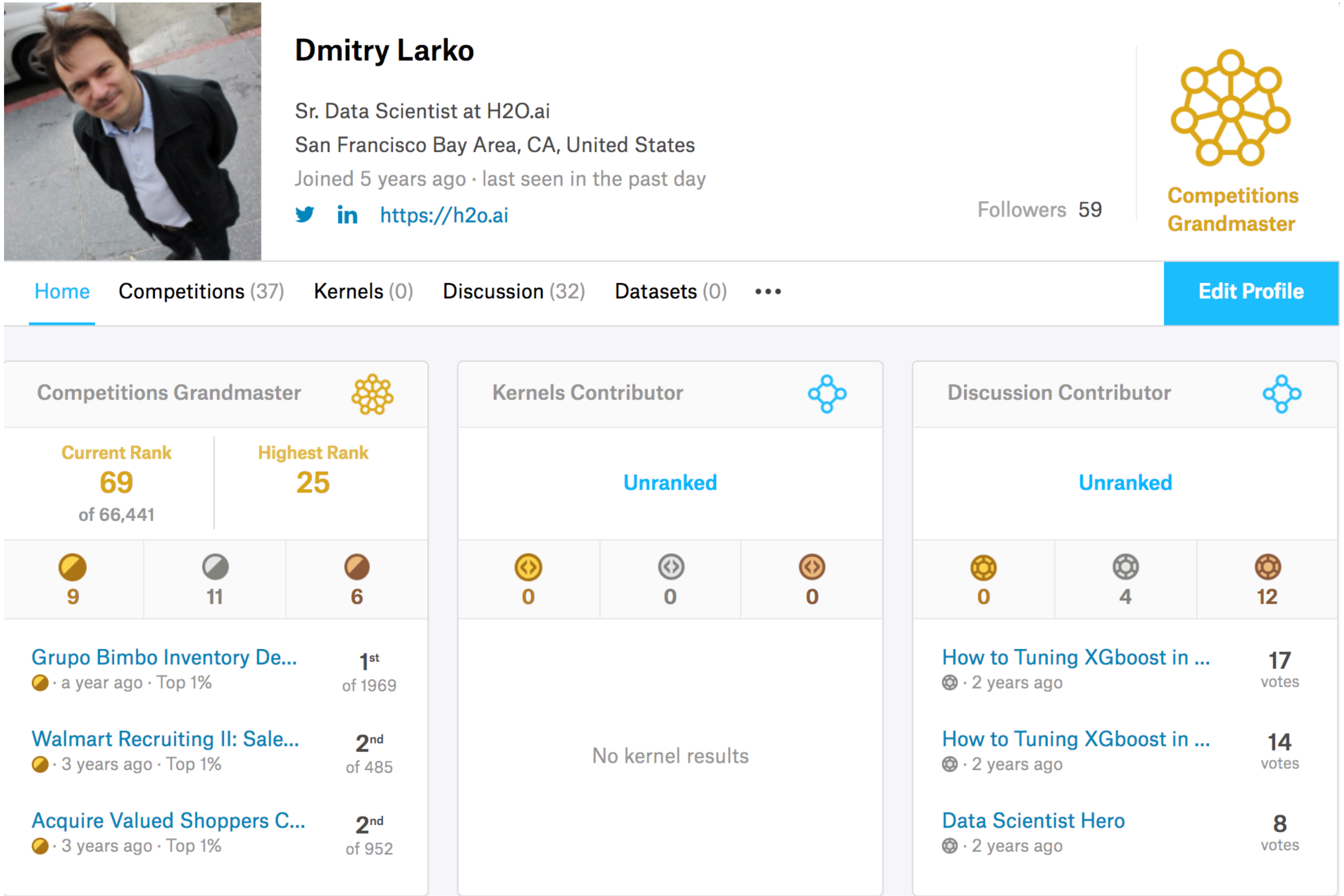
Feature Engineering

Dmitry Larko, Sr. Data Scientist @ H2O.ai

dmitry@h2o.ai

About me



Dmitry Larko

Sr. Data Scientist at H2O.ai
San Francisco Bay Area, CA, United States
Joined 5 years ago · last seen in the past day
https://h2o.ai

Followers 59

Competitions Grandmaster

Home Competitions (37) Kernels (0) Discussion (32) Datasets (0) •••

Edit Profile

Competitions Grandmaster

Current Rank	Highest Rank
69 of 66,441	25

●	●	●
9	11	6

Competition	Rank
Grupo Bimbo Inventory De... ● · a year ago · Top 1%	1st of 1969
Walmart Recruiting II: Sale... ● · 3 years ago · Top 1%	2nd of 485
Acquire Valued Shoppers C... ● · 3 years ago · Top 1%	2nd of 952

Kernels Contributor

Unranked

●	●	●
0	0	0

No kernel results

Discussion Contributor

Unranked

●	●	●
0	4	12

Topic	Votes
How to Tuning XGboost in ... ● · 2 years ago	17 votes
How to Tuning XGboost in ... ● · 2 years ago	14 votes
Data Scientist Hero ● · 2 years ago	8 votes

Feature Engineering

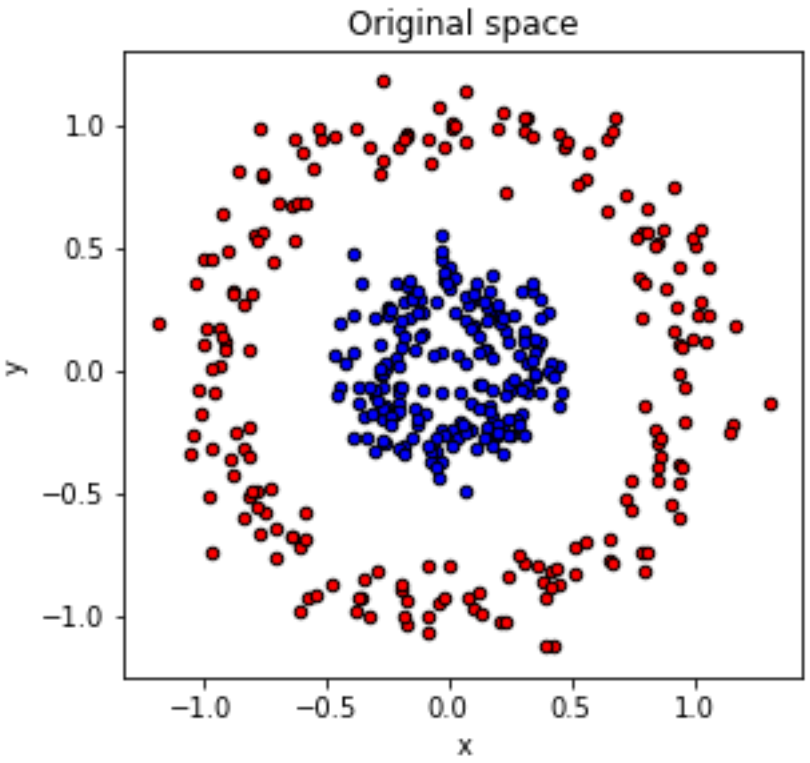
"Coming up with features is difficult, time-consuming, requires expert knowledge. “Applied machine learning” is basically feature engineering." ~ Andrew Ng

“… some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.” ~ Pedro Domingos

“Good data preparation and feature engineering is integral to better prediction” ~ Marios Michailidis (KazAnova), Kaggle GrandMaster, Kaggle #3, former #1

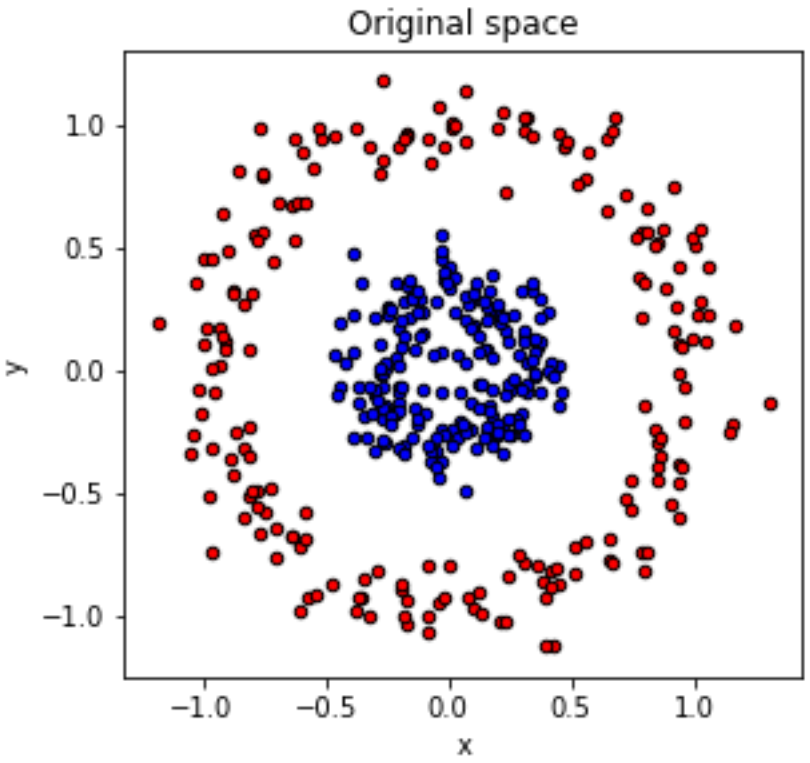
”*you have to turn your inputs into things the algorithm can understand*” ~ Shayne Miel, answer to “What is the intuitive explanation of feature engineering in machine learning?”

What is feature engineering

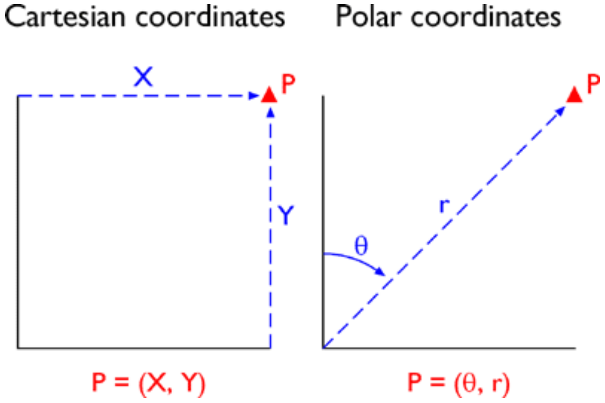


Not possible to separate using linear classifier

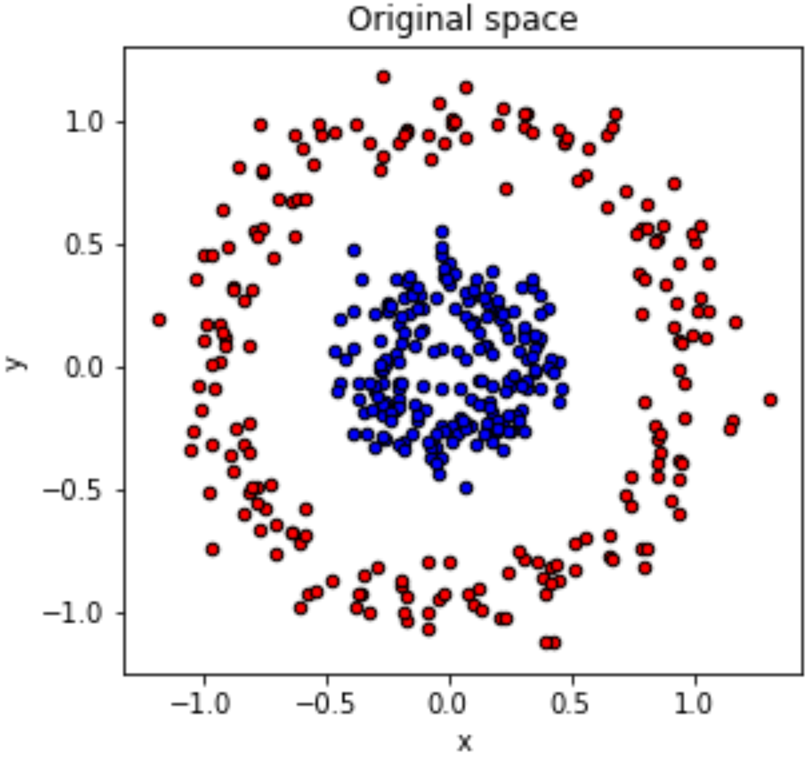
What is feature engineering

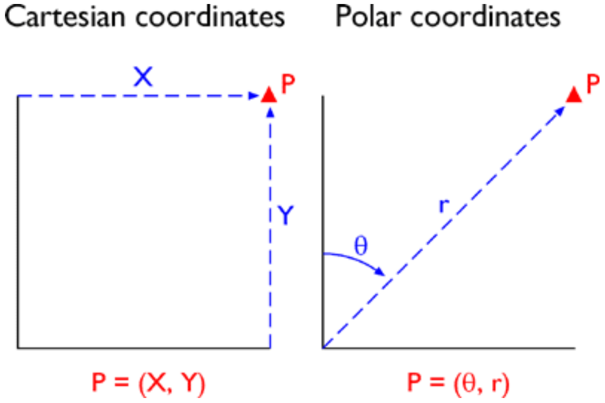


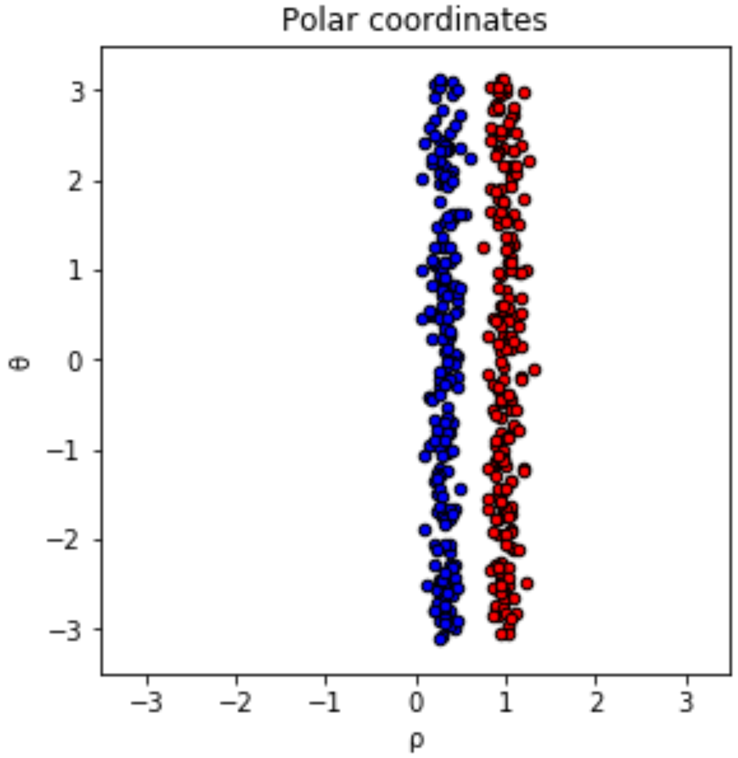
What if we use polar coordinates instead?



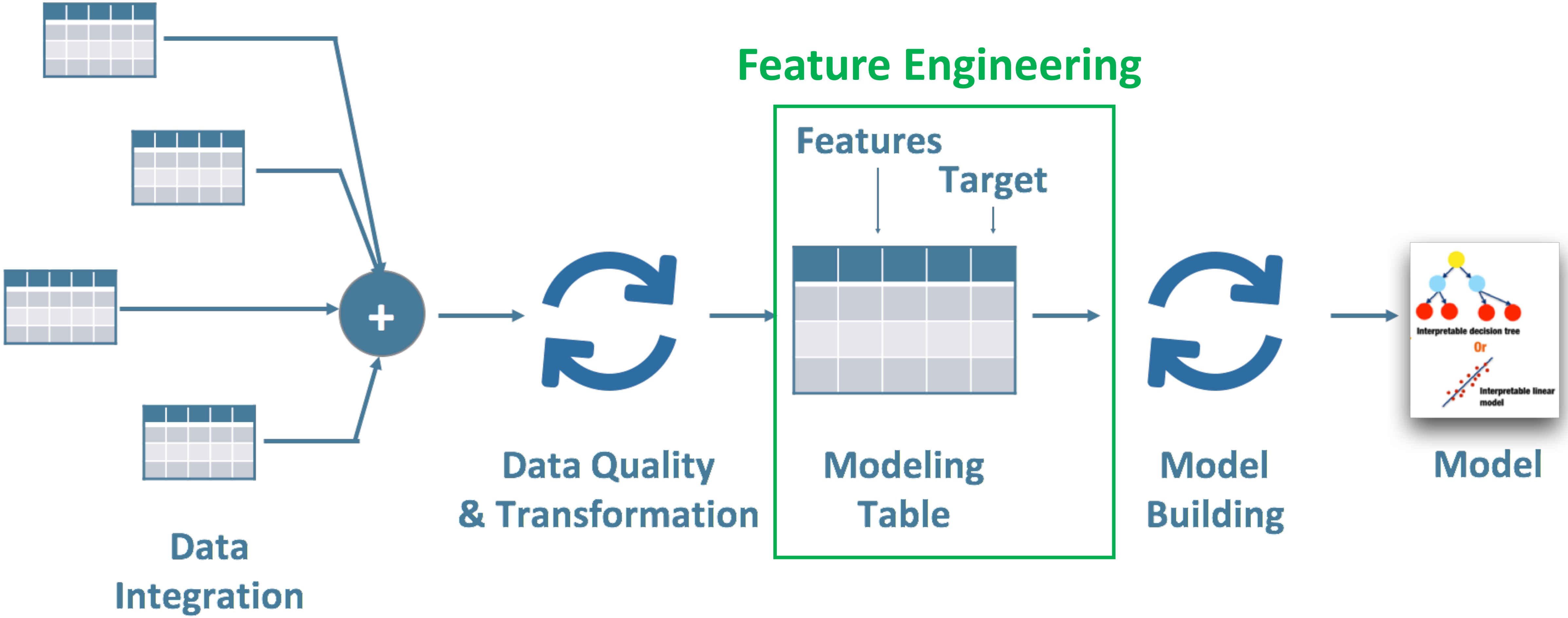
What is feature engineering







Typical Enterprise Machine Learning Workflow



Feature Engineering

Features

Target

Data Integration

Data Quality & Transformation

Modeling Table

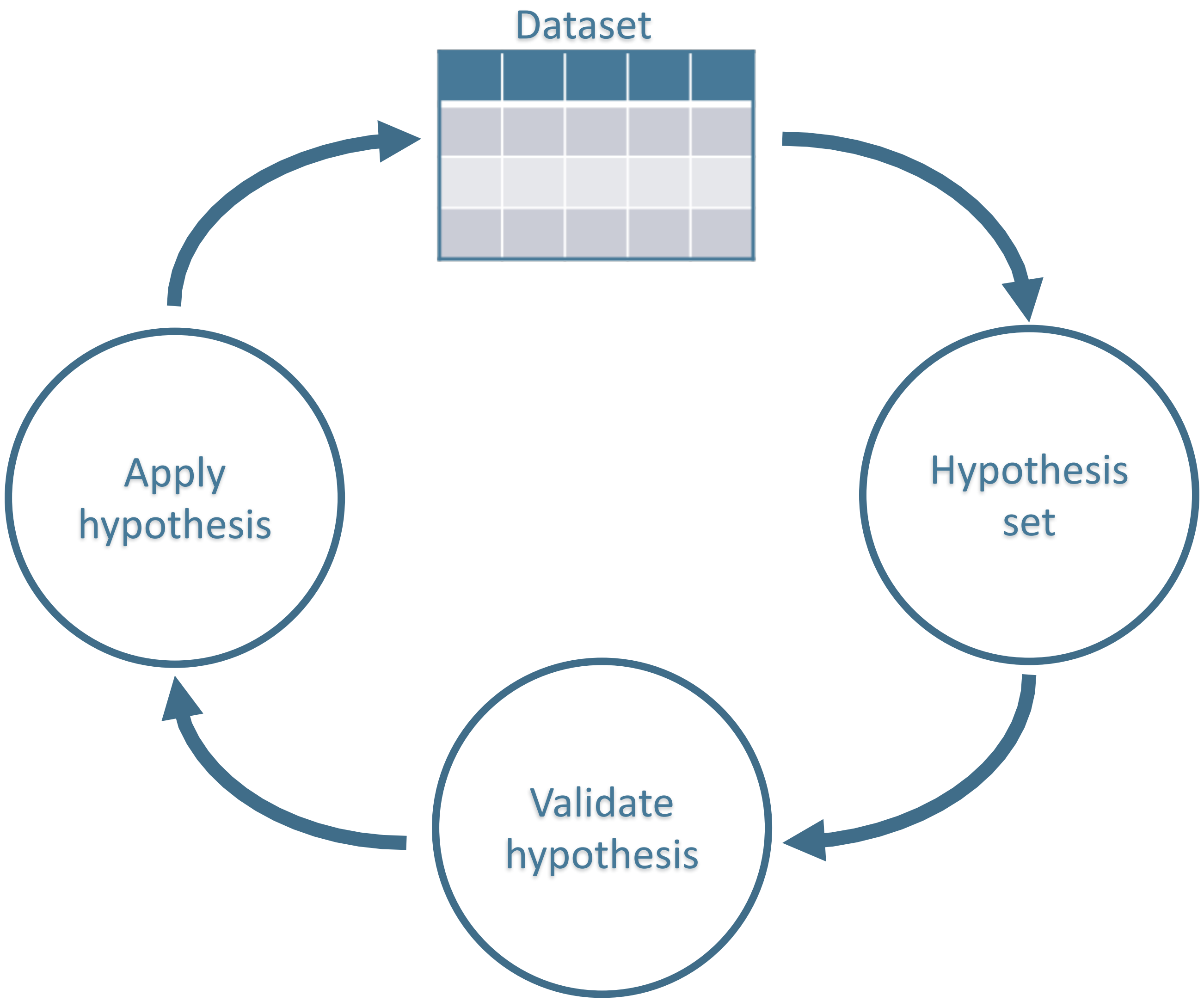
Model Building

Model

That’s NOT Feature Engineering

- Initial data collection
- Creating the target variable.
- Removing duplicates, handling missing values, or fixing mislabeled classes, it’s data cleaning.
- Scaling or normalization
- Feature selection, although I’m going to mention it in this talk ☺

Feature Engineering cycle



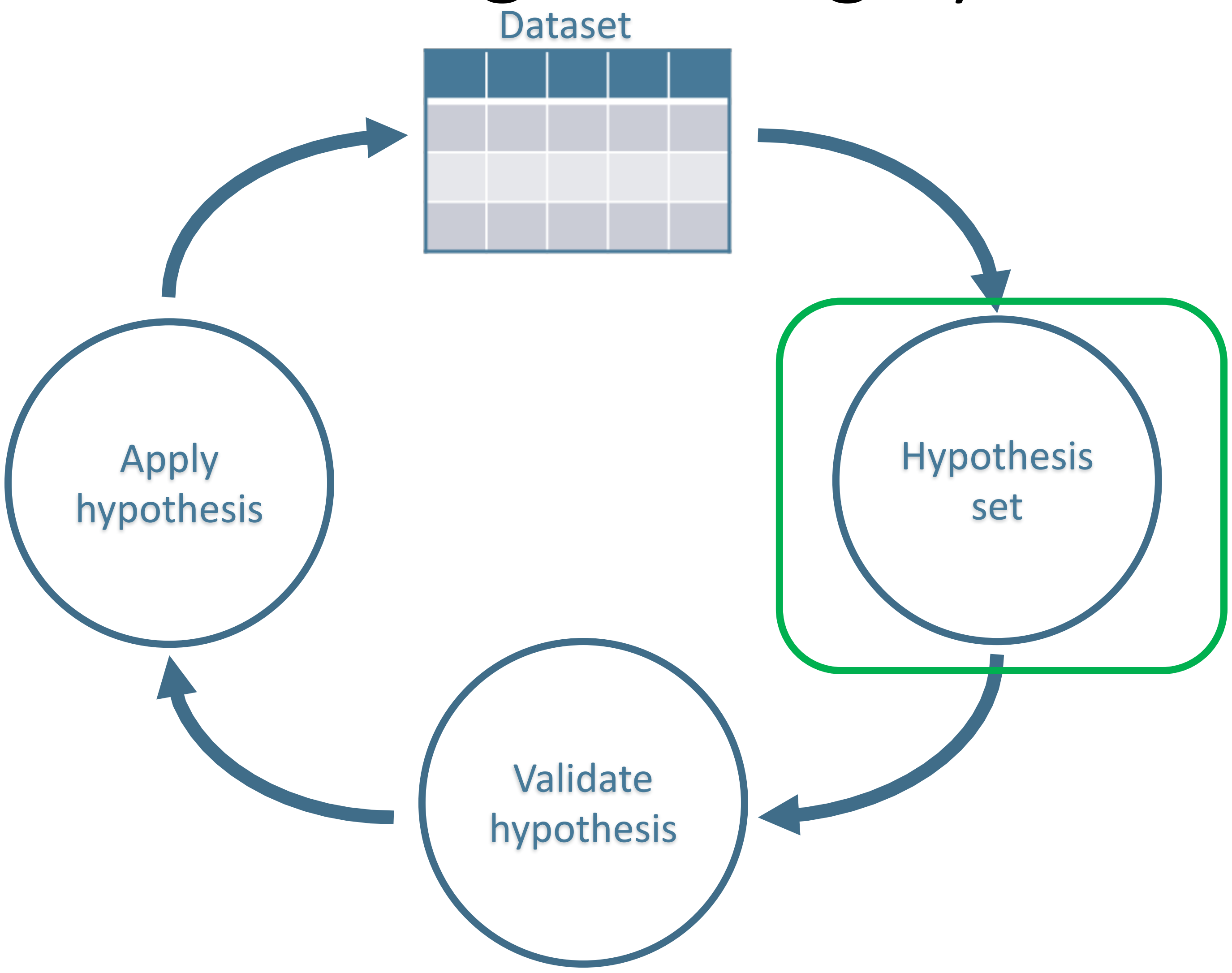
Dataset

Hypothesis set

Validate hypothesis

Apply hypothesis

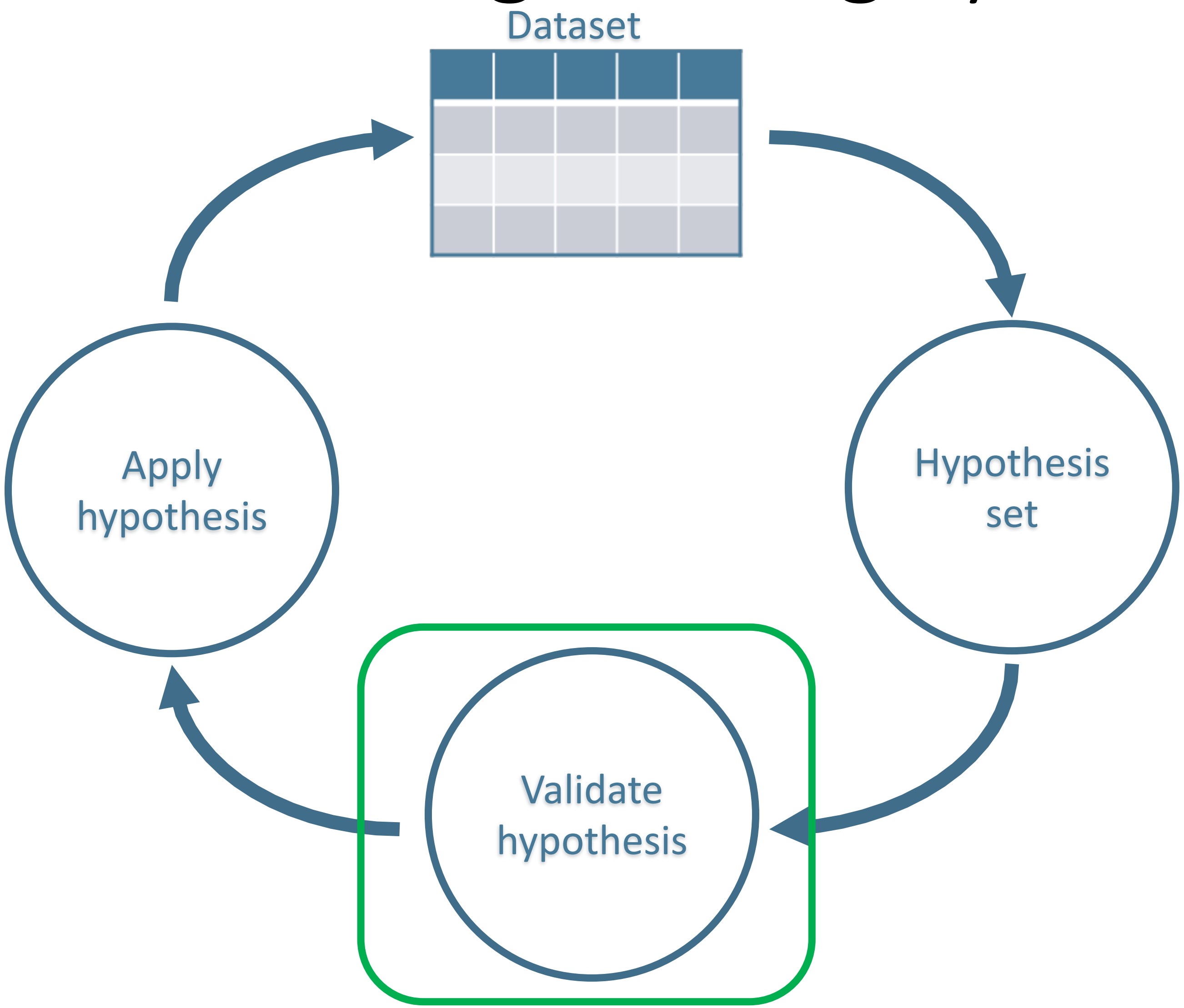
Feature Engineering cycle



How?

- Domain knowledge
- Prior experience
- EDA
- ML model feedback

Feature Engineering cycle



How?

- Cross-validation
- Measurement of desired metrics
- Avoid leakage

Why feature engineering is hard

- Powerful feature transformations (like target encoding) can introduce leakage when applied wrong
- Usually requires domain knowledge about how features interact with each other
- Time-consuming, need to run thousand of experiments

Feature Engineering

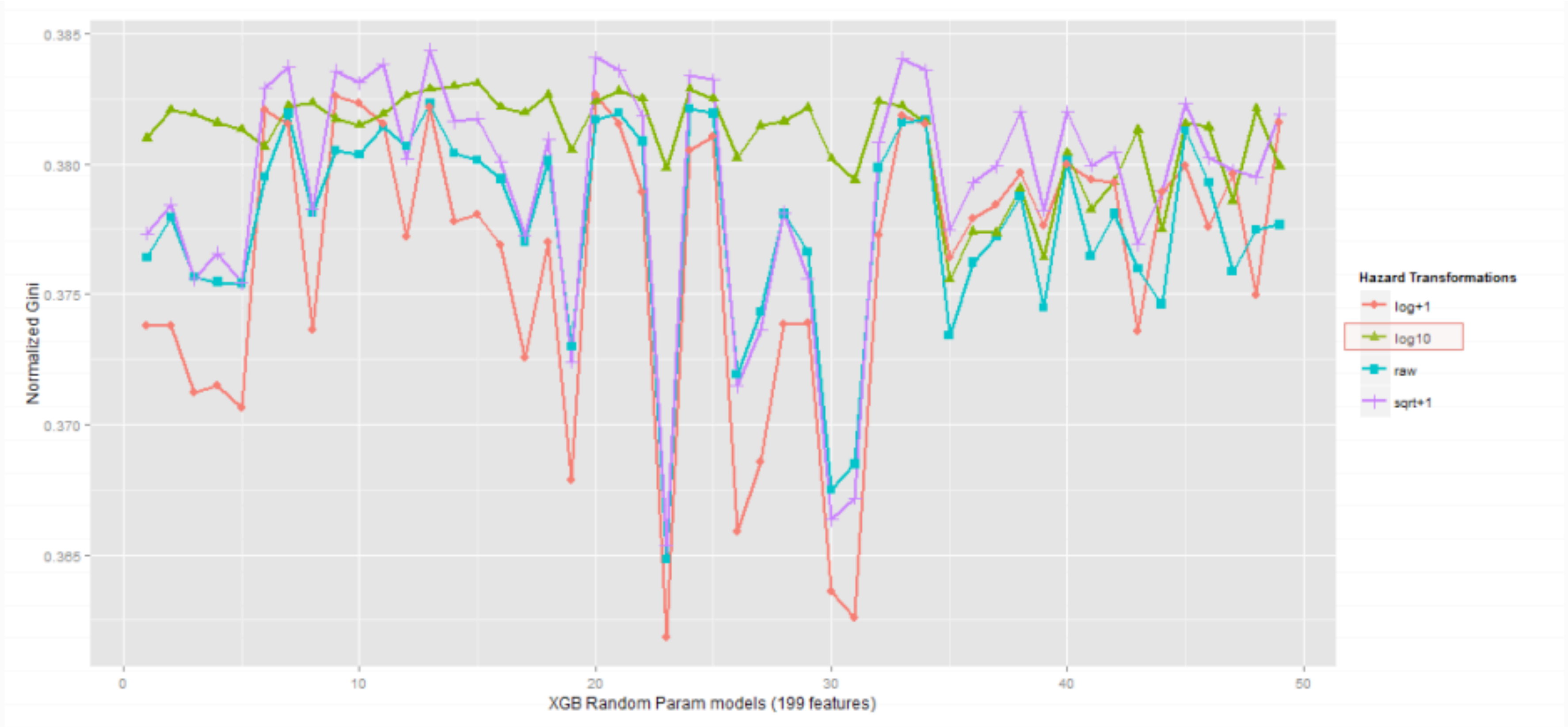
- Extract more new gold features, remove irrelevant or noisy features
  - Simpler models with better results
- Key Elements
  - Target Transformation
  - Feature Encoding
  - Feature Extraction

Target Transformation

- Predictor/Response Variable Transformation
  - Use it when variable shows a skewed distribution make the residuals more close to “normal distribution” (bell curve)
  - Can improve model fit
    - log(x), log(x+1), sqrt(x), sqrt(x+1), etc.

Target Transformation

In Liberty Mutual Group: Property Inspection Prediction



Different transformations might lead to different results

Feature Encoding

- Turn categorical features into numeric features to provide more fine-grained information
  - Help explicitly capture non-linear relationships and interactions between the values of features
  - Most of machine learning tools only accept numbers as their input
    - xgboost, gbm, glmnet, libsvm, liblinear, etc.

Feature Encoding

- Labeled Encoding
  - Interpret the categories as ordered integers (mostly wrong)
  - Python scikit-learn: LabelEncoder
  - Ok for tree-based methods
- One Hot Encoding
  - Transform categories into individual binary (0 or 1) features
  - Python scikit-learn: DictVectorizer, OneHotEncoder
  - Ok for K-means, Linear, NNs, etc.

Feature Encoding

- Labeled Encoding

A	0
B	1
C	2

Feature 1	Encoded Feature 1
A	0
A	0
A	0
A	0
B	1
B	1
B	1
C	2
C	2

Feature Encoding

- One Hot Encoding

A	**1**	0	0
B	0	**1**	0
C	0	0	**1**

Feature	Feature = A	Feature = B	Feature = C
A	**1**	0	0
A	**1**	0	0
A	**1**	0	0
A	**1**	0	0
B	0	**1**	0
B	0	**1**	0
B	0	**1**	0
C	0	0	**1**
C	0	0	**1**

Feature Encoding

- Frequency Encoding
    - Encoding of categorical levels of feature to values between 0 and 1 based on their relative frequency

A	0.44 (4 out of 9)
B	0.33 (3 out of 9)
C	0.22 (2 out of 9)

Feature	Encoded Feature
A	0.44
A	0.44
A	0.44
A	0.44
B	0.33
B	0.33
B	0.33
C	0.22
C	0.22

Feature Encoding - Target mean encoding

- Instead of dummy encoding of categorical variables and increasing the number of features we can encode each level as the mean of the response.

A	0.75 (3 out of 4)
B	0.66 (2 out of 3)
C	1.00 (2 out of 2)

Feature	Outcome	MeanEncode
A	1	0.75
A	0	0.75
A	1	0.75
A	1	0.75
B	1	0.66
B	1	0.66
B	0	0.66
C	1	1.00
C	1	1.00

Feature Encoding - Target mean encoding

- Also it is better to calculate weighted average of the overall mean of the training set and the mean of the level:

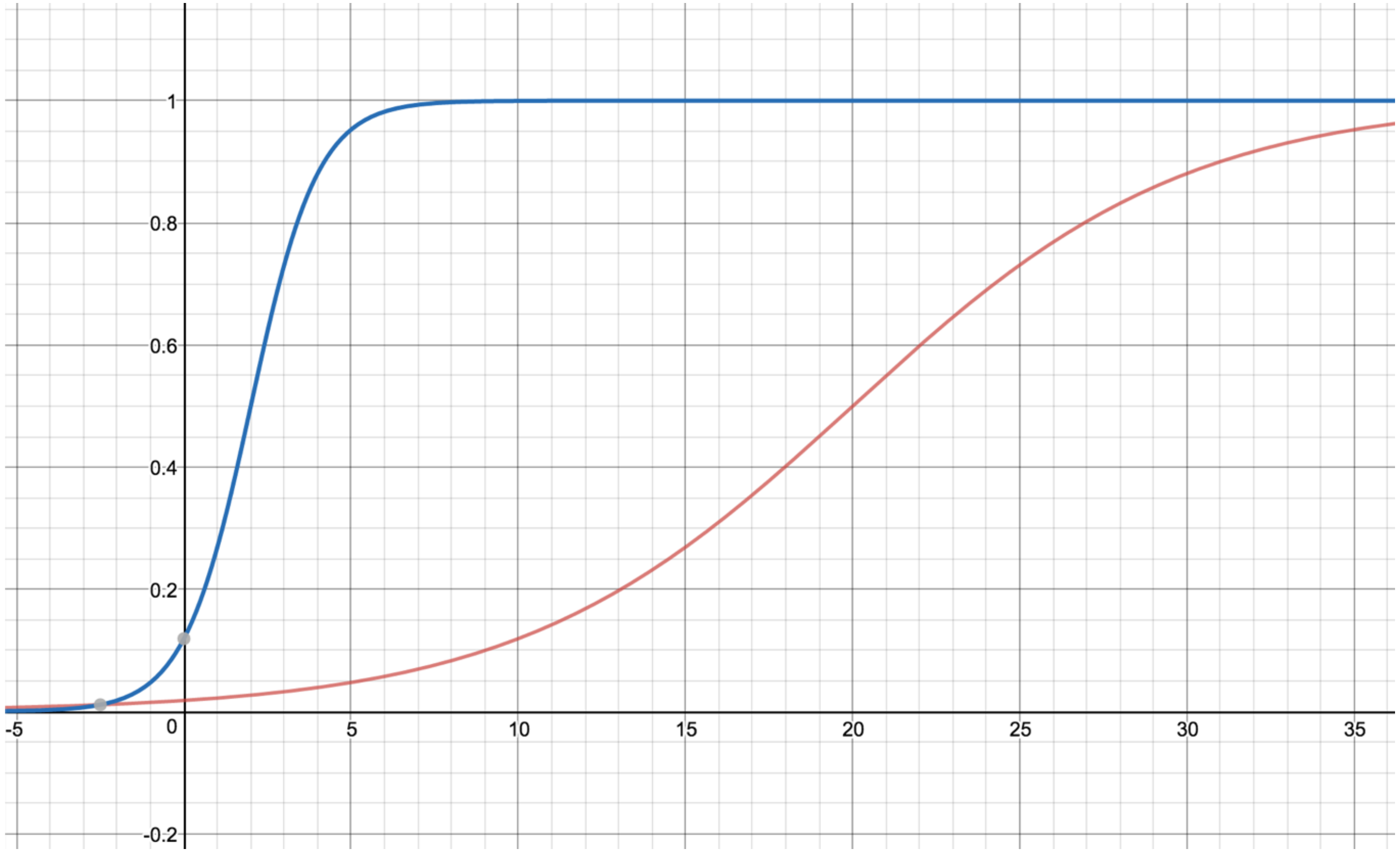
$$\lambda(n) * mean(level) + \left(1 - \lambda(n)\right) * mean(dataset)$$

- The weights are based on the frequency of the levels i.e. if a category only appears a few times in the dataset then its encoded value will be close to the overall mean instead of the mean of that level.

Feature Encoding – Target mean encoding $\lambda(n)$ example

$$\frac{1}{1 + exp(\frac{-(x-k)}{f})}$$

x = frequency
k = inflection point
f = steepness



https://www.desmos.com/calculator

Feature Encoding - Target mean encoding - Smoothing

$$\lambda = \frac{1}{1 + \exp(-\frac{(x-2)}{0.25})}$$

	x	level	dataset	λ	
A	4	0.75	0.77	0.99	0.99*0.75 + 0.01*0.77 = 0.7502
B	3	0.66	0.77	0.98	0.98*0.66 + 0.02*0.77 = 0.6622
C	2	1.00	0.77	0.5	0.5*1.0 + 0.5*0.77 = 0.885

$$\lambda = \frac{1}{1 + \exp(-\frac{(x-3)}{0.25})}$$

	x	level	dataset	λ	
A	4	0.75	0.77	0.98	0.98*0.75 + 0.01*0.77 = 0.7427
B	3	0.66	0.77	0.5	0.5*0.66 + 0.5*0.77 = 0.715
C	2	1.00	0.77	0.017	0.017*1.0 + 0.983*0.77 = 0.773

Feature	Outcome
A	1
A	0
A	1
A	1
B	1
B	1
B	0
C	1
C	1

Feature Encoding - Target mean encoding

- Instead of dummy encoding of categorical variables and increasing the number of features we can encode each level as the mean of the response.

A	0.75 (3 out of 4)
B	0.66 (2 out of 3)
C	1.00 (2 out of 2)

Feature	Outcome	MeanEncode
A	1	0.75
A	0	0.75
A	1	0.75
A	1	0.75
B	1	0.66
B	1	0.66
B	0	0.66
C	1	1.00
C	1	1.00

Feature Encoding - Target mean encoding

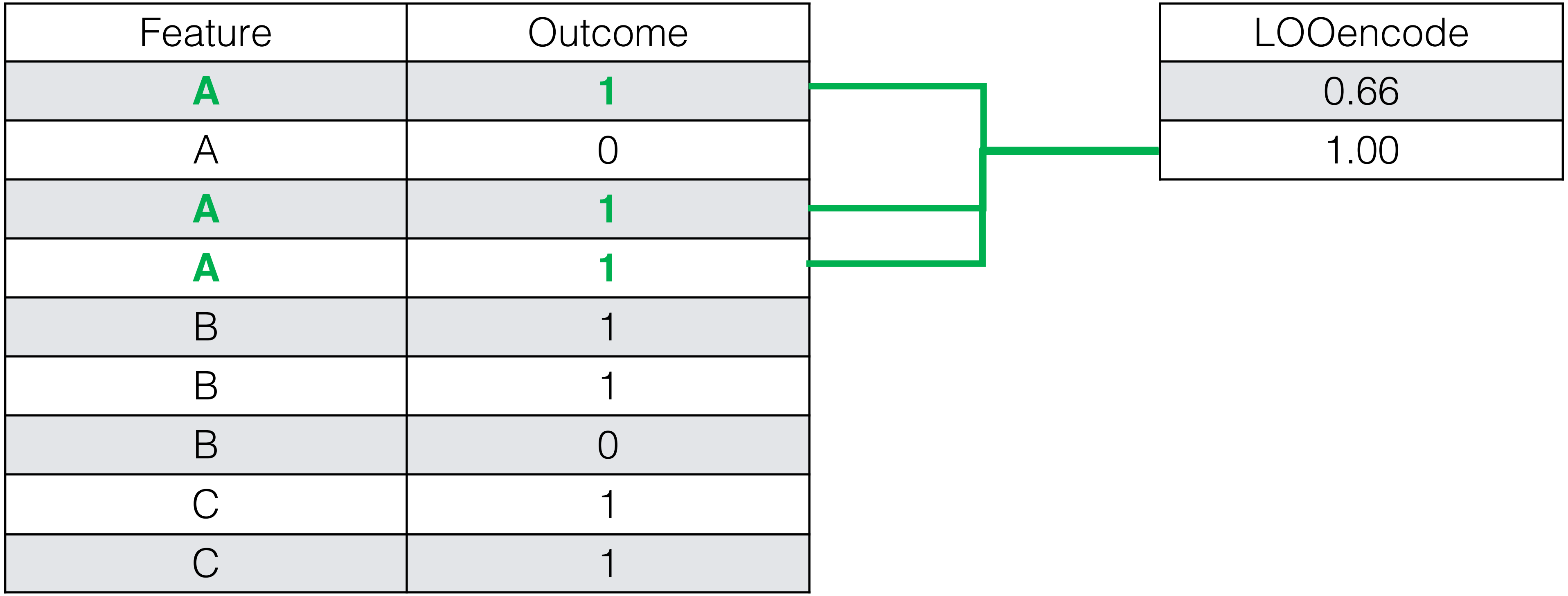
- To avoid overfitting we could use leave-one-out schema

Feature	Outcome
A	1
A	**0**
A	**1**
A	**1**
B	1
B	1
B	0
C	1
C	1

LOOencode
0.66

Feature Encoding - Target mean encoding

- To avoid overfitting we could use leave-one-out schema



Feature	Outcome
A	**1**
A	0
A	**1**
A	**1**
B	1
B	1
B	0
C	1
C	1

LOOencode
0.66
1.00

Feature Encoding - Target mean encoding

- To avoid overfitting we could use leave-one-out schema

Feature	Outcome
A	**1**
A	**0**
A	1
A	**1**
B	1
B	1
B	0
C	1
C	1

LOOencode
0.66
1.00
0.66

Feature Encoding - Target mean encoding

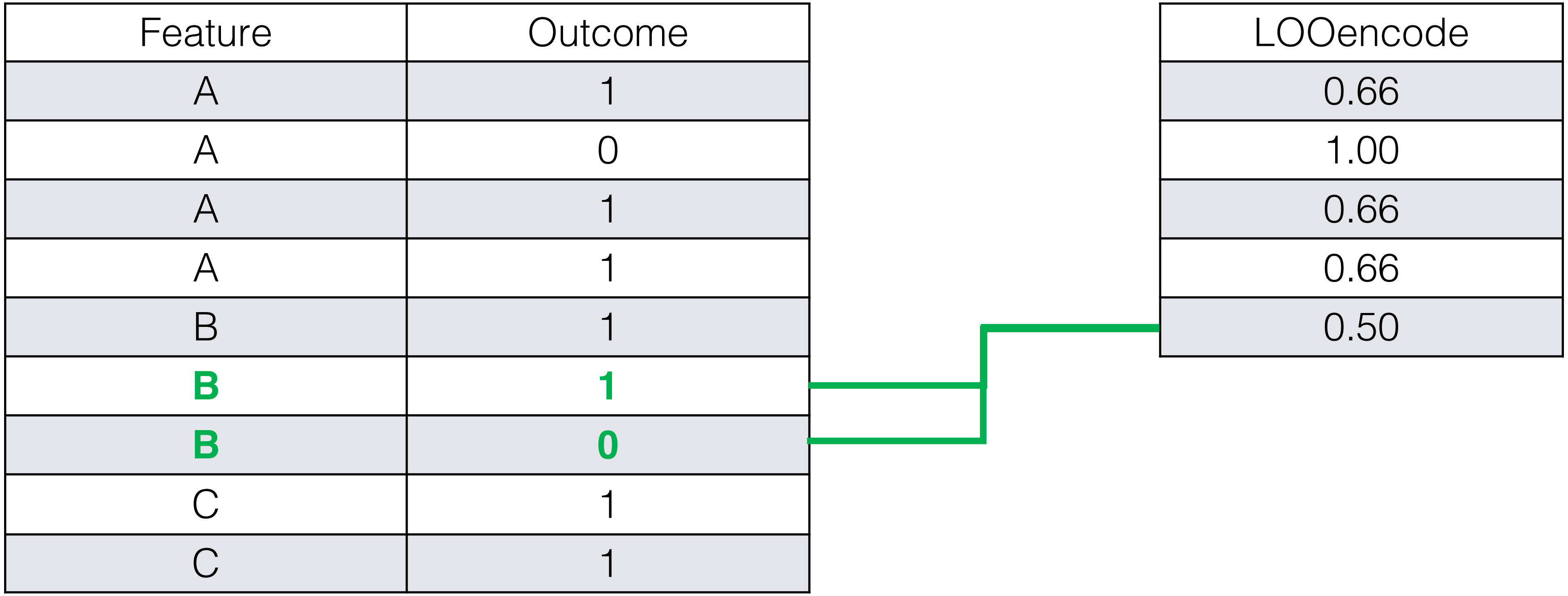
- To avoid overfitting we could use leave-one-out schema

Feature	Outcome
A	**1**
A	**0**
A	**1**
A	1
B	1
B	1
B	0
C	1
C	1

LOOencode
0.66
1.00
0.66
0.66

Feature Encoding - Target mean encoding

- To avoid overfitting we could use leave-one-out schema



Feature	Outcome
A	1
A	0
A	1
A	1
B	1
B	**1**
B	**0**
C	1
C	1

LOOencode
0.66
1.00
0.66
0.66
0.50

Feature Encoding - Target mean encoding

- To avoid overfitting we could use leave-one-out schema

Feature	Outcome
A	1
A	0
A	1
A	1
B	**1**
B	1
B	**0**
C	1
C	1

LOOencode
0.66
1.00
0.66
0.66
0.50
0.50

Feature Encoding - Target mean encoding

- To avoid overfitting we could use leave-one-out schema

Feature	Outcome
A	1
A	0
A	1
A	1
B	**1**
B	**1**
B	0
C	1
C	1

LOOencode
0.66
1.00
0.66
0.66
0.50
0.50
1.00

Feature Encoding - Target mean encoding

- To avoid overfitting we could use leave-one-out schema

Feature	Outcome
A	1
A	0
A	1
A	1
B	1
B	1
B	0
C	1
C	**1**

LOOencode
0.66
1.00
0.66
0.66
0.50
0.50
1.00
1.00

Feature Encoding - Target mean encoding

- To avoid overfitting we could use leave-one-out schema

Feature	Outcome
A	1
A	0
A	1
A	1
B	1
B	1
B	0
C	**1**
C	1

LOOencode
0.66
1.00
0.66
0.66
0.50
0.50
1.00
1.00
1.00

Feature Encoding – Weight of Evidence

$$WoE = \ln(\frac{\%\, non - events}{\%\, events})$$

To avoid division by zero

$$WoE_{adj} = \ln(\frac{\text{Number of non-events in a group + 0.5} \big/ \text{Number of non-events}}{\text{Number of events in a group + 0.5} \big/ \text{Number of events}})$$

Feature Encoding – Weight of Evidence

	Non-events	Events	% of non-events	% of events	WoE
A	1	3	50	42	$\ln\left(\frac{(1+0.5)/_{2}}{(3+0.5)/_{7}}\right)=0.4$
B	1	2	50	29	$\ln\left(\frac{(1+0.5)/_{2}}{(2+0.5)/_{7}}\right)=0.74$
C	0	2	0	29	$\ln\left(\frac{(0+0.5)/_{2}}{(2+0.5)/_{7}}\right)=-0.35$

Feature	Outcome	WoE
A	1	0.4
A	0	0.4
A	1	0.4
A	1	0.4
B	1	0.74
B	1	0.74
B	0	0.74
C	1	-0.35
C	1	-0.35

Feature Encoding – Weight of Evidence and Information Value

$$IV = \sum(\%\, non - events\ \ - \%\, events) * WoE$$

	Non-events	Events	% of non-events	% of events	WoE	IV
A	1	3	50	42	$\ln\left(\frac{(1+0.5)/_2}{(3+0.5)/_7}\right) = 0.4$	$(0.5\ \ - 0.42) * 0.4 = 0.032$
B	1	2	50	29	$\ln\left(\frac{(1+0.5)/_2}{(2+0.5)/_7}\right) = 0.74$	$(0.5\ \ - 0.29) * 0.4 = 0.084$
C	0	2	0	29	$\ln\left(\frac{(0+0.5)/_2}{(2+0.5)/_7}\right) = -0.35$	$(0\ \ - 0.29) * -0.35 = 0.105$
						0.221

Feature	Outcome	WoE
A	1	0.4
A	0	0.4
A	1	0.4
A	1	0.4
B	1	0.74
B	1	0.74
B	0	0.74
C	1	-0.35
C	1	-0.35

Feature Encoding – Weight of Evidence and Information Value

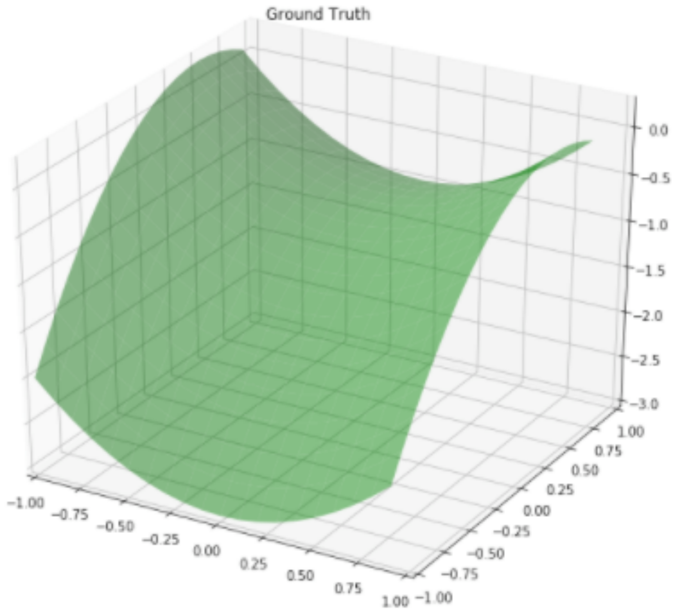
Information Value	Variable Predictiveness
Less than 0.02	Not useful for prediction
0.02 to 0.1	Weak predictive Power
0.1 to 0.3	Medium predictive Power
0.3 to 0.5	Strong predictive Power
>0.5	Suspicious Predictive Power

Feature Encoding – Numerical features

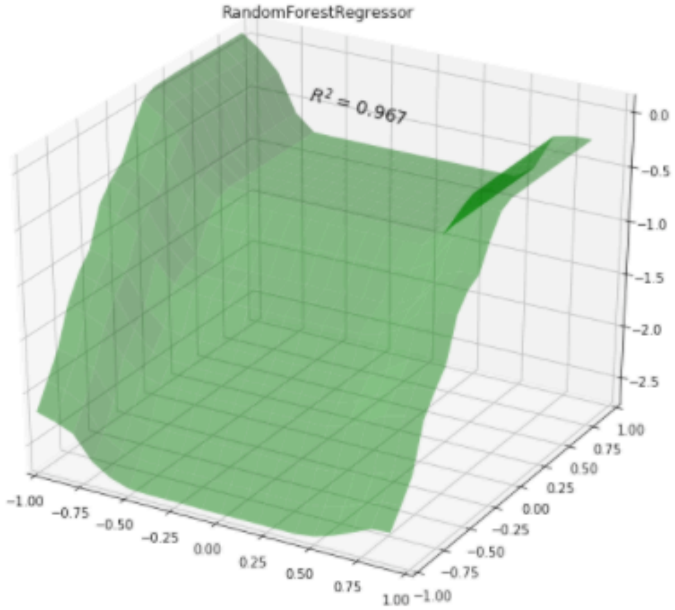
- Binning using quantiles (population of the same size in each bin) or histograms (bins of same size)
  - Replace with bin's mean or median
  - Treat bin id as a category level and use any categorical encoding schema
- Dimensionality reduction techniques – SVD and PCA
- Clustering and using cluster IDs or/and distances to cluster centers as new features

Feature Interaction

- $y = \ x_1{}^2 - x_2{}^2 + x_2 - 1$

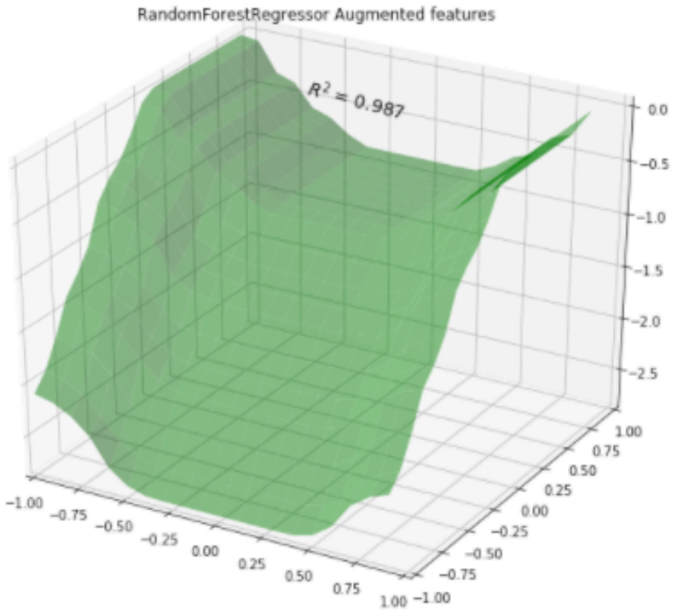


Ground Truth



RandomForestRegressor

Adding $x_1{}^2$ and $x_2{}^2$ as new features



RandomForestRegressor Augmented features

Feature Interaction – how to find?

- Domain knowledge
- ML algorithm behavior (for example analyzing GBM splits or linear regressor weights)

Feature Interaction – how to model?

- Clustering and kNN for numerical features
- Target encoding for pairs (or even triplets and etc.) of categorical features
- Encode categorical features by stats of numerical features

Feature Extraction

- There usually have some meaningful features inside existing features, you need to extract them manually
- Some examples
  - Location
    - Address, city, state and zip code .... (categorical or numeric)
  - Time
    - Year, month, day, hour, minute, time ranges, ... (numeric)
    - Weekdays or weekend (binary)
    - Morning, noon, afternoon, evening, ... (categorical)
  - Numbers
    - Turn age numbers into ranges (ordinal or categorical)

Feature Extraction: Textual Data

- Bag-of-Words: extract tokens from text and use their occurrences (or TF/IDF weights) as features
- Require some NLP techniques to aggregate token counts more precisely
  - Split token into sub-tokens by delimiters or case changes
  - N-grams at word (often 2-5 grams) or character level
  - Stemming for English words
  - Remove stop words (not always necessary)
  - Convert all words to lower case
- Bag-of-Words Tools
  - Python: CountVectorizer, TfidfTransformer in scikit-learn package

Feature Extraction: Textual Data

- Deep Learning for textual data
  - Turn each token into a vector of predefined size
  - Help compute "semantic distance" between tokens/words
    - For example, the semantic distance between user query and product titles in search results (how relevant?)
  - Greatly reduce the number of text features used for training
    - Use average vector of all words in given text
    - Vector size: 100~300 is often enough
  - Tools
    - Word2vec, Doc2vec, GloVe

End