Machine Learning in Production

Hello!

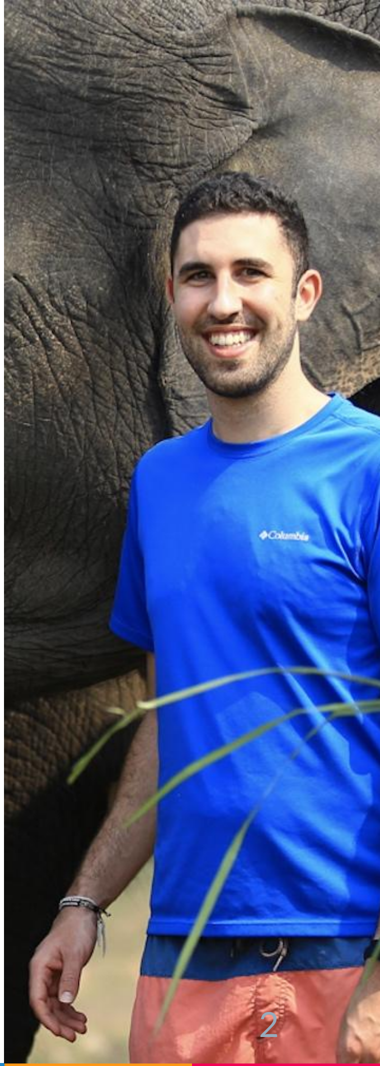
I am Albert Jiménez

Electrical Engineer @ UPC - Barcelona

4+ years experience with Computer Vision, Deep Learning and Python

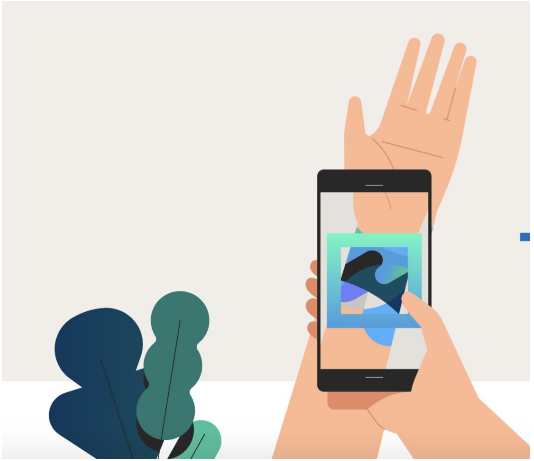
2+ years working @ Triage (Healthcare Startup)

Github: @jsalbert



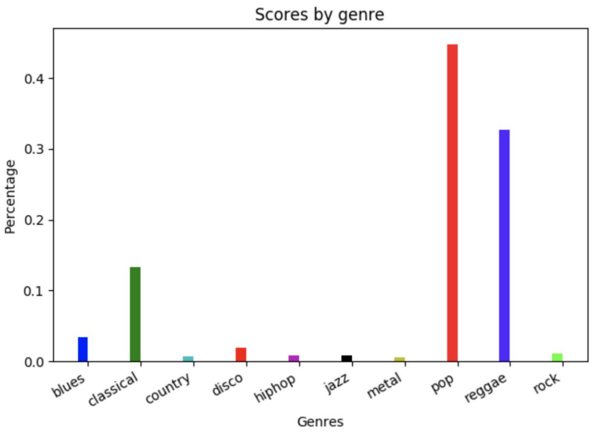
Some of the projects I've done…

Image Retrieval



Skin Lesions Recognition

Some of the projects I've done…



Music Genre Recognition

In my search of inspiration I build a robot
Hoping it would give me the perfect love song
It told me that she was the one I'd imagined
She smiled sweetly
She said that I couldn't ask for a better lover
But it was hard to love you
I recall her fondly
The day we met she led me off
As our passion blew apart
We split apart
My robot passion
She said that we were going to have to fight
I said "No you don't have to fight
I know we can't afford to lose your love"
But she said "You don't have to argue
I know you can't afford to lose your love"

It wasn't hard to love you
It wasn't hard to understand
All that was necessary to be my lover
Was my love for you

Days that I used to laugh I don't know how
They all came and fell
I miss your touch so

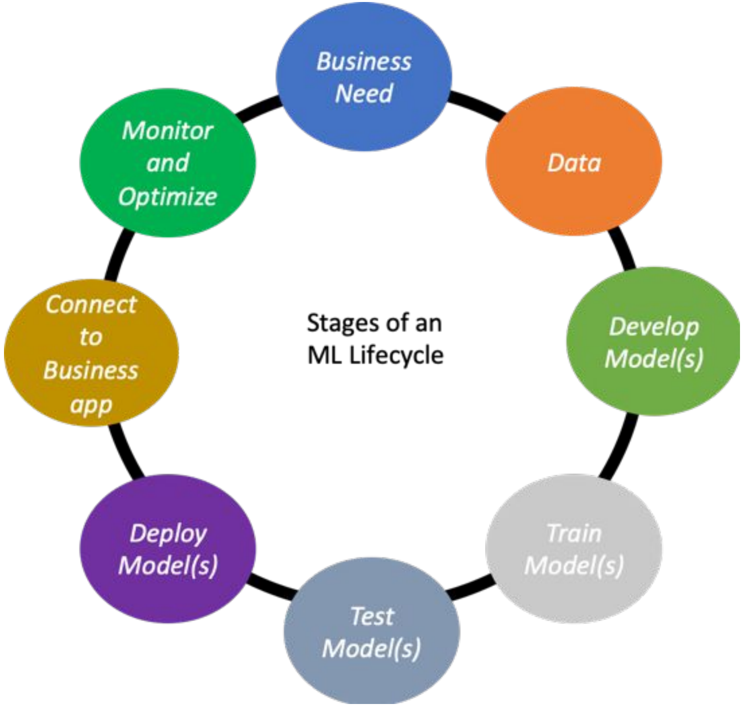
Lyrics Generation

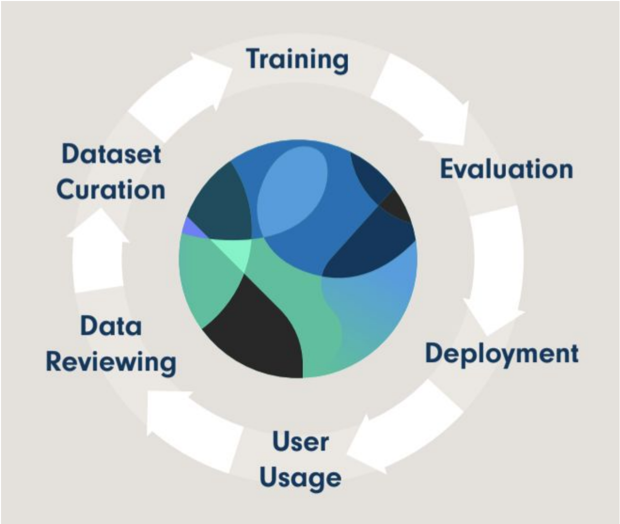
Agenda

1. Introduction
2. Best Practices for ML in Production
3. Palladium

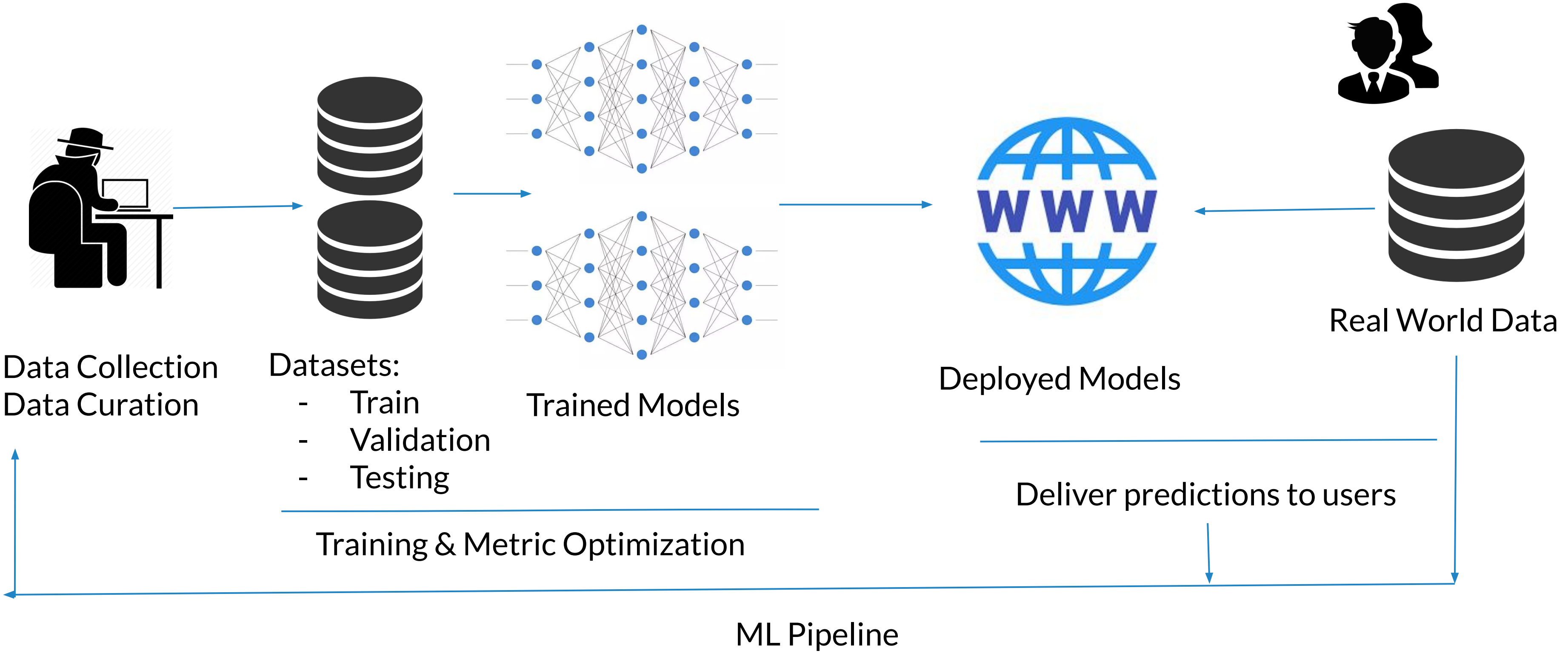
1. Introduction

ML Life Cycle





Machine Learning Pipeline



2. Best Practices for ML in Production

When starting a new product...

- Limited access to:
    - Data sources
    - Computational resources
    - User base

- Limited time to work (24h days)

- **Planning ahead and ability to iterate fast is key to success**

Best Practices of ML

To make great products:

Do machine learning like the great engineer you are, not like the great machine learning expert you aren't.

https://developers.google.com/machine-learning/guides/rules-of-ml/ by Martin Zinkevich

Basic Approach

1. Have a solid pipeline (end to end)
    a. Data Collection - Train - Test - Deployment
2. Start with a reasonable objective
    a. Choose a metric that reflects well performance
3. Add features in a simple way
4. Make sure the pipeline stays solid and allows for quick iterations

Before ML

1. Design and implement metrics

- Metrics will help you to:
  - Measure and track the performance of your system over time
  - Allow you to compare different models and make decisions
  - Indicate in which areas of your pipeline you should allocate more resources
    - For instance, data collection for particular cases

Before ML

1.Design and implement metrics

- Metric example
  - You have built a skin cancer binary image classifier and you have a testset containing:
    - 10000 cases of benign lesions (healthy moles)
    - 100 cases of melanomas (malignant skin cancer)

  - **Which metric would you choose to test your model performance?**

Before ML

1.Design and implement metrics

- Accuracy is the most used metric for image classifiers but in our case it has little utility as our dataset is very unbalanced.
- If our model classifies all the images as benign we would have an impressive accuracy of 10000/10100 = 99%

Accuracy computes the proportion of correct predictions divided by the total number of samples:

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{number of samples}}$$

Before ML

1.Design and implement metrics

- Sensitivity for the melanoma class would be our perfect metric
- For the previous case we would get a sensitivity value of 0! Which would make obvious that our model is not really doing a good job.

Sensitivity, also known as recall, computes the ratio of cases correctly identified as a particular skin condition (true positives) to the total number of images with that condition:

$$\text{Sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}} \tag{2}$$

Before ML

1.Design and implement metrics

- For the same example, which metric would you use to compare 2 different models?

Before ML

1.Design and implement metrics

<table>
<tr><td colspan="2"></td><td colspan="2">True condition</td><td colspan="3"></td></tr>
<tr><td></td><td>Total population</td><td>Condition positive</td><td>Condition negative</td><td>Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$</td><td colspan="2">Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$</td></tr>
<tr><td rowspan="2">Predicted condition</td><td>Predicted condition positive</td><td>True positive</td><td>False positive,
Type I error</td><td>Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$</td><td colspan="2">False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$</td></tr>
<tr><td>Predicted condition negative</td><td>False negative,
Type II error</td><td>True negative</td><td>False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$</td><td colspan="2">Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$</td></tr>
<tr><td colspan="2" rowspan="2"></td><td>True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$</td><td>False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$</td><td>Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$</td><td rowspan="2">Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$</td><td rowspan="2">$F_1$ score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$</td></tr>
<tr><td>False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$</td><td>Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$</td><td>Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$</td></tr>
</table>

Before ML

1.Design and implement metrics

		True condition				
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$	F_1 score $= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$		

https://en.wikipedia.org/wiki/Confusion_matrix

Before ML

1.Design and implement metrics

- Important to add them sooner than later:
    - Easier to gain user's permission early on
    - Getting historical data for possible concerning areas in the future
    - Storing metrics on database will save you to look and parse long logs
    - You will notice trends over time

Before ML

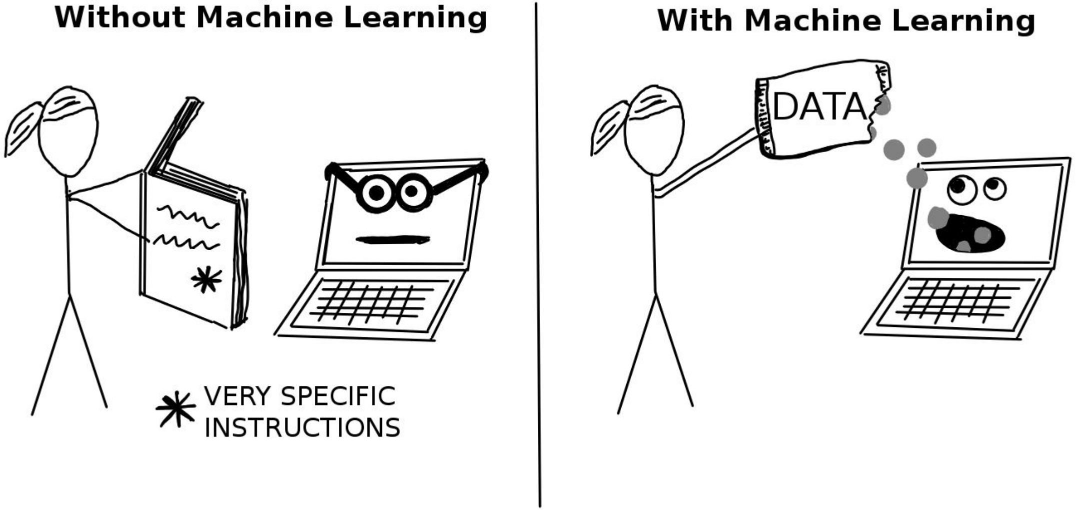
1.Design and implement metrics

- Sometimes it is very hard to find the perfect metric for a product

- Tracking a good variety of them is a good idea to see how they interoperate

Before ML

2.Don't be afraid to launch a product without machine learning

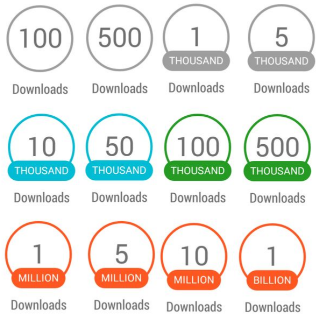


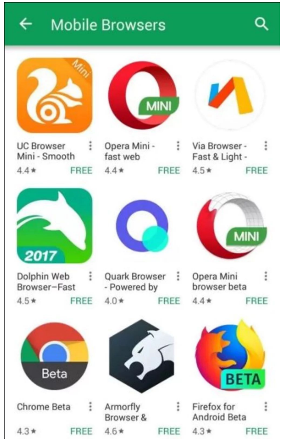
https://christophm.github.io/interpretable-ml-book/terminology.html

Before ML

2.Don't be afraid to launch a product without machine learning

- Use simple heuristics first
    - Very easy to see if correlation with your metrics exist
- Ranking / Suggesting Apps in Google Play or App Store
    - Use number of downloads
    - Use user scores



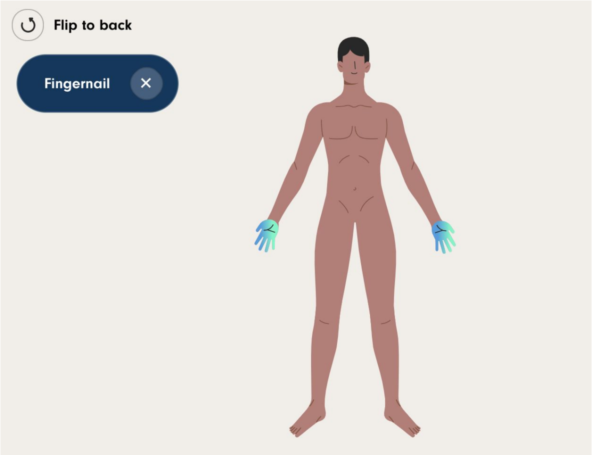


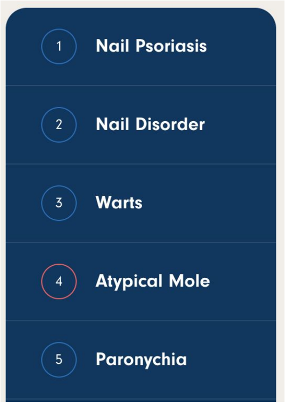
Before ML

2.Don’t be afraid to launch a product without machine learning

- Use a rule based system
    - System that by definition must work that way
- Predicting skin diseases
    - Have body locations rules per each disease

User input





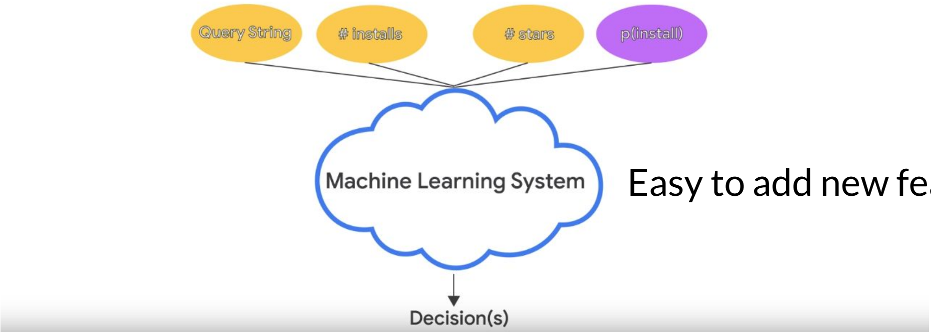
First steps with ML

3.Choose Machine Learning over a complex heuristic

Heuristic 1	# of views
Heuristic 2	# of stars

$(\#\ \text{stars})^{2/3} + 2(\#\ \text{installs})$

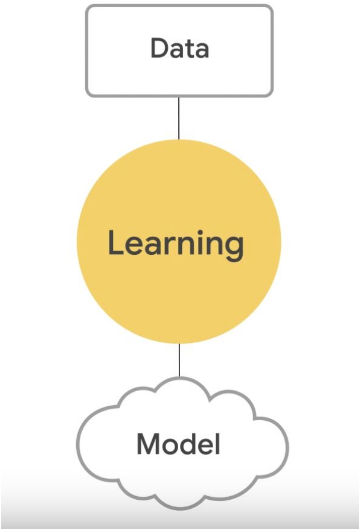
Hard to maintain & iterate

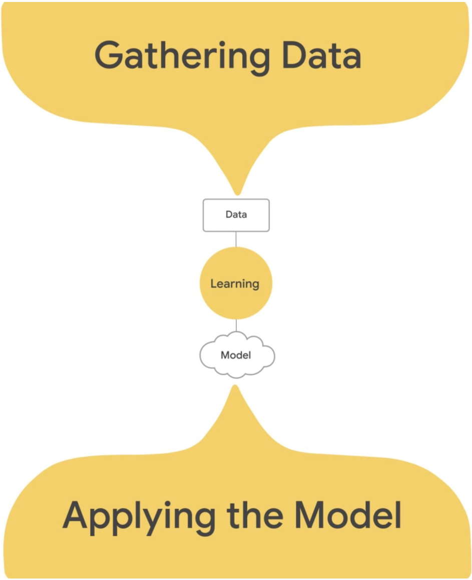


Easy to add new features

First steps with ML

4.Keep the first model simple and the infrastructure right





“Perfect is the enemy of the good”

First steps with ML

4.Keep the first model simple and the infrastructure right

- Before creating a fancy machine learning system
    - Focus on how are you going to collect and curate the data
    - How are you going to integrate the model within your application
        - Model deployment - (Docker, Kubernetes, GCloud, AWS)
        - How are the predictions going to be computed (Offline/Online)
            - Will you be able to have enough bandwidth
        - How are you going to store metrics, user inputs… in a database

First steps with ML

4.Keep the first model simple and the infrastructure right

- Make sure
  - You have a good way of validating if your model is good: reliable test set + metrics you trust
  - You can replace models easily in deployment
  - You write complete integration tests, unit tests
  - You have model versioning, data versioning documentation

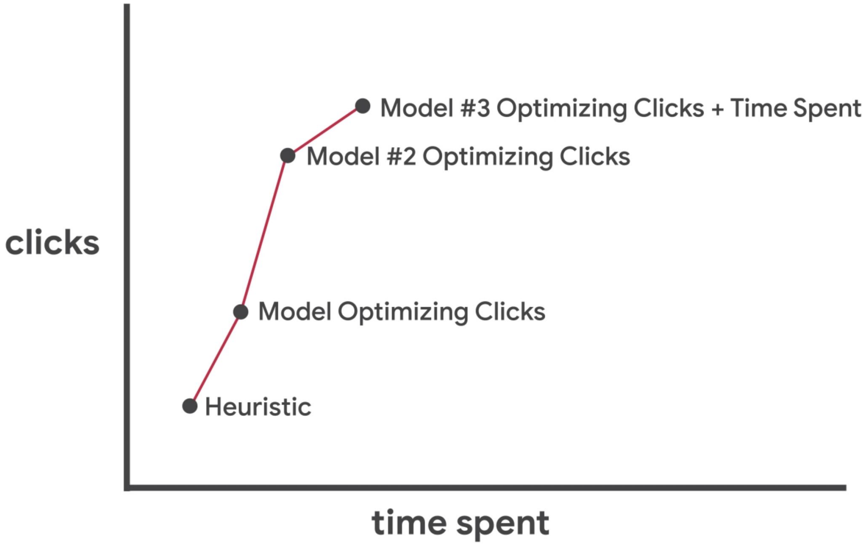
First steps with ML

5.Know the refresh rate requirements of your system

- Monitor the performance of your model over time
    - E.g. search engines and recommendation algorithms should be updated more often than a dog breed classifier model

First steps with ML

6.Don’t overthink the machine learning objective that you choose to optimize



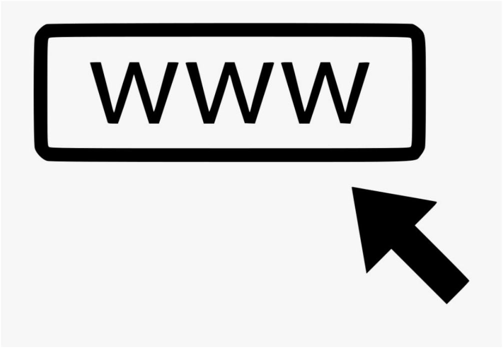
- An objective or cost function is what your machine learning model is trying to optimize for during training

- Probably optimizing for one metric will provide an increase on the rest (correlation)

First steps with ML

7.Choose a simple, observable and attributable metric for your first objective

VS



User Happiness

Number of Clicks

First steps with ML

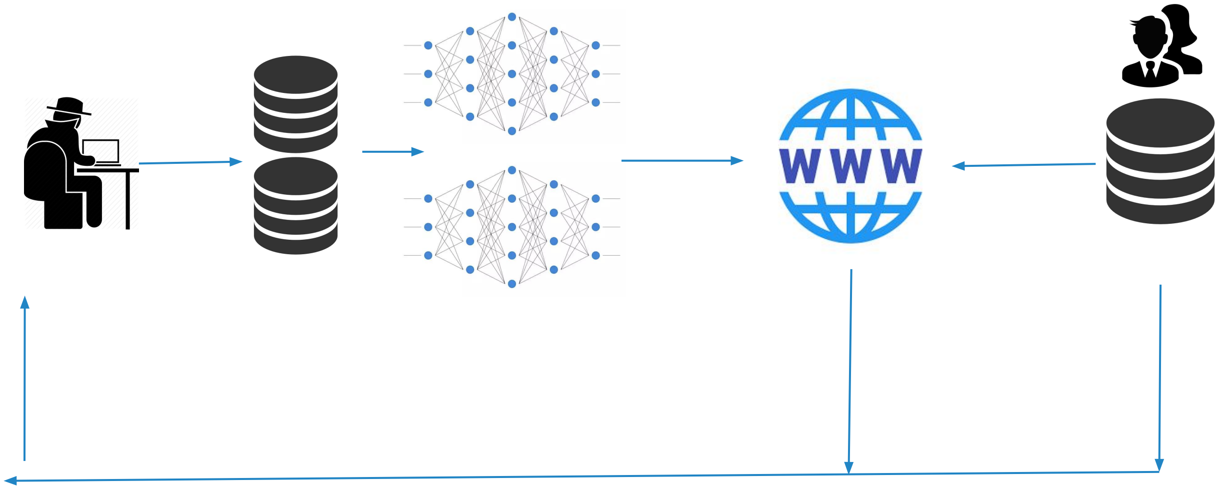
8.Do prior research on the topic! don’t try to reinvent the wheel

- You’ll find that many people probably have worked on the topic before!
- Datasets, metrics, objective functions, algorithms, pre-trained models, common failure scenarios
- https://toolbox.google.com/datasetsearch, https://www.kaggle.com/
- https://arxiv.org/, http://www.arxiv-sanity.com/
- https://github.com, https://paperswithcode.com

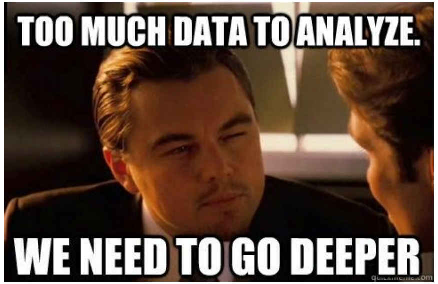
First steps with ML

8.Do prior research on the topic! don’t try to reinvent the wheel

After First Steps



End to End ML Pipeline



Phase 2: Feature Engineering

Feature Engineering

9.Aim for the low-hanging fruit

- Pull as many features as you can and combine them in intuitive ways (if possible)
  - Is there any new state of the art model, new feature transformations, new encodings you can implement?
- It is almost always a good approach to spend resources on more data collection
- All metrics should be rising at this stage

Feature Engineering

10.Combine features and modify existing features to create new features in human-understandable ways

- Use prior knowledge / expert knowledge or data analysis statistics
- Discretization of continuous variables
    - E.g. Age in age groups -- Feature 1: Age < 2 years old, Feature 2: age > 65 years old
- Crosses / Combine features into groups
    - E.g. Specific population group: {Male, Canadian}, {Female, Spanish, Older than 35}

Feature Engineering

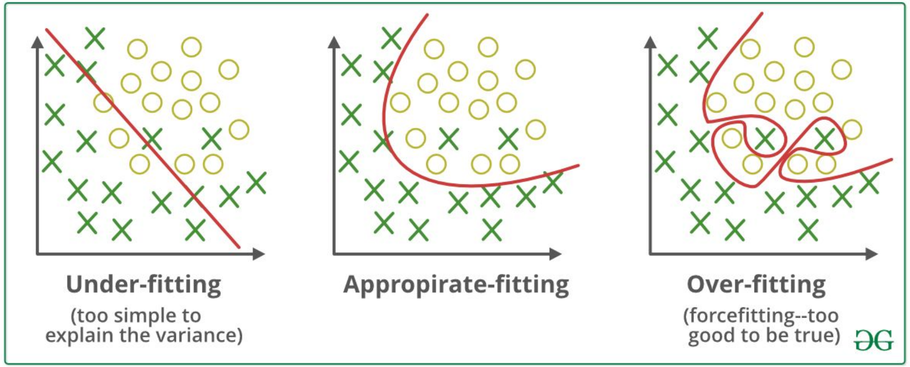
11. Look for patterns and analyze your data

- Perform error analysis to find:
    - Data biases
    - Errors in your own datasets
        - Wrong data labels
        - Users are making mistakes when inputting data
    - Features that are not contributing to optimize the objective function

Feature Engineering

12.Use regularization techniques

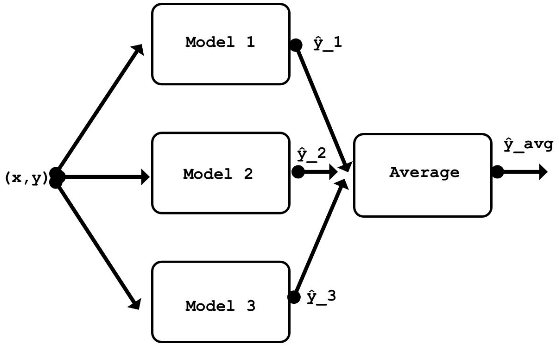
- Regularization can be a great help against overfitting
  - Dropout, Batch Normalization, L1, L2



Feature Engineering

13.Use model ensembles

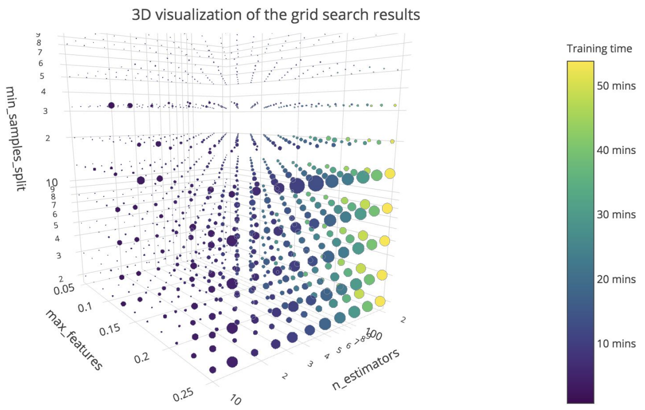
- If you have the computational resources using ensembles is always a good idea to get more accurate predictions.



Feature Engineering

14.Use hyperparameter search

- If you have the computational resources using a search to look for the best parameters for your model can help getting improvements



Feature Engineering

15.Monitor the Train / Serving skew

- There will be always difference between training and test data distribution

- Be able to detect drops in metric performances

- If you train a model with data until 24 September 2019, using test data from 25 September 2019 will help predicting production behaviour

After Feature Engineering Phase

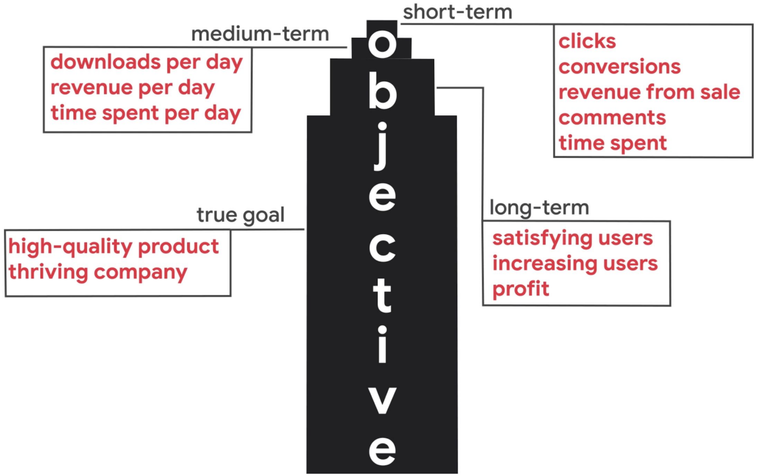
You will know you are here if

- You have a really good pipeline in place
- Gains in performance start to diminish
- You see trade-offs between metrics

After Feature Engineering Phase

To get to the next level

- You will need a more sophisticated machine learning system
- Custom ML models
- Redefine new goals, focus on long term objectives



3. Palladium

Palladium

These are the requirements that Palladium fulfills:

- Smooth transition from prototypes to machine learning models in production
- Avoid boilerplate in ML projects
- High scalability
- Avoid license costs

Why Palladium?

- We will see the development cycle and putting models into production scalably
- Covers common tasks in machine learning projects

Go to

https://jsalbert.github.io/ml_production_2019/