Pytorch Lightning
Model Monitoring
Hyper-parameter optimization

Pytorch Lightning

Clean and Modular Code

PyTorch Lightning enforces a standard, modular structure:

- forward() → defines the model's forward pass.
- training_step() → defines one training iteration.
- validation_step() → defines validation logic.
- configure_optimizers() → sets up optimizers/schedulers.

This separation keeps research logic (model) and engineering logic (training loop) cleanly decoupled.

https://lightning.ai/lightning-ai/studios/image-segmentation-with-pytorch-lightning

Lightning handles the repetitive “boilerplate” code

- Training and validation loops
- Epoch and batch management
- Logging and checkpointing
- GPU/TPU/multi-node setup

Lightning Module and Callbacks

https://lightning.ai/docs/pytorch/LTS/common/lightning_module.html

https://lightning.ai/docs/pytorch/stable/extensions/callbacks.html
https://lightning.ai/docs/pytorch/stable/api_references.html#callbacks

Multi-GPU and TPU Training

```
trainer = pl.Trainer(accelerator="gpu", devices=4)
```

Built-in Logging and Checkpointing

More on that later

Reproducibility and Debugging Tools

Provides utilities for deterministic training and seed control.

Trainer(fast_dev_run=True) allows you to **debug your pipeline quickly** using just one batch.

PyTorch Lightning = “PyTorch, but cleaner, faster, and easier to scale.”

Model monitoring

When do we monitor DL models? (In what stages of life-cycle?)

When do we monitor DL models? (In what stages of life-cycle?)

- Training
- Inference

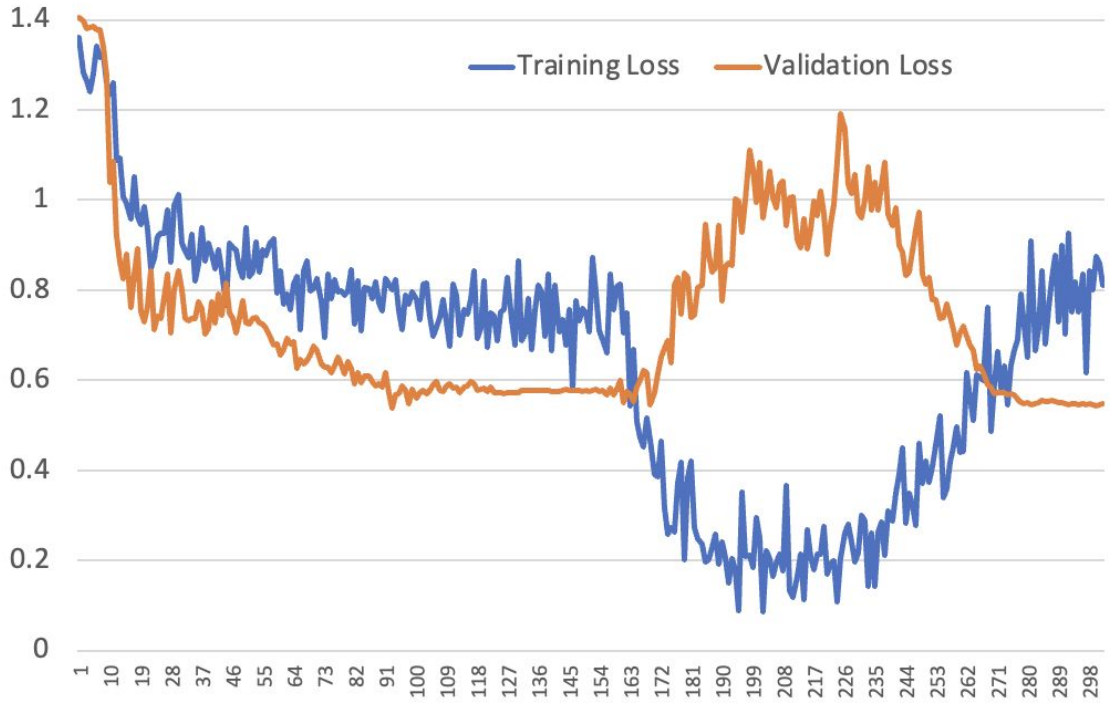
Model training

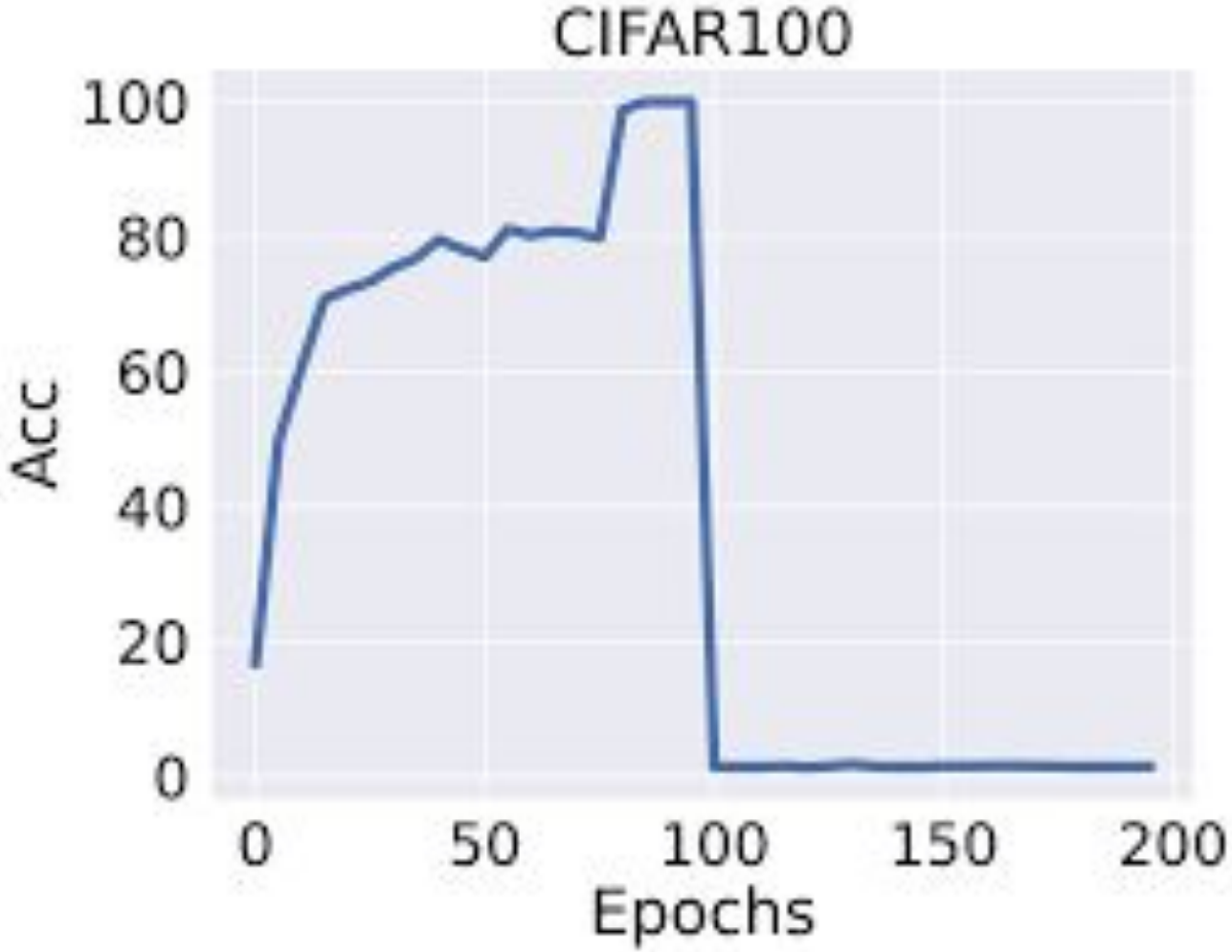
What and why can we monitor during DL training?

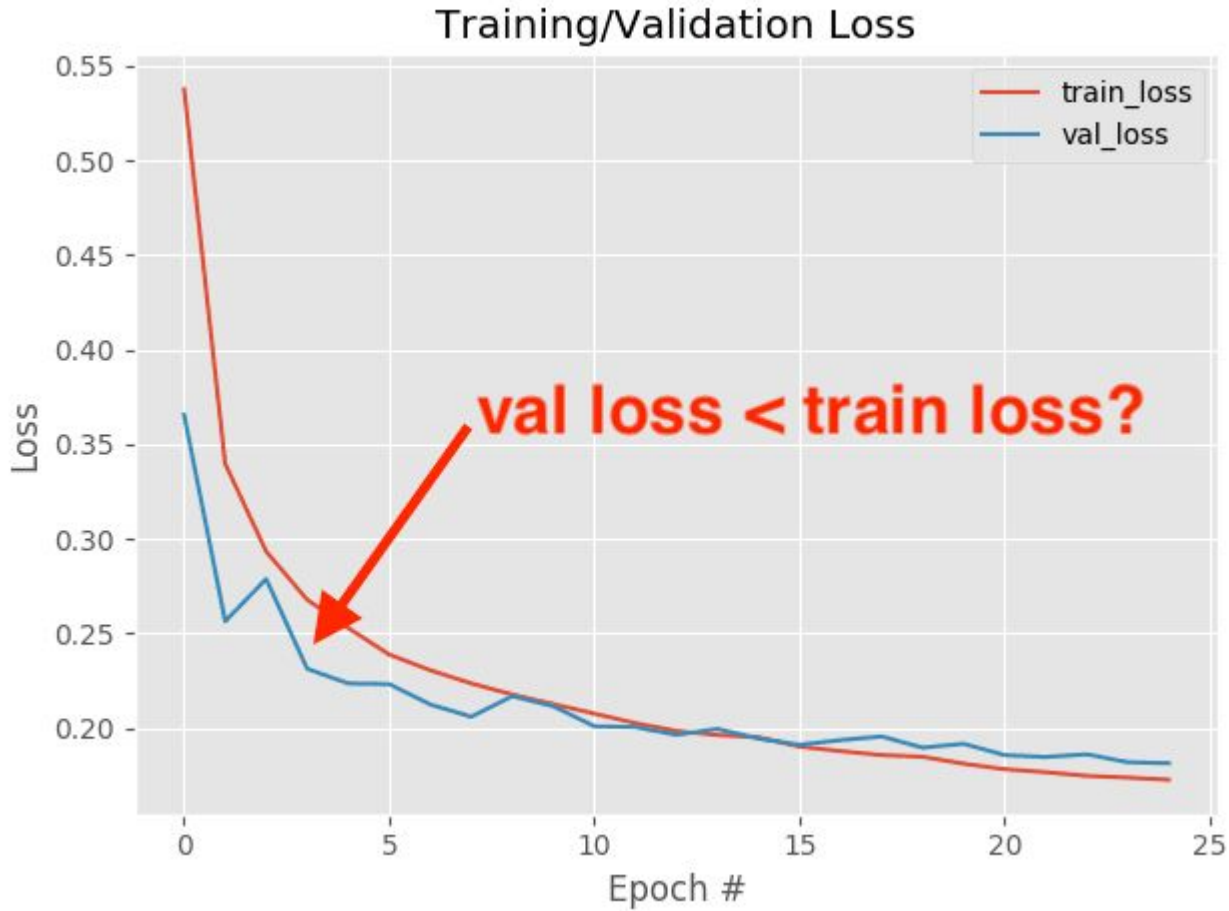
-

What and why can we monitor during training?

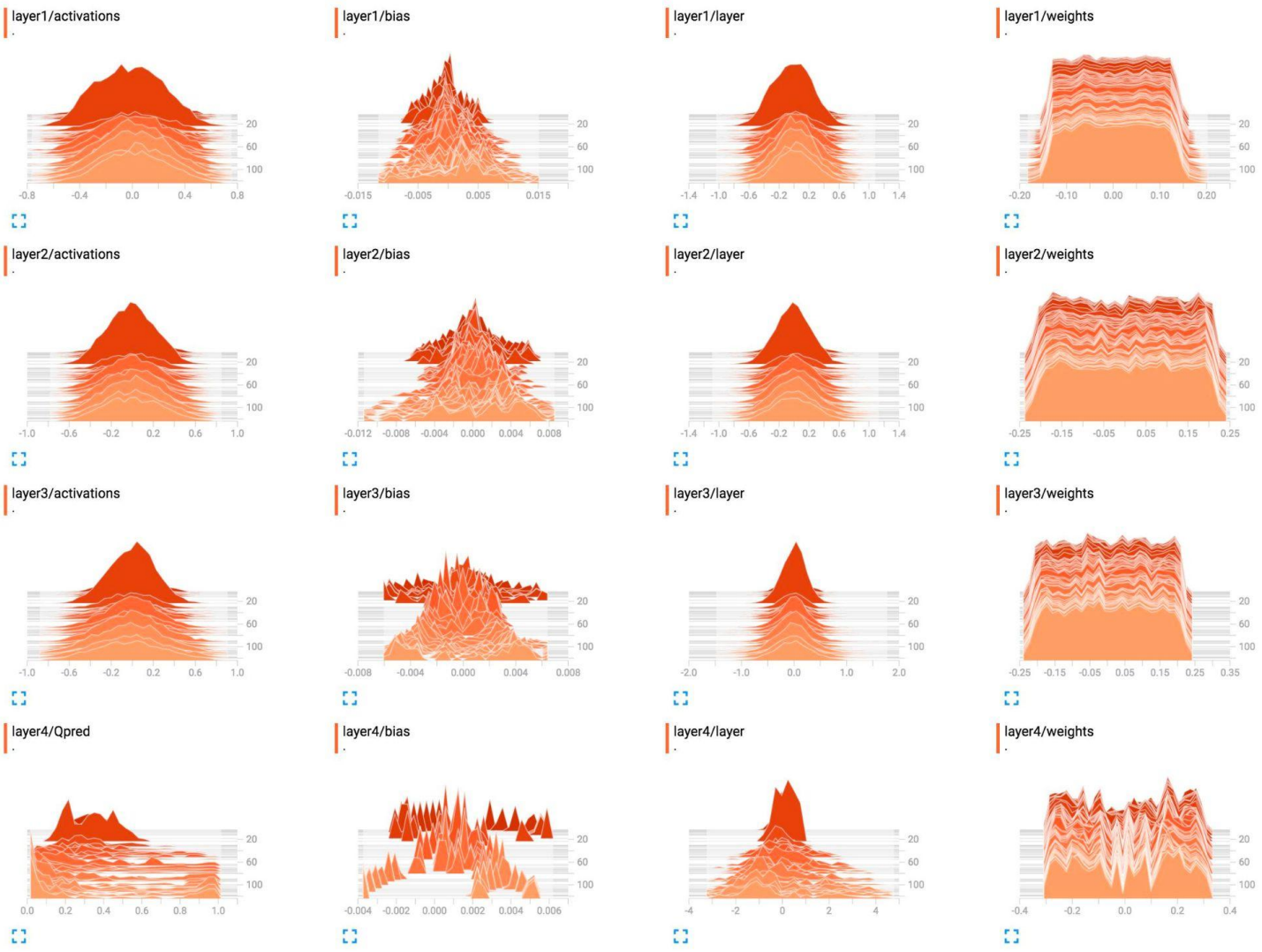
- Training and validation **loss** -> Is model learning?
- Training and validation **metrics**
- Number of **batches** (steps) and consequently **epochs**
- Log **hyper-parameters**:
  - Monitor the one that change: **Learning rate** / Learning rates
- **Hardware** usage
- I encourage you to **visualize sample** predictions!!
- At the end, all the same but for test set.


Training Loss
Validation Loss
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
10
19
28
37
46
55
64
73
82
91
100
109
118
127
136
145
154
163
172
181
190
199
208
217
226
235
244
253
262
271
280
289
298


CIFAR100
100
80
60
40
20
0
Acc
0
50
100
150
200
Epochs


Training/Validation Loss
train_loss
val_loss
val loss < train loss?
Loss
Epoch #
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0
5
10
15
20
25

TensorBoard



Examples of tools used for model monitoring

Weights & Biases

mlflow

neptune.ai

Using model training monitoring tools

https://docs.wandb.ai/models/tutorials/pytorch

https://docs.wandb.ai/models/integrations/lightning

Live example of results

Model inference

Why do we have to monitor model after deployment?

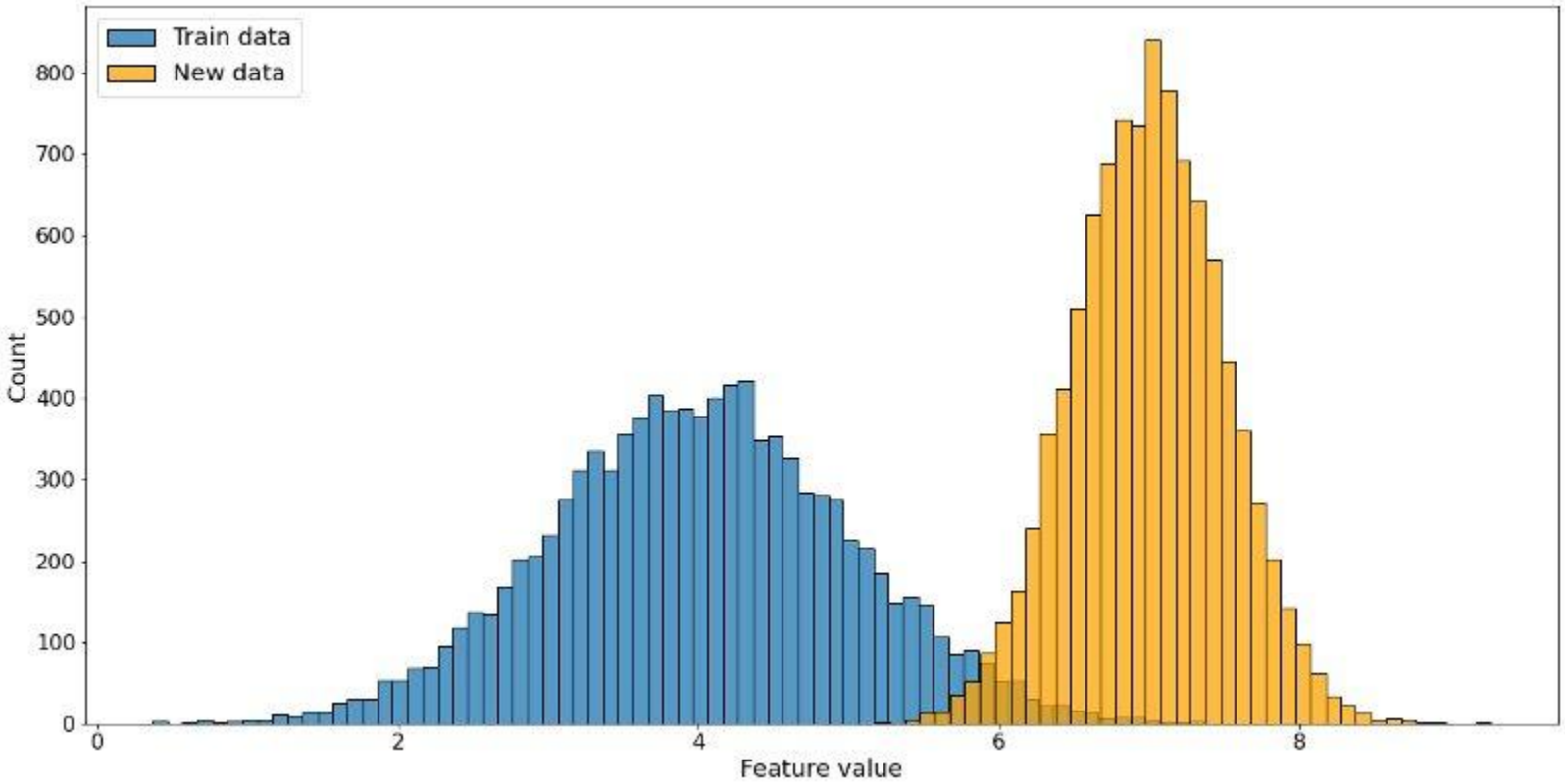
What should we monitor?

Why do we have to monitor model after deployment?

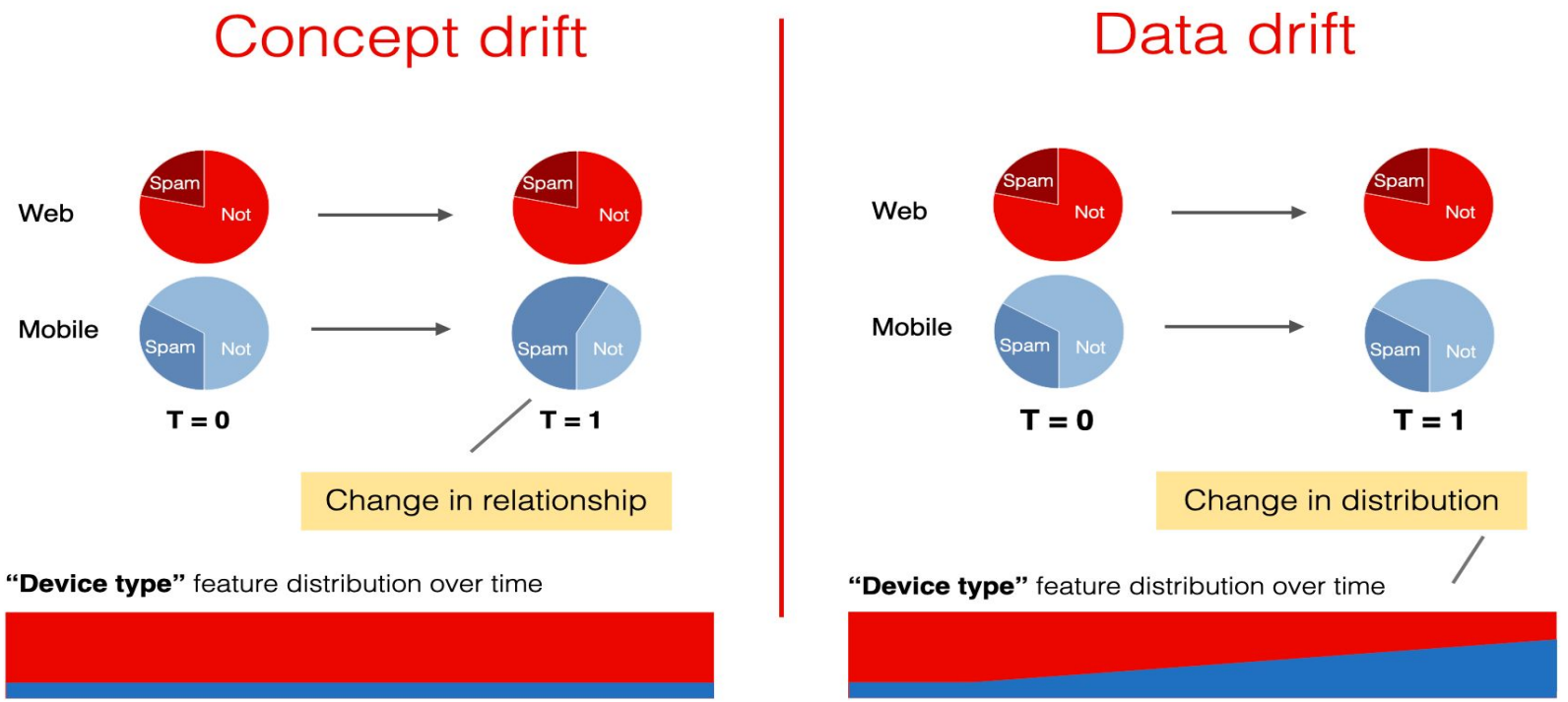
What should we monitor?

- Distribution of **input**
- Distribution of **outputs**

Data drift


Train data
New data
Count
Feature value
0
100
200
300
400
500
600
700
800
0
2
4
6
8

Concept drift



https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-model-monitor-feature-attribution-drift.html

Feature Attribution Drift

College Admission Hypothetical Scenario

Feature	Attribution in training data	Attribution in live data
SAT score	0.70	0.10
GPA	0.50	0.20
Class rank	0.05	0.70

Feature Attribution Metrics

SHAP (SHapley Additive exPlanations)

LIME (Local Interpretable Model-agnostic Explanations)

Integrated Gradients (for deep models)

Option A — Rank correlation

Compute **Spearman's rank correlation coefficient** between the training and live feature importance rankings.

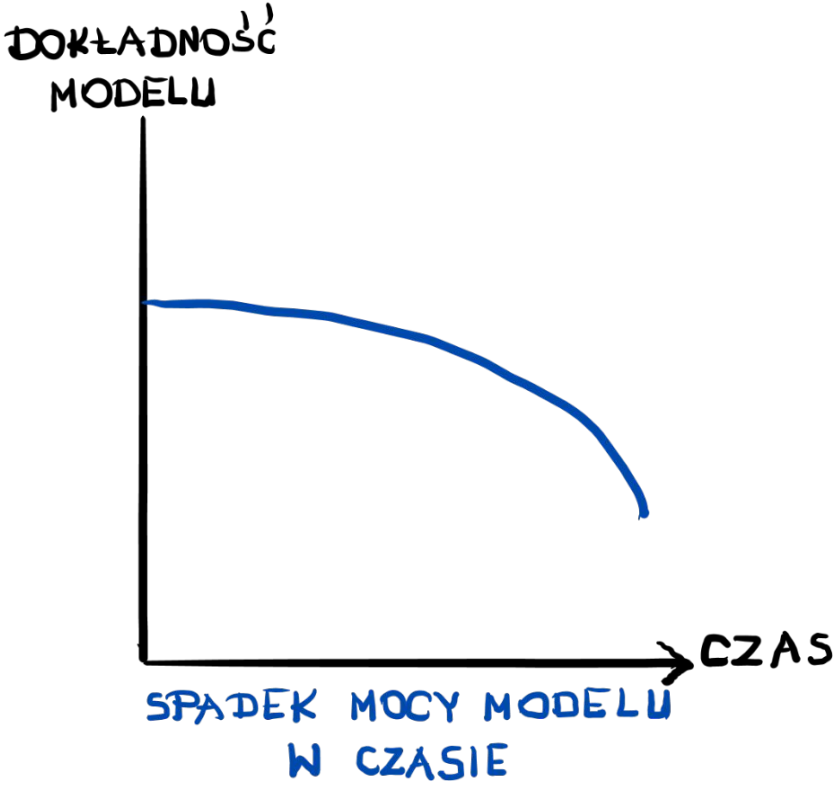
python

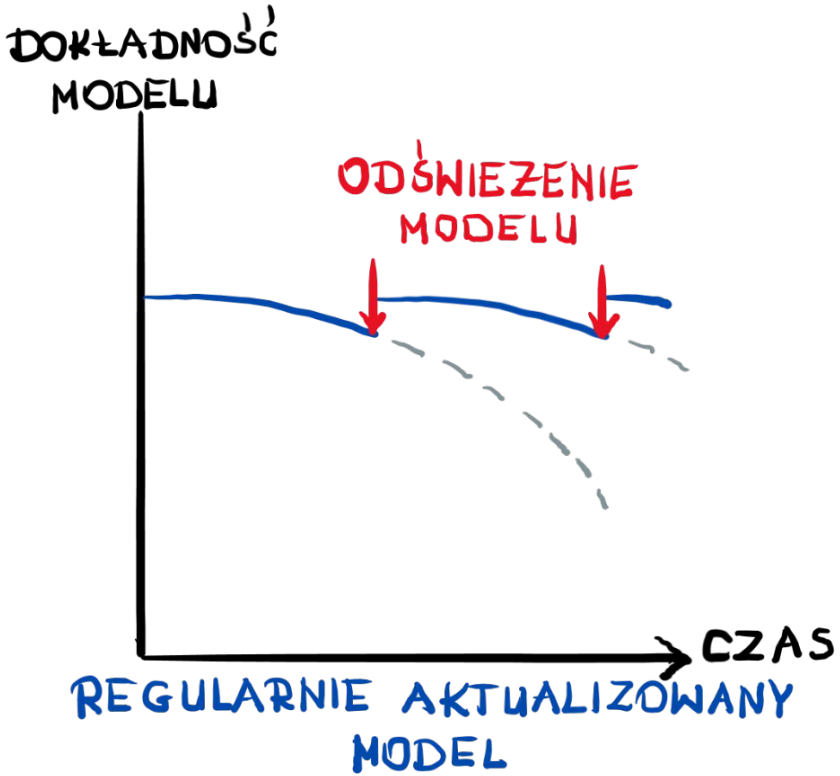
Copy code

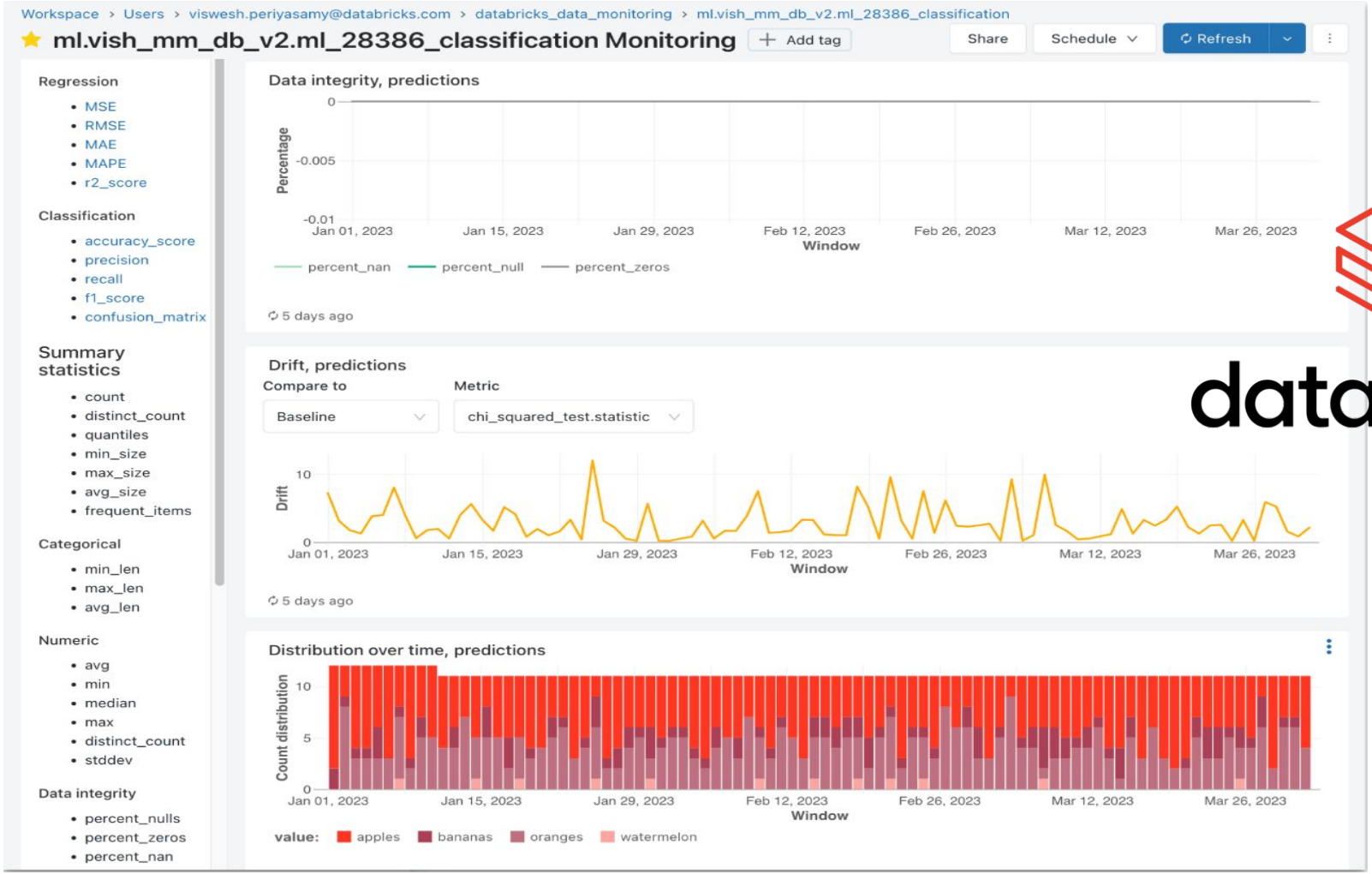
```python
import pandas as pd
from scipy.stats import spearmanr

train = pd.Series({'SAT': 0.70, 'GPA': 0.50, 'Rank': 0.05})
live = pd.Series({'SAT': 0.10, 'GPA': 0.20, 'Rank': 0.70})

corr, _ = spearmanr(train.rank(), live.rank())
print(f"Spearman correlation: {corr:.2f}")
```

What to do if the model is failing?

Model drift = Concept Drift + Data Drift






Workspace › Users › viswesh.periyasamy@databricks.com › databricks_data_monitoring › ml.vish_mm_db_v2.ml_28386_classification
ml.vish_mm_db_v2.ml_28386_classification Monitoring
Add tag
Share
Schedule
Refresh
Regression
MSE
RMSE
MAE
MAPE
r2_score
Classification
accuracy_score
precision
recall
f1_score
confusion_matrix
Summary statistics
count
distinct_count
quantiles
min_size
max_size
avg_size
frequent_items
Categorical
min_len
max_len
avg_len
Numeric
avg
min
median
max
distinct_count
stddev
Data integrity
percent_nulls
percent_zeros
percent_nan
Data integrity, predictions
Percentage
0
-0.005
-0.01
Jan 01, 2023
Jan 15, 2023
Jan 29, 2023
Feb 12, 2023
Feb 26, 2023
Mar 12, 2023
Mar 26, 2023
Window
percent_nan
percent_null
percent_zeros
5 days ago
Drift, predictions
Compare to
Baseline
Metric
chi_squared_test.statistic
Drift
10
0
Window
5 days ago
Distribution over time, predictions
Count distribution
10
5
0
Window
value:
apples
bananas
oranges
watermelon

https://github.com/aws-samples/amazon-sagemaker-drift-detection

https://aws.amazon.com/blogs/machine-learning/bring-your-own-container-to-project-model-accuracy-drift-with-amazon-sagemaker-model-monitor/

Hyper-parameters

Hyper-parameter optimization

- What kind of hyper-parameters there are?
- And why do we want to know them?
- And why do we want to observe them in subsequent training runs?

Hyper-parameters

- Learning rate
- Optimizers and their parameters, schedulers and their parameters
- Transformations and their parameters
- Architectures type, activation functions, number of layers, freezing, backbone types, head types
- Loss functions and their parameters
- Training Procedure: Number of epochs, gradien accumulation
- Presence and type of regularization
- Let's look at my Wandb for more examples

Hyper-parameter optimization approaches

- Grid search (everything with everything)
- Random search
- Bayesian optimization
- Tree-structured Parzen Estimator
- Genetic: For Optuna: NSGA-II

Tools for hyper-parameter optimization

https://github.com/optuna/optuna-examples/blob/main/pytorch/pytorch_lightning_simple.py

Thank you for attention

- Why use it?
  - *“PyTorch Lightning is just organized PyTorch - Lightning disentangles PyTorch code to decouple the science from the engineering.”*
  - https://lightning.ai/lightning-ai/studios/image-segmentation-with-pytorch-lightning
  - https://lightning.ai/docs/pytorch/stable/api_references.html#callbacks
  - https://lightning.ai/docs/pytorch/stable/notebooks/course_UvA-DL/07-deep-energy-based-generative-models.html#Sampling-buffer