Copyright Notice

DeepLearning.AI

Model Analysis

Welcome

DeepLearning.AI

Model Analysis Overview

Model Performance Analysis

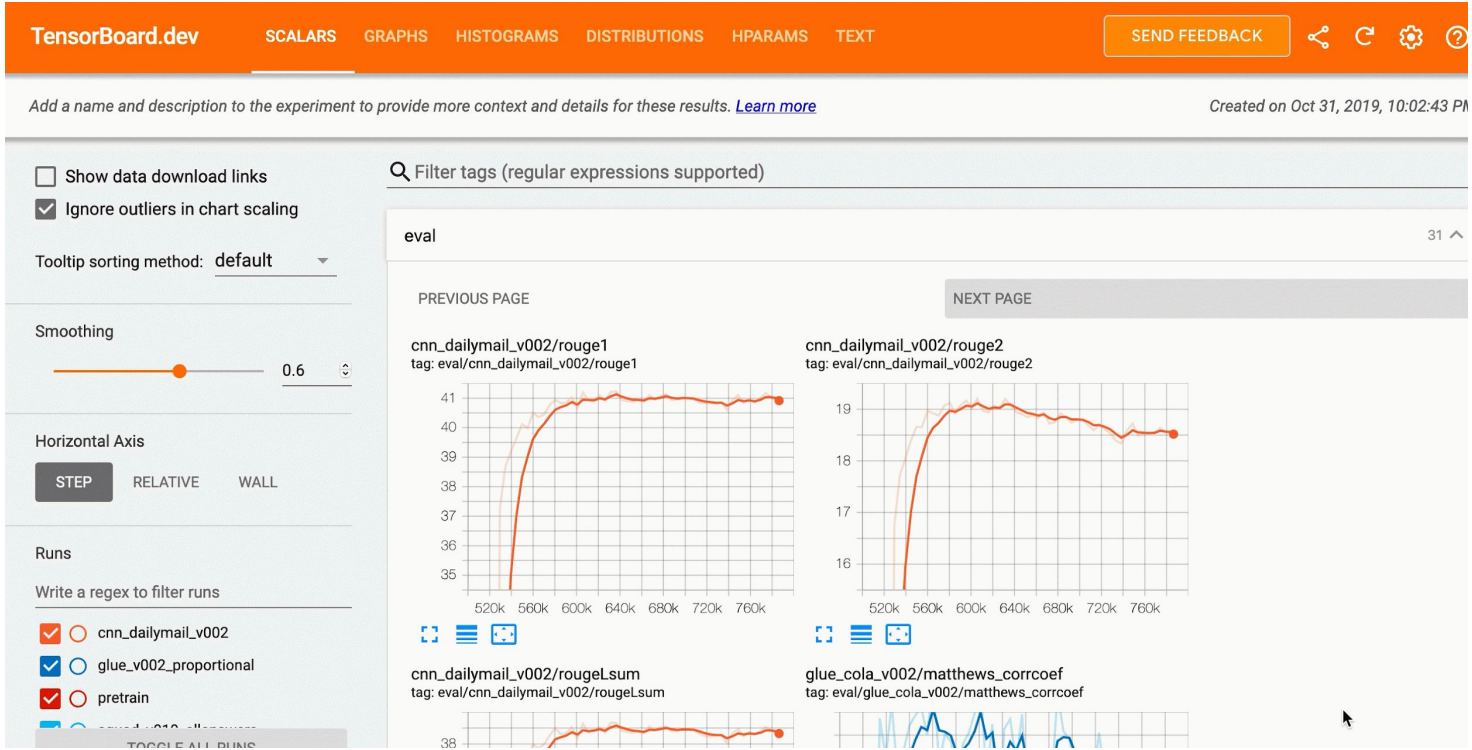
What is next after model training/deployment?

- Is model performing well?
- Is there scope for improvement?
- Can the data change in future?
- Has the data changed since you created your training dataset?

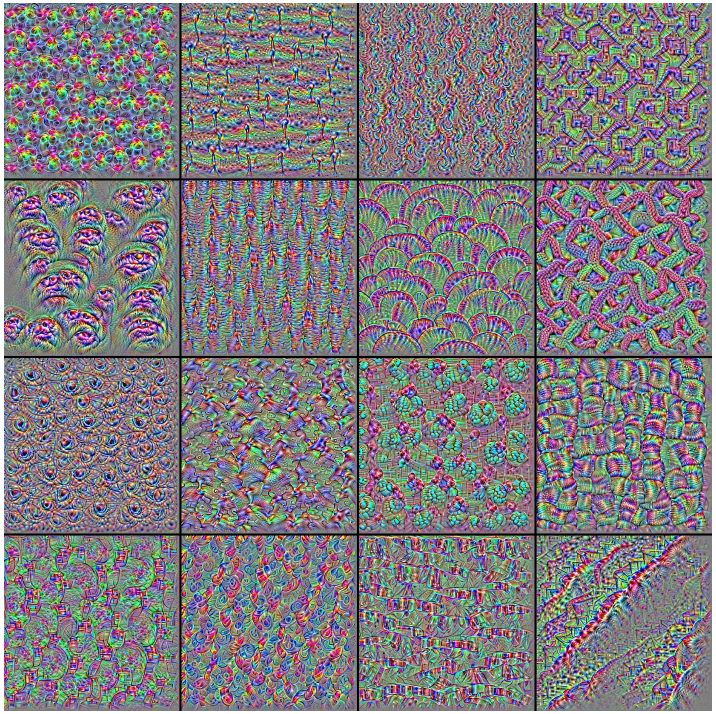
Black box evaluation vs model introspection

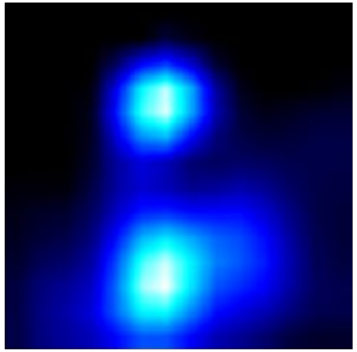
- Models can be tested for metrics like accuracy and losses like test error without knowing internal details
- For finer evaluation, models can be inspected part by part

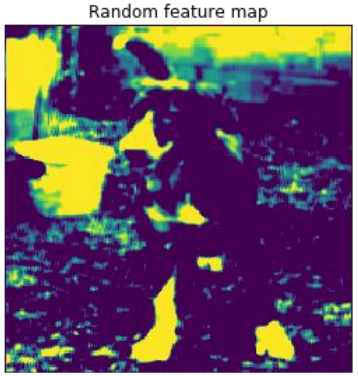
Black box evaluation



Model introspection

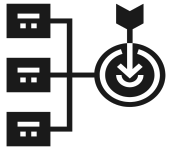






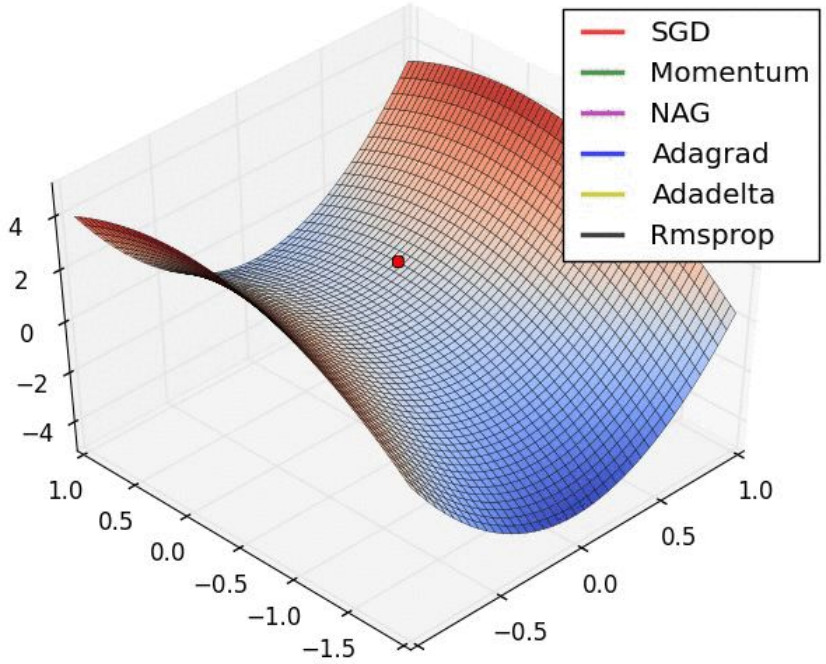
Performance metrics vs optimization objectives

- Performance metrics will vary based on the task like regression, classification, etc.
- Within a type of task, based on the end-goal, your performance metrics may be different
- Performance is measured after a round of optimization



- Machine Learning formulates the problem statement into an objective function
- Learning algorithms find optimum values for each variable to converge into local/global minima

Performance metrics vs optimization objectives



https://cs231n.github.io/neural-networks-3/

Top level aggregate metrics vs slicing

- Most of the time, metrics are calculated on the entire dataset
- Slicing deals with understanding how the model is performing on each subset of data

DeepLearning.AI

Advanced Model Analysis and Debugging

Introduction to TensorFlow Model Analysis

Why should you slice your data?

Your top-level metrics may hide problems

- Your model may not perform well for particular [customers | products | stores | days of the week | etc.]

Each prediction request is an individual event, maybe an individual customer

- For example, customers may have a bad experience
- For example, some stores may perform badly

TensorFlow Model Analysis (TFMA)

Scalable framework

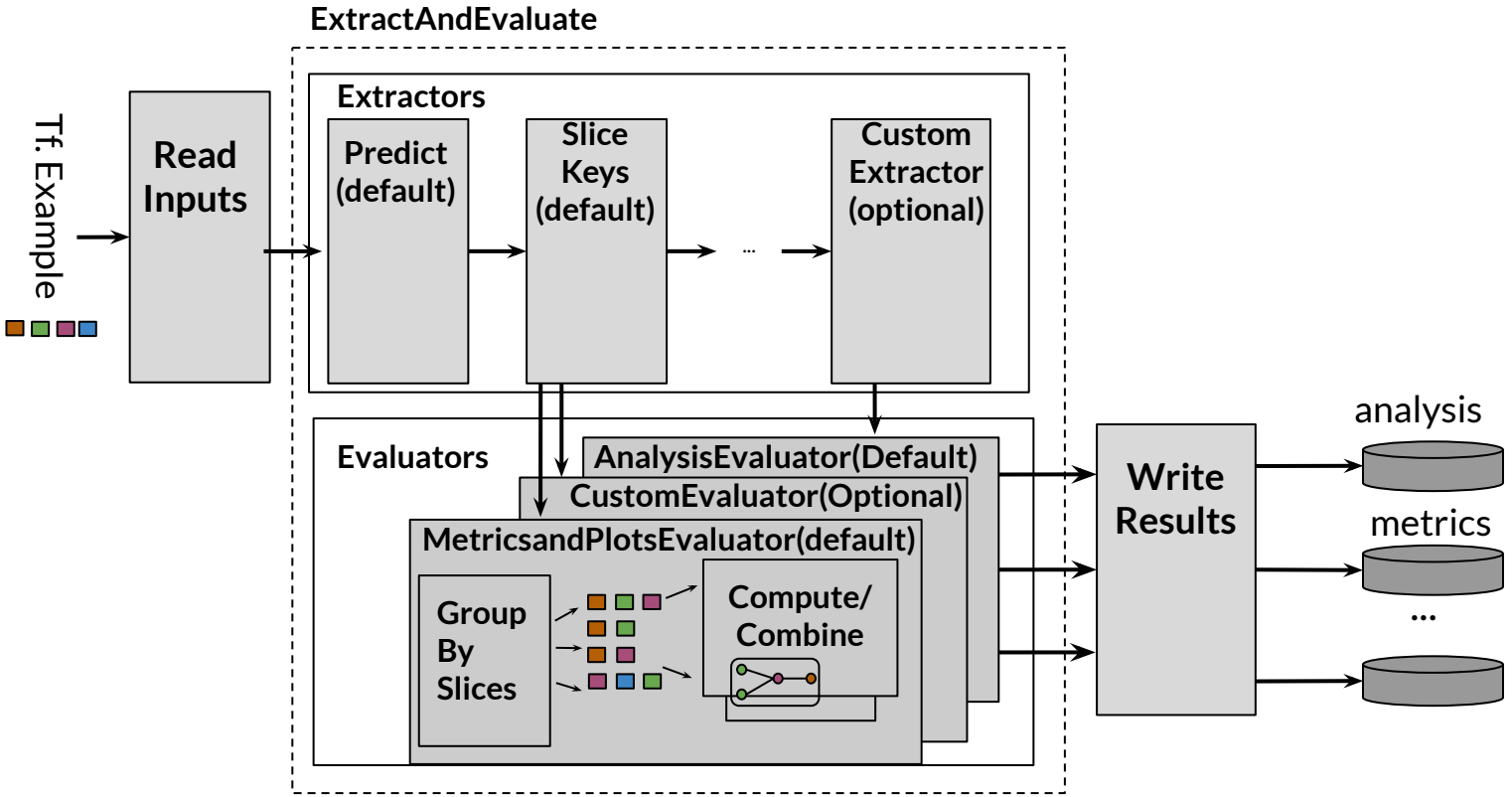
Open source library

Ensures models meet required quality thresholds

Used to compute and visualize evaluation metrics

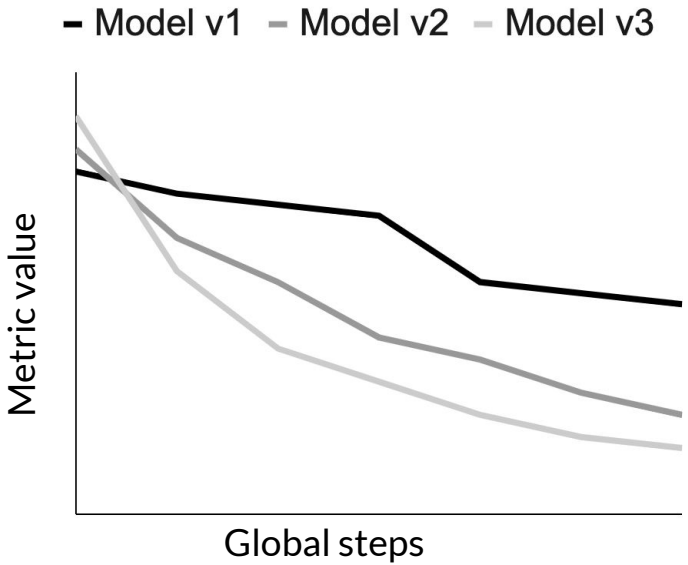
Inspect model's performance against different slices of data

Architecture

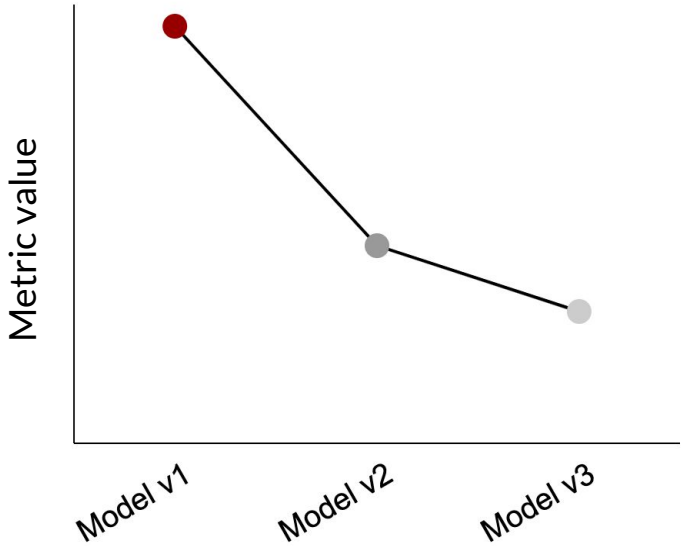


One model vs multiple models over time

TensorFlow metrics in TensorBoard

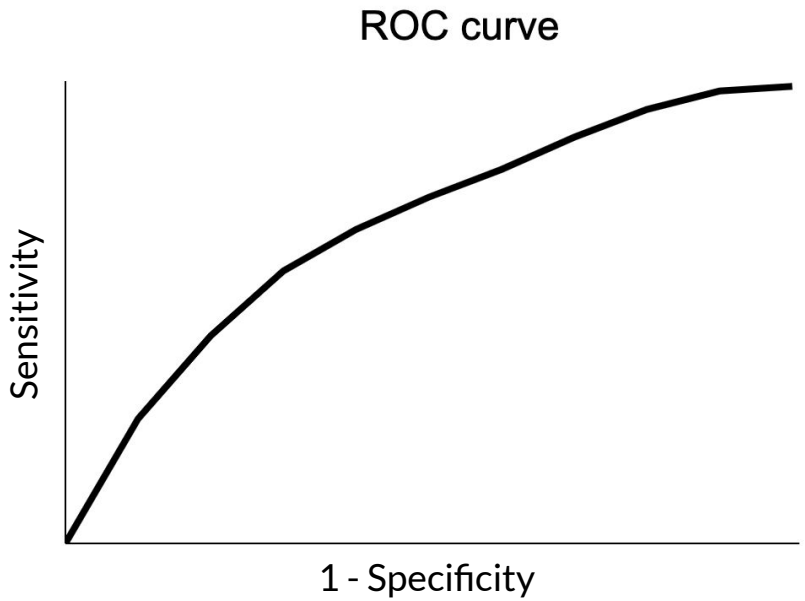


TensorFlow metrics in TensorFlow model analysis

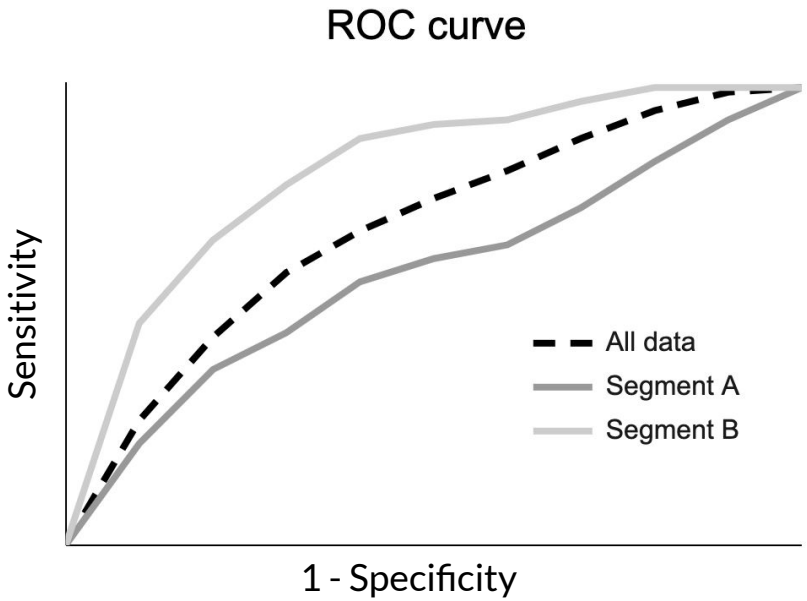


Aggregate vs sliced metrics

Aggregate metric computed over entire eval dataset



Metric “sliced” by different segments of the eval dataset



Streaming vs full-pass metrics

Streaming metrics are approximations computed on mini-batches of data

TensorBoard visualizes metrics through mini-batches

TFMA gives evaluation results after running through entire dataset

Apache Beam is used for scaling on large datasets

DeepLearning.AI

Advanced Model Analysis and Debugging

TFMA in Practice

TFMA in practice

- Analyse impact of different slices of data over various metrics
- How to track metrics over time?

Step 1: Export EvalSavedModel for TFMA

```
import tensorflow as tf
import tensorflow_transform as tft
import tensorflow_model_analysis as tfma

def get_serve_tf_examples_fn(model, tf_transform_output):
  # Return a function that parses a serialized tf.Example and applies TFT

  tf_transform_output = tft.TFTransformOutput(transform_output_dir)
signatures = {
  'serving_default': get_serve_tf_examples_fn(model, tf_transform_output)
                         .get_concrete_function(tf.TensorSpec(...)),
}

model.save(serving_model_dir_path, save_format='tf', signatures=signatures)
```

Step 2: Create EvalConfig

```
# Specify slicing spec
slice_spec = [slicer.SingleSliceSpec(columns=[‘column_name’]), ...]
# Define metrics
metrics = [tf.keras.metrics.Accuracy(name='accuracy'),
           tfma.metrics.MeanPrediction(name='mean_prediction'), ...]
metrics_specs = tfma.metrics.specs_from_metrics(metrics)

eval_config = tfma.EvalConfig(
                    model_specs=[tfma.ModelSpec(label_key=features.LABEL_KEY)],
                    slicing_specs=slice_spec,
                    metrics_specs=metrics_specs, ...)
```

Step 3: Analyze model

```
# Specify the path to the eval graph and to where the result should be written
eval_model_dir = ...
result_path = ...

eval_shared_model = tfma.default_eval_shared_model(
                            eval_saved_model_path=eval_model_dir,
                            eval_config=eval_config)
# Run TensorFlow Model Analysis
eval_result = tfma.run_model_analysis(eval_shared_model=eval_shared_model,
                                      output_path=result_path,
                                      ...)
```

Step 4: Visualizing metrics

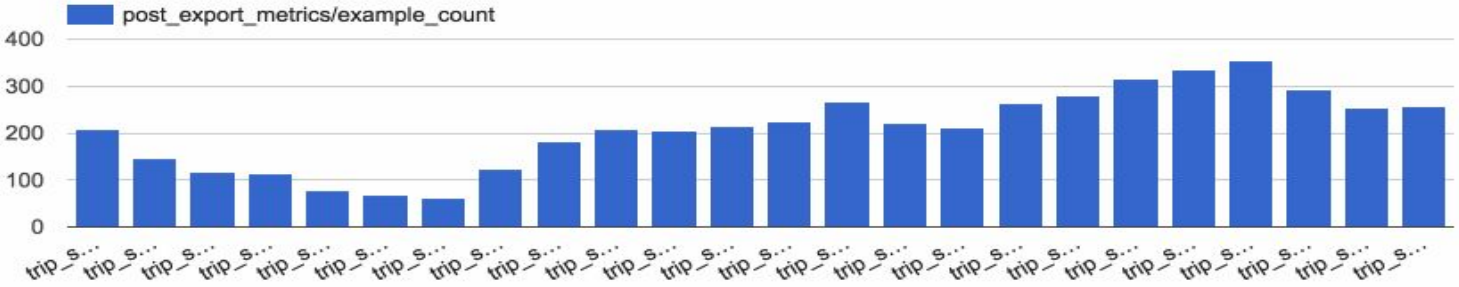
```
# render results
tfma.viewer.render_slicing_metrics(result)
```

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_loss
trip_start_hour:19	0.63582	0.59104	0.64311	0.56092	0.64626
trip_start_hour:14	0.67117	0.65766	0.63793	0.49112	0.61667
trip_start_hour:2	0.66102	0.63559	0.58527	0.47002	0.65236
trip_start_hour:12	0.69643	0.65625	0.68270	0.54122	0.59538
trip_start_hour:0	0.66184	0.66667	0.63773	0.45081	0.61634
trip_start_hour:23	0.65625	0.64844	0.58357	0.43514	0.64315

DeepLearning.AI

Advanced Model Analysis and Debugging

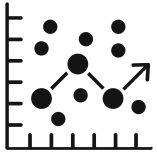
Model Debugging Overview

Model robustness

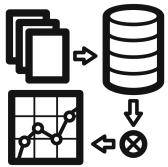
- Robustness is much more than generalization
- Is the model accurate even for slightly corrupted input data?

Robustness metrics

Robustness measurement shouldn't take place during training



Split data in to train/val/dev sets



Specific metrics for regression and classification problems

Model debugging

- Deals with detecting and dealing with problems in ML systems
- Applies mainstream software engineering practices to ML models

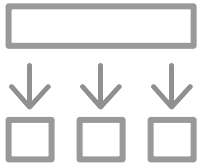
Model Debugging Objectives

Opaqueness

Social discrimination

Security vulnerabilities

Privacy harms

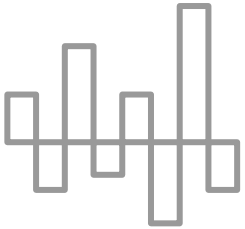


Model decay

Model Debugging Techniques

Benchmark models

Sensitivity analysis



Residual analysis

DeepLearning.AI

Advanced Model Analysis and Debugging

Benchmark Models

Benchmark models

Simple, trusted and interpretable models solving the same problem

Compare your ML model against these models

Benchmark model is the starting point of ML development

DeepLearning.AI

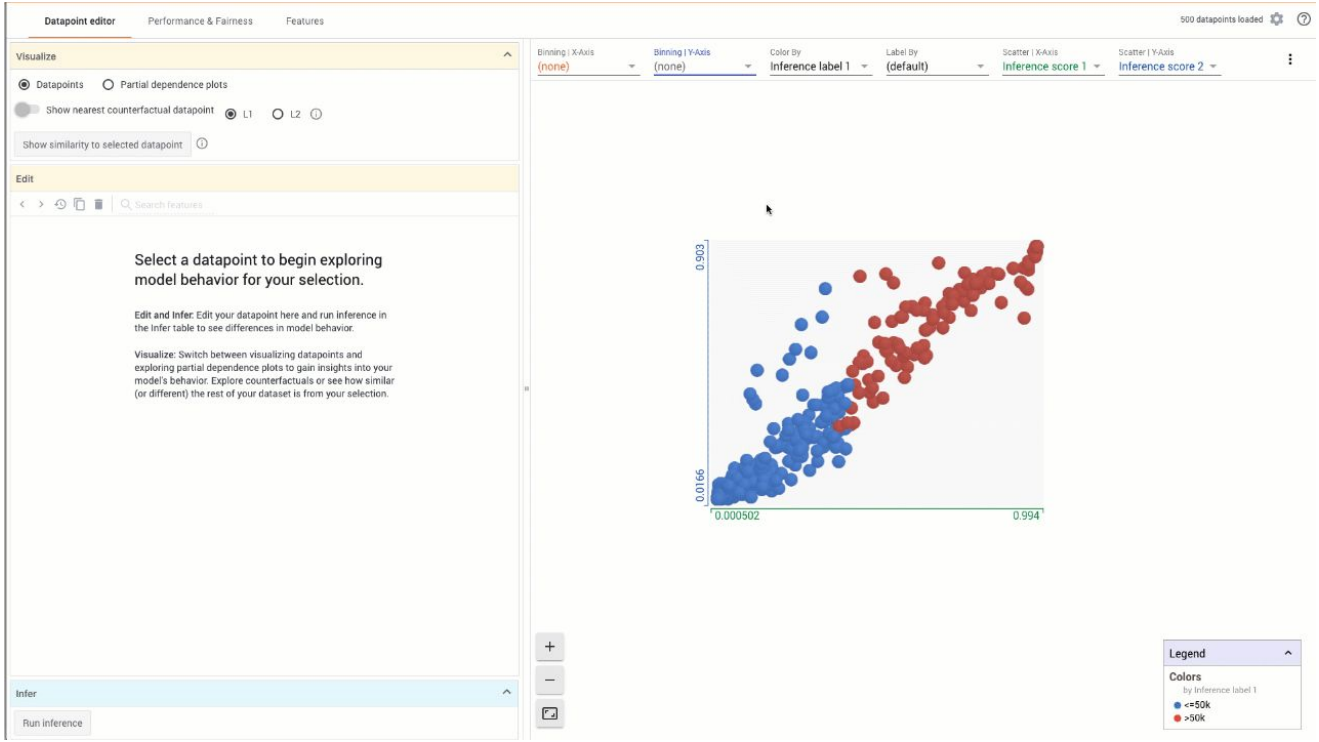
Advanced Model Analysis and Debugging

Sensitivity Analysis and Adversarial Attacks

Sensitivity analysis

- Simulate data of your choice and see what your model predicts
- See how model reacts to data which has never been used before

What-If Tool for sensitivity analysis



Random Attacks

- Expose models to high volumes of random input data
- Exploits the unexpected software and math bugs
- Great way to start debugging

Partial dependence plots

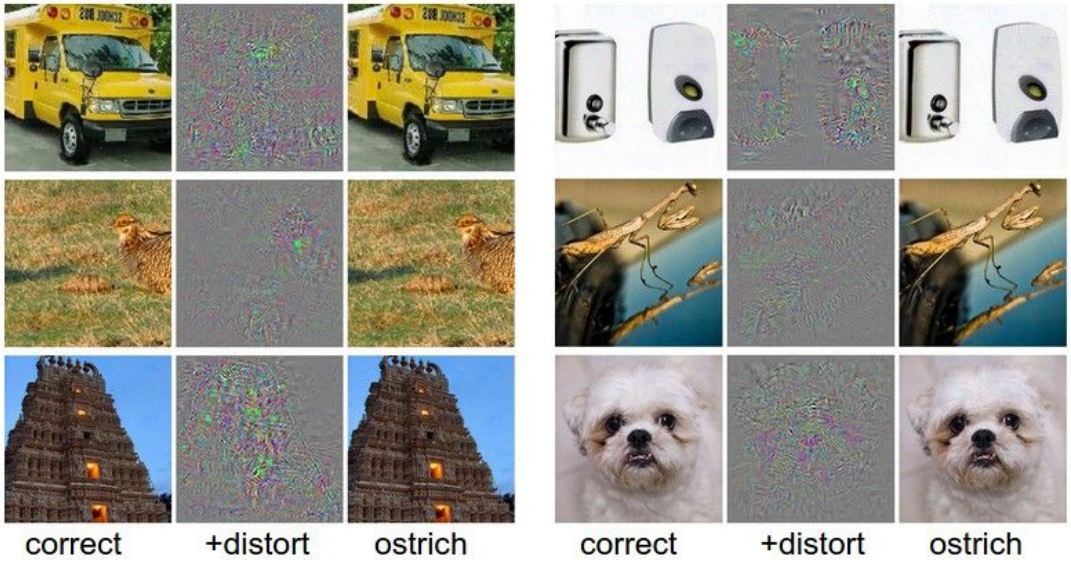
- Visualize the effects of changing one or more variables in your model
- PDPbox and PyCEbox open source packages

How vulnerable to attacks is your model?

Sensitivity can mean vulnerability

- Attacks are aimed at fooling your model
- Successful attacks could be catastrophic
- Test adversarial examples
- Harden your model

A Famous Example: Ostrich



How vulnerable to attacks is your model?

Example:

A self-driving car crashes because black and white stickers applied to a stop sign cause a classifier to interpret it as a Speed Limit 45 sign.

How vulnerable to attacks is your model?

Example:

A spam detector fails to classify an email as spam. The spam mail has been designed to look like a normal email, but is actually phishing.

How vulnerable to attacks is your model?

Example:

A machine-learning powered scanner scans suitcases for weapons at an airport. A knife was developed to avoid detection by making the system think it is an umbrella.

Informational and Behavioral Harms

- Informational Harm: Leakage of information
- Behavioral Harm: Manipulating the behavior of the model

Informational Harms

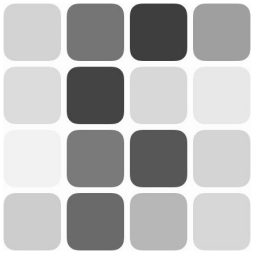
- Membership Inference: was this person's data used for training?
- Model Inversion: recreate the training data
- Model Extraction: recreate the model

Behavioral Harms

- Poisoning: insert malicious data into training data
- Evasion: input data that causes the model to intentionally misclassify that data

Measuring your vulnerability to attack

Cleverhans:
an open-source Python library to benchmark machine learning systems' vulnerability to adversarial examples

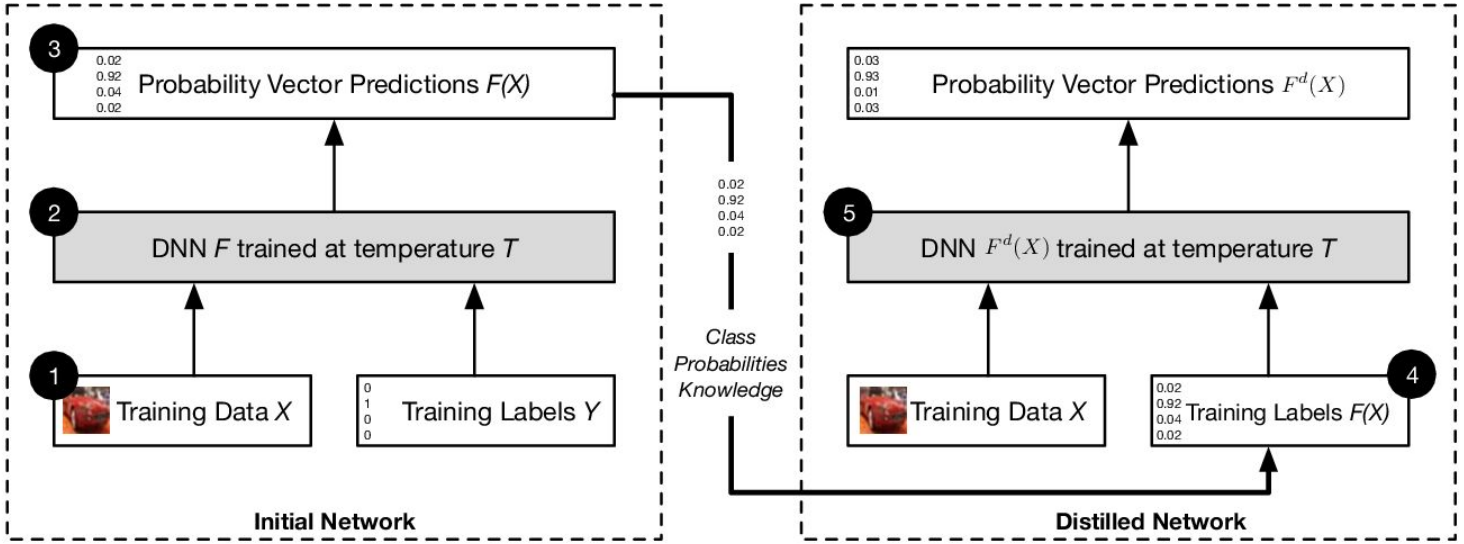


Foolbox:
an open-source Python library that lets you easily run adversarial attacks against machine learning models

Adversarial example searches

Attempted defenses against adversarial examples

- Defensive distillation



DeepLearning.AI

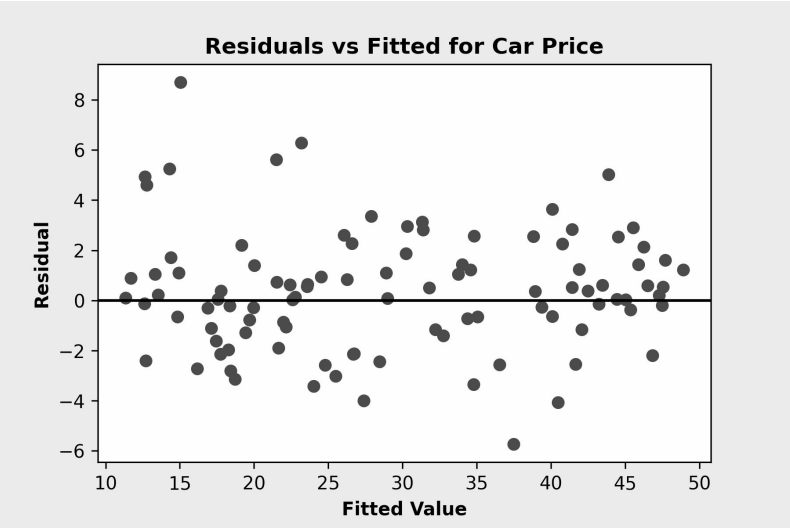
Advanced Model Analysis and Debugging

Residual Analysis

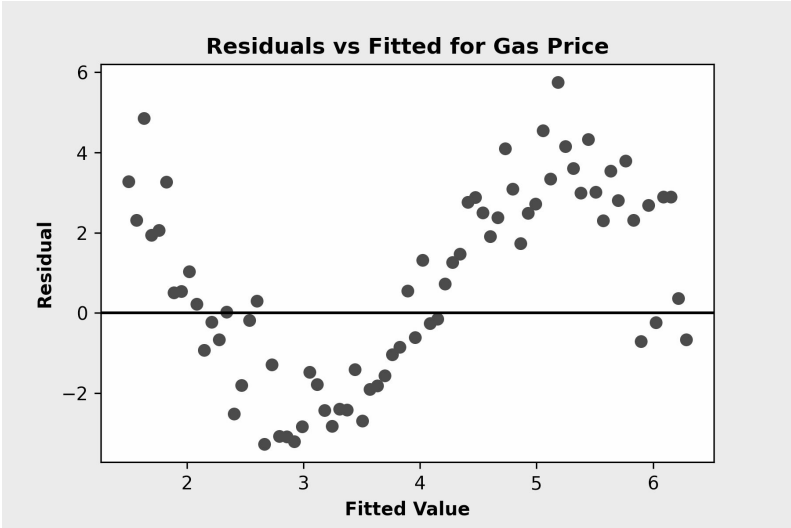
Residual analysis

- Measures the difference between model’s predictions and ground truth
- Randomly distributed errors are good
- Correlated or systematic errors show that a model can be improved

Residual analysis



Random = Good



Systematic = Bad

Residual analysis

- Residuals should not be correlated with another feature
- Adjacent residuals should not be correlated with each other (autocorrelation)

DeepLearning.AI

Advanced Model Analysis and Debugging

Model Remediation

Remediation techniques

Data augmentation

- Adding synthetic data into training set
- Helps correct for unbalanced training data

Interpretable and explainable ML

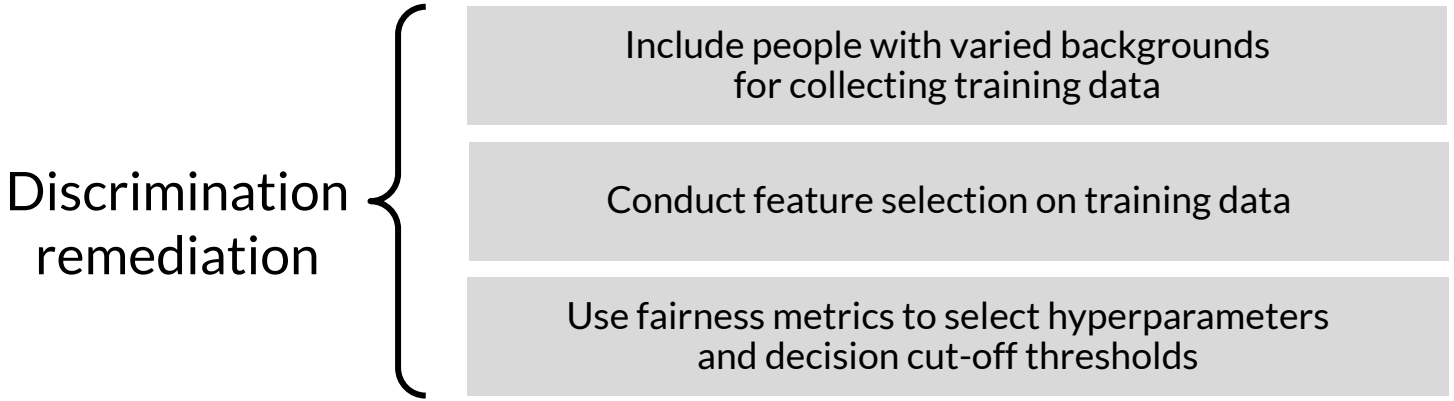
- Overcome myth of neural networks as black box
- Understand how data is getting transformed

Remediation techniques

- Model editing:
  - Applies to decision trees
  - Manual tweaks to adapt your use case

- Model assertions:
  - Implement business rules that override model predictions

Remediation techniques



Remediation techniques

Model monitoring

- Conduct model debugging at regular intervals
- Inspect accuracy, fairness, security problems, etc

Anomaly detection

- Anomalies can be a warning of an attack
- Enforce data integrity constraints on incoming data

DeepLearning.AI

Advanced Model Analysis and Debugging

Fairness

Fairness indicators

- Open source library to compute fairness metrics
- Easily scales across dataset of any size
- Built in top of TFMA

What does fairness indicators do?

- Compute commonly-identified fairness metrics for classification models
- Compare model performance across subgroups to other models
- No remediation tools provided

Evaluate at individual slices

- Overall metrics can hide poor performance for certain parts of data
- Some metrics may fare well over others

Aspects to consider

- Establish context and different user types
- Seek domain experts help
- Use data slicing widely and wisely

General guidelines

- Compute performance metrics at all slices of data
- Evaluate your metrics across multiple thresholds
- If decision margin is small, report in more detail

DeepLearning.AI

Advanced Model Analysis and Debugging

Measuring Fairness

Positive rate / Negative rate

- Percentage data points classified as positive/negative
- Independent of ground truth
- Use case: having equal final percentages of groups is important

True positive rate (TPR) / False negative rate (FNR)

- TPR: percentage of positive data points that are correctly labeled positive
- FNR: percentage of positive data points that are incorrectly labeled negative
- Measures equality of opportunity, when the **positive class** should be equal across subgroups
- Use case: where it is important that same percent of qualified candidates are rated positive in each group

True negative rate (TNR) / False positive rate (FPR)

- TNR: percentage of negative data points that are correctly labeled negative
- FPR: percentage of negative data points that are incorrectly labeled positive
- Measures equality of opportunity, when the **negative class** should be equal across subgroup
- Use case: where misclassifying something as positive are more concerning than classifying the positives

Accuracy & Area under the curve (AUC)

- Accuracy: The percentage of data points that are correctly labeled
- AUC: The percentage of data points that are correctly labeled when each class is given equal weight independent of number of samples
- Metrics related to predictive parity
- Use case: when precision is critical

Tips

Unfair skews if there is a gap in a metric between two groups

Good fairness indicators doesn't always mean the model is fair

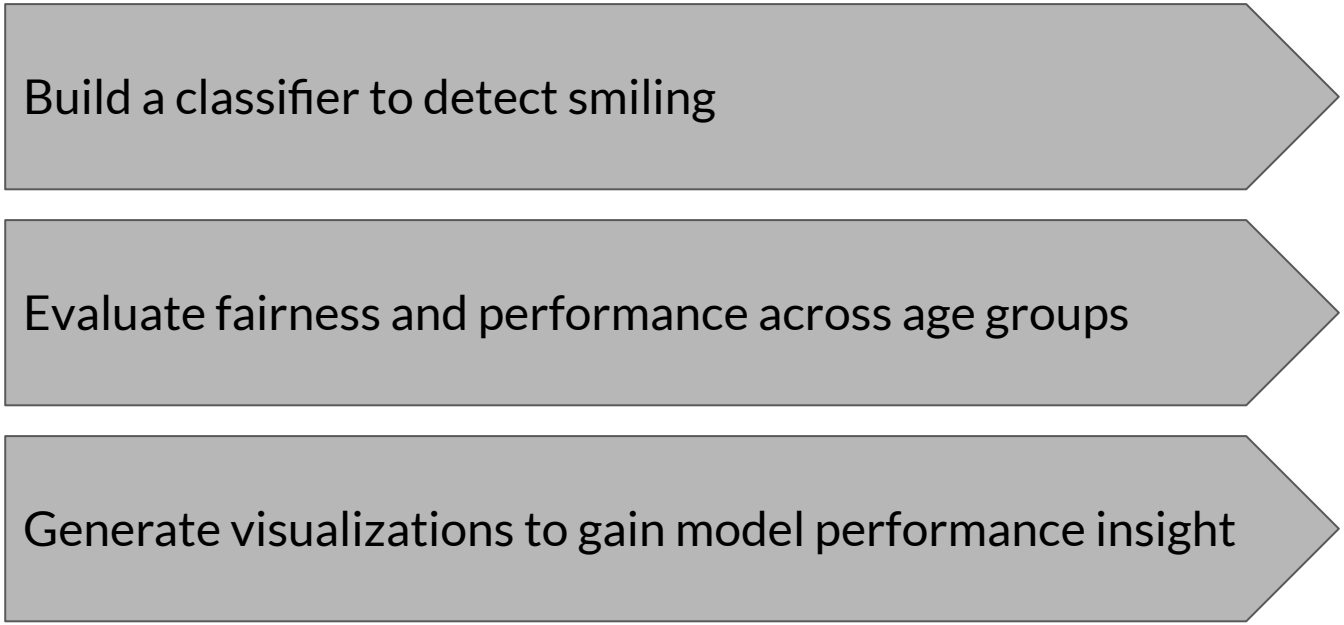
Continuous evaluation throughout development and deployment

Conduct adversarial testing for rare, malicious examples

About the CelebA dataset

- 200K celebrity images
- Each image has 40 attribute annotations
- Each image has 5 landmark locations
- Assumption on smiling attribute

Fairness indicators in practice



DeepLearning.AI

Continuous Evaluation and Monitoring

Continuous evaluation and monitoring

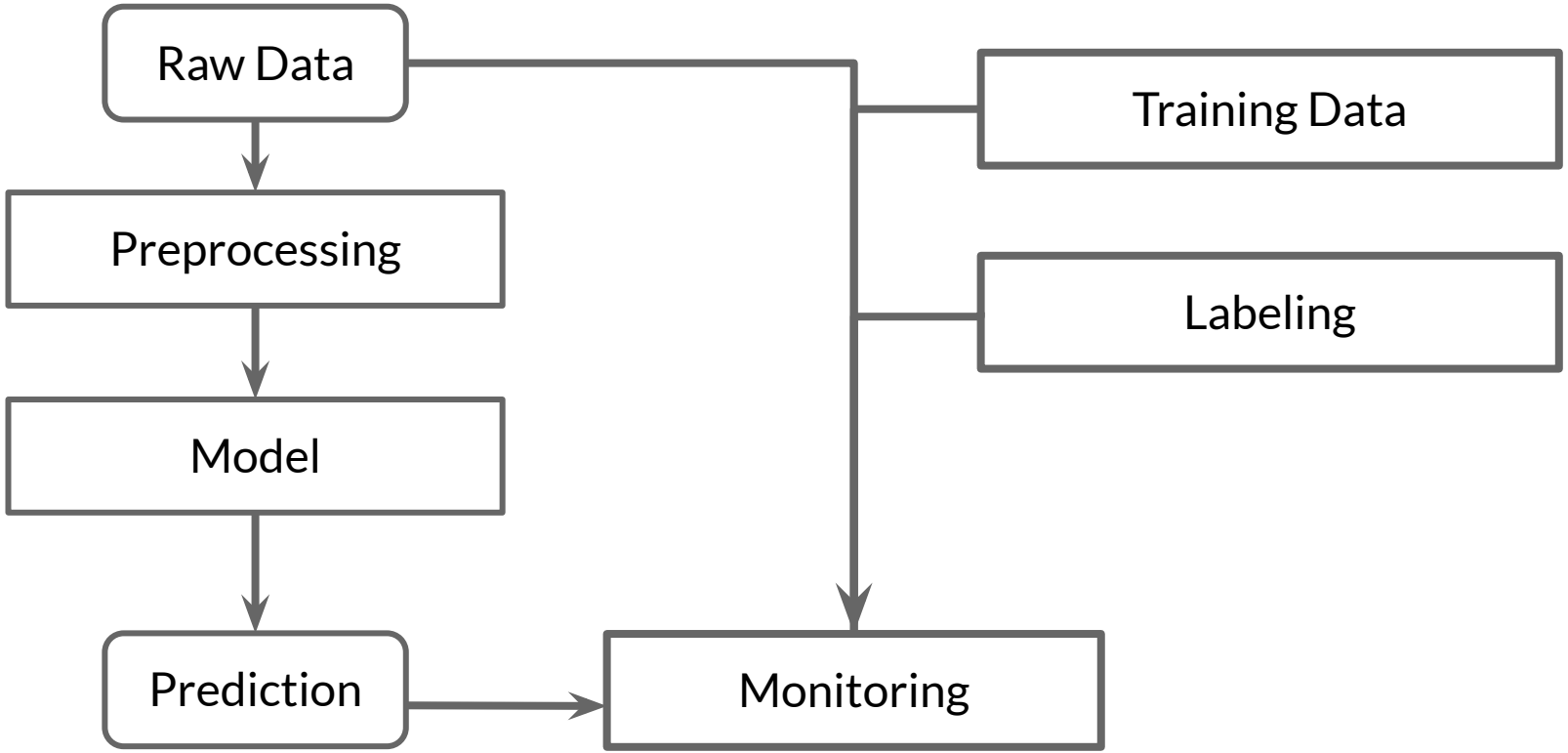
Why do models need to be monitored?

- Training data is a snapshot of the world at a point in time
- Many types of data change over time, some quickly
- ML Models do not get better with age
- As model performance degrades, you want an early warning

Data drift and shift

- Concept drift: loss of prediction quality
- Concept Emergence: new type of data distribution
- Types of dataset shift:
  - covariate shift
  - prior probability shift

How are models monitored?



Statistical process control

Method used is drift detection method

Models number of error as binomial random variable

$$\mu = np_t \qquad \sigma = \sqrt{\frac{p_t(1-p_t)}{n}}$$

Alert rule

$$p_t + \sigma_t \geq p_{min} + 3\sigma_{min}$$

Sequential analysis

Method used is linear four rates

If data is stationary, contingency table should remain constant

$$P_{npv} = \frac{TN}{TN+FN} \quad P_{precision} = \frac{TP}{TP+FP} \quad P_{recall} = \frac{TP}{TP+FN} \quad P_{specificity} = \frac{TN}{TN+FP}$$

$$P_*^t \leftarrow \eta_* P_*^{t-1} + (1-\eta_*) I_{y_t=\hat{y}_t}$$

Error distribution monitoring

Method used is Adaptive Windowing (ADWIN)

Calculate mean error rate at every window of data

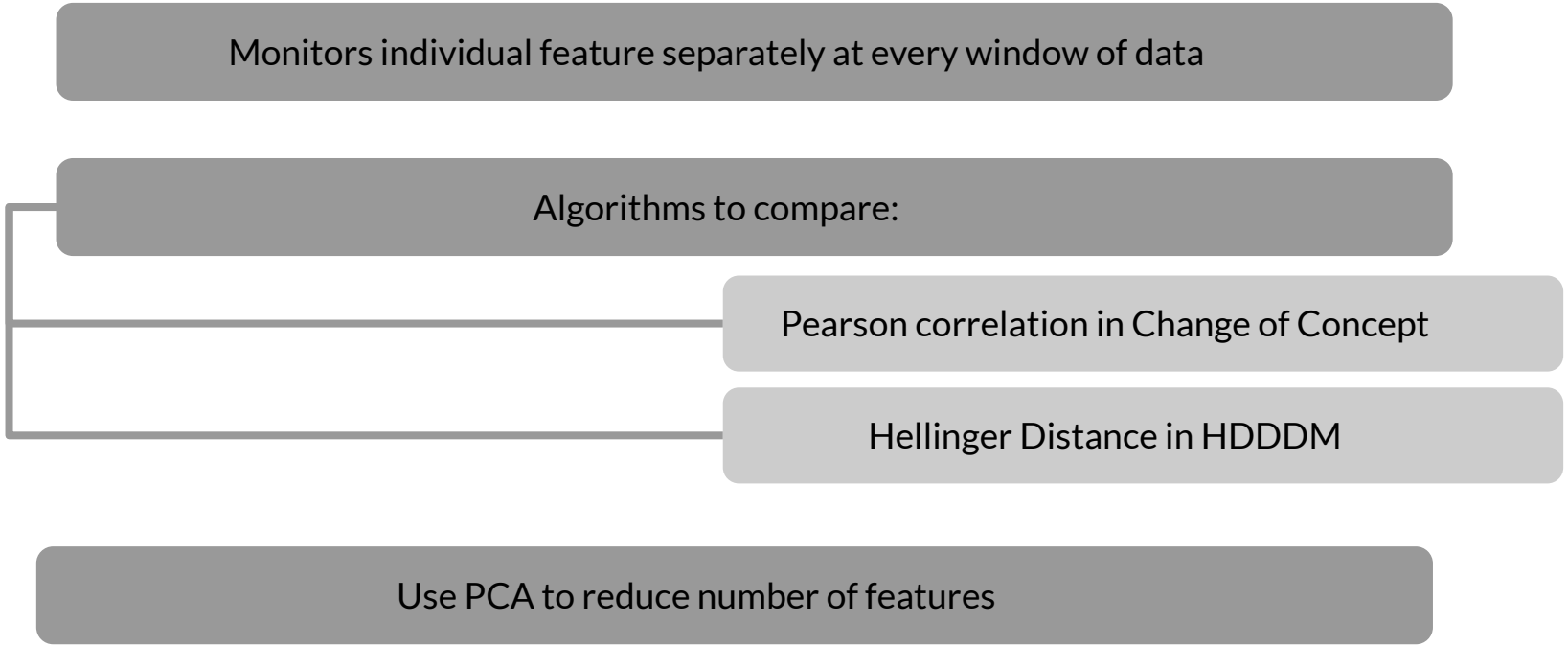
Size of window adapts, becoming shorter when data is not stationary

$$|\mu_0 - \mu_1| > \Theta_{Hoeffding}$$

Clustering/novelty detection

- Assign data to known cluster or detect emerging concept
- Multiple algorithms available: OLINDDA, MINAS, ECSMiner, and GC3
- Susceptible to curse of dimensionality
- Does not detect population level changes

Feature distribution monitoring



Model-dependent monitoring

- Concentrate efforts near decision margin in latent space
- One algorithm is Margin Density Drift Detection (MD3)
- Area in latent space where classifiers have low confidence matter more
- Reduces false alarm rate effectively

Google Cloud AI Continuous Evaluation

- Leverages AI Platform Prediction and Data Labeling services
- Deploy your model to AI Platform Prediction with model version
- Create evaluation job
- Input and output are saved in BigQuery table
- Run evaluation job on few of these samples
- View the evaluation metrics on Google Cloud console

How often should you retrain?

- Depends on the rate of change
- If possible, automate the management of detecting model drift and triggering model retraining

How often should you retrain?