model drift with MLOps processes

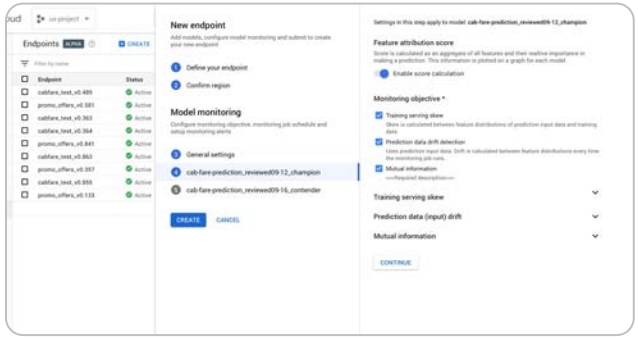
Continuous Evaluation

- Sample your model's predictions
- Compare the model's predictions to "ground truth"
- Assess the accuracy
- Send notification if threshold reached

Continuous Training

- Deploy an ML pipeline that automatically:
  - Extracts latest data from sources
  - Trains a model
  - Tests the model
  - Deploys to production

Vertex Model Monitoring



Automatically alert your data scientists and ML engineers when model performance changes

Detect **drift** and **training-serving skew**

Provides confidence in model **reliability**

10. Model inference isn't scaling well in production

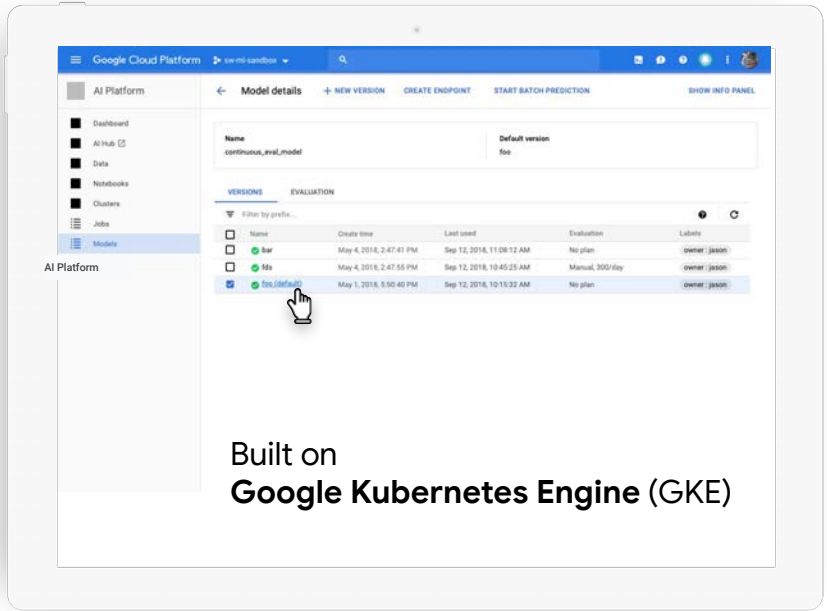
Your ML model is a success.

It solves the problem well, and will be integrated into a widely used application.

How will you host the model and serve all of the requests?

Robust and reliable model hosting with Vertex Prediction

- Serve **online** endpoints for low-latency predictions, or predictions on massive **batches** of data
- Scale **automatically** based on your traffic
- Log prediction requests and responses to **BigQuery** for monitoring and debugging
- Choose from a variety of compute options, including the inference-optimized **NVIDIA T4 GPU**, for faster predictions

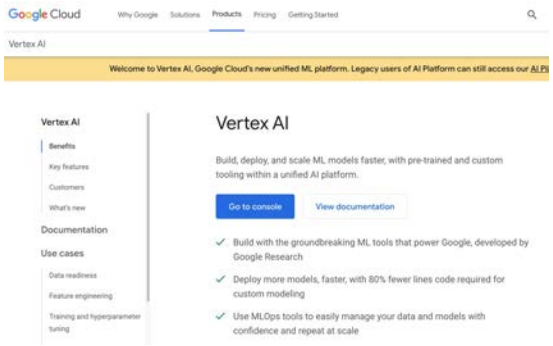


AI Platform

Built on
Google Kubernetes Engine (GKE)

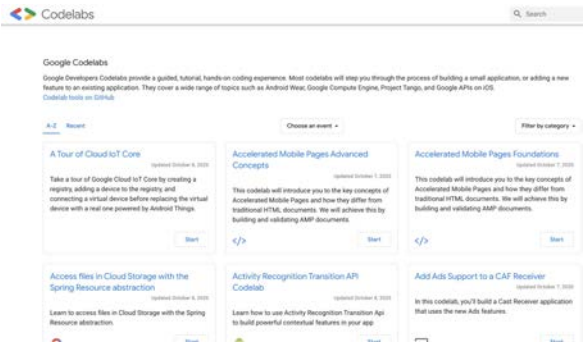
Resources

Resources



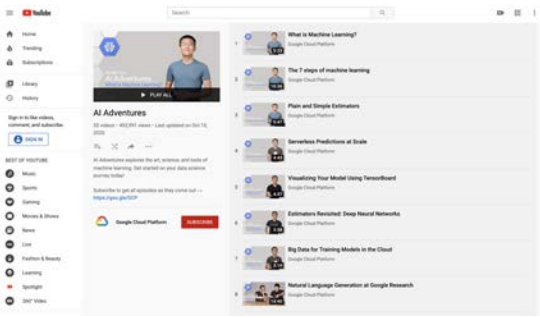
Vertex AI

Try out the end-to-end platform for data science and machine learning on Google Cloud



Codelabs

Hands-on labs that enable you to learn machine learning concepts and tools on Google Cloud



AI Adventures

50+ Youtube videos covering machine learning concepts and how to use Cloud AI Platform

Thank you!

Have questions or suggestions? We want to hear from you! Please contact @kweinmeister!