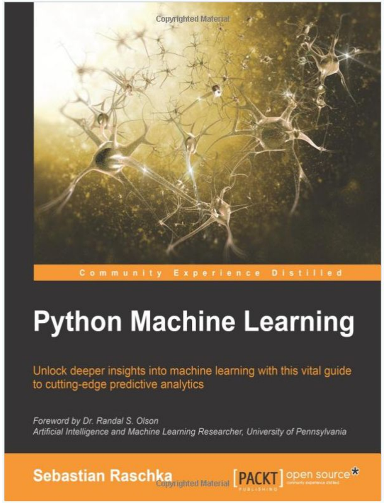
CMG imPACt19

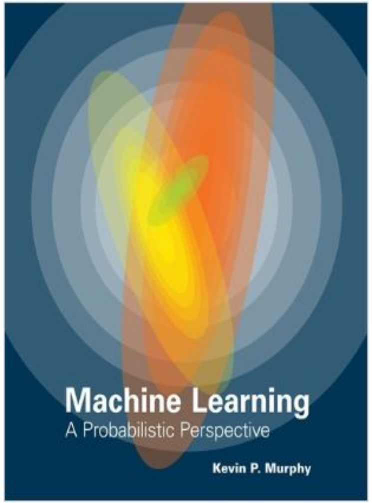
Introduction to Machine Learning

Some Books

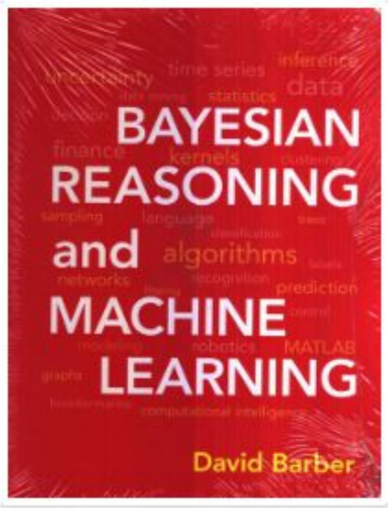
https://amzn.to/2PB81hB



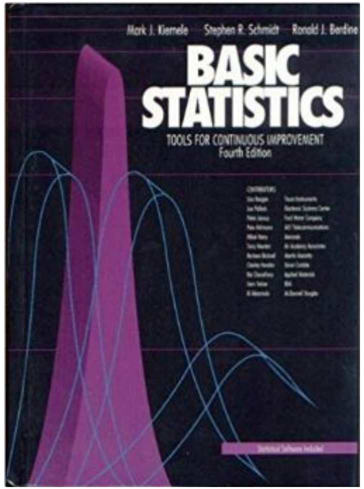
https://bit.ly/2A3t9lM



https://bit.ly/2QP4oBd



https://amzn.to/2SjlqIt



Some General Information and Ideas

- Machine Learning vs. Statistical Learning vs. Scientific Programming
- Ideal setup:
  - I7 CPU (or GPU) 16+ GB RAM
  - R v3.4.x + RStudio
  - Anaconda python 3.6+
  - Tensorflow
  - Keras or Cafe or Torch
- Non-programmer tools:
  - KNIME
  - Rattle
  - Wordij
  - Gephi
  - Tableau - if you still want to be in the driver's seat of your algorithms.
- Hypothesis Testing
- No Free Lunch Theorem
- Bias/Variance tradeoff
- Ensemble Modeling
- Bayesian Approach
- Patrick Winston's theory of Incremental Learning (vs. Locke's Tabula Rasa theory)
- "Anybody can learn to code. And everyone should give it a try"(Bill Gates on Twitter). So we will.

Flow

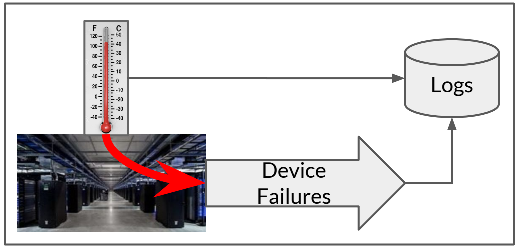
1. Device Failure Prediction:
    a. Regression and Logistic Regression
    b. Line fit and significance of parameters

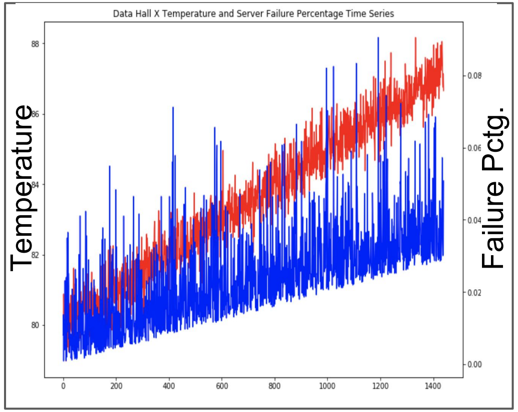
2. Queue Assignment based on Response Times:
    a. Distributions
    b. Clustering
    c. Classification
    d. Accuracy

3. ML Workflow:
    a. Data Engineering
    b. Split
    c. Train
    d. Test

4. Model Quality

Example 1: Predicting Device Failure Probability

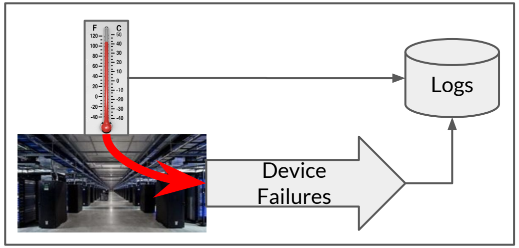




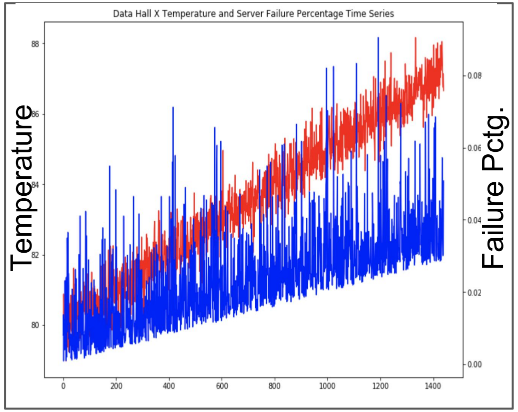
Is there a correlation between temperature and failure probability? If there is, can we use it to predict ***Pr{fail}***?

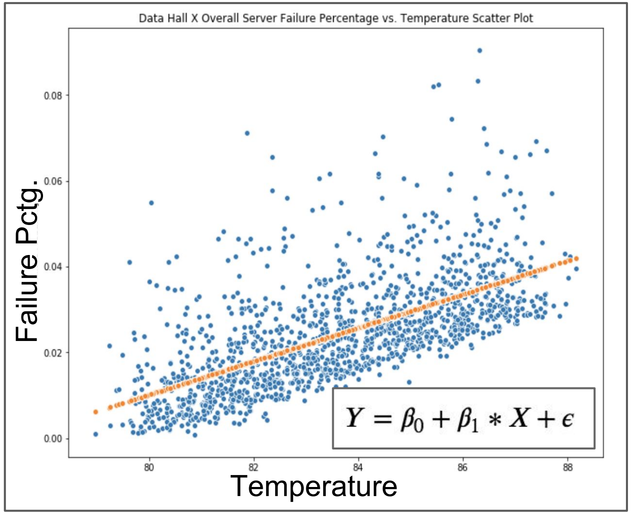
The Jupyter Notebook is [here]()

Example 1: Predicting Device Failure Probability



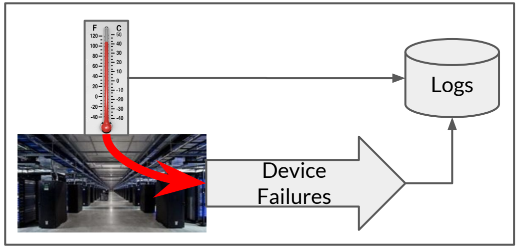
Is there a correlation between temperature and failure probability? If there is, can we use it to predict ***Pr{fail}***?



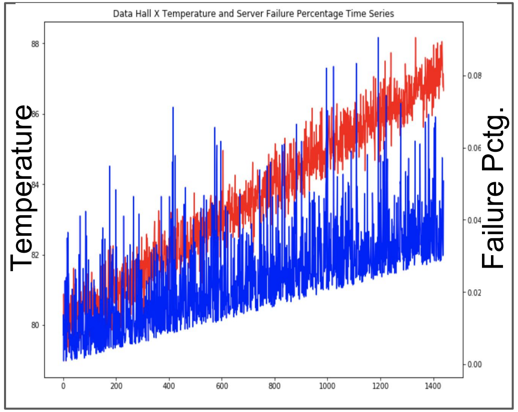


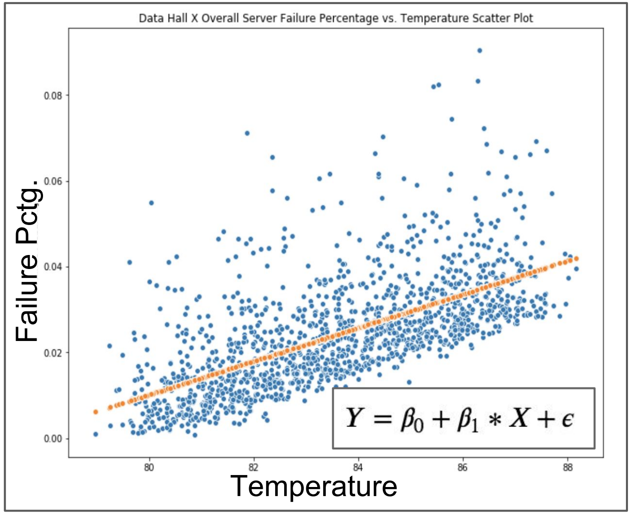
The Jupyter Notebook is [here]()

Example 1: Predicting Device Failure Probability



Is there a correlation between temperature and failure probability? If there is, can we use it to predict ***Pr{fail}***?



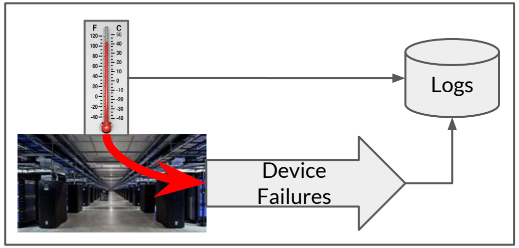


Dep. Variable:	failure_pctg	**R-squared:**	0.414
Model:	OLS	**Adj. R-squared:**	0.413
Method:	Least Squares	**F-statistic:**	1015.
Date:	Fri, 07 Dec 2018	**Prob (F-statistic):**	6.25e-169
Time:	10:23:28	**Log-Likelihood:**	4607.1
No. Observations:	1440	**AIC:**	-9210.
Df Residuals:	1438	**BIC:**	-9200.
Df Model:	1		
Covariance Type:	nonrobust		

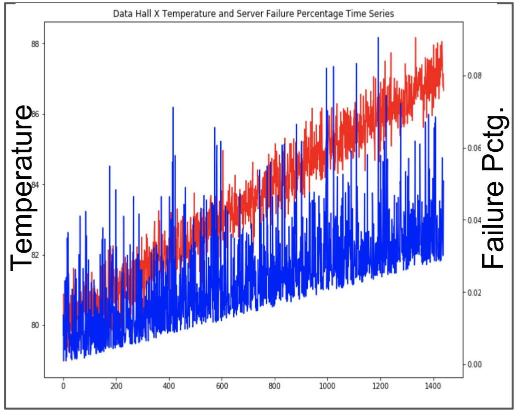
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-0.3024	0.010	-29.482	0.000	-0.323	-0.282
temperature	0.0039	0.000	31.853	0.000	0.004	0.004

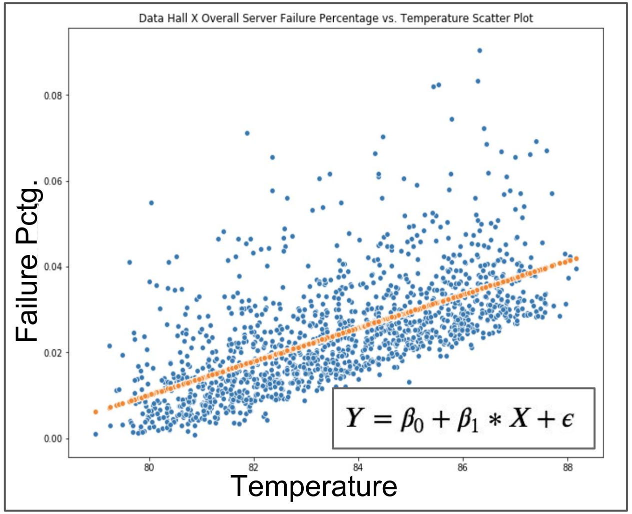
The Jupyter Notebook is [here](#)

Example 1: Predicting Device Failure Probability



Is there a correlation between temperature and failure probability? If there is, can we use it to predict ***Pr{fail}***?





Dep. Variable:	failure_pctg	**R-squared:**	0.414
Model:	OLS	**Adj. R-squared:**	0.413
Method:	Least Squares	**F-statistic:**	1015.
Date:	Fri, 07 Dec 2018	**Prob (F-statistic):**	6.25e-169
Time:	10:23:28	**Log-Likelihood:**	4607.1
No. Observations:	1440	**AIC:**	-9210.
Df Residuals:	1438	**BIC:**	-9200.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-0.3024	0.010	-29.482	0.000	-0.323	-0.282
temperature	0.0039	0.000	31.853	0.000	0.004	0.004

$$R^2 = 41.4\%$$

$$\beta_1 = \rho_{Y\_X} * \frac{\sigma_Y}{\sigma_X}$$

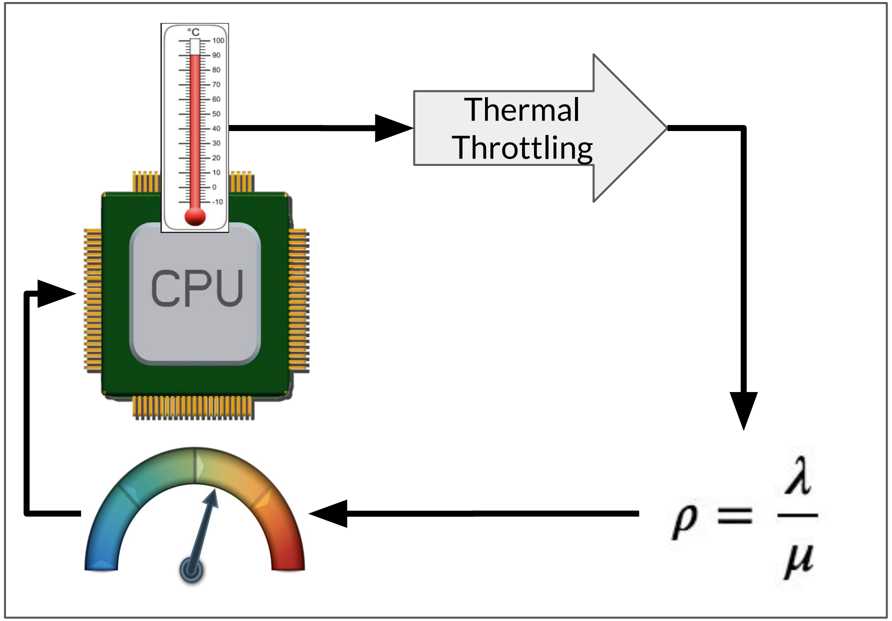
$$T = \frac{\beta_1 - 0}{\frac{\sigma_{resid}}{\sqrt{N_{df}}}}$$

```
regression.pvalues

Intercept      8.019322e-150
temperature    6.246228e-169
```

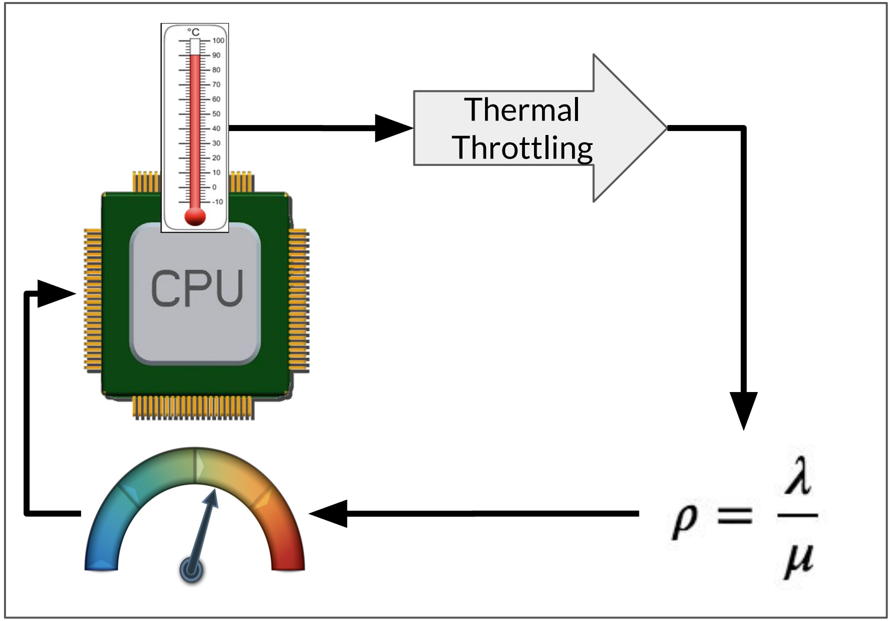
The Jupyter Notebook is here

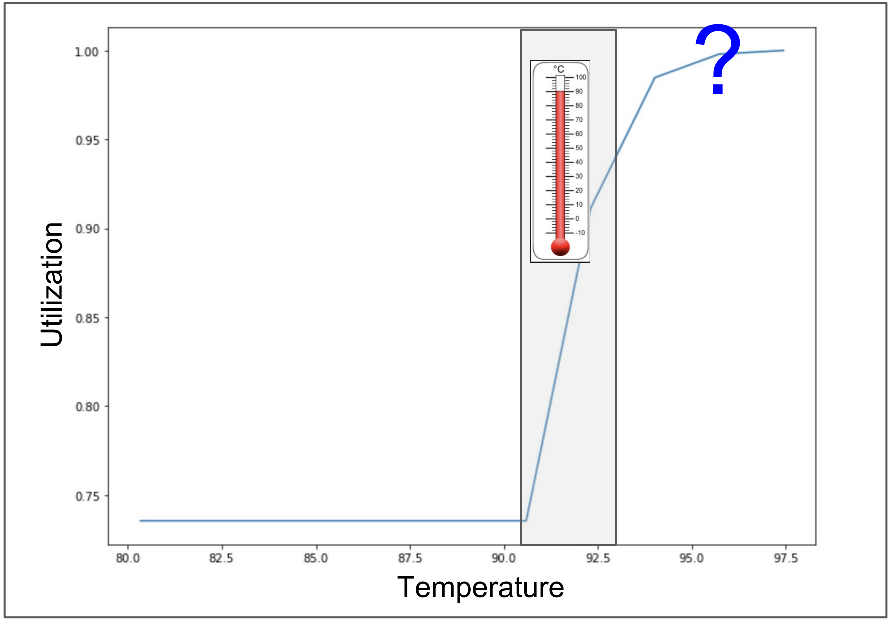
Example 2: Logistic Regression



The Jupyter Notebook is [here]

Example 2: Logistic Regression

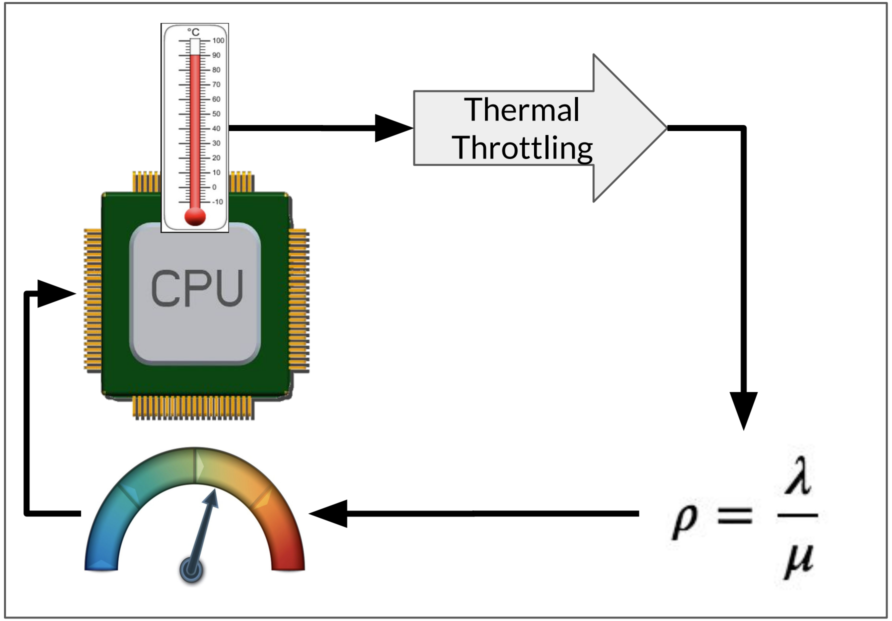


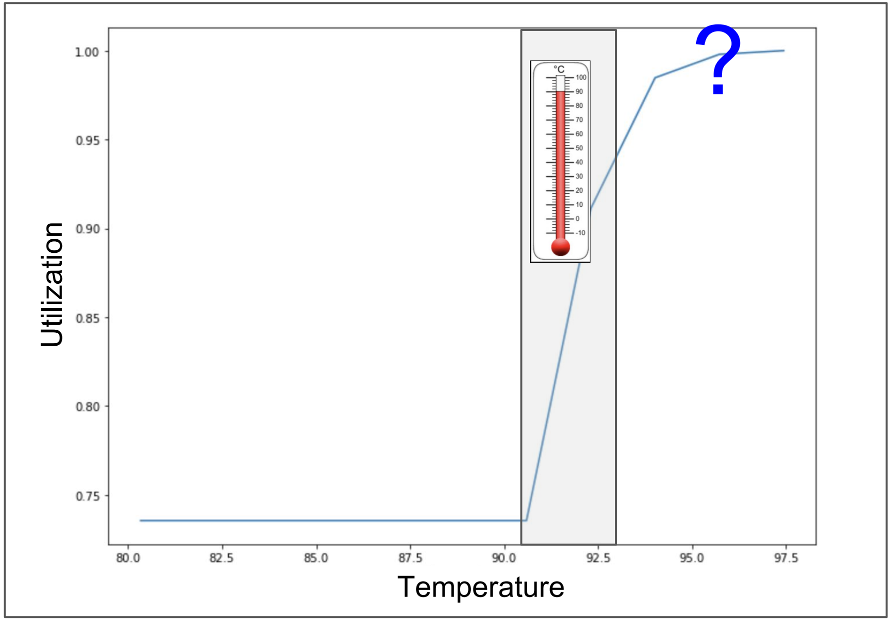


Given utilization measured between 90 and 93 degrees, how far will utilization grow due to throttling if the temperature rises to 97 degrees?

The Jupyter Notebook is [here]

Example 2: Logistic Regression





Given utilization measured between 90 and 93 degrees, how far will utilization grow due to throttling if the temperature rises to 97 degrees?

$$Y' = \ln\left[\frac{\rho}{1-\rho}\right] = \beta_0 + \beta_1 * T$$

$$\rho = \frac{e^{\beta_0} * e^{\beta_1 * T}}{1 + e^{\beta_0} * e^{\beta_1 * T}}$$

The Jupyter Notebook is here

Example 3: Clustering

Grouping services by Response Times: services svc1-svc4 are funneling into one interface, and we want to make sure that faster services are not waiting for the slower ones.

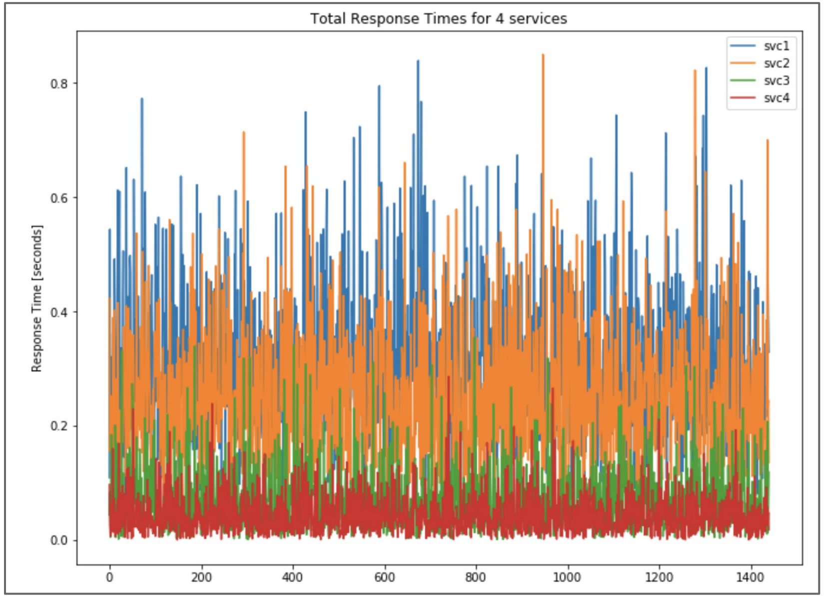
Due to resource constraints, we cannot allocate more than 2 instances of the interface. So we need to cluster the services into two groups.

The Jupyter Notebook is here

Example 3: Clustering

Grouping services by Response Times: services svc1-svc4 are funneling into one interface, and we want to make sure that faster services are not waiting for the slower ones.

Due to resource constraints, we cannot allocate more than 2 instances of the interface. So we need to cluster the services into two groups.

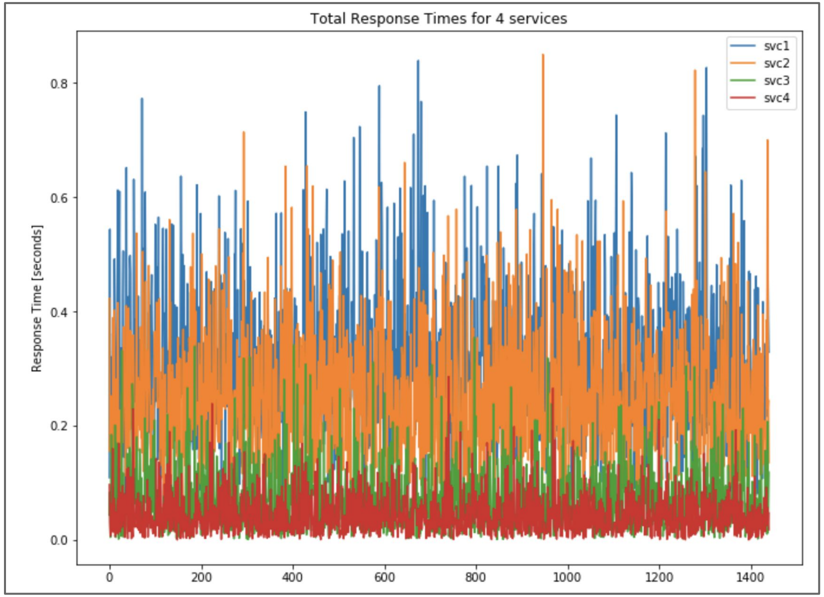


The Jupyter Notebook is here

Example 3: Clustering

Grouping services by Response Times: services svc1-svc4 are funneling into one interface, and we want to make sure that faster services are not waiting for the slower ones.

Due to resource constraints, we cannot allocate more than 2 instances of the interface. So we need to cluster the services into two groups.



svc3 and svc4 seem to belong in a different group than svc1 and svc2. We can use a clustering technique (e.g., k-means, with k = 2), to assign services to clusters.

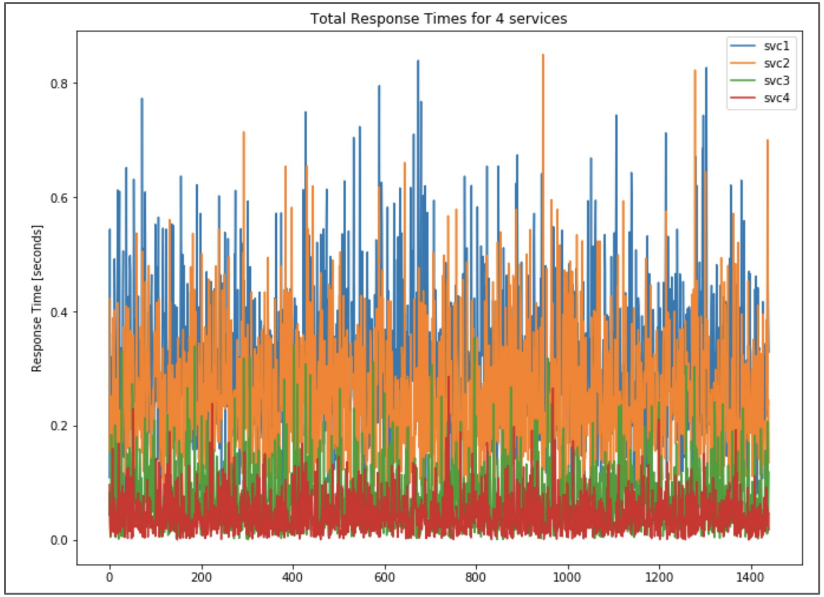
Find Clusters → Assign Services to Clusters

The Jupyter Notebook is [here]

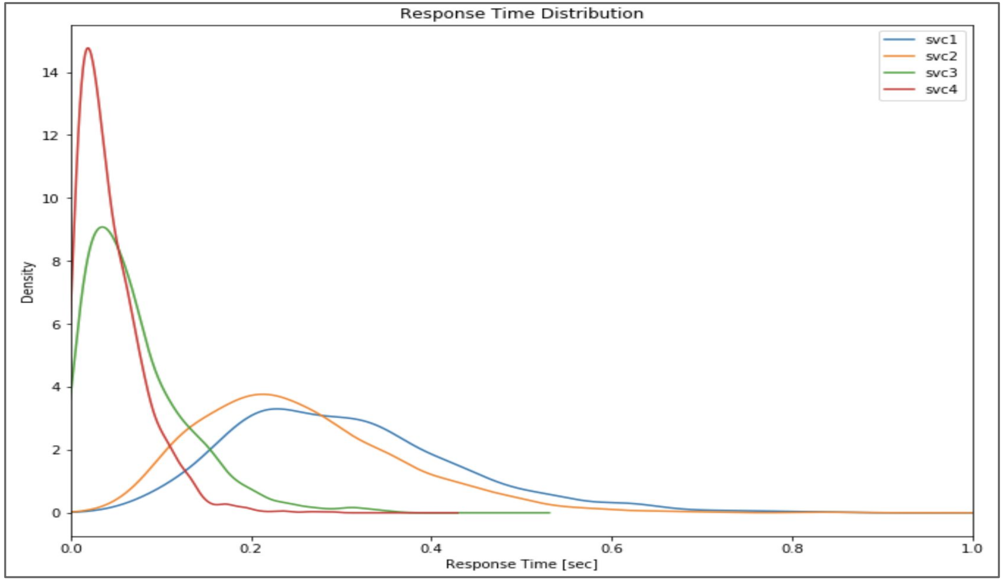
Example 3: Clustering

Grouping services by Response Times: services svc1-svc4 are funneling into one interface, and we want to make sure that faster services are not waiting for the slower ones.

Due to resource constraints, we cannot allocate more than 2 instances of the interface. So we need to cluster the services into two groups.



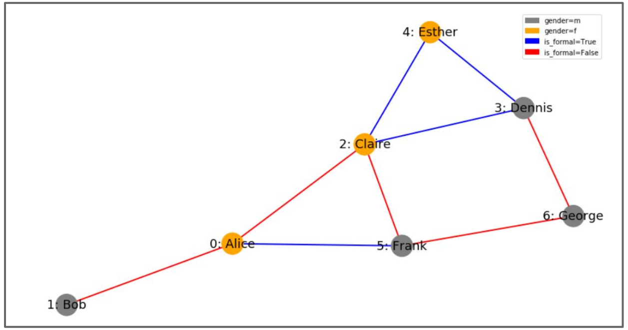
svc3 and svc4 seem to belong in a different group than svc1 and svc2. We can use a clustering technique (e.g., k-means, with k = 2), to assign services to clusters.

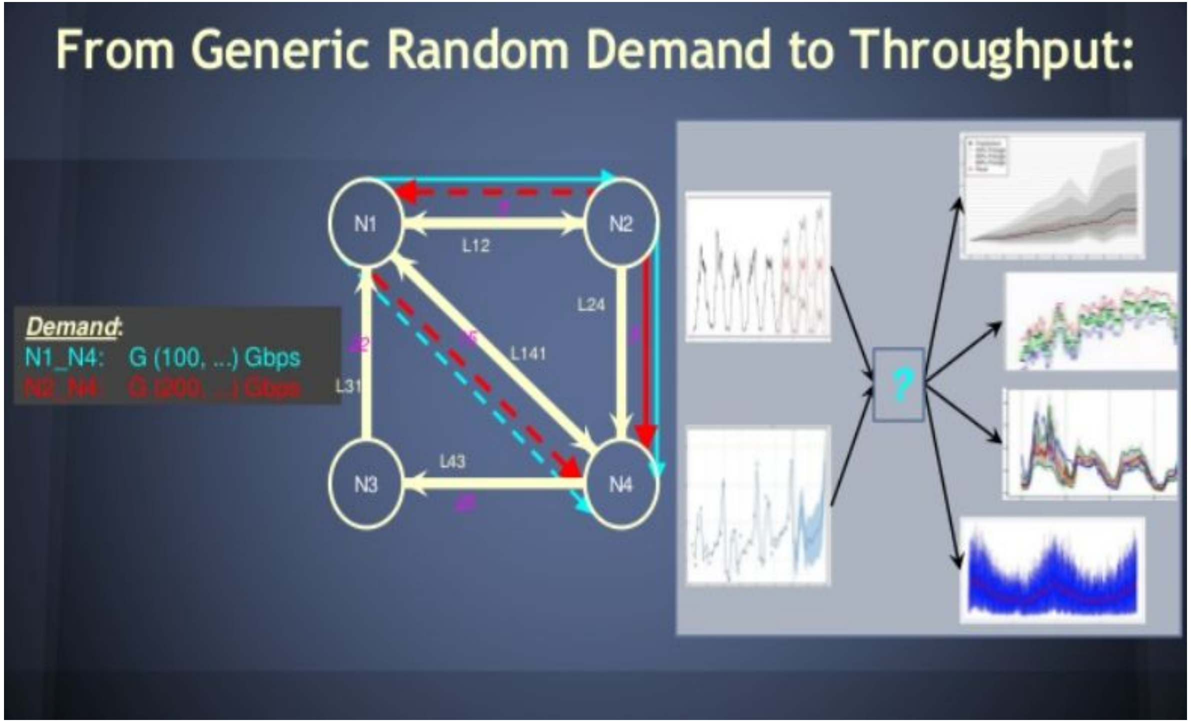


Find Clusters → Assign Services to Clusters

The Jupyter Notebook is [here]()

Network Analysis Introduction





A slide from a CMG'14 paper: https://bit.ly/2BO2BGc (a shameless act of self-advertisement)

Node Attributes

Degree Distribution

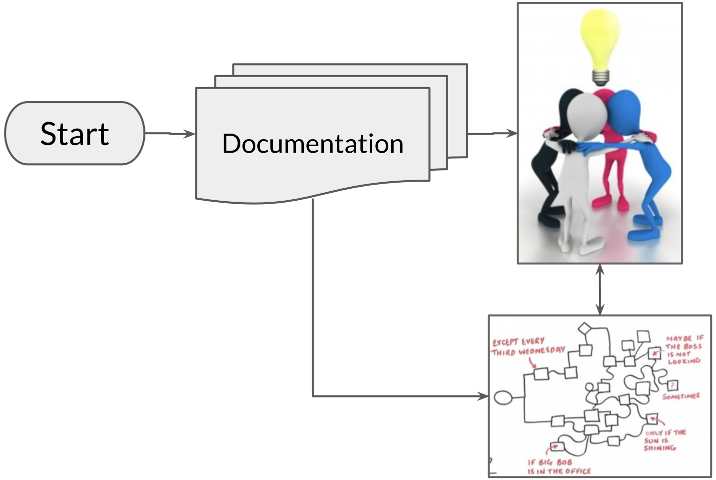
Shortest Path

Node Centrality

More on the topic is in this PDF

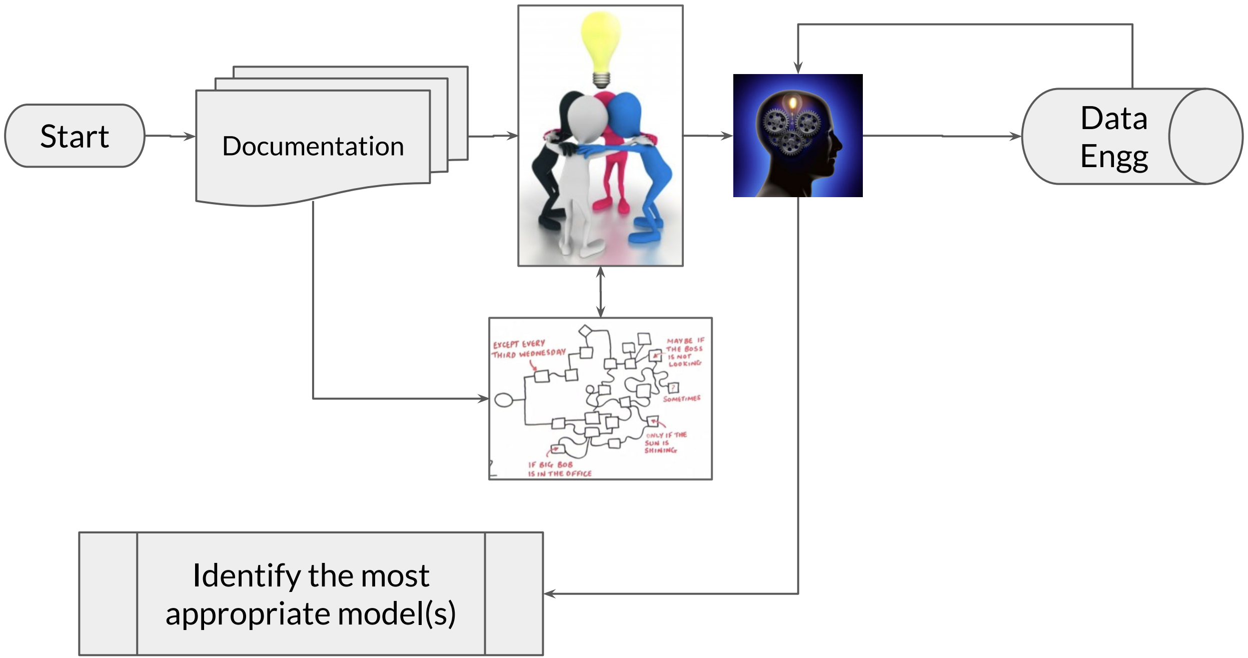
The Jupyter Notebook is here.

The Life of a Model: Waterfall



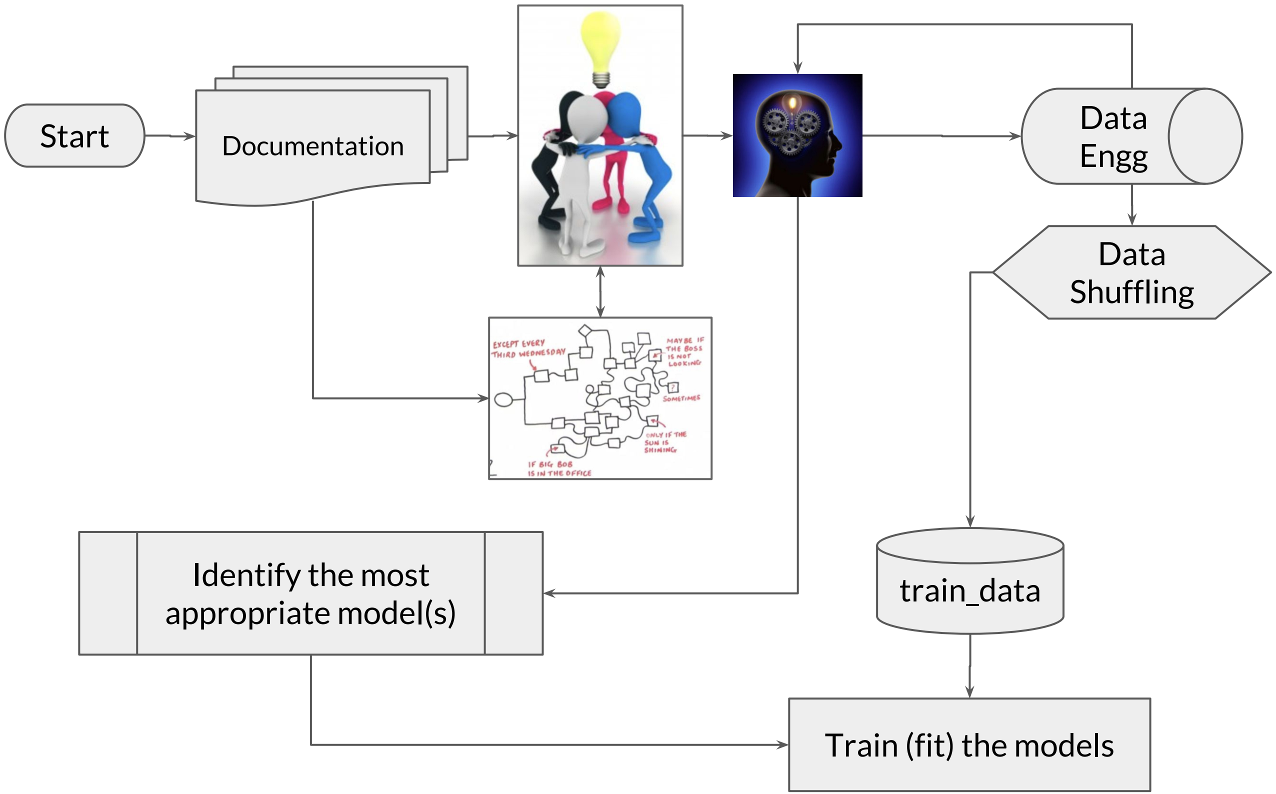
Also see here:
https://github.com/chemodan/ml_training_for_cmg_impact/blob/master/PDF/ML%20Process%20Flow%20Fit%20Metrics%20and%20Entropy.pdf

The Life of a Model: Waterfall



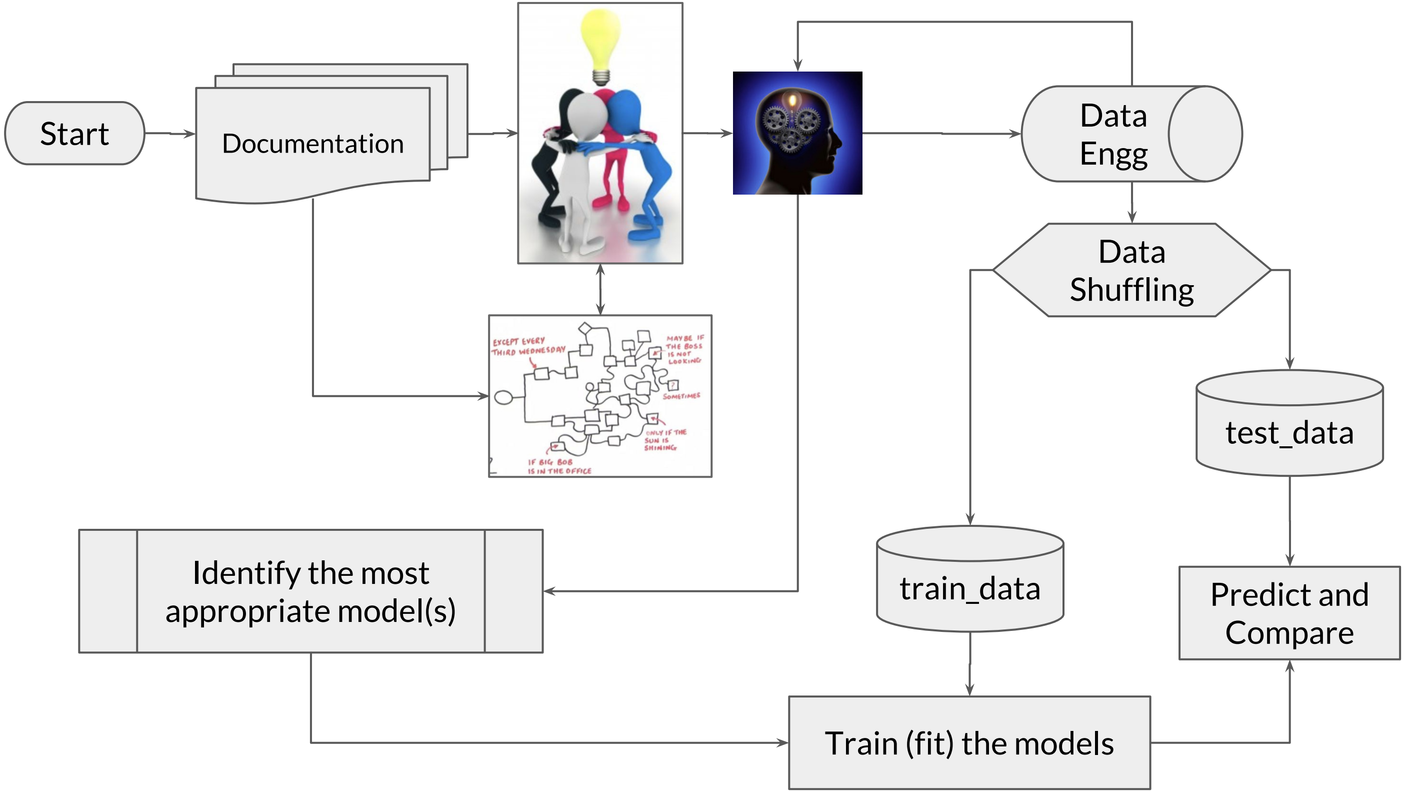
Also see here:
https://github.com/chemodan/ml_training_for_cmg_impact/blob/master/PDF/ML%20Process%20Flow%20Fit%20Metrics%20and%20Entropy.pdf

The Life of a Model: Waterfall



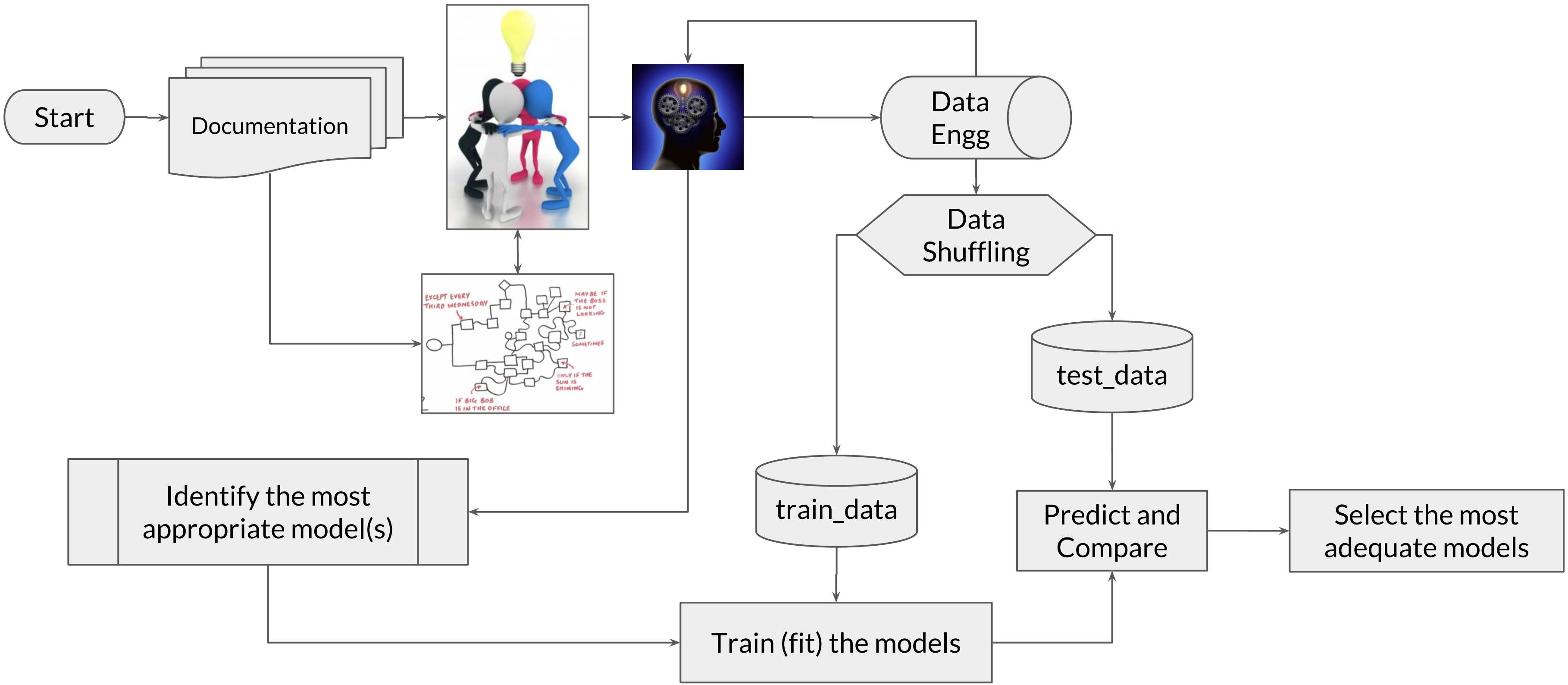
Also see here:
https://github.com/chemodan/ml_training_for_cmg_impact/blob/master/PDF/ML%20Process%20Flow%20Fit%20Metrics%20and%20Entropy.pdf

The Life of a Model: Waterfall



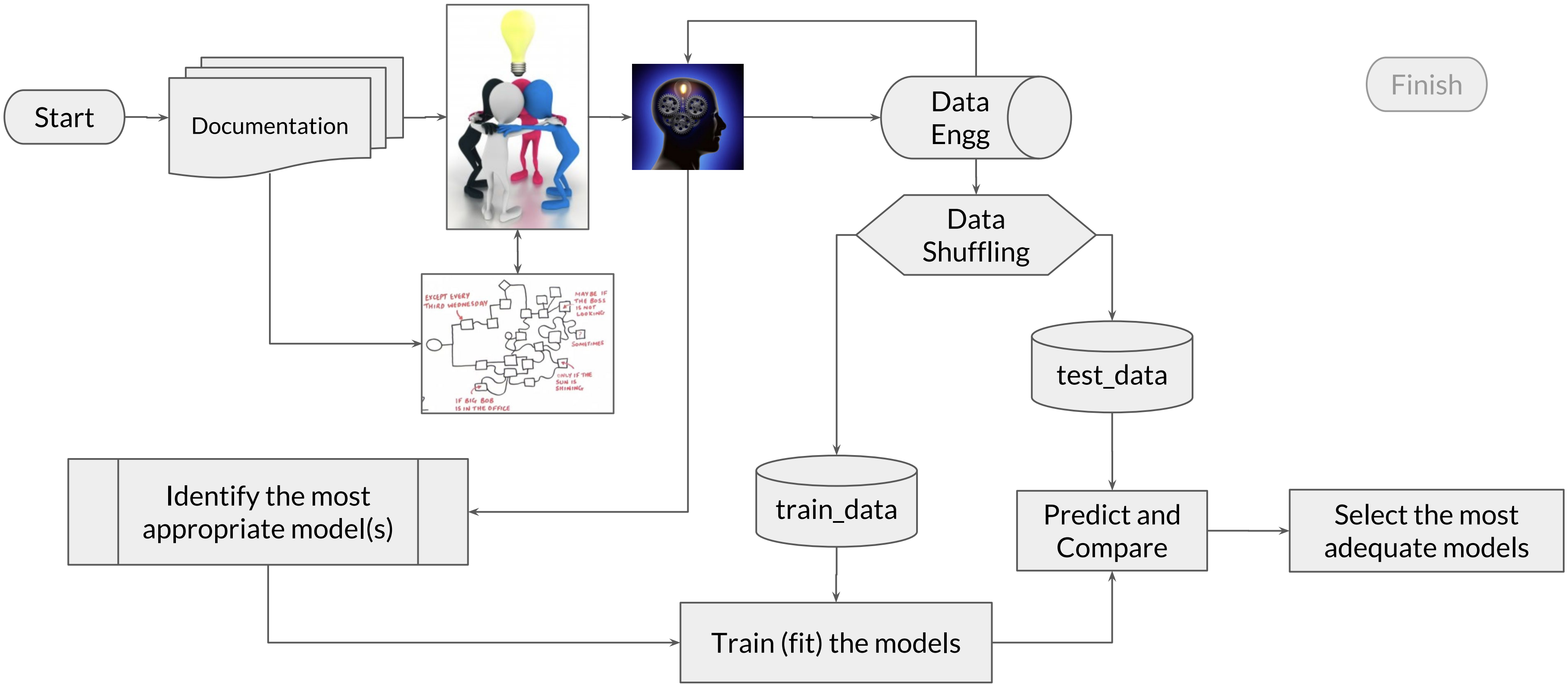
Also see here:
https://github.com/chemodan/ml_training_for_cmg_impact/blob/master/PDF/ML%20Process%20Flow%20Fit%20Metrics%20and%20Entropy.pdf

The Life of a Model: Waterfall



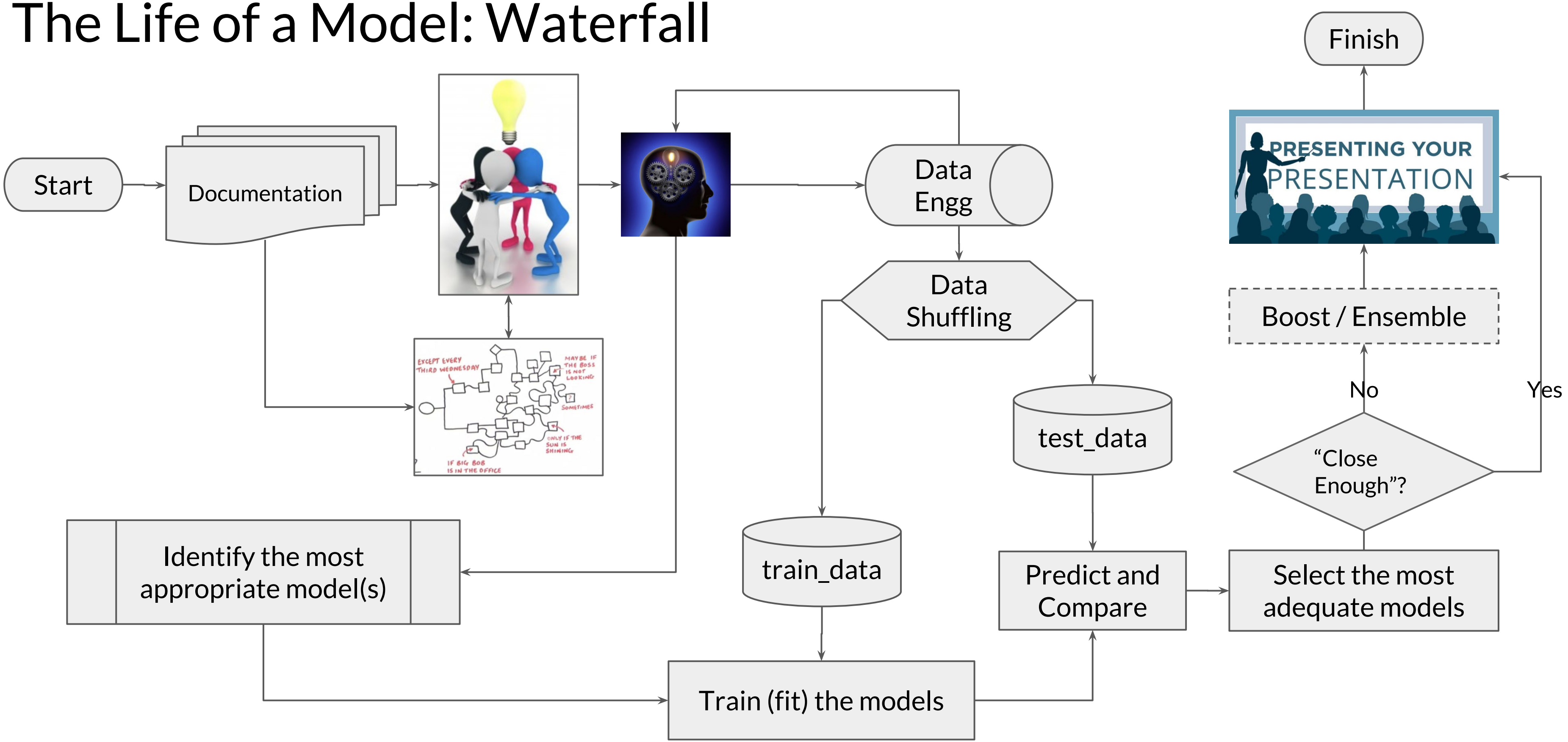
Also see here:
https://github.com/chemodan/ml_training_for_cmg_impact/blob/master/PDF/ML%20Process%20Flow%20Fit%20Metrics%20and%20Entropy.pdf

The Life of a Model: Waterfall



Also see here:
https://github.com/chemodan/ml_training_for_cmg_impact/blob/master/PDF/ML%20Process%20Flow%20Fit%20Metrics%20and%20Entropy.pdf

The Life of a Model: Waterfall



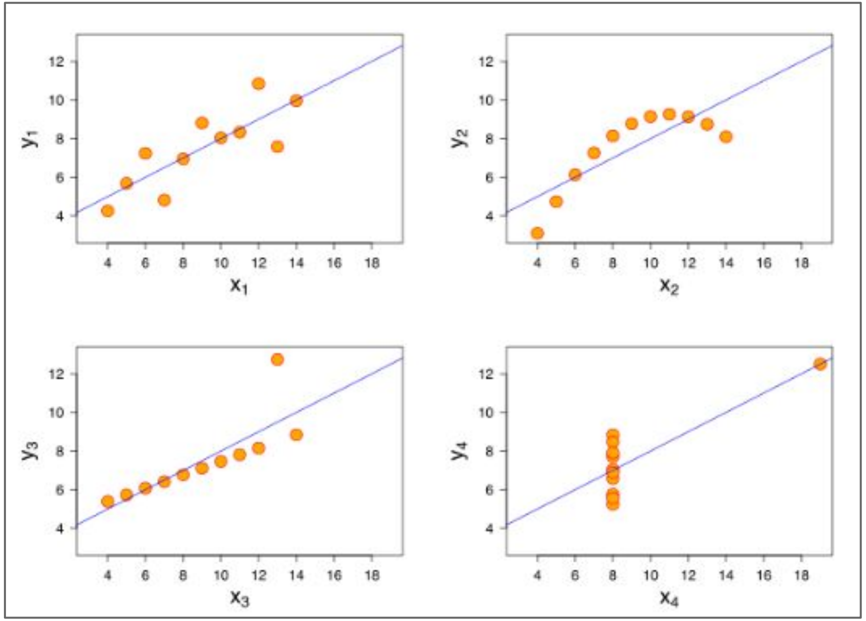
Also see here:
https://github.com/chemodan/ml_training_for_cmg_impact/blob/master/PDF/ML%20Process%20Flow%20Fit%20Metrics%20and%20Entropy.pdf

Regression: How to find the Best-Fitting Model

1. The 7 linearizable models
2. Multivariate Analysis
3. Fine points of model selection

Details are here.
More details are here.

Regression: One Thing to Beware



Anscombe's Quartet

Property	Value	Accuracy
Mean of *x*	9	exact
Sample variance of *x*	11	exact
Mean of *y*	7.50	to 2 decimal places
Sample variance of *y*	4.125	±0.003
Correlation between *x* and *y*	0.816	to 3 decimal places
Linear regression line	$y = 3.00 + 0.500x$	to 2 and 3 decimal places, respectively
Coefficient of determination of the linear regression	0.67	to 2 decimal places

Classification Model Quality:
Accuracy

True Value	Happy	Unhappy	Total Classified
Happy	6	3	9
Unhappy	4	15	19
Total True	10	18	$\boxed{28}$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

= 76%

Classification Model Quality:
Precision

True Value	Happy	Unhappy	Total Classified
Happy	6	3	9
Unhappy	4	15	19
Total True	10	18	28

$$Precision = \frac{TP}{TP + FP}$$

= 67% for Happy
= 79% for Unhappy

Classification Model Quality:
Specificity

True Value	Happy	Unhappy	Total Classified
Happy	6	3	9
Unhappy	4	15	19
Total True	10	18	28

$$Specificity = \frac{TN}{TN + FP}$$

= 33% for Happy
= 21% for Unhappy

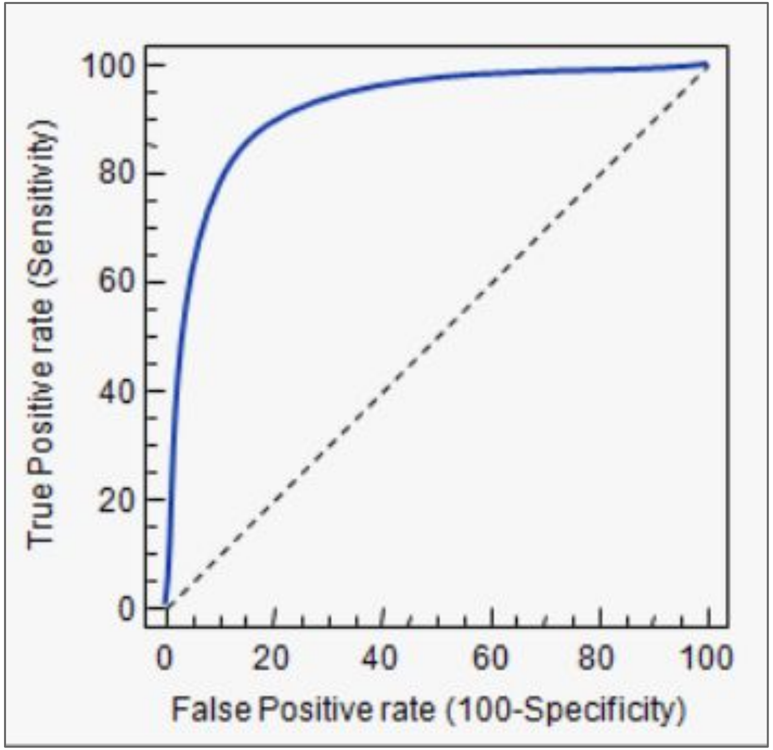
Classification Model Quality:
Recall (Sensitivity)

True Value	Happy	Unhappy	Total Classified
Happy	6	3	9
Unhappy	4	15	19
Total True	10	18	28

$$Recall = Sensitivity = \frac{TP}{TP + FN}$$

= 83% (Unhappy)
= 60% (Happy)

Classification Model Quality:
AUC (Area Under Curve)

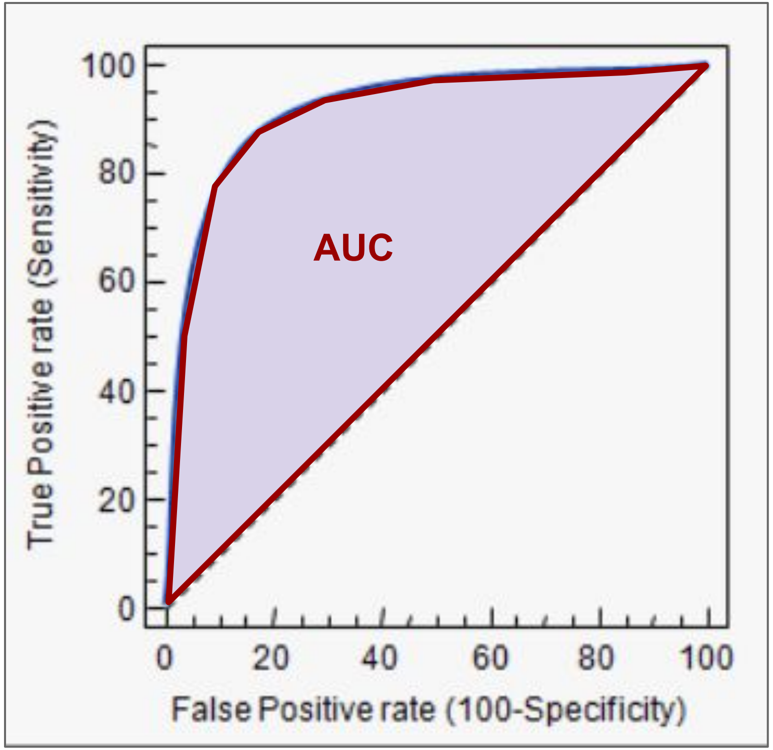


The higher the AUC the more likely we are to recognize the right label and the less likely we are to have a false positive conclusion.

Very useful for multi-label classification.

A great Wikipedia article here: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Classification Model Quality:
AUC (Area Under Curve)

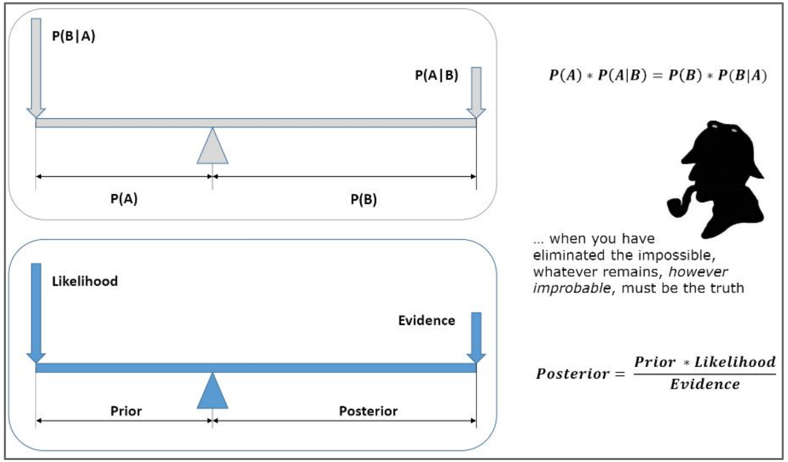


The higher the AUC the more likely we are to recognize the right label and the less likely we are to have a false positive conclusion.

Very useful for multi-label classification.

A great Wikipedia article here: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

In Lieu of Epilogue:



1. **Bayesian Principle**
2. **Neural Networks:**
   a. A very short overview of what neural networks are is [here](#).
   b. Naftali Tishby: the only existing [mathematical explanation for why Deep Neural Networks work](#).
3. **Dimensionality Reduction:** [Principal Component Analysis (PCA)](#) ([a more serious video](#))

Appendix

A lot of useful PDFs: https://github.com/chemodan/ml_training_for_cmg_impact/tree/master/PDF

An excellent blog: https://sebastianraschka.com/blog/index.html

Q&A sources: Quora; StackOverflow; StackExchange

A good explanation of confidence intervals of variances: (milefoot)

What We Have Discussed

1. Literature
2. Examples:
    a. Linear Regression
    b. Logistic Regression
    c. Classification
    d. Clustering
3. Workflow
4. Model Quality:
    a. Regression
    b. Classification

Where We Skimmed the Surface & What We Left Out

1. Dimensionality Reduction Techniques
2. Neural Networks
3. Bayesian Approach
4. Time Series Analysis
    a. Stationary
    b. Nonstationary
5. Statistical Process Control

Some Additional Useful Links

GitHub repository for this presentation: https://github.com/chemodan/ml_training_for_cmg_impact

Tools (No Programming Needed):

- https://www.knime.com/ - a generic ML platform (GUI-based ML workflow creation tool)
- https://www.youtube.com/watch?v=7IpvQW360js - Word analysis: Wordij (a 25-min demo)
- https://gephi.org/ - a network visualization tool

Thank you!