Google Cloud

Quality of Machine Learning Models

Speakers

Giovanni Marchetti
AI/ML Customer Engineer

Mona Mona
AI/ML Customer Engineer

Google

Proprietary + Confidential

Agenda

1. What does a model do?
2. How do you differentiate between a good and a bad model?
3. Common measures for common problems
4. Testing for model quality
5. Automation
6. How Vertex AI can help

- Vertex AI pipelines
- Vertex AI Model Monitoring
- Vertex AI Explainable AI
- Model cards

A Model

$$f : X \to Y$$

A model maps a set of inputs to a set of outputs.

In ML, the function is learned from the data, not given.

A test set is a set (X,Y) of known desired outcomes for given inputs.

Training and Evaluation

Good vs. Bad Models

1. Models are parts of systems
2. A good model will support the system goals better than a bad one
3. A good model will make fewer mistakes than a bad one

"Essentially, all models are wrong, but some are useful."

- George Box

Classification Measures

Precision

P=TP/(TP+FP)

Recall

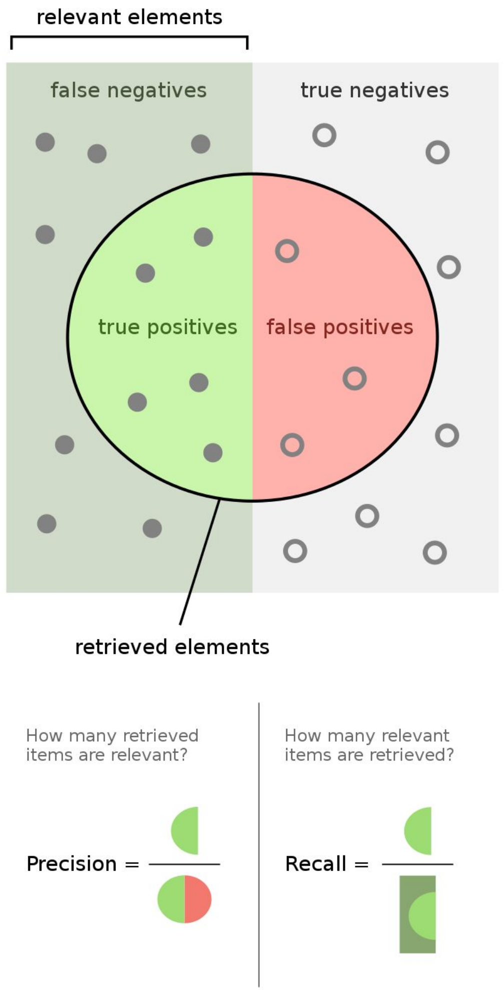
R=TP/(TP+FN)

Accuracy

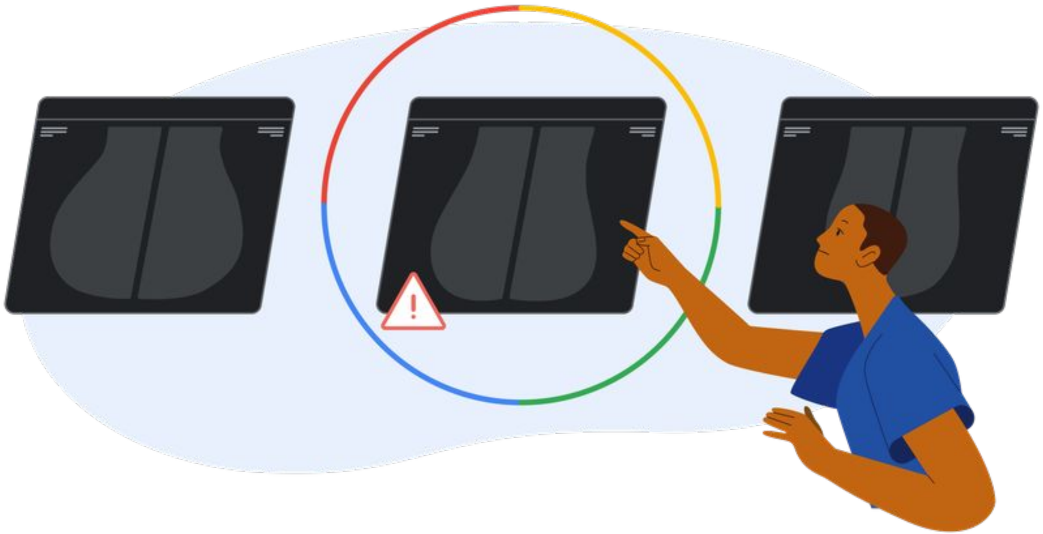
A=(TP+TN)/(TP+FP+TN+FN)

F1 Score

F1=2(R*P)/(R+P)



Example: AI for Breast Cancer Screening



Regression Measures

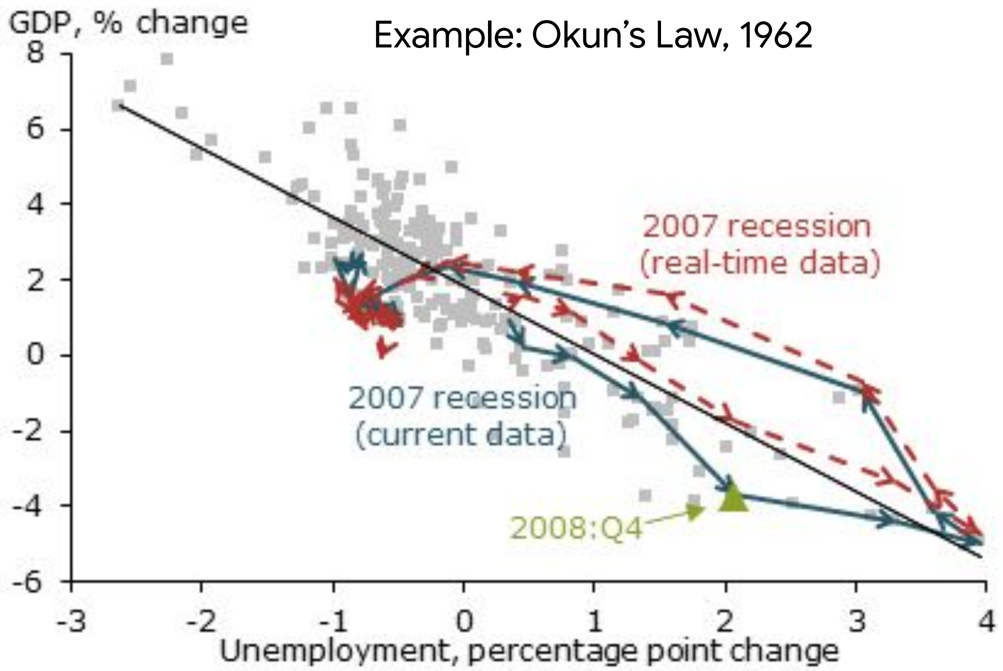
$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right|$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

Example: Okun's Law, 1962

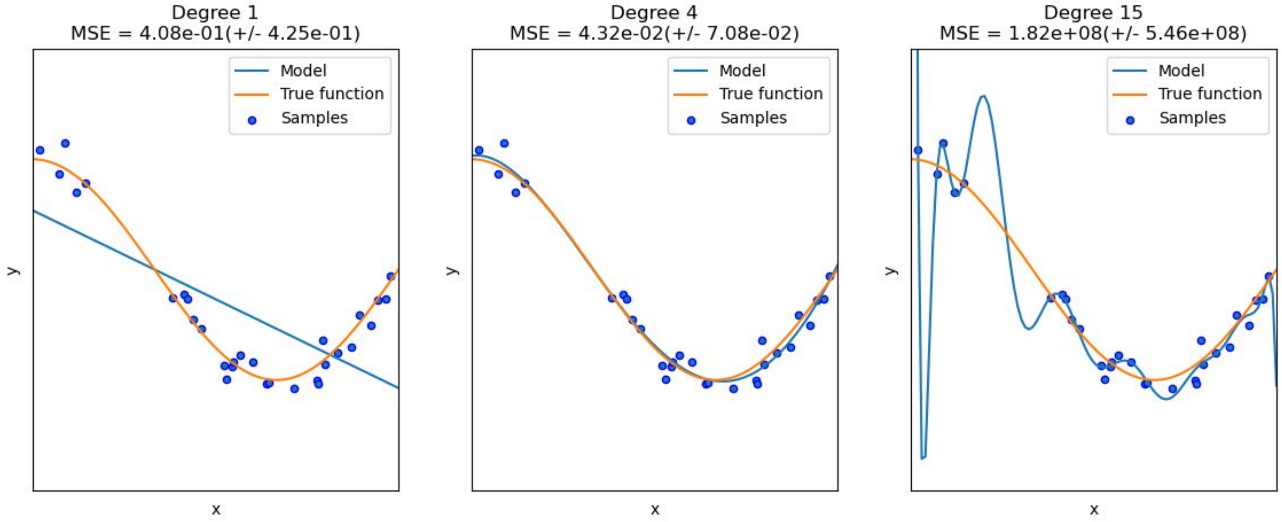


[FRBSF Economic Letters, April 21st 2014]

Generalization

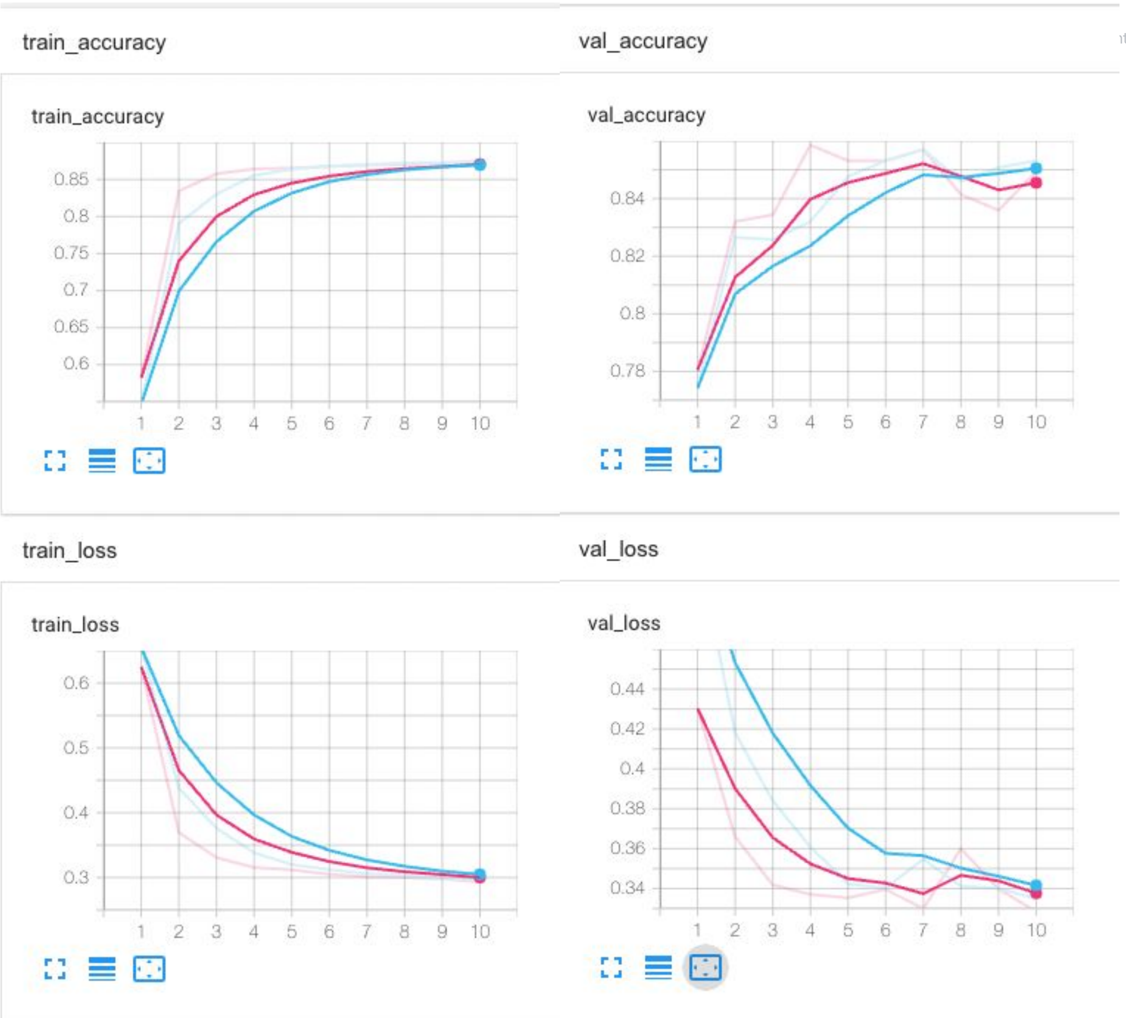
Overfitting: model learns from the training set, but performs poorly on the test set

Underfitting: model cannot learn from training set, performs poorly on test set too.



Monitor runs

For overfitting, underfitting, convergence



Compare experiments

gm-demos-351301-classification-xgboost-20220830233815 DELETE REFRESH

Compare metrics, parameters and artifacts to identify the best run. Learn more

Runs

Filter Enter property name or value

Name	Status	Type	Created	Parameter: boost_rounds	Parameter: label_uri	Parameter: learning_rate	Parameter: max_depth	Parameter: model_uri	Parameter: train_uri	Metric: accurancy
custom-training-pipeline-20220831012034	✔	Pipeline run	August 30, 2022	30	gs://gm-experiment-demos3/iris/iris_target.csv	0.4	5	gs://gm-experiment-demos3/model	gs://gm-experiment-demos3/iris/iris_data.csv	0.9
custom-training-pipeline-20220831011609	✔	Pipeline run	August 30, 2022	40	gs://gm-experiment-demos3/iris/iris_target.csv	0.5	6	gs://gm-experiment-demos3/model	gs://gm-experiment-demos3/iris/iris_data.csv	0.9
custom-training-pipeline-20220831011606	✔	Pipeline run	August 30, 2022	30	gs://gm-experiment-demos3/iris/iris_target.csv	0.1	3	gs://gm-experiment-demos3/model	gs://gm-experiment-demos3/iris/iris_data.csv	0.9
custom-training-pipeline-20220830235455	✔	Pipeline run	August 30, 2022	20	gs://gm-experiment-demos3/iris/iris_target.csv	0.3	5	gs://gm-experiment-demos3/model	gs://gm-experiment-demos3/iris/iris_data.csv	0.8666666667
custom-training-pipeline-20220830234206	✔	Pipeline run	August 30, 2022	10	gs://gm-experiment-demos3/iris/iris_target.csv	0.2	4	gs://gm-experiment-demos3/model	gs://gm-experiment-demos3/iris/iris_data.csv	0.9

Testing

Proprietary + Confidential

Testing

Programs have known specifications:

- Given test cases, get expected results
- Consistent, but incomplete

Models have no formal specifications: data, code, hyperparameters and training make a model. Alas, we have to test them all.

- We expect % of results to be wrong
- Inputs may be noisy or mislabeled (inconsistent)
- Inputs may not cause all possible outcomes (incomplete)
- Models may be non-linear (e.g. neural networks) and possibly chaotic

Coverage

1. Representative data for all “important” cases and population groups
2. Independent train, test, validation data
    a. Prevent “leakage”, e.g. w. time series
3. Algorithm coverage
    a. All paths of a decision tree
    b. All neurons in a neural network

Keep in mind that datasets are incomplete and may be inconsistent

Evaluate the test results

1. Regression: Model should perform well consistently on important inputs (e.g. recognize family in pictures)

2. Fairness: Model should have comparable performance across subpopulations (e.g. “age blind”)

3. Reality check: It is acceptable for the model to have lower performance on outliers (e.g. dark or fuzzy pictures)

Compare Evaluations

Which model is a better fit for the intended purpose?

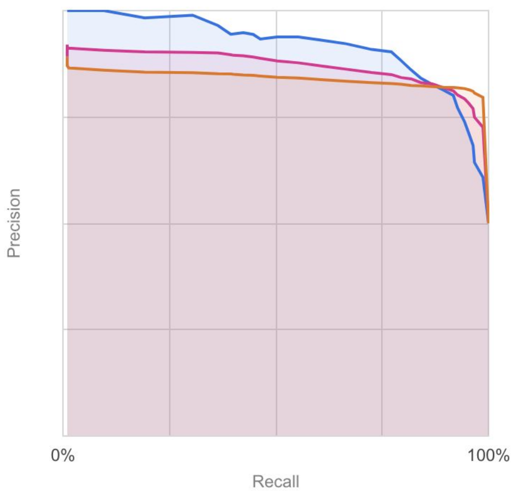
Hint: we are looking at X-Rays

Compare evaluations PREVIEW

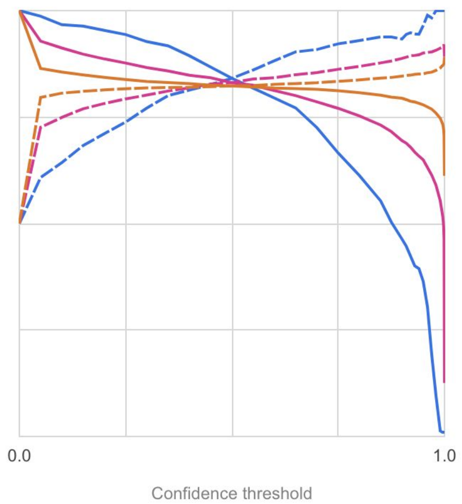
Confidence threshold 0.35

	Evaluation	PR AUC	ROC AUC	Log loss	F1 score	Precision	Recall
Version 1	untitled_5609292572078899200	0.92	—	0.365	0	80.06%	91.72%
Version 1	untitled_2349771573974138888	0.87	—	1.058	0	81.12%	85.83%
Version 1	untitled_6514146661273436160	0.853	—	3.012	0	81.96%	83.12%

Precision-recall curve



Precision-recall by threshold

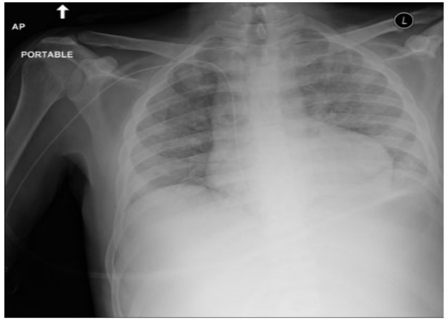


Distribution of Mistakes

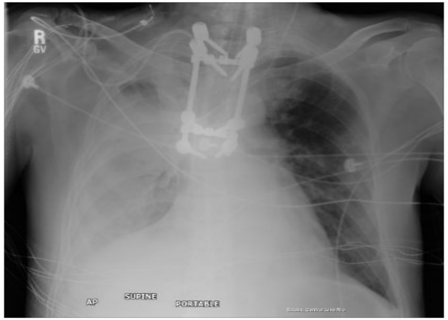
- Not all mistakes are random.
    - Monitor for bias in input and results
    - Evaluate separately for population groups
- Not all mistakes have the same impact
    - Monitor for input skew, drift
    - Monitor for amplitude and frequency
- Not all models are fair - in fact most are not
    - Explain the results, look for feature impact

False negatives

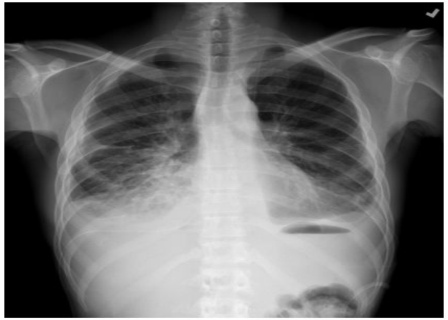
Your model should have predicted 1 for these images:



Score: 0.492



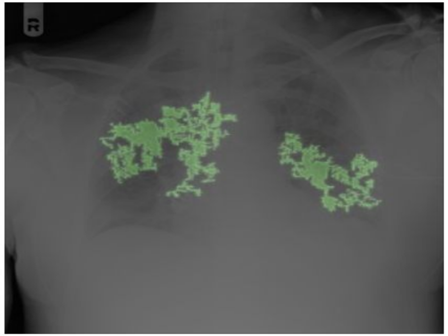
Score: 0.475


Score: 0.463

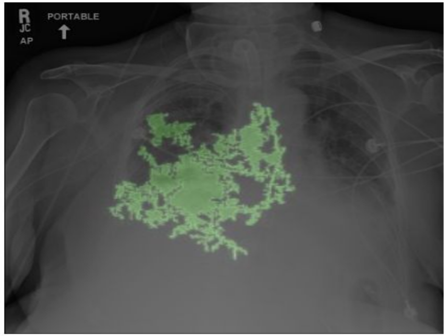

Score: 0.445

True positives

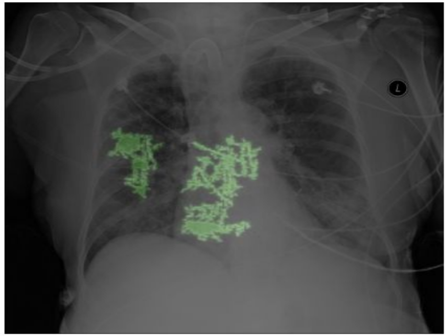
Your model correctly predicted 1 on these images:



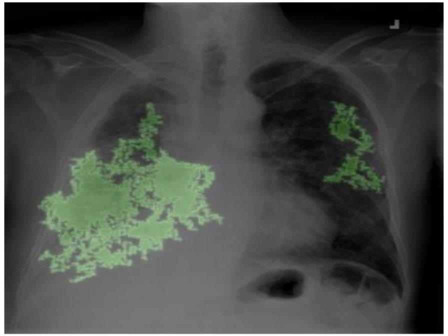
Score: 0.501



Score: 0.53

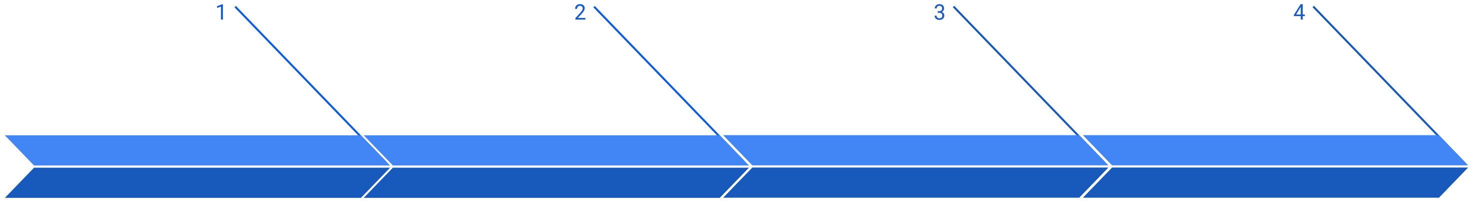


Score: 0.533


Score: 0.534

Process & Tools

A Process



Curate

Multiple train, validation and test datasets for subpopulations or critical regressions

Version control for data

Automate

Pipelines for automation repeatability

Version control for hyperparameters & metadata

Cross-validation for robustness

Evaluate and Compare

Store evaluation results across runs / experiments

Compare for fitness

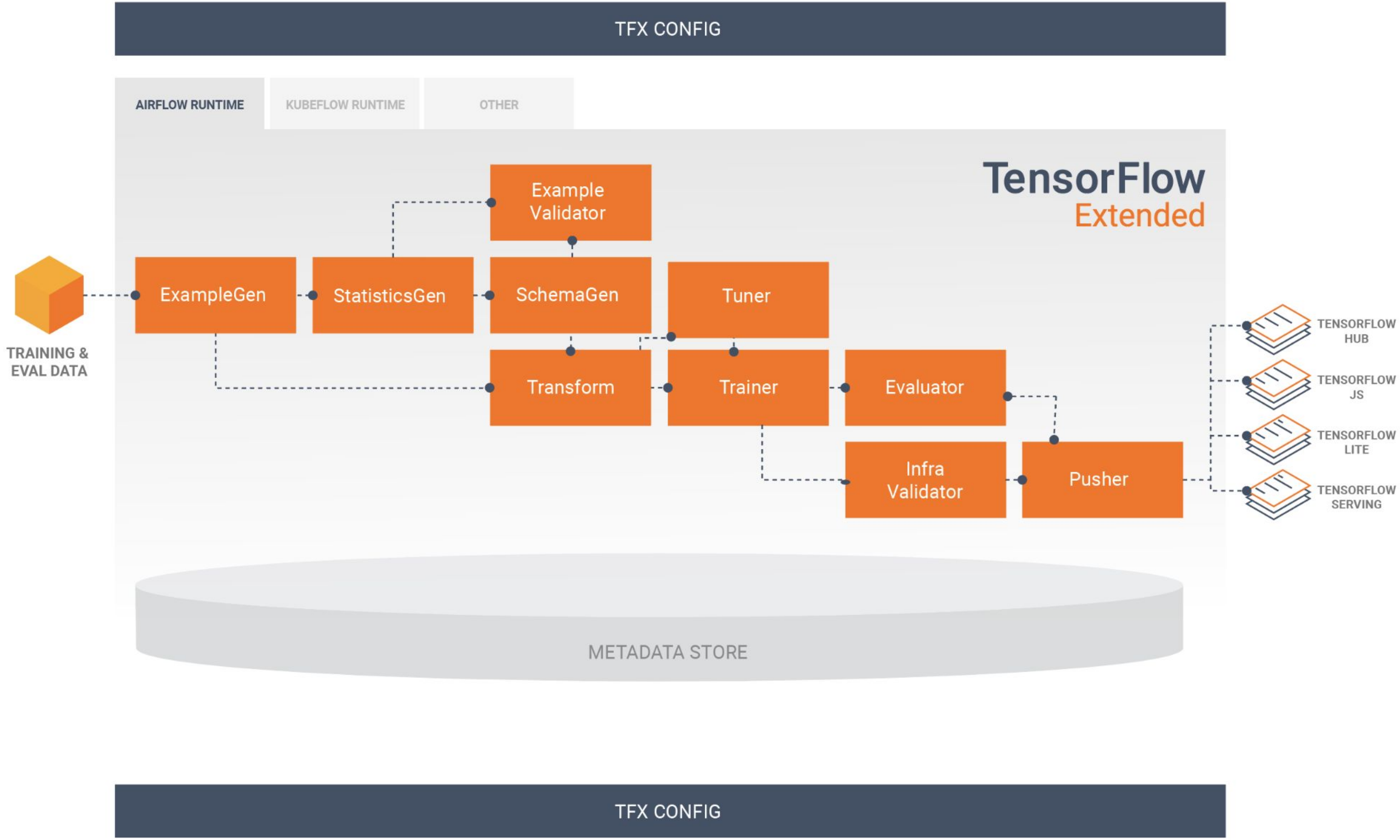
Continuous evaluation in production

Explain

Attribute feature importance

Evaluate in testing

Monitor for drift and skew in production



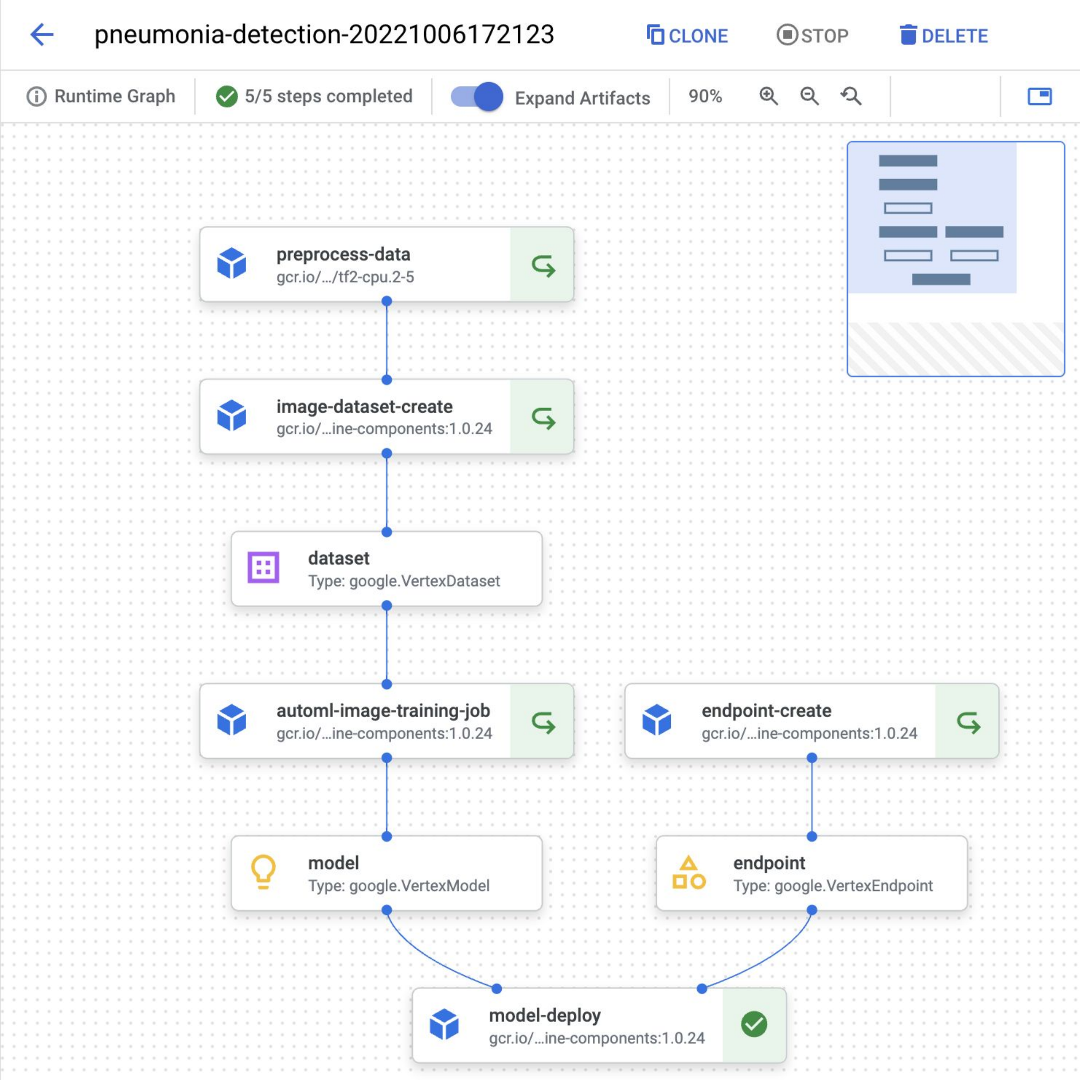
TFX CONFIG
AIRFLOW RUNTIME
KUBEFLOW RUNTIME
OTHER
TensorFlow
Extended
TRAINING &
EVAL DATA
ExampleGen
StatisticsGen
Example Validator
SchemaGen
Tuner
Transform
Trainer
Evaluator
Infra Validator
Pusher
TENSORFLOW HUB
TENSORFLOW JS
TENSORFLOW LITE
TENSORFLOW SERVING
METADATA STORE
TFX CONFIG

Pipelines

Repeatable, parametrized processes

Run on serverless infrastructure

Store all metadata for traceability and comparison



Why is ML testing hard?

You need to test the data on which the model is trained.

You need to test the model itself

You need to test the model code

Test the deployment

Test the model in production

How to test model in deployment

Test Model Updates with Reproducible Training

Testing Model Updates to Specs and API calls

Write Integration tests for Pipeline Components

Validate Model Quality before Serving

Validate Model-Infra Compatibility before Serving

How to test model in production

Check for Training-Serving Skew

Monitor Model Age Throughout Pipeline

Test Model Weights

Monitor Model Performance

Test Quality of Live Model on Served Data

How to know Quality of the model?

Fairness

Bias

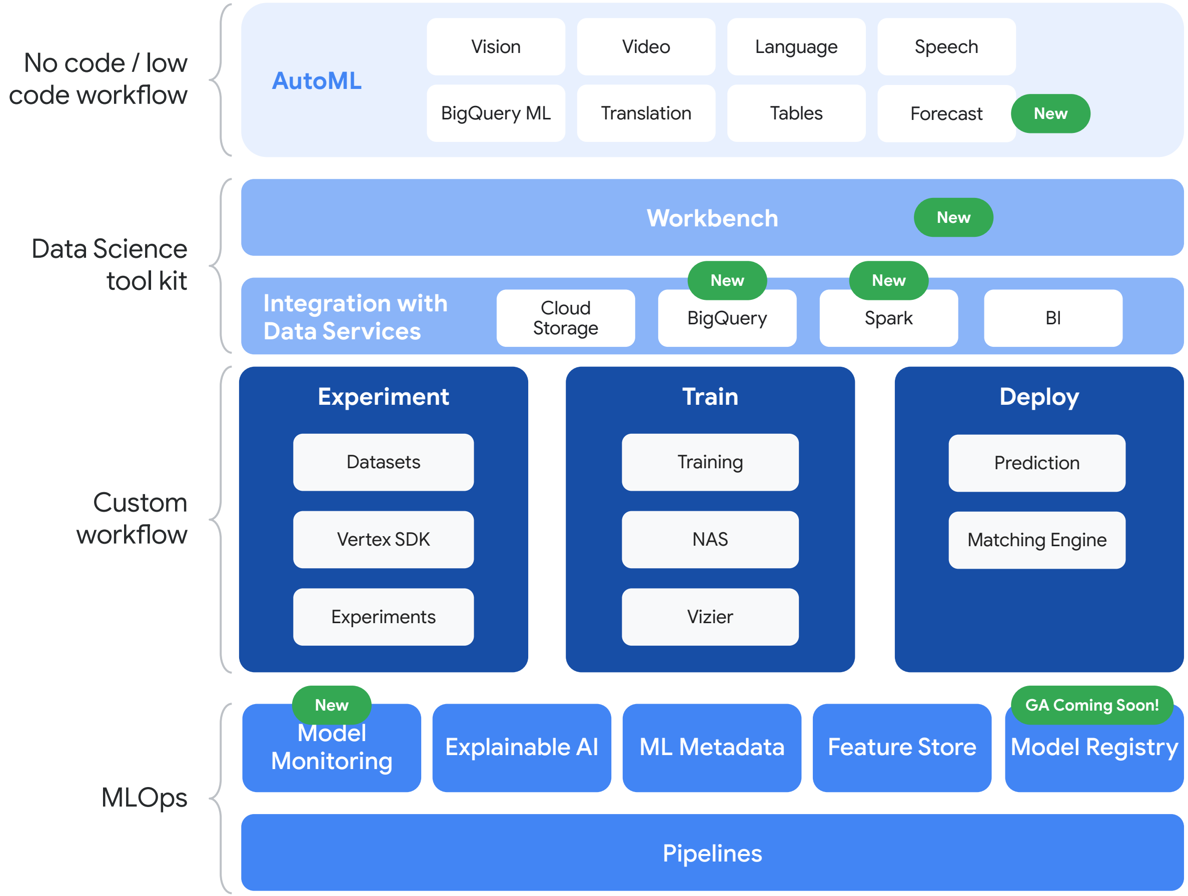
Explainability

How Vertex AI can help with Model Quality

Proprietary + Confidential

Vertex AI

- Unified development and deployment platform for data science and machine learning
- Increase productivity of data scientists and ML engineers



Managing model quality in deployment and production

Vertex AI Pipelines

Vertex AI Experiments

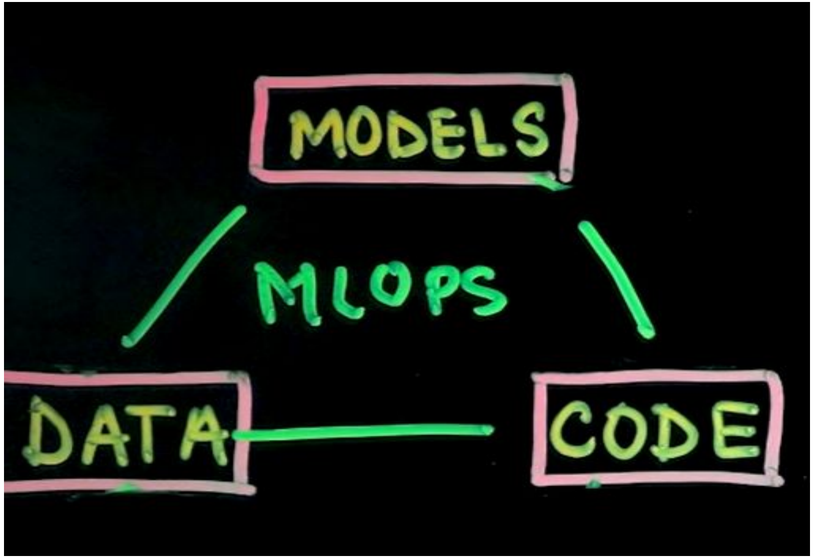
Vertex AI Model monitoring

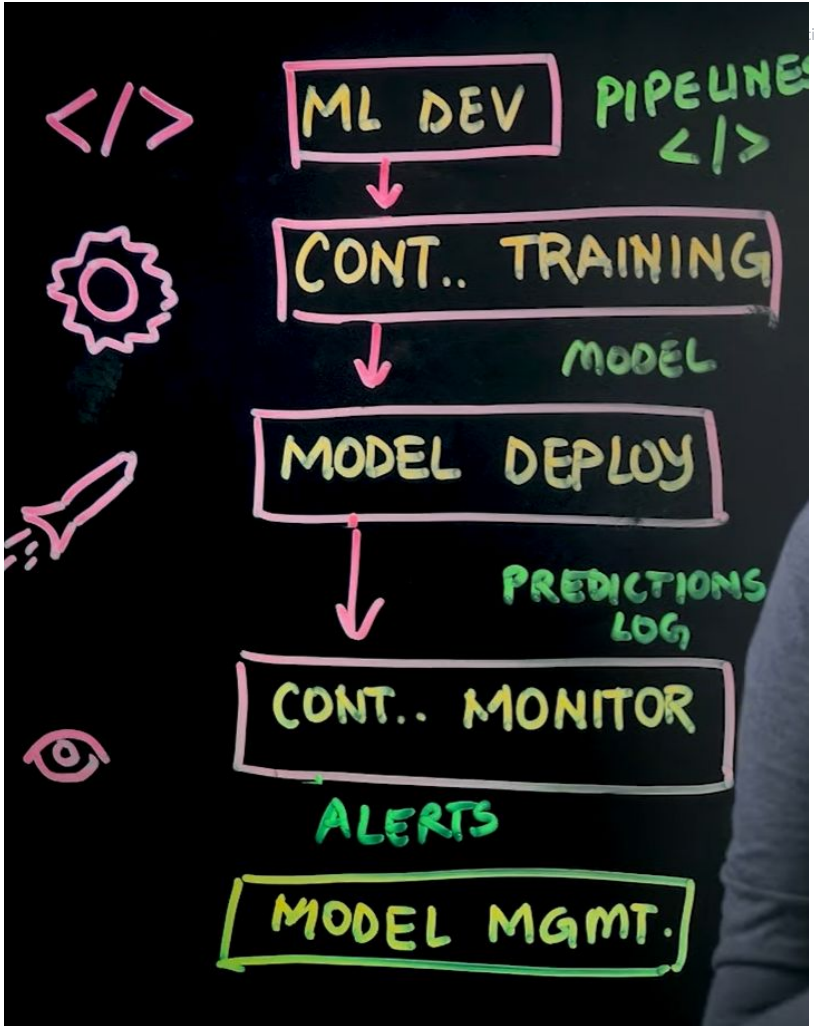
Vertex AI Explainable AI

Model Cards

Vertex AI pipelines

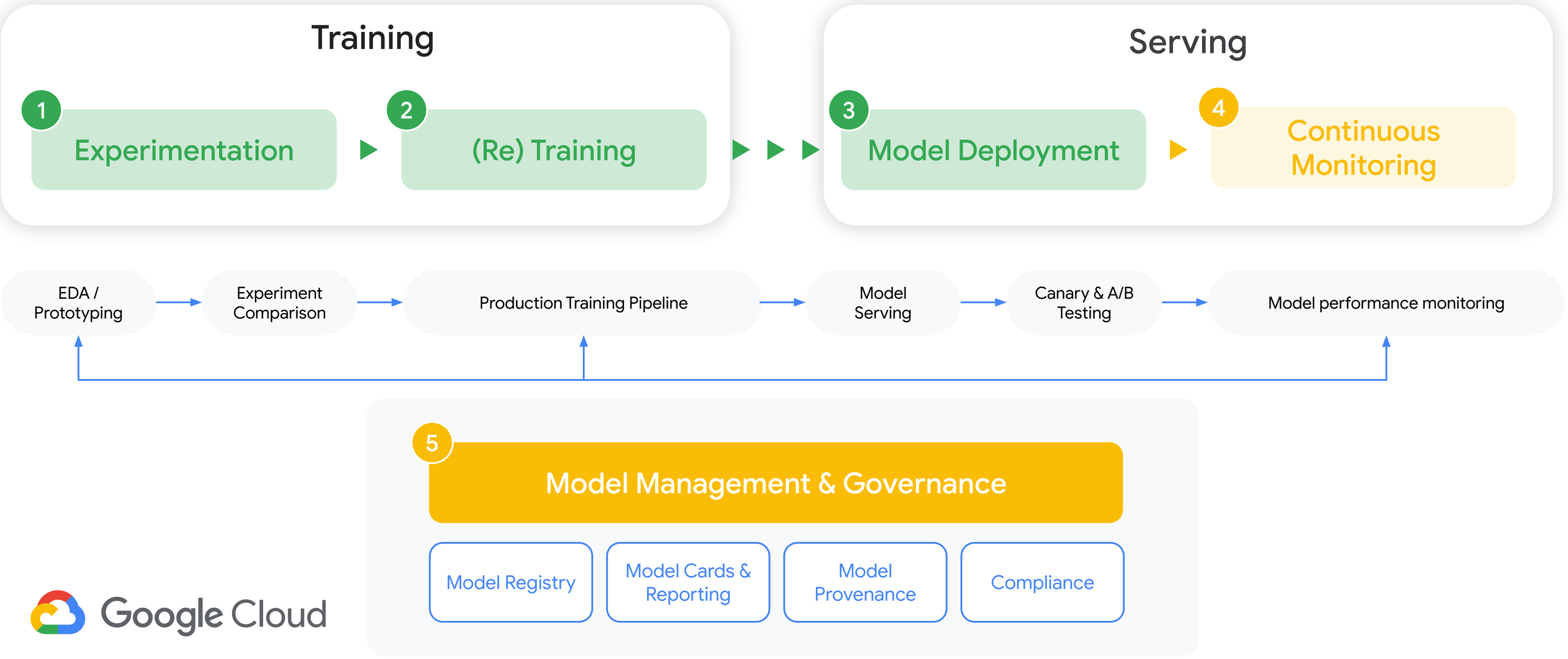
MLOps on Vertex AI





Efficient and responsible AI requires end-to-end MLOps

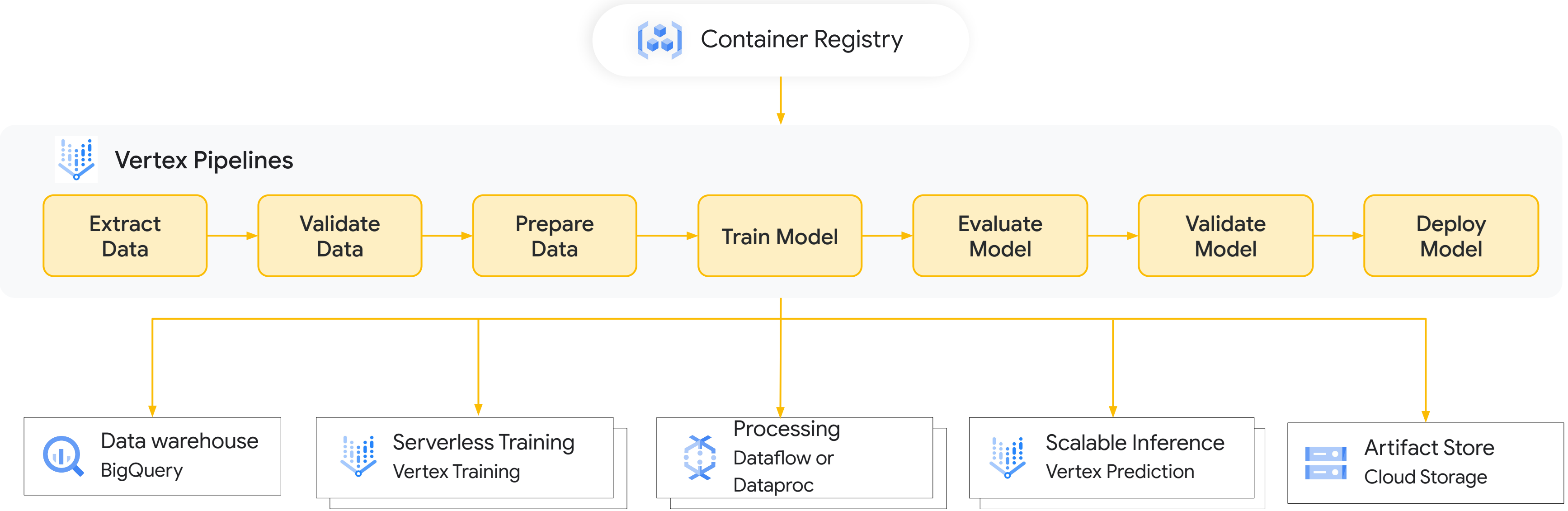
Vertex AI's end-to-end MLOps enables data scientists and ML engineers to efficiently and responsibly manage, monitor, govern, and explain ML projects throughout the entire development lifecycle.



Google Cloud

Manage

Simplify MLOps with Vertex AI Pipelines



Etsy

*"We're estimating a **~50% reduction in the time** it takes to go from idea to live ML experiment. [with **Vertex AI Pipelines**]"*

Govern

Manage and govern your ML models with Feature Store, ML Metadata, and Model Registry

Feature Store

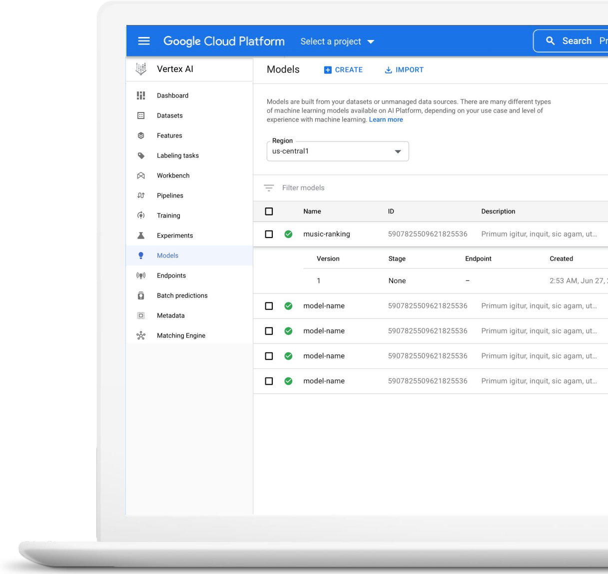
- **Share** and **reuse** ML features across use cases
- Serve ML Features **at scale** with **low latency**
- **Alleviate** training serving skew

ML Metadata

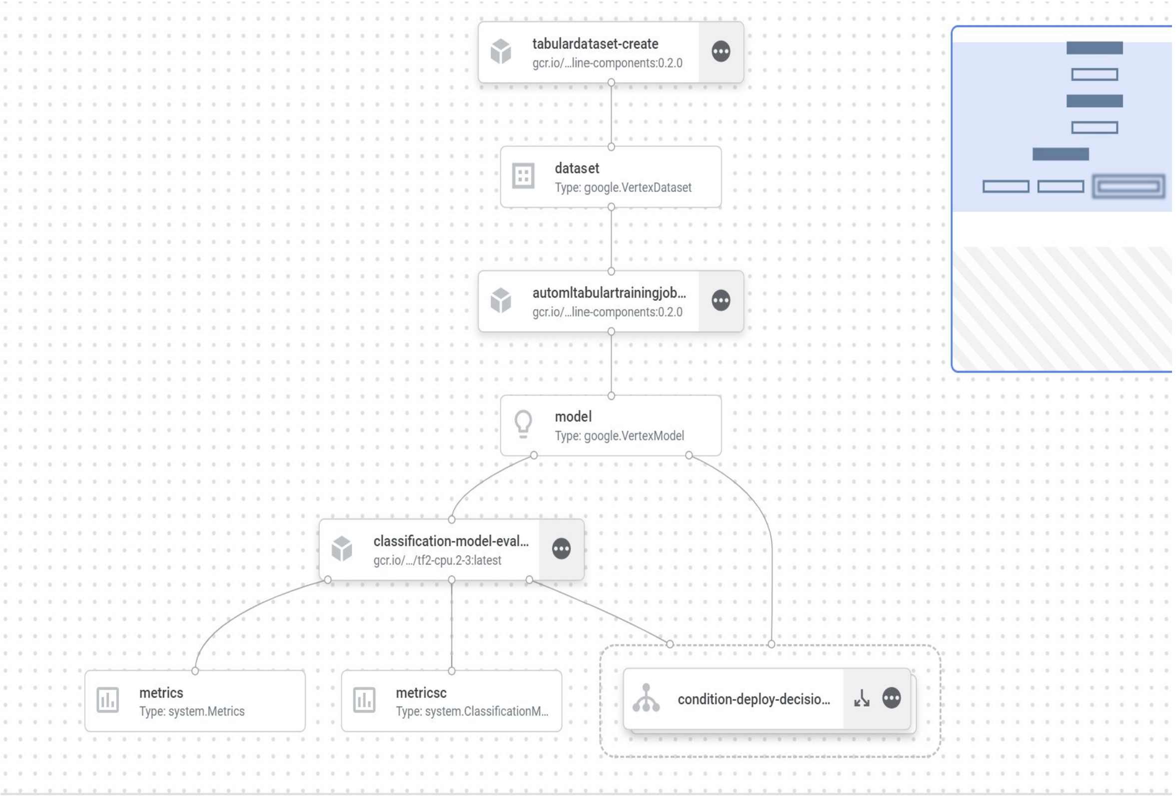
- **Automatically track** inputs / outputs to all components
- Track custom metadata **directly from your code**
- **Visualize**, **analyze**, and **compare** detailed ML lineage

Model Registry

- **Register, organize, track,** and **version** your trained and deployed ML models.
- **Govern** the model launch process
- **Maintain** model documentation and reporting



Deploy when threshold is met




```
    # Use the given metrics threshold(s) to determine whether the model is
    # accurate enough to deploy.
    def classification_thresholds_check(metrics_dict, thresholds_dict):
        for k, v in thresholds_dict.items():

            if k in ["auRoc", "auPrc"]:  # higher is better
                if metrics_dict[k] < v:  # if under threshold, don't deploy
                    logging.info("{} < {}; returning False".format(metrics_dict[k], v))
                    return False
        logging.info("threshold checks passed.")
        return True
```

Pipelines demo

Vertex AI model monitoring

Monitor

Proactively monitoring model performance with Model Monitoring

Monitor and alert

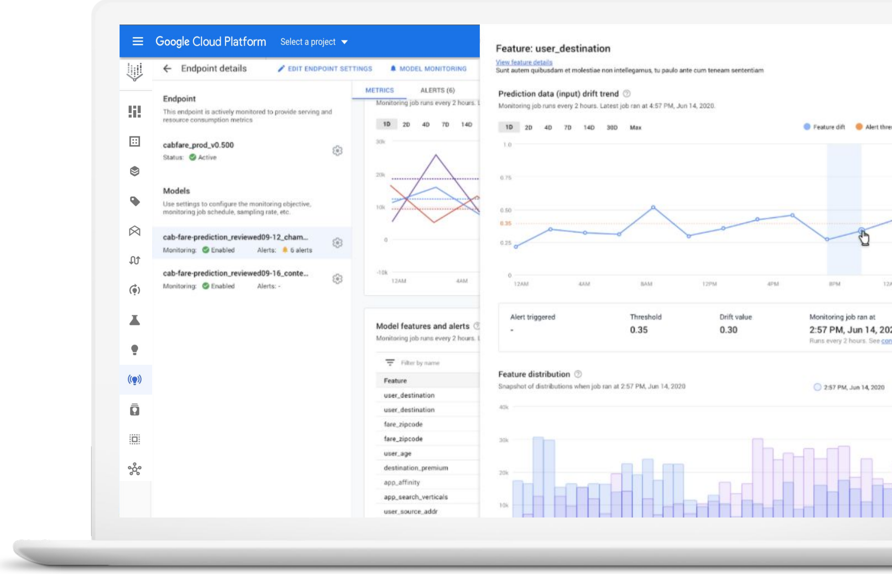
Monitor signals for model's predictive performance, and alert when those signals deviate.

Diagnose

Help identify the cause for the deviation i.e. what changed, how and how much?

Update Model

Trigger model re-training pipeline or collect relevant training data to address performance degradation.



Monitoring Objective

New endpoint

- Define your endpoint
- Model settings
- Model monitoring
- 4 Monitoring objectives

CREATE CANCEL

Model monitoring applies to **all models** deployed on this endpoint

Monitoring objective

- Training-serving skew detection
  Training-serving skew occurs when the feature data distribution in production is different from the feature data distribution in model training
- Prediction drift detection
  Prediction drift occurs when feature data distribution in production changes significantly over time

Training-serving skew detection

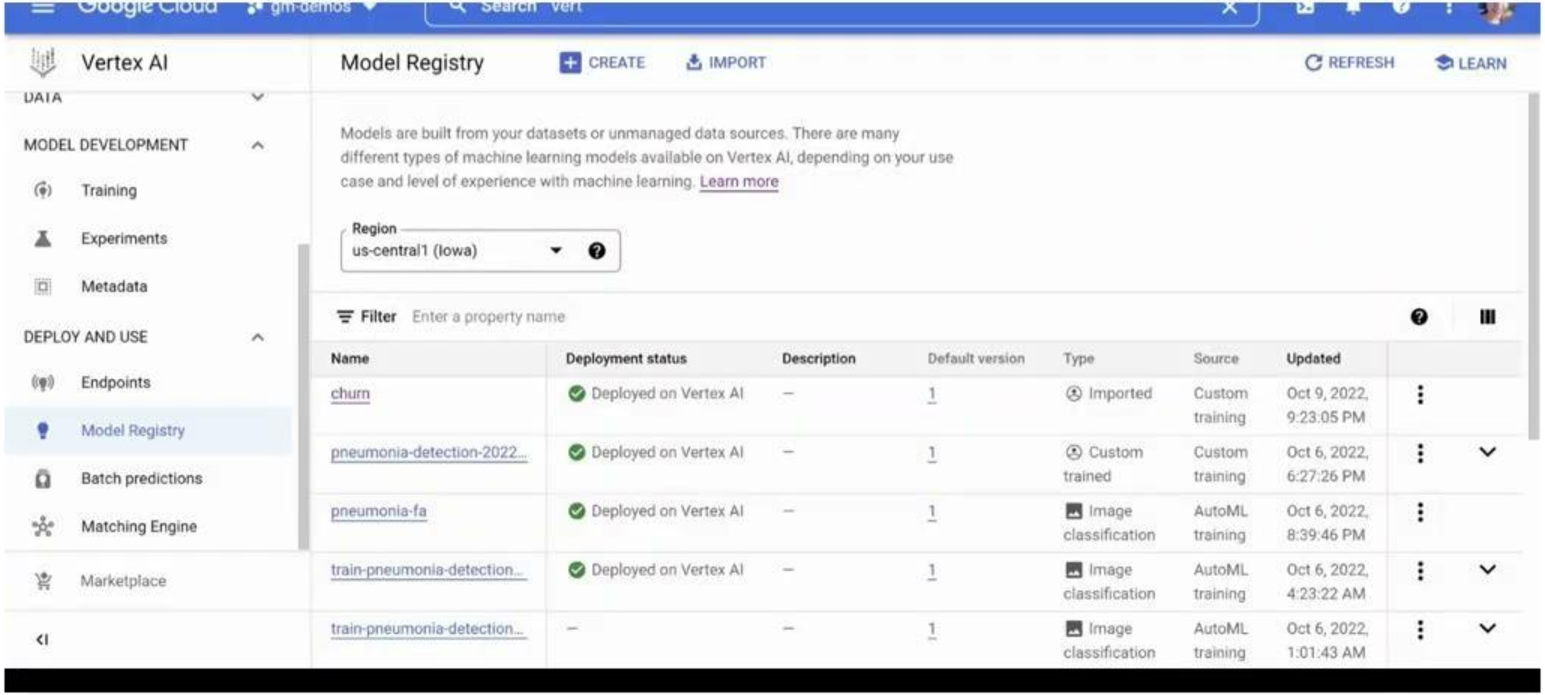
Training data source

To detect training-serving skew, the monitoring job needs to compare the model training data to the dataset used to train the model

- Cloud Storage bucket
- BigQuery table
- Vertex AI dataset

BigQuery path * BROWSE

Model Monitoring demo



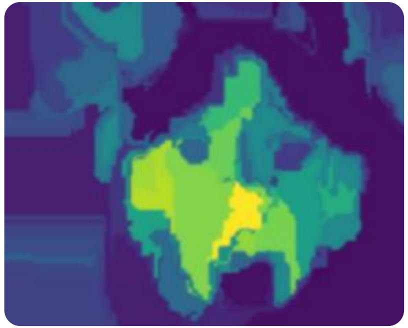
Demo How to setup monitoring using Console

Vertex AI Explainable AI

Proprietary + Confidential

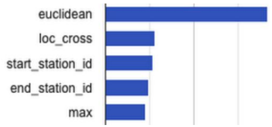
Explainable AI tells you how important each input feature is

Images



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



Text

The cake tastes delicious!

`Sentiment score: 0.9`

Explanations tell you:

What **image pixels or regions** most contributed to the model's classification?

How much did each **feature column** contribute to a single prediction or the model overall?

How much did each **word** or **token** contribute to the text classification?

Proprietary + Confidential

Explainable AI feature set

1

Robust

Three explainability methods based on established research*

- Sampled Shapley
- Integrated Gradients
- XRAI

Intuitive for data scientists & end-users

* See our AI Explainability Whitepaper for details

2

Flexible

Supports multiple model types:

- Tabular classification & regression
- Image classification
- Text classification

Support Online and Batch Processing

ML framework-agnostic: compatible with any model deployed as a Custom Container

3

Seamlessly integrated

XAI currently available in:

- AutoML Tables
- Vertex Prediction
- Vertex Notebooks

- Continuous Monitoring
- Others...

4

Easy to use & scale

Explainable SDK enables quick set-up

Managed, serverless service

Significantly faster and more resource-efficient than OSS packages

New

Vertex AI Example-based explanations Preview

Build better models

Mislabeled Examples	Look for examples in the training data where similar examples have a different label.
Active Learning	Look for unlabeled examples where neighbors have a variety of labels. Label these & add them to the training data.
Misclassification Analysis	Look at examples from the training set that are 'nearby' the misclassified instance to identify if new data is needed or existing examples are mislabeled/noisy.

Loop in stakeholders

Decision Support	Provide a rationale for an ML-generated prediction/decision by surfacing previous relevant predictions or similar data points.

We trained an image classification model on a subset of the STL-10 dataset, using only images of birds and planes. We noted some images of birds being misclassified as planes. For one such image, we used Example-based Explanations to retrieve other images in the training data that appeared most similar to this misclassified bird image in the latent space.

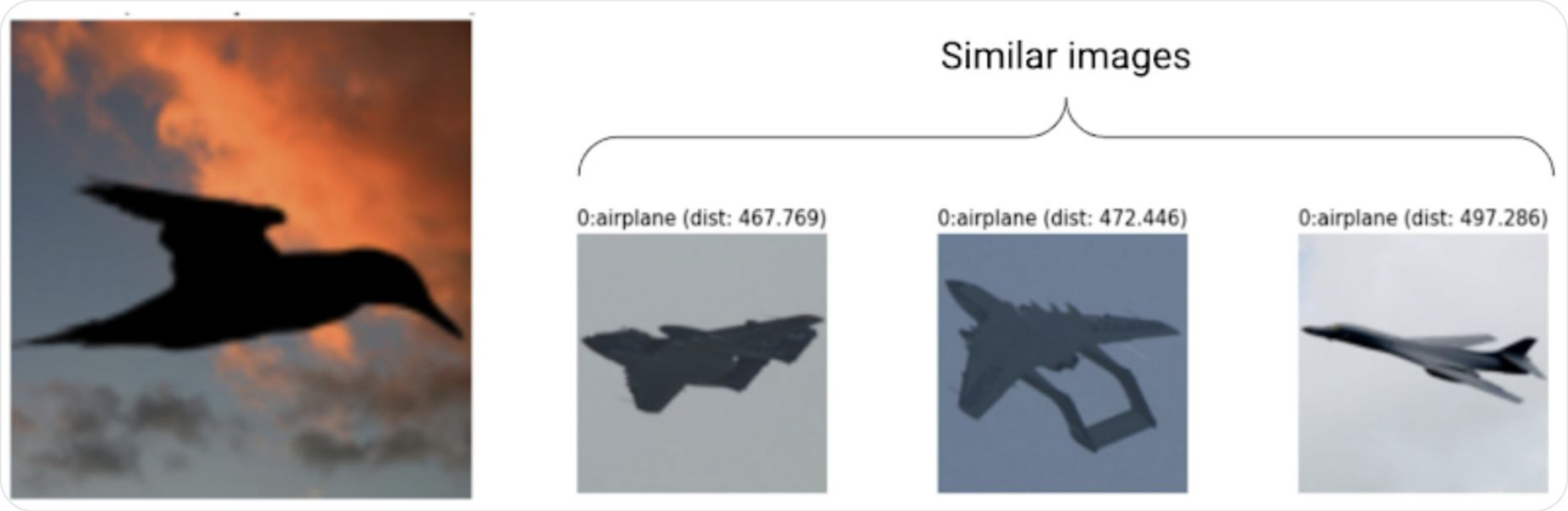
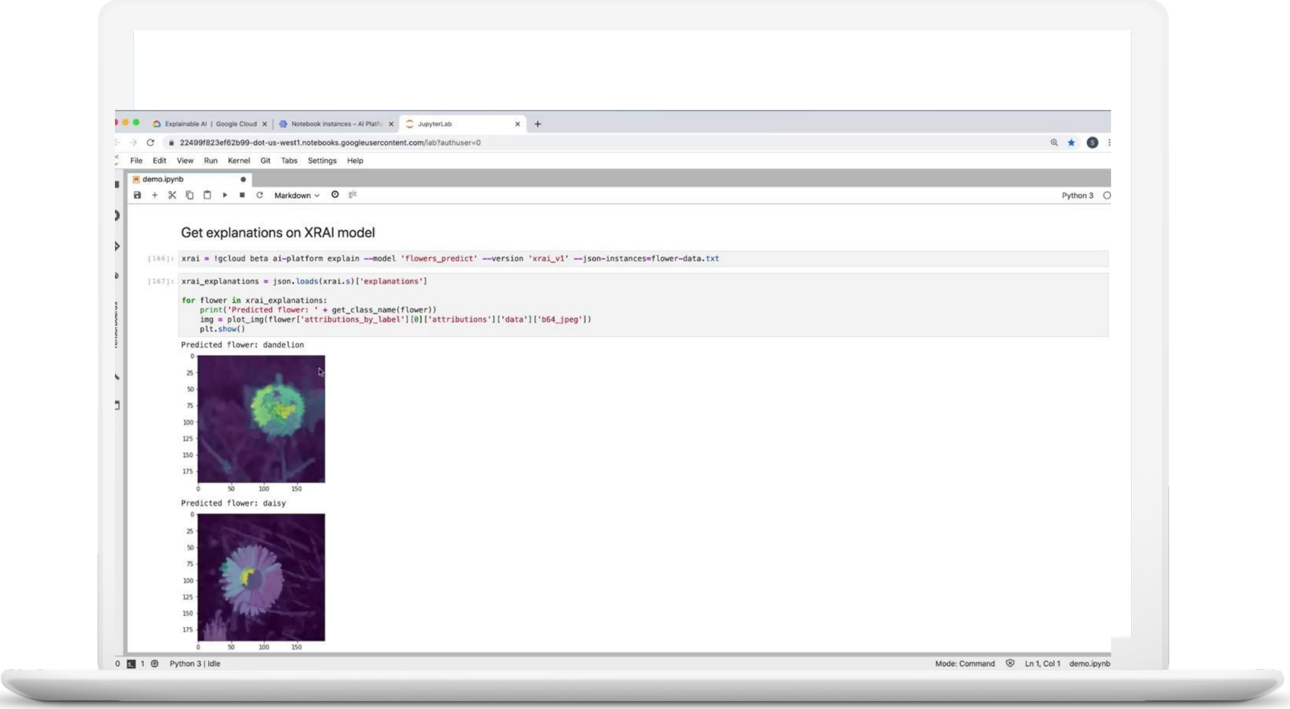


Figure 2. Use Example-based Explanations for misclassification analysis

Proprietary + Confidential

Demo

AI Platform Prediction & Notebooks
(7min 29sec)

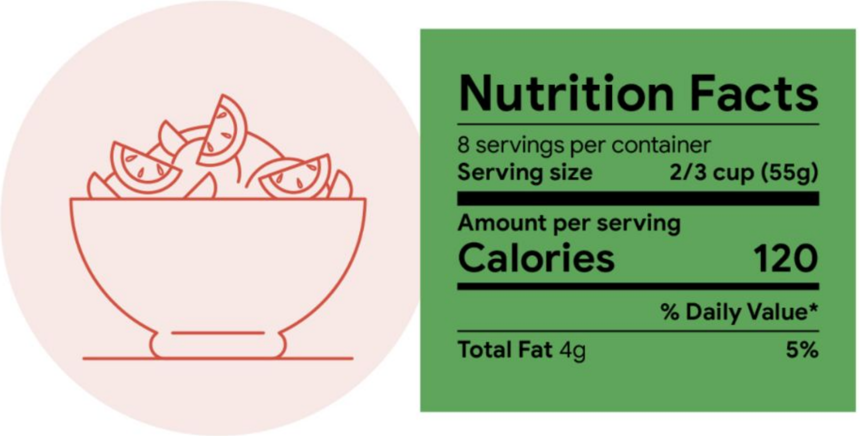


Model Card

What and why of Model Card?

under what conditions does the model perform best and most consistently? Does it have blind spots? If so, where?

Does a model perform consistently across a diverse range of people, or does it vary in unintended ways as characteristics like skin color or region change?

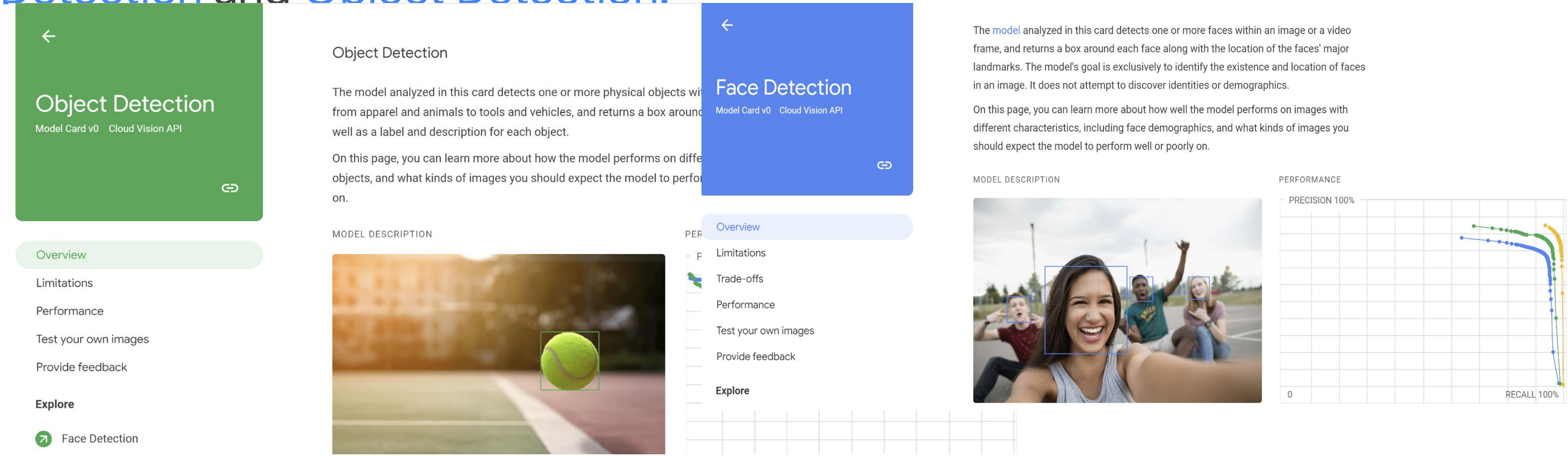


Benefits of using a model card

- Content
- Process
- Experience
- Fairness in Machine Learning
- Privacy

Google Cloud Model Cards

To explore the possibilities of model cards in the real world, we've designed examples for two features of our Cloud Vision API, Face Detection and Object Detection.



Thank You

https://www.linkedin.com/in/mona-mona/

https://www.linkedin.com/in/giovanni-marchetti/