

NYU FRE 7773 - Week 10

Machine Learning in Financial Engineering

Jacopo Tagliabue

Building Trust

Machine Learning in Financial Engineering
Jacopo Tagliabue

Part 2: Building trust

Data

Architecture

Tuning

In the life of real-world ML systems, what is the most important factor in determining the final performance?

Part 2: Building trust



Data

1. Data is the most important factor, but it is hard to automate (data change all of the time, data contains domain assumptions, data quality depends on collection best practices etc.).
2. Architectures are getting increasingly commoditized.
3. Tuning is conceptually simple, but may be expensive in practice.

Part 2: Building trust

A three steps plan:

1. To trust your model you need to trust your data -> data checks, integrity checks, etc.
2. To trust your model you need to trust your training routine -> hyper tuning, experiment tracking, quantitative objective, etc.
3. To trust your model you need to trust it in edge cases, or cases that are particularly interesting to you -> error analysis, “black-box” testing, etc.

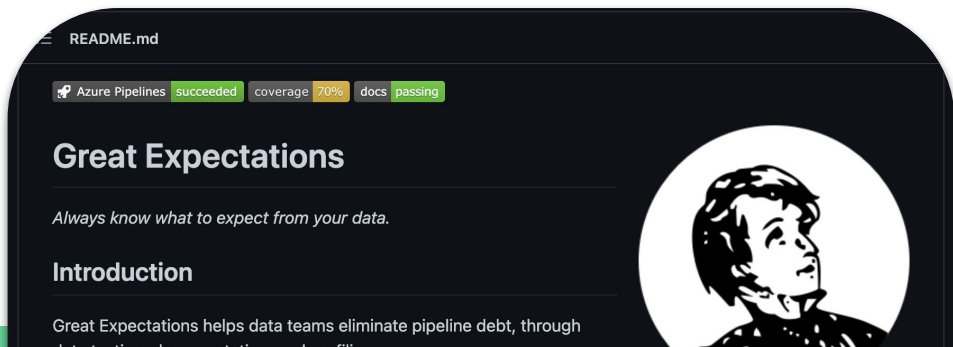
Do you trust your data?

Trusting your data

To trust your model you need to trust your data

In academic settings (and in your homeworks!) data is given to you, often prepared, cleaned and (up to a point) normalized for your analysis.

This is not what happens in the real world: data collection may be a very messy process and *before* doing ML it is important to make sure our “data expectations” hold.



Trusting your data

To trust your model you need to trust your data

Some questions we may want to ask our data:

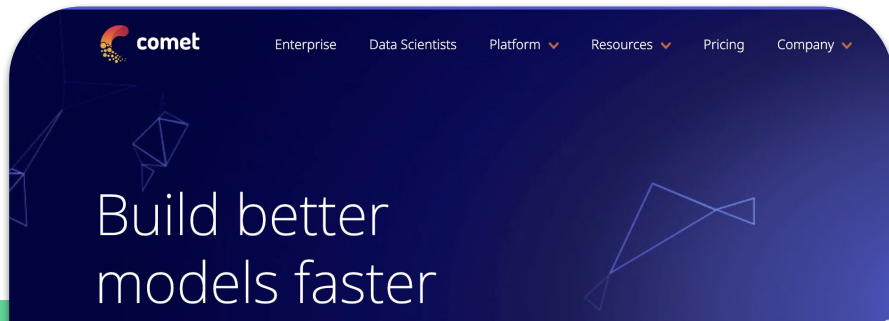
- Are there some missing values? (If yes, what do we do with it?)
- Is the dataset imbalanced? (If yes, what do we do with it?)
- Is the value range for feature X reasonable? For example, we expect an “age” column to have only positive values, up to 120.
- Is the value mean / median for feature X reasonable? For example, we expect an “IQ” column to have mean around 100, if the dataset reflects the general population.

Do you trust your training?

Trusting your training

To trust your model you need to trust your training routine

- Make sure your train, validation, test split are correct (Q: how do we split a dataset about historical stock prices?)
- Make sure to identify the relevant hyperparameters and optimize them properly: use an experiment tracking system to track and organize experiments
- Make sure to version artifacts (data, models), so that outcomes can be reproduced (Q: how do we deal with randomness?)
- Make sure the final metrics on the test set are satisfying, considering your use case.



Do you trust your evaluation?

Train vs Test: Why?

- You are now familiar with basic flow of training and evaluating a model:
 - Split dataset in train and test
 - Train model on the training set
 - Test model on the test set and report metrics (e.g. precision and recall)
- Why reporting metrics on the *test* set?

Train vs Test: Why?

- You are now familiar with basic flow of training and evaluating a model:
 - Split dataset in train and test
 - Train model on the training set
 - Test model on the test set and report metrics (e.g. precision and recall)
- Why reporting metrics on the *test* set?
 - Because the test set is our natural proxy for generalization: **how well will the model do when facing new inputs?**
 - In real life it means: how well will the model do **when we deploy it in front of customers?**
 - If the model is doing “well according to our metrics”, we *expect* it to be doing well when deployed.

Metrics: Why?

- You have encountered by now different types of metrics:
 - **Regression metrics:** for example, R^2 , MSE etc.
 - **Classification metrics:** precision, recall etc.
- What is the use of *metrics*?

$$R^2 = 1 - \frac{SS_{Regression}}{SS_{Total}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

Metrics: Why?

- You have encountered by now different types of metrics:

- **Regression metrics:** for example, R^2 , MSE etc.
- **Classification metrics:** precision, recall etc.

- What is the use of *metrics*?

- Summarize a very complex object in few numbers
- Make models comparable
 - Give us a sense of “progress”, individually and as a field

$$R^2 = 1 - \frac{SS_{Regression}}{SS_{Total}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

You can't improve
what you don't
measure

Metrics: Why?

- You have encountered by now different types of metrics:

- **Regression metrics:** for example, R^2 , MSE etc.
- **Classification metrics:** precision, recall etc.

- What is the use of *metrics*?

- Summarize a very complex object in few numbers
- Make models comparable
 - Give us a sense of “progress”, individually and as a field

- What is a good *metric*?

$$R^2 = 1 - \frac{SS_{Regression}}{SS_{Total}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

Metrics: Why?

- You have encountered by now different types of metrics:

- **Regression metrics:** for example, R^2 , MSE etc.
- **Classification metrics:** precision, recall etc.

$$R^2 = 1 - \frac{SS_{Regression}}{SS_{Total}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

- What is the use of *metrics*?

- Summarize a very complex object in few numbers
- Make models comparable
 - Give us a sense of “progress”, individually and as a field

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

- What is a good *metric*?

- Correlated with the *true phenomenon we care about*
 - When metric is improve, “my world is better”
- Hard to game
- Easy to understand

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

Metrics in Practice

- **TL;DR:** there is no a priori right / wrong metric for your project (as long as you're not using a regression metric for classification!) - even the same problem can be framed and evaluated differently depending on what we care about.
- **Example 1:**
 - Model: a binary classifier for healthcare
 - Input: a X-Ray
 - Output: cancer diagnosis
 - If I ask you to pick between reporting precision and recall, what would you do? **What is more important, to be always right or to not miss anything?**

Metrics in Practice

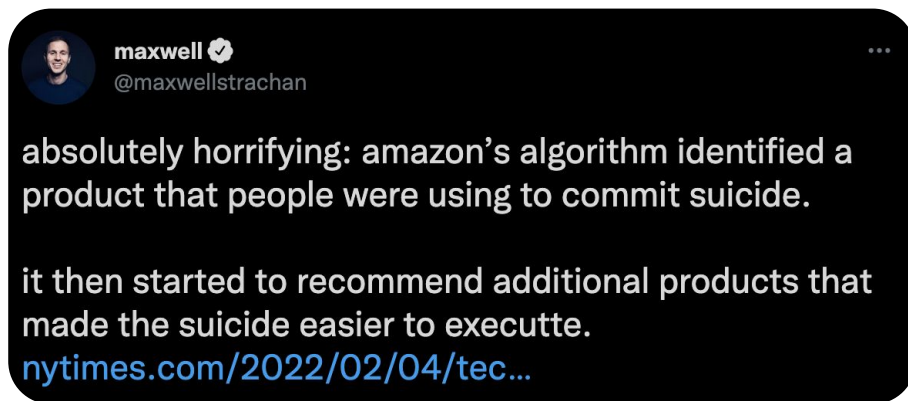
- **TL;DR:** there is no a priori right / wrong metric for your project (as long as you're not using a regression metric for classification!) - even the same problem can be framed and evaluated differently depending on what we care about.
- **Example 2:**
 - Model: a binary classifier for law enforcement (sci-fi example)
 - Input: evidence from the crime
 - Output: guilty / innocent judgement
 - If I ask you to pick between reporting precision and recall, what would you do? **What is more important, to be always right or to not miss anything?**

“Not everything that
can be counted counts
and not everything that
counts can be
counted” (A. Einstein)

Use case: recSys

Even good Recommender Systems (RS) may be bad...

- Even when RS are *mostly right*, one mistake may have disastrous consequences for:
 - the final users...
 - ...and the RS makers



Recommendations in IR

- RS evaluation is traditionally performed using standard IR metrics over the test set (i.e. held-out data points):
 - recommendations are a *ranking* problem!

Test Input



Test Truth



Model Output



Recommendations in IR

- RS evaluation is traditionally performed using standard IR metrics over the test set (i.e. held-out data points):
 - recommendations are a *ranking* problem!

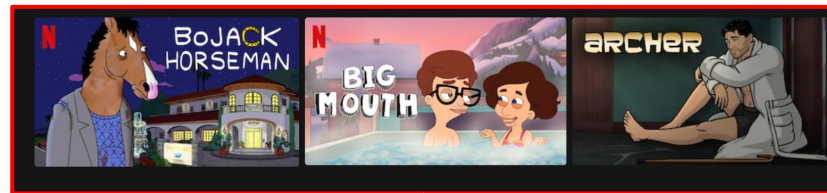
Test Input



Test Truth



Model Output



Standard metrics

- Hit Rate@k
 - Ask the model to predict k movies
 - If the target is inside the k predictions, increase hit count
 - Divide hit count by total predictions

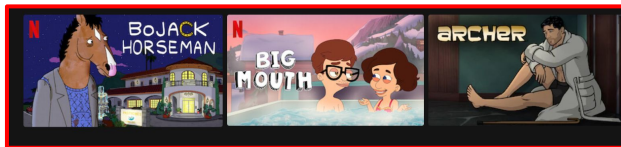
Standard metrics

- Hit Rate@k
 - Ask the model to predict k movies
 - If the target is inside the k predictions, increase hit count
 - Divide hit count by total predictions

Test Truth



Model Output



$$\text{HR@3} = 1 / 2 = 0.5$$

Q: wait a sec, does ranking matter?

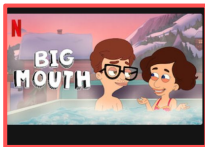
Standard metrics

- MRR

- Calculate reciprocal rank for all predictions
- Average the RR
- Intuition: position matters!

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}.$$

Test Truth



Model Output

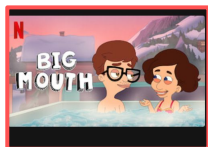


Standard metrics

- MRR
 - Calculate reciprocal rank for all predictions
 - Average the RR

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}.$$

Test Truth



Model Output



$$\text{MRR} = ((1 / 2) + 0) / 2 = 0.25$$


... and their flaws

- IR metrics won't generally considering anything above the fact that the target item is in the top-k recommendations.
 - *Sometimes recommendations are not just bad, but deeply illogical*

New jobs for CTO

Santa Monica, CA

HIGH PAYING

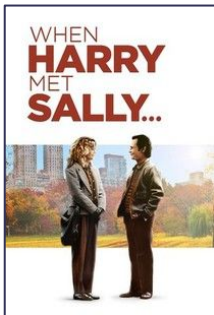
**Executive Assistant**
Recharge Payments - Los Angeles, CA
\$54K - \$112K (Glassdoor est.)
4.7 ★
Easy Apply
Highly Rated: Career Opportunity, Compensation & Benefits, Work/Life Balance, Culture & Values, Senior Leadership

See All Jobs

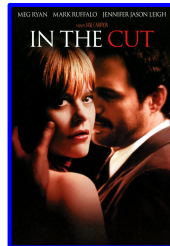
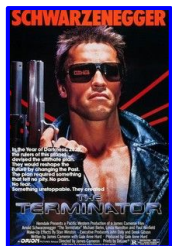
... and their flaws

- IR metrics won't generally considering anything above the fact that the target item is in the top-k recommendations.
 - Sometimes pseudo-feedback is not the whole truth: both models below have $HR@3=1!$*

Test Truth



Model #1



Model #2



What is behavioral testing?

1. Define expected output for a given (set of) input(s), *regardless of the composition of the test set*.
2. Test model compliance with the expected output, *regardless of the model inner workings*.

Beyond Accuracy: Behavioral Testing of NLP Models with CHECKLIST

Marco Tulio Ribeiro
Microsoft Research
marcotcr@microsoft.com

Tongshuang Wu
Univ. of Washington
wtshuang@cs.uw.edu

Carlos Guestrin
Univ. of Washington
guestrin@cs.uw.edu

Sameer Singh
Univ. of California, Irvine
sameer@uci.edu

Abstract

Although measuring held-out accuracy has

A number of additional evaluation approaches have been proposed, such as evaluating robust-

Examples from NLP

Sentence

Sentiment

*I am a [PROTECTED]
[NOUN]*

Template filling

I am a black women

???

I am an asian man

???

Sentence

Sentiment

*The pilot wasn't
fantastic.*

Negative

Parse in (question, "no") form

*Was the pilot
fantastic? No.*

???

Examples from NLP

Sentence

Sentiment

BERT has >90%
accuracy on sentiment
analysis dataset.

How well can it do?

I am an asian man

???

Sentence

Sentiment

*The pilot wasn't
fantastic.*

Negative

Parse in (question, "no") form

*Was the pilot
fantastic? No.*

???

Examples from NLP

Sentence

Sentiment

I am a [PROTECTED]

BERT-base fails **100%** of
the time

I am a black women

???

I am an asian man

???

Sentence

Sentiment

The pilot wasn't

Microsoft fails **96.8%** of the
time, Amazon **81.6%**,
BERT-base **55.4%**

fantastic? No.

???

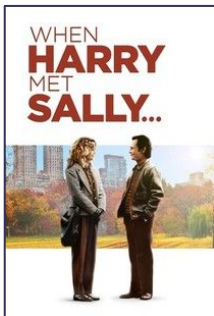
The bitter lesson from behavioral testing

- Remember what we said earlier?
 - If the model is doing “well according to our metrics”, we expect it to be doing well when deployed.
- Behavioral testing - and in general, nuanced evaluation - put our faith in many standard metrics into question: **even if we are doing “well”, the model is susceptible to many “silly” mistakes.**
 - Our datasets may not capture it: behavioral tests expand the range of use cases!
 - Our metrics may be too simplistic: a more nuanced evaluation is needed.

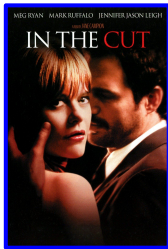
Some RecSys behavioral principles

Principle #1: some mistakes are *worse than others*.

Test Truth



Model #1



Model #2



This is a reasonable mistake!

Some RecSys behavioral principles

Principle #2: the ~~heart~~ power-law is deceitful above all things (mind the niches!)

HR: $57 / 110 = 0.52$



#: 100, H: 50, HR: 0.5



#: 10, H: 7, HR: 0.7

VS

HR: $57 / 110 = 0.52$



#: 100, H: 56, HR: 0.56



#: 10, H: 1, HR: 0.1

Evaluation recap

Beyond RecSys

- The same principles apply outside of RecSys:
 - Is your classification model for financial news **good across all news types**?
 - Is your regression model penalizing a **certain type of people**?
- **You will be asked to do “nuanced evaluation” for your final project!**

Testing checklist

- Quantitative metrics
 - What is a good metric to get an overall sense of progress? For example, Precision vs Recall in classification.
- Sliced-based metrics
 - What are important groups in my input / output distribution? How well does my model do on different groups? Why? Is it “fair”?
- Behavioral tests
 - What are some out-of-distribution, rare, qualitatively important cases I wish to test the model on? Is my model robust to perturbation in the input? Is my model being reasonable with huge outliers?

Remember: good tests help with error analysis and building trust in your model!

BONUS: document your model!

- Making your model accessible to technical and non-technical users is very important!
 - In [Github](#), we use [Streamlit](#) to showcase how artifacts from Metaflow can be made available through a UI.
- Model / DAG cards are useful ways to recap information about data, training and testing, for auditing, discussion etc.
 - Metaflow [support custom report with cards](#), which get automatically versioned every time you run the Flow!

Model Cards for Model Reporting

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, Timnit Gebru
(mmitchellai,simonewu,andrewzaldivar,parkerbarnes,lucyvasserman,benhutch,espitzer,tgebru}@google.com
deborah.raji@mail.utoronto.ca

ABSTRACT
Trained machine learning models are increasingly used to perform high-impact tasks in areas such as law enforcement, medicine, education, and employment. In order to clarify the intended use cases of machine learning models and minimize their usage in contexts for which they are not well suited, we recommend that released models be accompanied by documentation detailing their performance characteristics. In this paper, we propose a framework that we call model cards, to encourage such transparent model reporting. Model cards are short documents accompanying trained machine learning models that provide benchmarked evaluation in a variety of conditions, such as across different cultural, demographic, or phenotype groups (e.g., race, geographic location, sex, Fitzpatrick skin type [15]) and intersectional groups (e.g., age and race, or sex and Fitzpatrick skin type) that are relevant to the intended application domains. Model cards also disclose the context in which models are intended to be used, details of the performance evaluation procedures, and other relevant information. While we focus primarily on machine learning models, the framework can be extended to other domains.

KEYWORDS
datasheets, model cards, documentation, disaggregated evaluation, fairness evaluation, ML model evaluation, ethical considerations

ACM Reference Format:
Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, Timnit Gebru. 2019. Model Cards for Model Reporting. In *FAccT '19: Conference on Fairness, Accountability, and Transparency*, January 29–31, 2019, Atlanta, GA, USA. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3287566.3287596>

1 INTRODUCTION
Currently, there are no standardized documentation procedures to communicate the performance characteristics of trained machine learning (ML) and artificial intelligence (AI) models. This lack of documentation is especially problematic when models are used in applications that have serious impacts on people's lives, such as in health care [14, 42, 44], employment [1, 13, 29], education [23, 45] and law enforcement [3, 7, 20, 34].

DAG Card is the new Model Card

Jacopo Tagliabue* Coveo Labs jtagliabue@coveo.com	Ville Tunkos Outerbounds ville@outerbounds.co
Ciro Greco Coveo Labs cgreco@coveo.com	Valay Dave Outerbounds valay@outerbounds.co

Abstract
With the progressive commoditization of modeling capabilities, data-centric AI recognizes that what happens before and after training becomes crucial for real-world deployments. Following the intuition behind *Model Cards*, we propose *DAG Cards* as a form of documentation encompassing the tenets of a data-centric point of view. We argue that Machine Learning pipelines (rather than models) are the most appropriate level of documentation for many practical use cases, and we share with the community an open implementation to generate cards from code.

1 Introduction