GDG Kolkata

LET'S BUILD

JANUARY 4TH 2020

Machine Learning Best Practices

Shweta Bhatt | ML GDE

Agenda

- Motivation
- ML Workflow
  - Data Collection and Preparation
  - Model Selection and Training
  - Model Validation and Error Analysis
  - Testing and Deployment
  - Retraining and Maintaining Models
- Bias vs Variance trade-off
- General best practices
- Summary

Why do we need ML best practices?

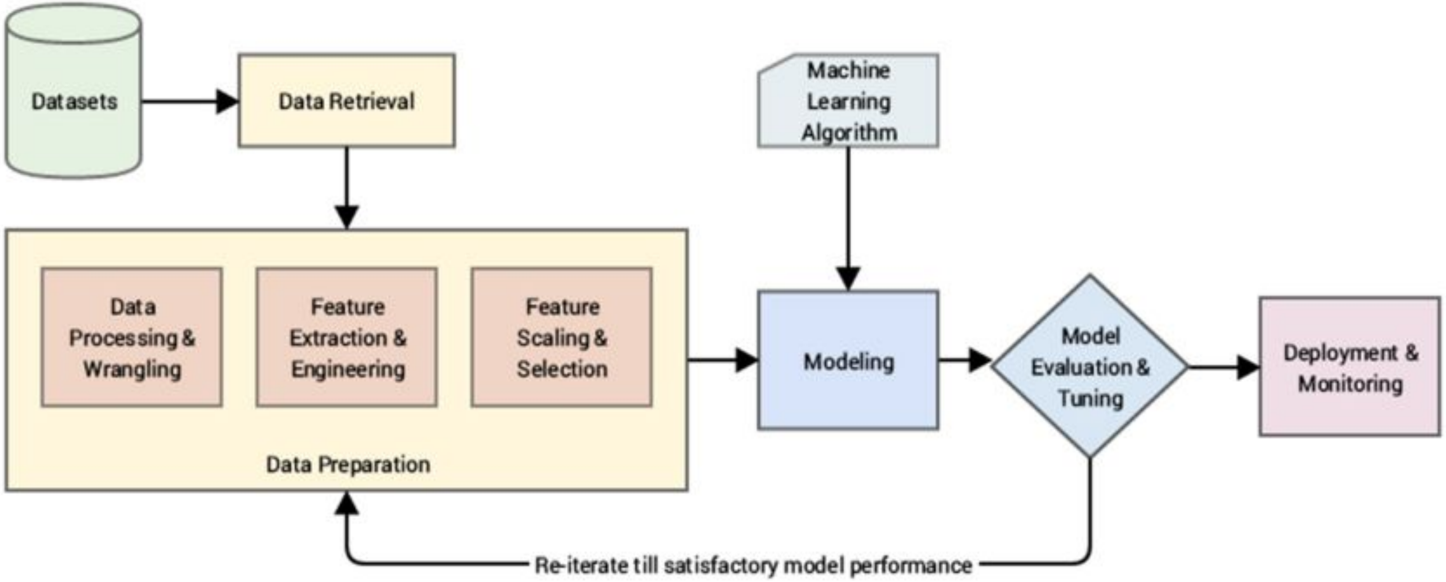
- Labelled vs unlabelled data?
- Feature Selection?
- Model Architecture?
- Model Complexity?
- Metrics?
- Hyperparameter Tuning
- Next Iteration?
- Retraining?

Image source

Our goal

Discuss some best practices that would aid in providing a direction that we can follow while making decisions when we are building ML systems.

ML Workflow: summary

- Problem Understanding and Formulation
- 

- Maintenance and Active Learning

Image source

Why is problem formulation important?

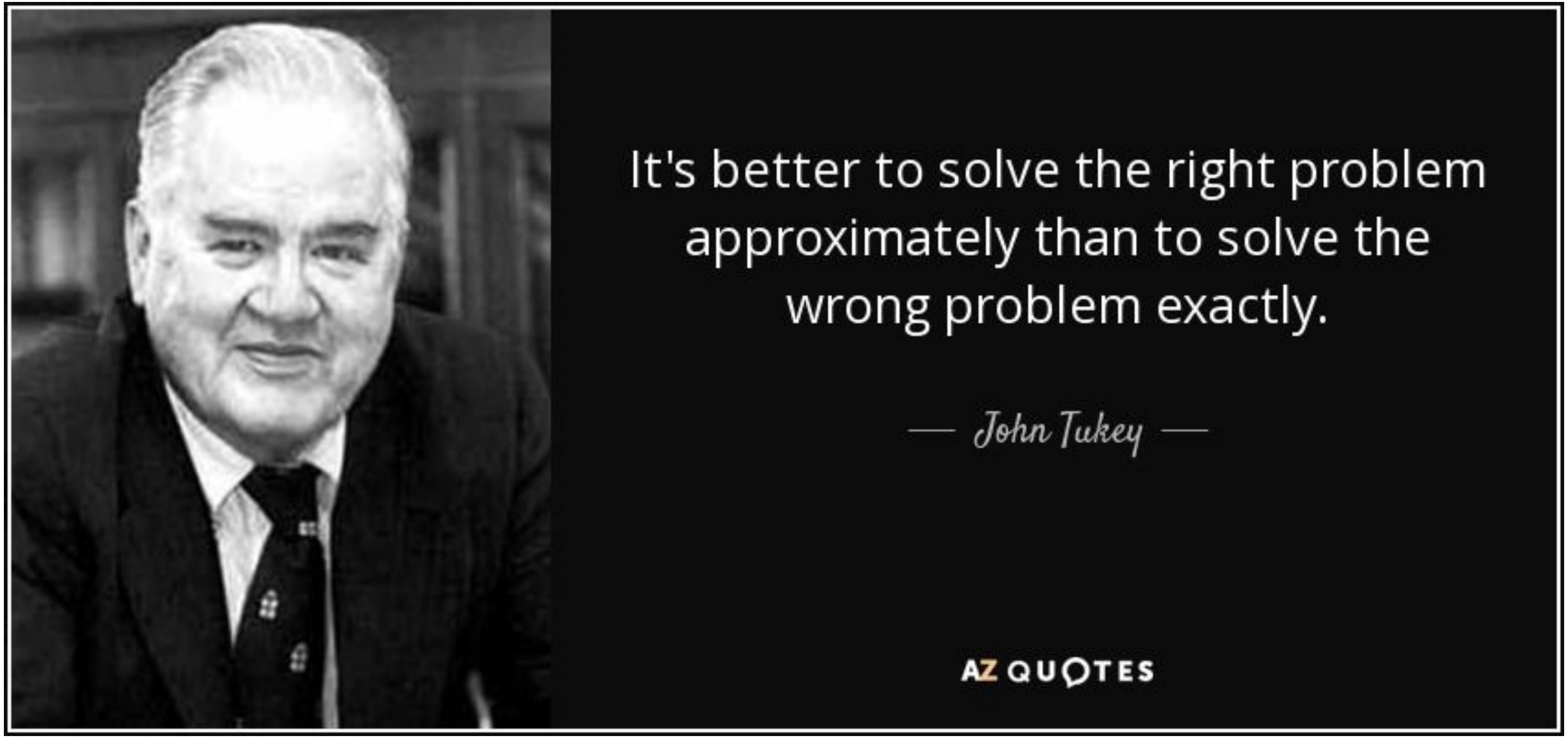


Image source

Problem Understanding and Formulation

- Define what ML problem to solve given the objectives, resources (data) and constraints
- Inputs and outcome of the model
- Metrics: ML, Business KPI
- Heuristics vs ML
- Formulation: start with a simple baseline, keep adding complexity later if required

It all starts with data

Image source

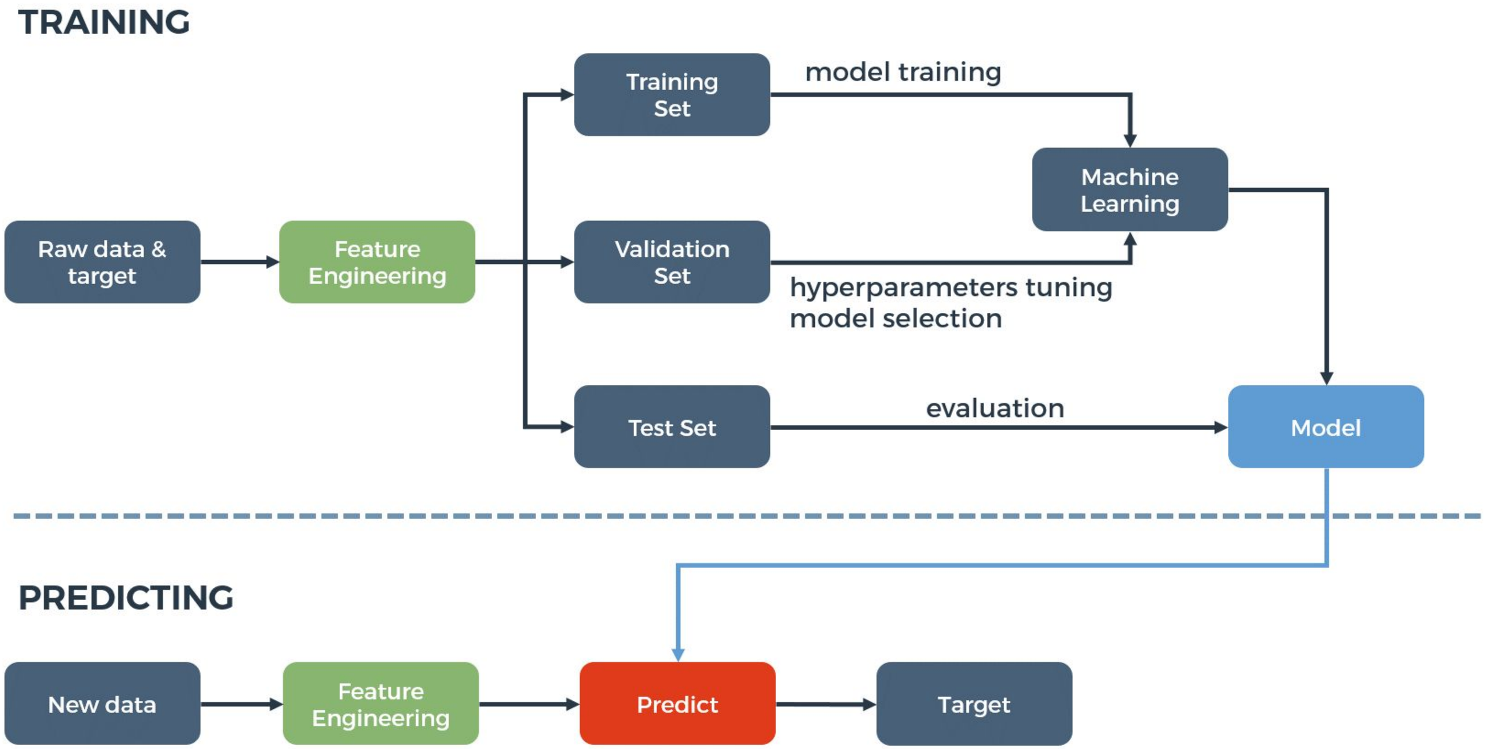
Data Collection and Preparation

- Collection: Scraping vs using an API
- Storage: Versioning, Database platform
- Processing: Cleaning + Formatting
- Verification and Validness: Consistency + Completeness

Where would you find an item quicker?

Image source

Training and prediction pipelines



Organizing Codebase

- Directory Structure
- Saving model names with parameters
- Experimentation notebooks
- Data Preprocessing
- Training scripts
- Inference/Prediction Scripts

```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
```

source

```
│
├── setup.py           <- Make this project pip installable with `pip install -e`
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.testrun.org
```

[source]()

Building an image classifier (apple vs orange)

- Large labelled dataset
- Train-test split: 70%-30%
- Testing accuracy: 95%
- Accuracy on the app (real world data) : 70%
- **What went wrong?**

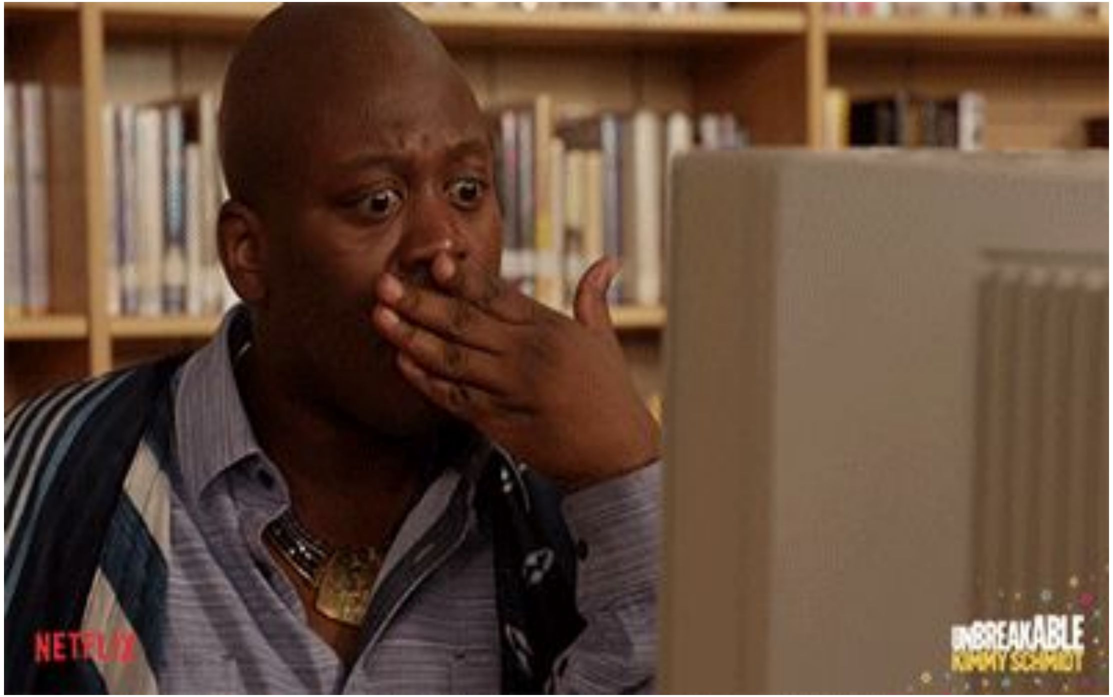


Image source

How to setup dev and test sets?

- **Distribution**
  - Should reflect the data you expect to get in future (unseen data)
  - Should come from the same distribution
- **Size**
  - Should be large enough to detect differences between different algorithms
  - 30% of the dataset vs 1000-10000 examples
- **Effort**
  - Come up with dev and test sets quickly (if not mature applications)
  - Change them quickly if you realize they are not meeting the mark

Model Building

- Feature Engineering & Selection
- Model Selection - follow general rule of thumb
- Model Training - early stopping, schedule training process accordingly, debugging and investigating, for example: use Tensorboard if you are using Tensorflow for implementation
- Model Validation - measure performance against a benchmark
- Visualizations - examine learning curves, visualize metrics

What other approaches to try?

Image source

Error Analysis

- Take a sample of 100 examples from your dev set that were misclassified by your model
- Evaluate these manually to understand the underlying causes of these errors
- Make a list of any patterns among the errors and try to categorize them
- Advantages:
  - You can evaluate how promising different directions and prioritize your ideas accordingly
  - Can save you a lot time and effort

How do you select the best model?

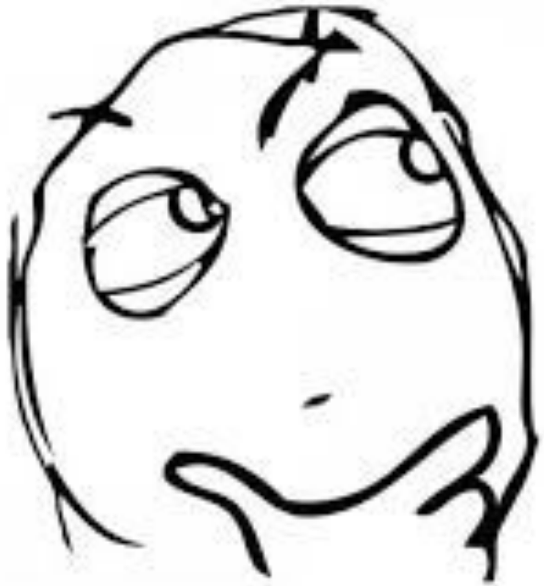


Image source

Evaluation Metrics

- Appropriate metrics for context and objectives of the system
- **Single-number evaluation metric** ⇒ decide which model works best
- **Derived metric**
  - E.g. F1 score, weighted average
- If N multiple metrics cannot be combined directly
  - **N-1 satisfying metrics** ⇒ they meet a certain value
  - **1 optimizing metric** ⇒ maximize performance over this
  - E.g. running time - satisfying metric, accuracy - optimizing metric

Iterative Process

- Don’t try to build the perfect solution in one go
- Start with something simple as quickly as possible
- Use error analysis to suggest you promising directions
- Use that to improve your existing solution

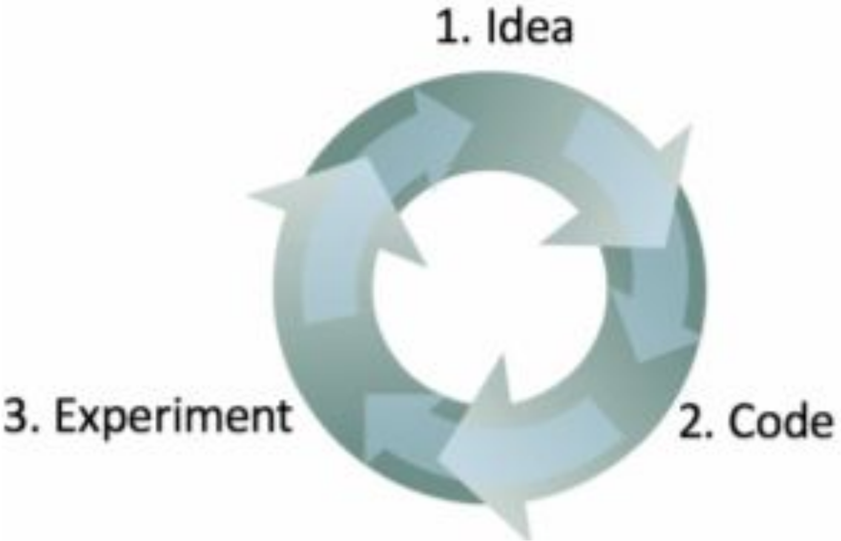
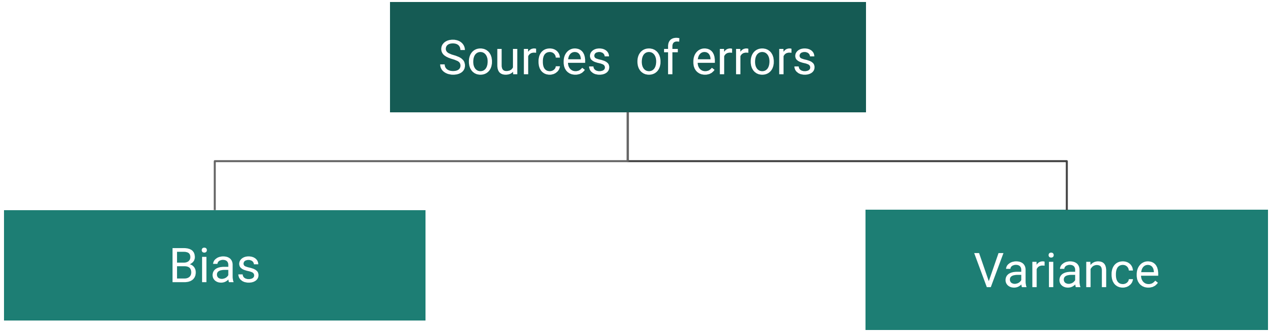


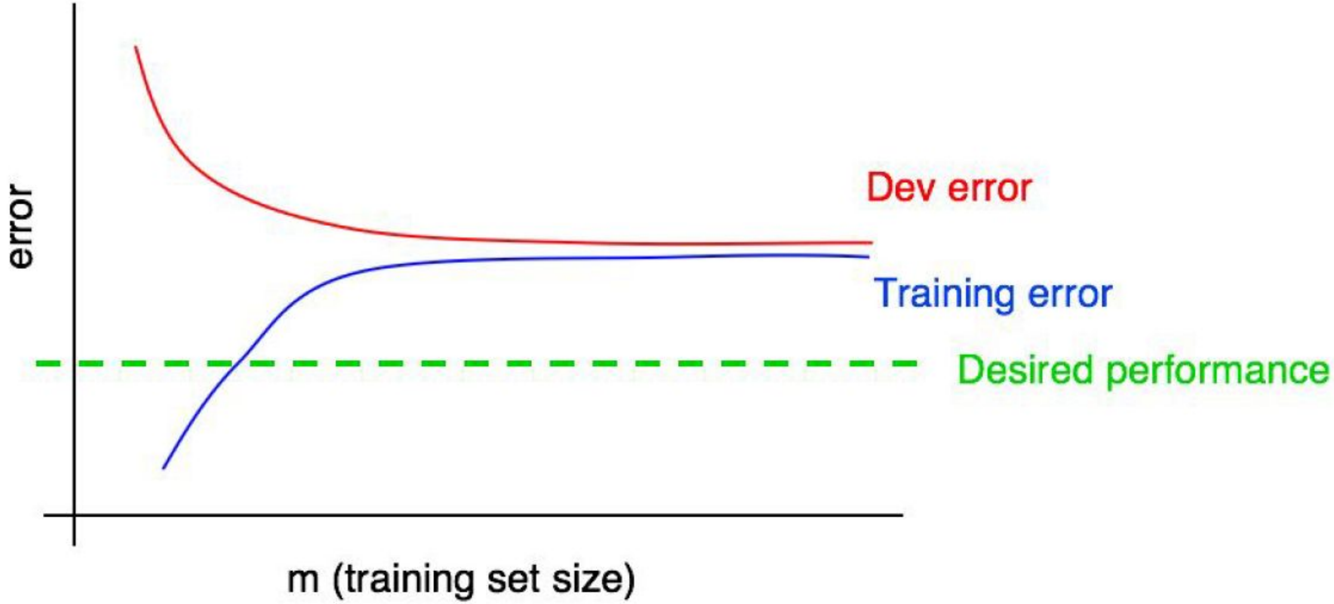
Image source

Bias vs Variance tradeoff



Our goal is to optimize for **low bias** and **low variance**

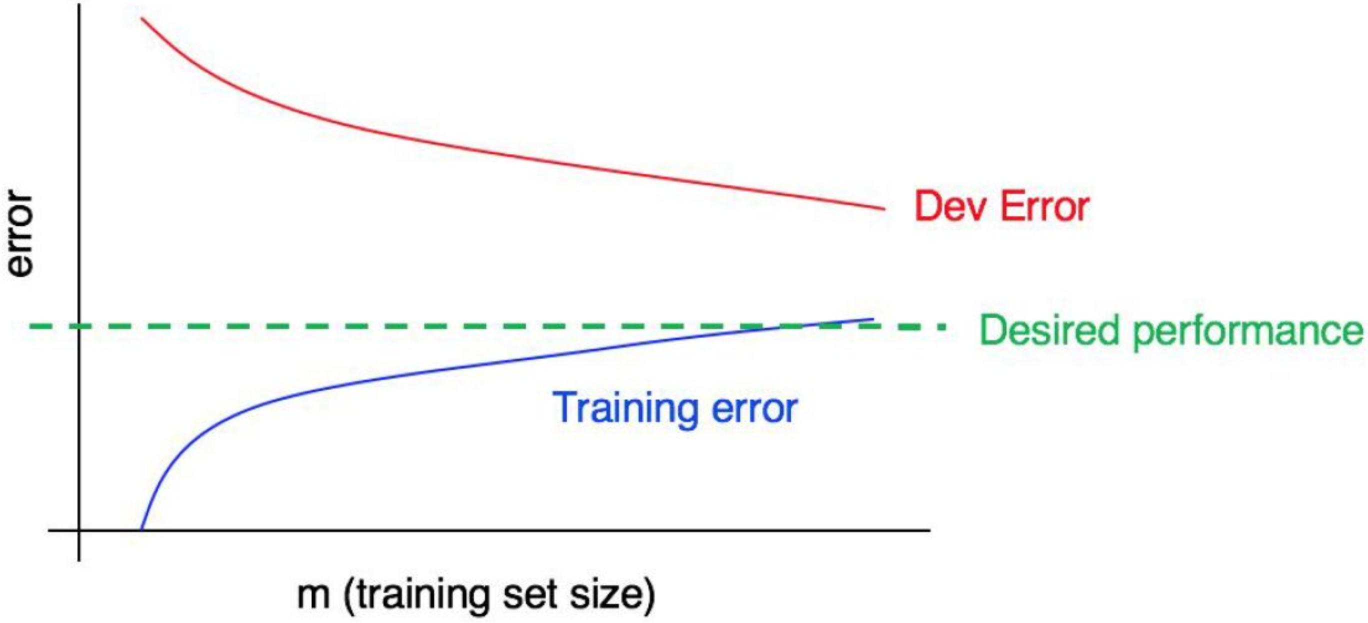
What do you interpret from this curve?



How to reduce bias?

- Increase model complexity (no. of layers/neurons) if computational power is not a limitation
- Update input features based on the feedback received from error analysis
- Reduce regularization
- Change the model architecture

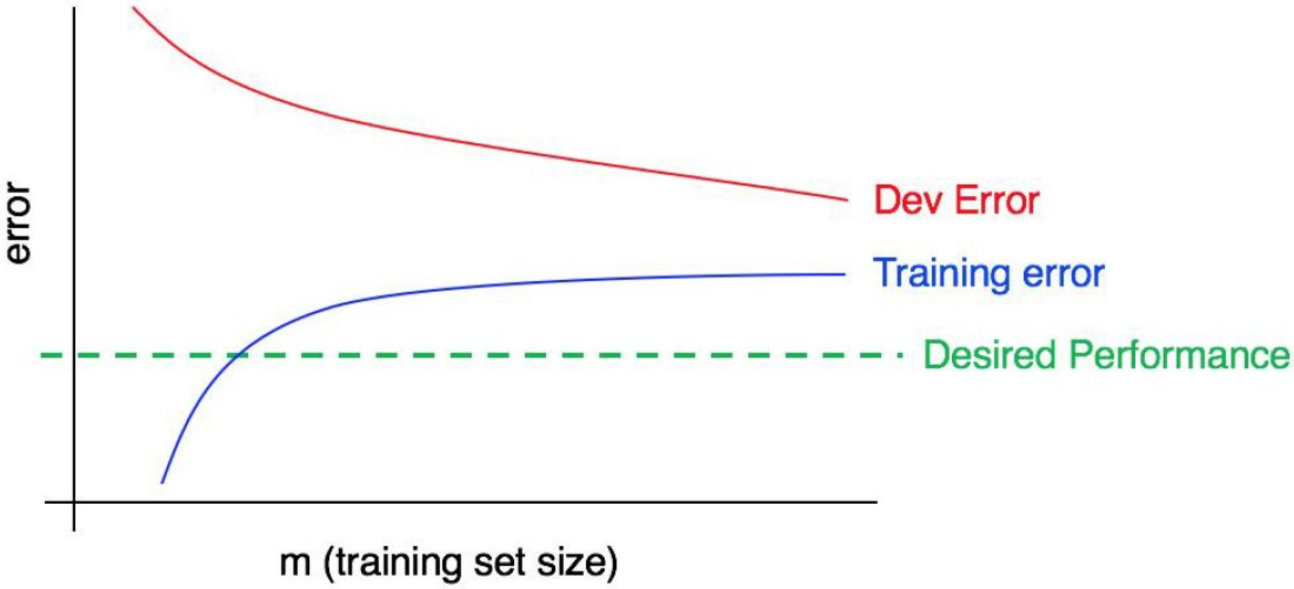
Again, what do you interpret from this curve?



How to reduce variance?

- Add more data
- Add regularization
- Early stopping
- Decrease number of input features
- Decrease model complexity (number of layers/neurons)
- Update input features based on the feedback received from error analysis
- Change the model architecture

And this one?


error
Dev Error
Training error
Desired Performance
m (training set size)

Testing and Deployment

- Unit testing - qualitative and quantitative
- Virtual Environment Setup
- Containers - E.g. Docker
- Do not reinvent the wheel - use open source code + tools

Maintenance and Active Learning

- Performance checks
- When to retrain?
- How to incorporate new data?

Human-centered Design Approach

- Clarity and control
- Engage with diverse set of users
- Incorporate feedback before and through project deployment

Examine raw data ⇒ Exploratory Data Analysis

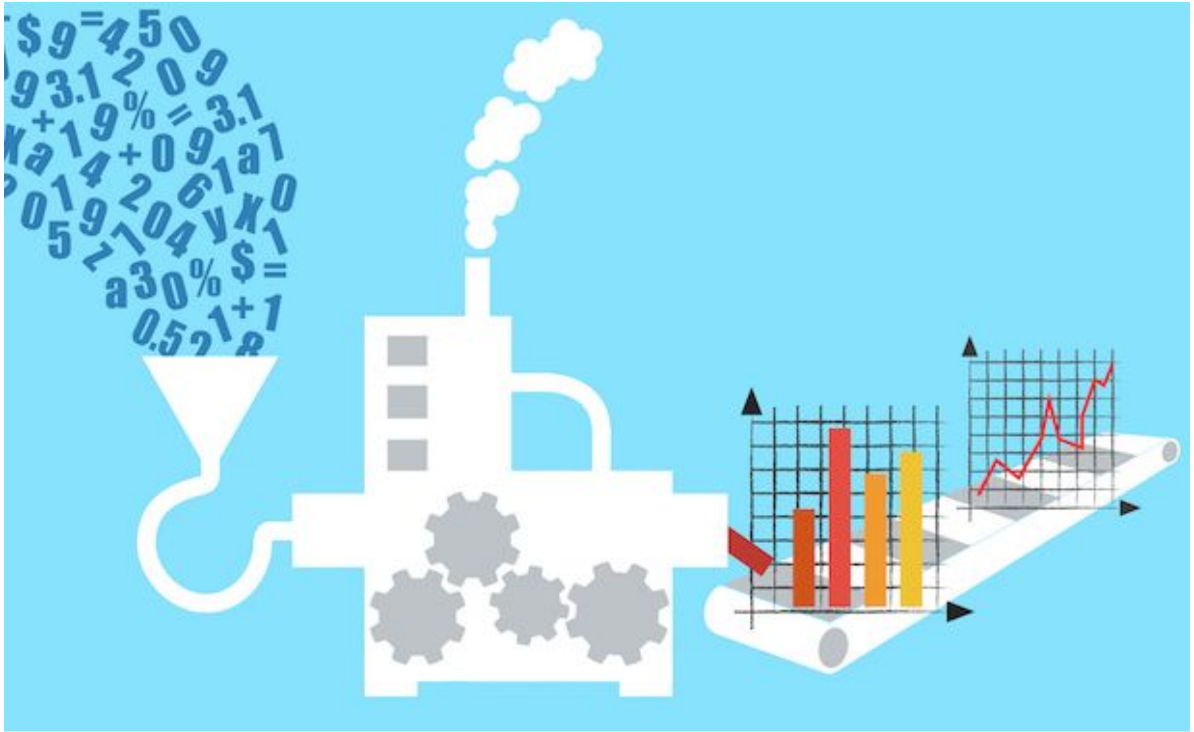


Image source

Understanding limitations

- What does your model solve?
- Where does it fall short?
- Can we overcome that with some heuristics?
- Communication with the team member and stakeholders

Test, test, test

- Unit tests
- Integration tests
- Update gold standard datasets
- Quality checks

System monitoring and update

- Real world performance + feedback
- Short term vs long term solution
- When to update?
- Effects of updating: system quality, user experience

Image source

Summary

- Understand the problem well and formulate it correctly
- Define dev and test sets and metrics
- Start with a simple baseline and add complexity when required
- Understand the data and clean it adequately
- Use feedback from data and results ⇒ Error analysis
- Do not reinvent the wheel ⇒ Use open source libraries, code, tools, literature
- Carry out a quick exhaustive research before jumping to implementation
- Use visualizations to make your life better
- Test your systems thoroughly
- Deploy as per your need ⇒ web/mobile

Thank you

Questions?

Let's connect-

@shweta_bhatt8

@shweta_bhatt

References

- Machine Learning Yearning by Andrew Ng
- Best Practices in Machine Learning Infrastructure-Algorithmia