Applied Machine Learning for Business Analytics

Lecture 9: Model Evaluation in Machine Learning

Lecturer: Zhao Rui

Logistics

- For kaggle competition, make sure your team name is student number (starting with A)
- We will keep online learning for the rest four lectures
  - 51% of you selected online learning
  - Due to covid situation
- Appreciate if you keeps video on!

Agenda

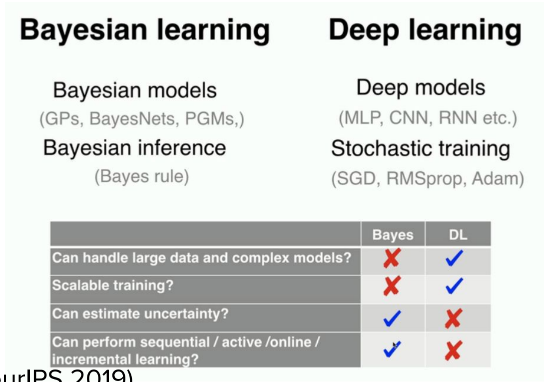
1. Baseline First
2. Model Evaluation
3. Experiment Tracking

1. Baseline First

Classical Machine Learning Algorithms

- Logistic regression
  - Very hard to beat baseline
- Naive Bayes
  - Suitable for high-dimensional data
- Tree-based models: random forest, bagging, boosting
  - XGBoost still one of the most popular winning algorithms on Kaggle
- K-nearest neighbor
  - Great for anomaly detection
- SVM
- …

Neural Networks



	Bayes	DL
Can handle large data and complex models?	✗	✓
Scalable training?	✗	✓
Can estimate uncertainty?	✓	✗
Can perform sequential / active /online / incremental learning?	✓	✗

Source: Deep Learning with Bayesian Principles (Emtiyaz Khan, NeurIPS 2019)

Architecture evolution

- Fancy models come and go
  - LSTM-RNNs: still used for time series (trading) but for text data, transformers is the first-choice

The fall of RNN and LSTM https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0

- Be solution-focused, not architecture/buzzword-focused

Model selection: baselines first

- **Random baseline**
  - Predict at random:
    - uniform
- **Zero rule baseline**
  - Predict the most common class always
- **Human baseline**
  - Human expert?
- **Simple heuristic**:
  - For example, if your device is linked to multiple accounts (10+), your account will have a high fraud risk.
- **Existing solutions**:
  - Existing APIs

Baselines

- Pave the way for iterative development
- Due to low model complexity
  - Rapid experimentation via hyperparameter tuning
  - Discover of data issues, false assumptions, bugs in ETL or code
- Build the benchmark performances
  - Slowly add complexity by addressing limitations and motivating representations and model architectures.

Case Study: Stackoverflow Classification

- Random
  - What is the random performance looks like
    - Binary Classification: np.random.randint(low=0, high=2)
  - All of our following trials should perform better than this
  - **Limitations**: no inputs information is used. No learning happened

Case Study: Stackoverflow Classification

- Random
  - No input information is used
- Rule-based
  - We would like to use signals from input data to make predictions
  - Domain knowledge and auxiliary data can be used here.
  - For example, if len(text) > 200 or code in text, the label will be positive
  - Let us guess how will the rule-based system perform?
    - High Precision low recall
    - Low Recall high recall
  - **Limitations**: Unable to generalize or capture patterns to make predictions

Case Study: Stackoverflow Classification

- Random
  - No input information is used
- Rule-based
  - Unable to generalize or capture patterns to make predictions
- Simple ML Systems
  - Representations: using TF-IDF (capture the importances of a token tc the labels)
  - Architecture: can use various classifiers to predict labels based on signals
  - **Limitations**:
    - TF-IDF is only counting tokens' frequency. We need to capture high-level semantic meaning
    - Models need to capture the meaning in a more contextual manner


```
{
  "logistic-regression": {
    "precision": 0.633369022127052,
    "recall": 0.2184154175588652,
    "f1": 0.3064204603390899
  },
  "k-nearest-neighbors": {
    "precision": 0.7410281119097024,
    "recall": 0.47109207708779444,
    "f1": 0.5559182508714337
  },
  "random-forest": {
    "precision": 0.772286671216075,
    "recall": 0.38329764453961457,
    "f1": 0.4852512297132596
  },
  "gradient-boosting-machine": {
    "precision": 0.8503271303309295,
    "recall": 0.6167023554603854,
    "f1": 0.7045318461336975
  },
  "support-vector-machine": {
    "precision": 0.8938397993500261,
    "recall": 0.5460385438972163,
    "f1": 0.6527334570244009
  }
}
```

Case Study: Stackoverflow Classification

- Random
- Rule-based
- Simple ML Systems
- CNN with word embeddings

In this process, we kind of motivate the need for slowly adding complexity from both the **representation** and **architecture**, as well as address the limitation at each step of the way.

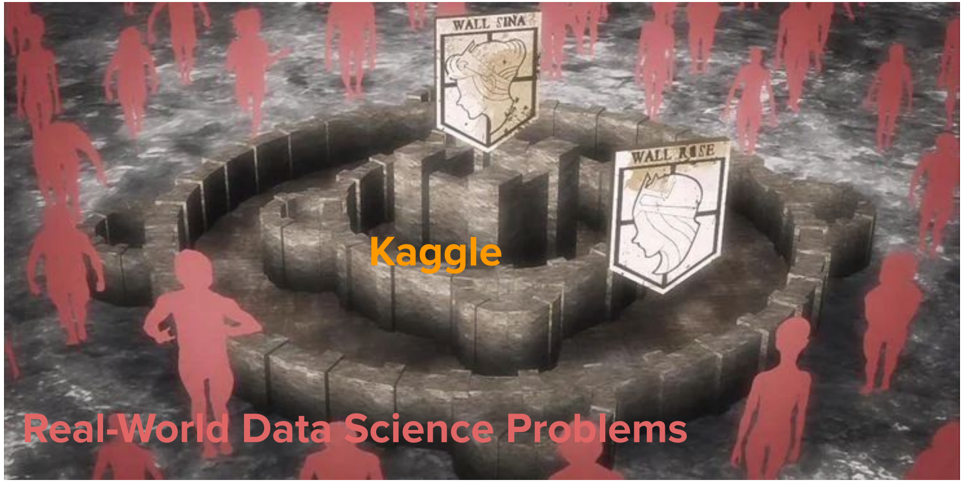
2. Model Evaluation

Model evaluation

- Offline evaluation:
  - Before deployment
  - Our focus today
- Online evaluation:
  - After deployment
  - ML model monitoring
  - https://christophergs.com/machine%20learning/2020/03/14/how-to-monitor-machine-learning-models/

ML offline evaluation

It is not simply computing the accuracy or other global metrics.



Intuition behind model evaluation

- Be clear about what metrics we are prioritizing
- Be careful not to over-optimize on any single metric
    - Trade-off is always there

Evaluation methods

1. Interpretability
2. Samples Inspection
3. Perturbation Tests
4. Directional Expectation Tests
5. Slice-based Evaluation
6. Model Calibration

Interpretability

- Interpretability methods such as LIME or SHAP can enable us to inspect the inputs to our models
- We can check:
    - Global level -> per class
    - Local level -> per single prediction

Samples Inspection

- Confusion Matrix:
  - True positives: prediction = ground-truth
    - Learn about where our model performans well
  - False positives: predict wrongly samples belongs to the class
    - Identify potentially mislabeled samples
  - False negatives: predict wrongly samples does not belongs to the class
    - Identify the model’s less performant areas to upsample later

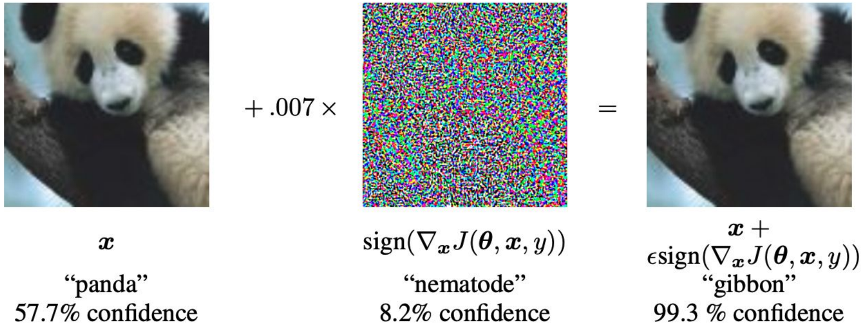
Check those FP/FN samples

Perturbation tests

- Motivation: users input might contain noise, making it different from test data
- Idea: randomly add small noise to test data to see how much outputs change

Perturbation tests

- Motivation: users input might contain noise, making it different from test data
- Idea: randomly add small noise to test data to see how much outputs change
- The more sensitive the model is to noise:
  - The harder it is to maintain
  - The more vulnerable the model is to adversarial attacks



Perturbation tests

- Motivation: users input might contain noise, making it different from test data
- Idea: randomly add small noise to test data to see how much outputs change
- If the model failed the perturbation tests, the solutions could be:
    - Add noise to training data
    - Add more training data
    - Select more robust model (simpler model)

Directional expectation tests

- Motivation: some changes to inputs should cause predictable changes in outputs
    - E.g. when predicting housing prices:
        - Increasing lot size shouldn’t decrease the predicted price
        - Decreasing square footage shouldn’t increase the predicted price

Directional expectation tests

- Motivation: some changes to inputs should cause predictable changes in outputs
- Idea: keep most features the same, but change certain features to see if outputs change predictably
- For example, if increasing lot size consistently reduces the predicted price, you might want to investigate why!

2.5 Slice-based Evaluation

Why not coarse-grained evaluation

- Overall metrics is a good start. However, it may hide:
  - Model biases
  - Potential for improvement
  - Which model will you select?

	Overall accuracy
Model A	96.2%
Model B	95%

Different performance on different slices

- Classes
  - Might perform worse on minority classes
- Subgroups
  - Gender
  - Location
  - Time of using the app
  - etc.

Fine-grained evaluation

- The date samples have:
  - Majority group: 90%
  - Minority group: 10%
- Then, which model will you chose?

	Majority accuracy	**Minority accuracy**	**Overall accuracy**
Model A	98%	80%	96.2%
Model B	95%	95%	95%

Same performance on different slices with different cost

- User churn prediction
  - Paying users are more critical
- Predicting adverse drug reactions
  - Patients with underlying conditions are more critical

Focusing on improving only overall metrics might hurt performance on subgroups

Slice-based evaluation

- Evaluate your model on different slices
  - E.g. when working with website traffic data, slice data among:
    - gender
    - mobile vs. desktop
    - browser
    - location
- Check for consistency over time
  - E.g. evaluate your model on data slices from each day

Slice-based evaluation

- Improve model's performance both overall and on critical data
- Help avoid biases
- Even when you don't think slices matter, slicing can:
  - give you confidence on your model (to convince your boss)
  - might reveal non-ML problems

Slices can be classes

- We need to check the same fine-grained metrics per class
  - As a general rule, the classes with fewer samples will have lower performance
  - We need to identify the class of data to improve the overall model performances
  - Plot the # of training samples per class vs the test performance

Source: https://madewithml.com/courses/mlops/evaluation/

How to identify slices?

- Manual Slices (based on subject matter expertise)
  - Classes
  - Features
  - Metadata
    - Timestamps, sources
  - Priority slices
    - Minority groups, high value customers

How to identify slices?

- Manual Slices (based on subject matter expertise)
- Slice finder
    - SliceLine
        - Use linear-algebra and pruning based method to find large slices that result in meaningful errors
    - Clustering

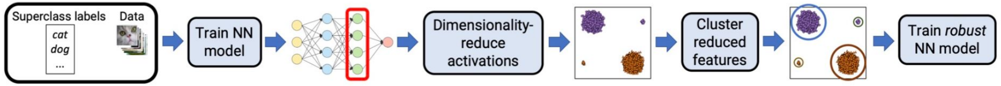
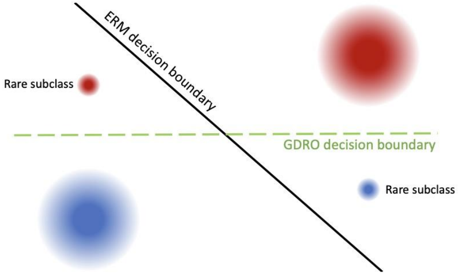


Figure 4: Schematic describing GEORGE. The inputs are the datapoints and superclass labels. First, a model is trained with ERM on the superclass classification task. The activations of the penultimate layer are then dimensionality-reduced, and clustering is applied to the resulting features to obtain estimated subclasses. Finally, a new model is trained using these clusters as groups for GDRO.



2.6 Model Calibration

Model calibration

"One of the most important tests of a forecast — I would argue that it is the single most important one — is called calibration."

Nate Silver, **The Signal and the Noise**

What is Calibration

- Assumption: the probability associated with the predicted class label should reflect its ground truth correctness likelihood
- Reality: complex models are no longer well-calibrated
  - Random Forest, SVMs, Naive Bayes, Deep Learning
- If model is well calibrated:
  - If you predict team A wins in A vs B match with 60% probability:
    - In 100 A vs. B match, A should win 60% of the time!
  - In binary classification, if the model’s predictions over 100 samples whose prob. score of positive class is 0.6
    - It means 60 samples here are positive (ground truth)

Why Calibration matters

- The high-level idea here is that with calibration, we can interpret the estimated probabilities as long-run frequencies.
- Estimated probabilities allow flexibility
- Model modularity

Model calibration: CTR

- The classifier is used to predict whether the user will click the add:
  - User A: ad 1 (20%) ad 2 (40%), ad 3 (8%), ad 4 (10%)
  - User B: ad 1 (30%) ad 2 (4%), ad 3 (80%), ad 4 (20%)
  - User C: ad 1 (15%) ad 2 (50%), ad 3 (10%), ad 4 (30%)
- Do we need to calibrate models if we want to rank ads for users (personalization)?

Model calibration: CTR

- The classifier is used to predict whether the user will click the add:
  - User A: ad 1 (20%) ad 2 (40%), ad 3 (8%), ad 4 (10%)
  - User B: ad 1 (30%) ad 2 (4%), ad 3 (80%), ad 4 (20%)
  - User C: ad 1 (15%) ad 2 (50%), ad 3 (10%), ad 4 (30%)
- Do we need to calibrate models if we want to rank ads for users (personalization)?
  - **No need to calibrate. The probability are only used for comparison.**

Model calibration: CTR

- The classifier is used to predict whether the user will click the add:
  - User A: ad 1 (20%) ad 2 (40%), ad 3 (8%), ad 4 (10%)
  - User B: ad 1 (30%) ad 2 (4%), ad 3 (80%), ad 4 (20%)
  - User C: ad 1 (15%) ad 2 (50%), ad 3 (10%), ad 4 (30%)
- Do we need to calibrate models if we want to calculate the expected number of clicks?
  - The expected clicks for ad1 is 0.2 + 0.3 + 0.15 + ...
  - The expected number can be used to estimated the revenue before we really launch the ads?

Model calibration: CTR

- The classifier is used to predict whether the user will click the add:
  - User A: ad 1 (20%) ad 2 (40%), ad 3 (8%), ad 4 (10%)
  - User B: ad 1 (30%) ad 2 (4%), ad 3 (80%), ad 4 (20%)
  - User C: ad 1 (15%) ad 2 (50%), ad 3 (10%), ad 4 (30%)
- Do we need to calibrate models if we want to calculate the expected number of clicks?
  - The expected clicks for ad1 is 0.2 + 0.3 + 0.15 + ...
  - The expected number can be used to estimated the revenue before we really launch the ads?
  - We need calibrated probabilities to estimate the expected number of clicks

Allow flexibility

Model calibration: Email Spam Detection

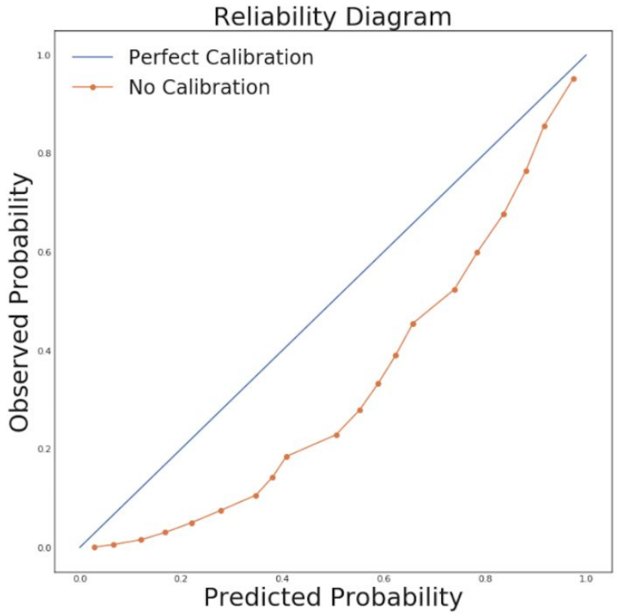
- In complex machine learning systems, models’ prob. scores are used as inputs to other machine learning models.
  - Email spam detection
    - Model A predicts the importance of the email and feed the prob(important) as the feature to the model b to predict the spam
  - Do we need to calibrate model A?

Model calibration: Email Spam Detection

- In complex machine learning systems, models’ prob. scores are used as inputs to other machine learning models.
  - Email spam detection
    - Model A predicts the importance of the email and feed the prob(important) as the feature to the model b to predict the spam
  - Do we need to calibrate model A?
  - We need calibrated probabilities
    - If the model a is miscalibrated and starts assigning too high of prob. Score for emails being important, the model b will under-predict spam

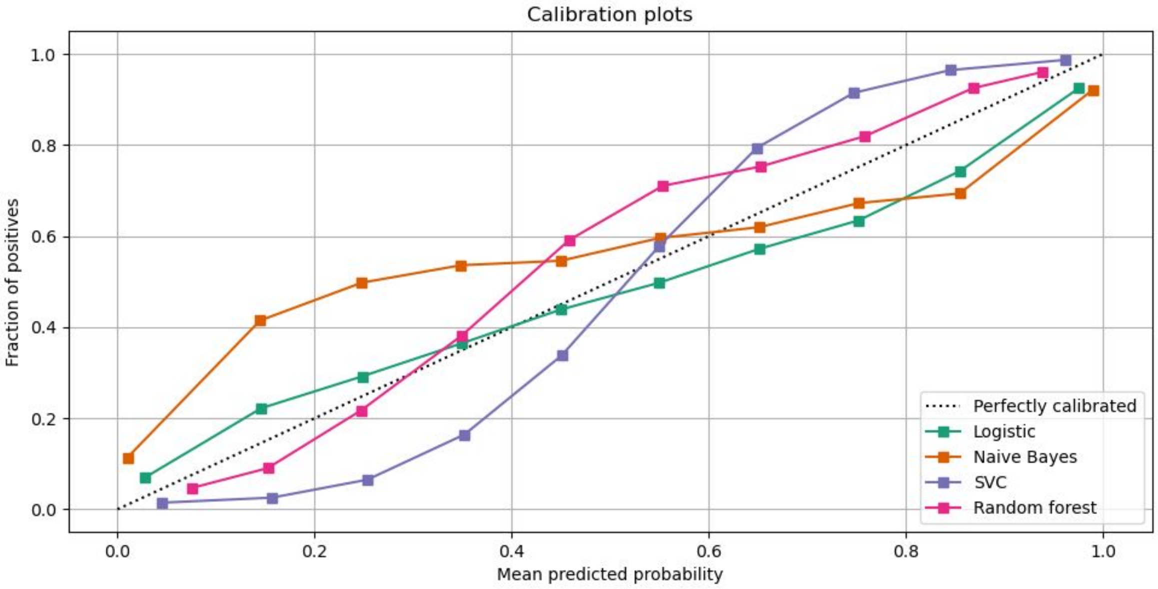
Reliability Plot

- Plot predicted probability against your empirical probability for some quantity buckets of the data



Tutorial:
https://www.youtube.com/watch?v=hWb-MIXKe-s

Case



Source: https://scikit-learn.org/stable/modules/calibration.html

Which machine learning model is the best calibrated one?

Calibration Methods

- View the classifier as a black-box and learn a calibration function which transforms your prob. output to be calibrated
  - Do you remember some previous methods we discussed?
- Different approaches for the calibration function:
  - Platt's scaling (Sklearn)
    - sklearn.calibration.CalibratedClassifierCV
    - https://github.com/gpleiss/temperature_scaling
  - Isotonic Regression (Sklearn)
  - Tensorflow Lattices

3. Experiment Tracking

Experiment Tracking

- In the life cycle of machine learning, we will train and evaluate tons of different machine learning models (representations, architectures, and hyperparameters)
- Experiment Tracking is the process to manage all experiments and their meta-data
  - Parameters
  - Metrics
  - Models
  - Other Artifacts

Experiment Tracking

- In the life cycle of machine learning, we will train and evaluate tons of different machine learning models (representations, architectures, and hyperparameters)
- Experiment Tracking is the process to manage all experiments and their meta-data
- With tracking, we can
  - Organize all the necessary components of a specific experiments
    - Where is my phone?
  - Reproduce past results using saved experiments
  - Log iterative improvements across time, data, ideas, teams, etc

Before Tracking Tools

TABLE II

CLASSIFICATION ACCURACIES (%) FOR COMPARED METHODS ON THE WHOLE FIVE ADOPTED DATASETS. BOLD FACE INDICATES HIGHEST ACCURACIES

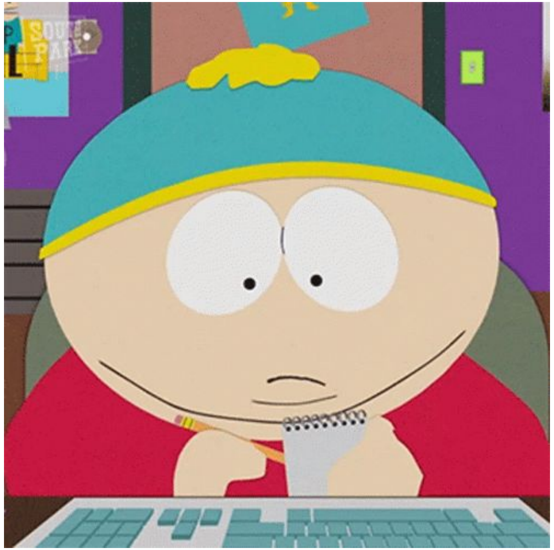
Category	Method	Datasets				
		MR	Subj	CR	MPQA	TREC
Text Classification Models	NBSVM	79.4	93.2	81.8	86.3	-
	MNB	79.0	93.6	80.0	86.3	-
	G-Dropout	79.0	93.4	82.1	86.1	-
	F-Dropout	79.1	93.6	81.9	86.3	-
CNN and its Variants	CNN	81.3	93.5	83.9	89.4	93.0
	CNN_{600}	79.3	92.0	81.6	87.5	91.9
	DCNN	-	-	-	-	93.0
	DSCNN	82.2	93.2	-	-	**95.6**
	P.V.	75.9	92.2	77.9	75.4	91.5
Other Deep Compositional Models	RAE	77.7	-	-	86.4	-
	MV-RNN	79.9	-	-	-	-
	RNN	77.2	90.9	71.8	88.6	83.8
	LSTM	79.5	93.3	80.4	88.8	89.4
	GRUs	80.5	93.5	82.1	89.0	91.8
	AdaSent	**83.1**	**95.5**	**86.3**	**93.3**	91.8
Our Models	$TopCNN_{word}$	81.7	93.4	84.9	89.9	92.5
	$TopCNN_{sen}$	81.3	93.4	84.8	90.3	92.0
	$TopCNN_{word\&sen}$	82.3	94.3	85.6	91.1	93.6
	$TopCNN_{ens}$	**83.0**	95.0	**86.4**	91.8	94.1
	$TopLSTMs_{word}$	81.2	94.1	82.6	89.6	91.5
	$TopLSTMs_{sen}$	80.6	93.7	81.6	89.1	90.5
	$TopLSTMs_{word\&sen}$	80.8	94.0	82.3	89.5	91.4
	$TopLSTMs_{ens}$	81.9	94.5	82.9	90.8	91.9

Source:
https://dr.ntu.edu.sg/bitstream/10356/83235/1/Topic-Aware%20Deep%20Compositional%20Models%20for%20Sentence%20Classification.pdf

Before Tracking Tools

```
(rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)
print("Elasticnet model (alpha=%f, l1_ratio=%f):" % (alpha, l1_ratio))
print("  RMSE: %s" % rmse)
print("  MAE: %s" % mae)
print("  R2: %s" % r2)
```

```
Elasticnet model (alpha=1.500000, l1_ratio=0.900000):
  RMSE: 0.8327481314145982
  MAE: 0.6751289812215555
  R2: 0.017435513620481347
Elasticnet model (alpha=0.500000, l1_ratio=0.020000):
  RMSE: 0.7364106074415193
  MAE: 0.5673052761841408
  R2: 0.23162398391500494
Elasticnet model (alpha=0.010000, l1_ratio=0.500000):
  RMSE: 0.6778557583356976
  MAE: 0.5190564939146215
  R2: 0.3489590462840657
```



Tracking Tools

- MLFow: 100% Free and open-source
  - Used by Azure, Facebook, Databricks
- Comet ML
  - Used by Google AI, HuggingFace
- Neptune
  - Used by NewYorkers
- Weights and Biases
  - Used by Open AI

Track Experiments - After MLFlow

▸ Description Edit

Refresh | Compare | Delete | Download CSV | ↓ Start Time | All time

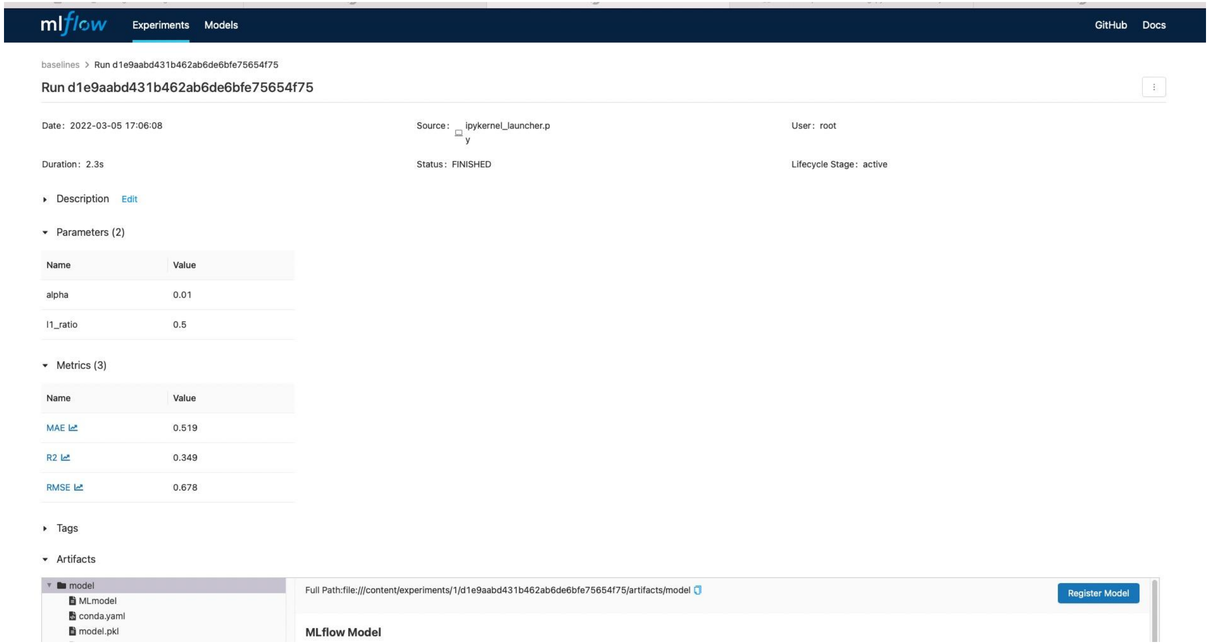
Columns | Only show differences | metrics.rmse < 1 and params.model = "tree" | Search | Filter | Clear

Showing 3 matching runs

							Metrics			Parameters	
↓ Start Time	Duration	Run Name	User	Source	Version	Models	MAE	R2	RMSE	alpha	l1_ratio
36 minutes ago	2.3s	-	root	ipykernel_	-	sklearn	0.519	0.349	0.678	0.01	0.5
36 minutes ago	2.2s	-	root	ipykernel_	-	sklearn	0.567	0.232	0.736	0.5	0.02
36 minutes ago	2.4s	-	root	ipykernel_	-	sklearn	0.675	0.017	0.833	1.5	0.9

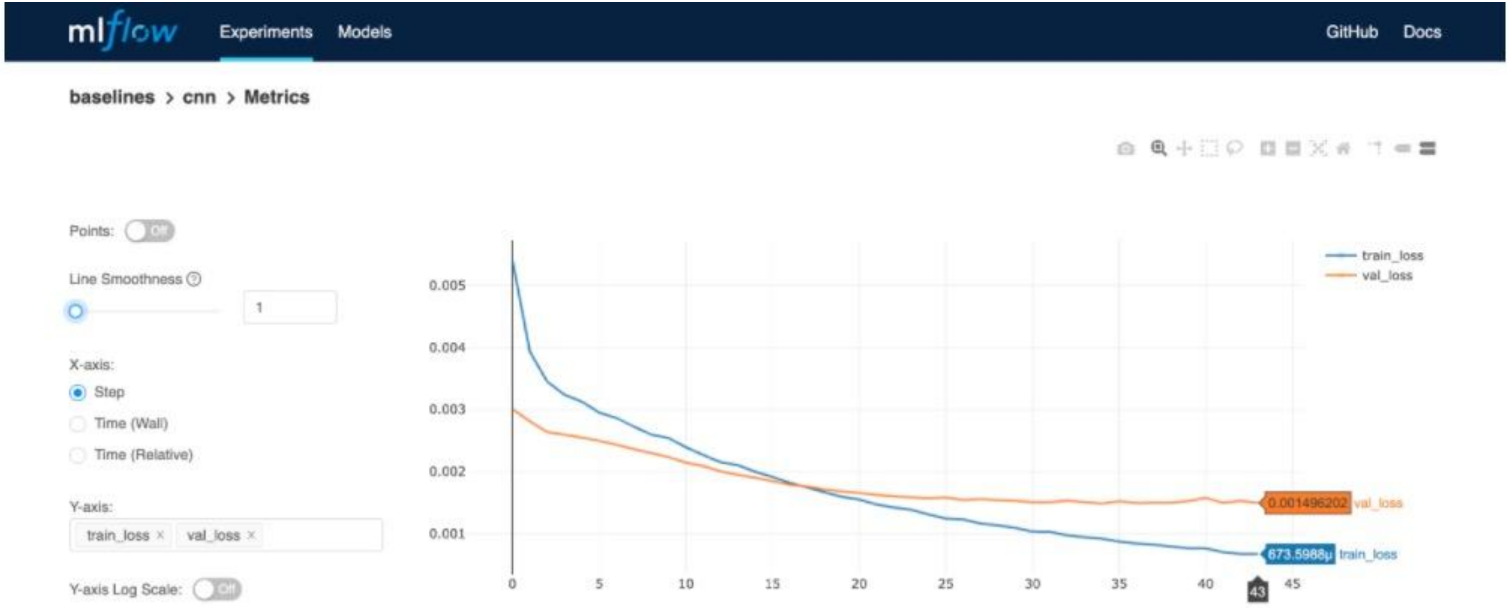
Load more

Track Experiments - After MLFlow



Track Experiments - After MLFlow

- For Deep Learning, the epoch performances can also be traced



Reproduce A Model - After MLFlow