How to win a Kaggle competition

Chris Chen@Calgary Data Science Academy

Prerequisites

- Basic knowledge in
  - Machine learning: linear regression, decision tree, model validation, bias/variance trade off etc.
  - Programming language, Python preferably.
  - Statistics, linear algebra, probability.
  - Prior Kaggle experience is not required but would be much beneficial

Environment

- Kaggle notebook (recommended)
  - Imagine a runnable Github repository with most of the machine learning packages pre-installed, powered by Google Cloud, and a FREE GPU!
- Local environment
  - Anaconda 3.7
  - XGBoost/ LightGBM/ TensorFlow/ Keras/ BayersianOptimization etc.

Syllabus

- Week 1: Exploratory Data Analysis and Basic Feature Engineering
- Week 2: Advanced Feature Engineering
- Week 3: Model Hyper-parameter Tuning/
- Week 4: Model ensemble/ Introduction to Multiple Layer Perceptron
- Week 5: Case Study: Avito Demand Prediction Challenge and Santander Customer Transaction Prediction

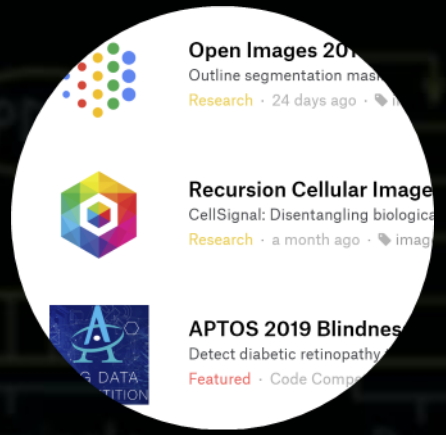
Your Home for Data Science

Kaggle helps you learn, work, and play

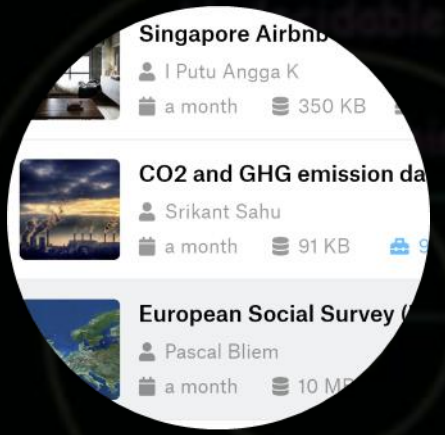
Create an account or Host a competition

- BATTLEFIELD for 120K talented data scientists from 190 countries
- Largest and most influencing data science COMMUNITY
- CRADLE of pioneering algorithms and approaches
- Best PLACE to learn data science in practice.

What Kaggle has to offer



Competitions



Datasets

Notebooks

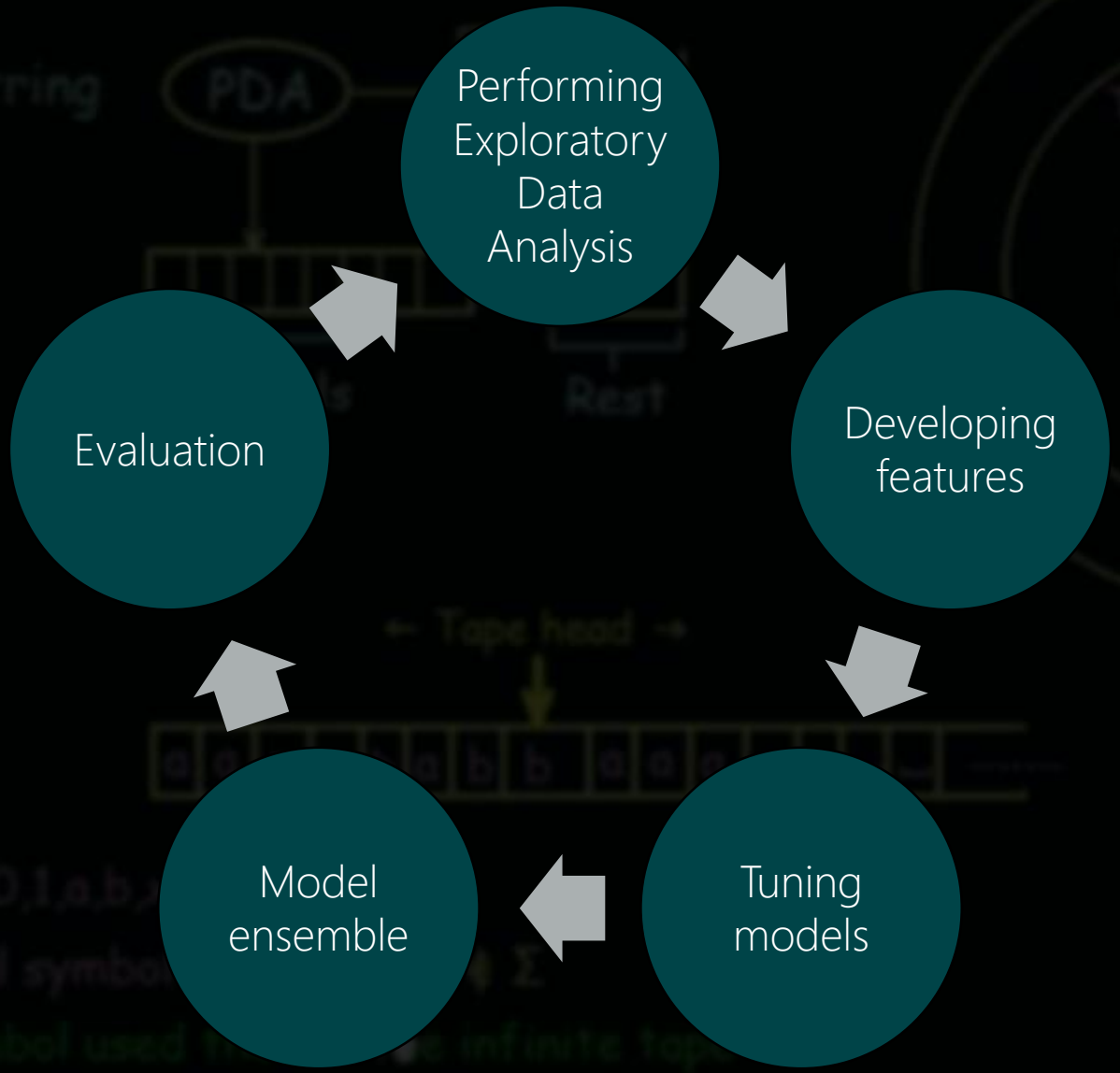
Courses

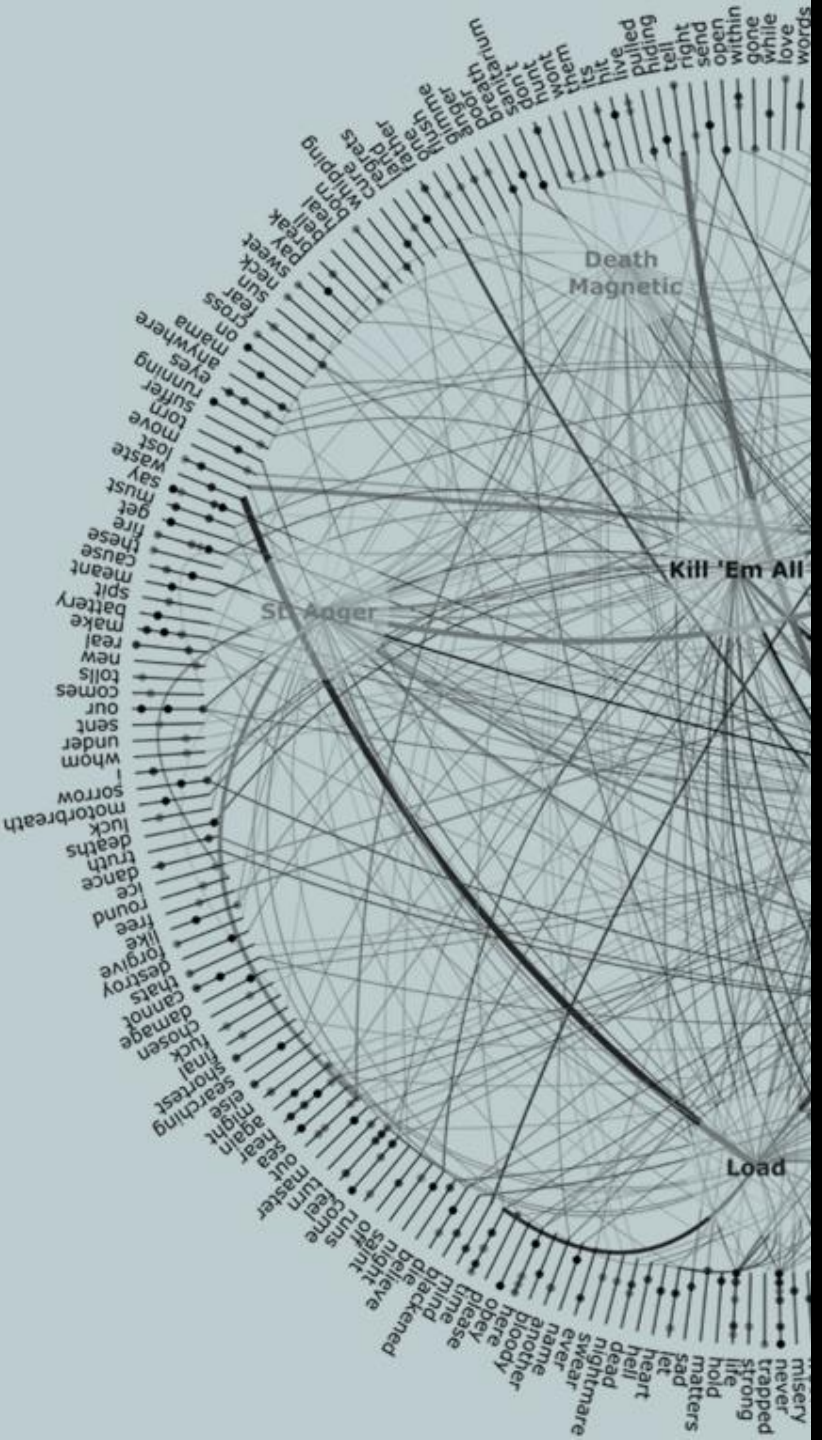
Two Sigma Connect: : Rental Listing Inquiries

How much interest will a new rental listing on RentHop receive?

Finding apartments for rent is usually a challenging task. RentHop, one of many websites that try to make the process more convenient, tries to help renters by sorting their listing by quality using data. To improve their methods and to better understand the needs and preferences of renters, **RentHop**, along with **Two Sigma**, hosted a competition on **Kaggle** to predict the number of inquiries a new listing will receive based on its features.

Data science competition process





Exploratory Data Analysis

- Maximize insights
- Uncover hidden patterns
- Extract important features
- Detect outliers and anomalies
- Validate underlying assumptions

Summary of data

Train data: 49,352 entries Test data: 74,659 entries

- bathrooms: number of bathrooms
- bedrooms: number of bedrooms
- building_id
- created
- description
- display_address
- features: a list of features about this apartment
- latitude
- listing_id
- longitude
- manager_id
- photos: a list of photo links. You are welcome to download the pictures yourselves from renthop's site, but they are the same as imgs.zip.
- price: in USD
- street_address
- interest_level: this is the target variable. It has 3 categories: 'high', 'medium', 'low'

Descriptive Data Analysis

Statistical measures of the samples

- Measures of central tendency/average
  - Mean
  - Median
  - Mode
- Measures of spread/ dispersion
  - Range (min, max, quartiles)
  - Variance
  - Standard deviation
- Measures of relationship
  - Pearson correlation
  - Spearman correlation

Tool to use (Python):

- scipy
- Pandas
- StatModel

Visual Data Analysis (Data Visualization)

- Distribution plot
- Box plot
- Violin plot
- Scatter (bubble) plot
- Pair plot

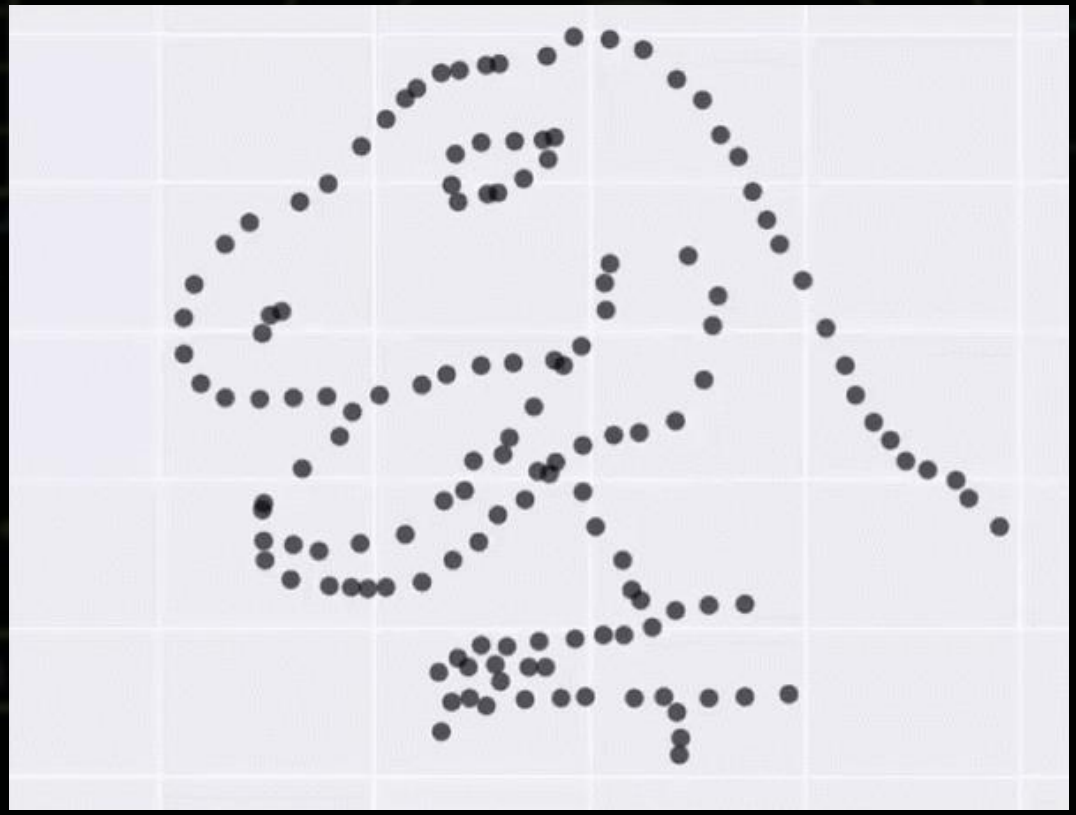
Tool to use:

- Matplotlib
- Seaborn
- Bokeh
- plotly

Why visualizing data is important?

Suppose we are analyzing a data set with following statistics, what do you think the data would look like?

X Mean :	54.26
Y Mean :	47.83
X SD :	16.76
Y SD :	26.93
Corr. :	-0.06



Retrieved from https://www.autodeskresearch.com/publications/samestats

Feature Engineering

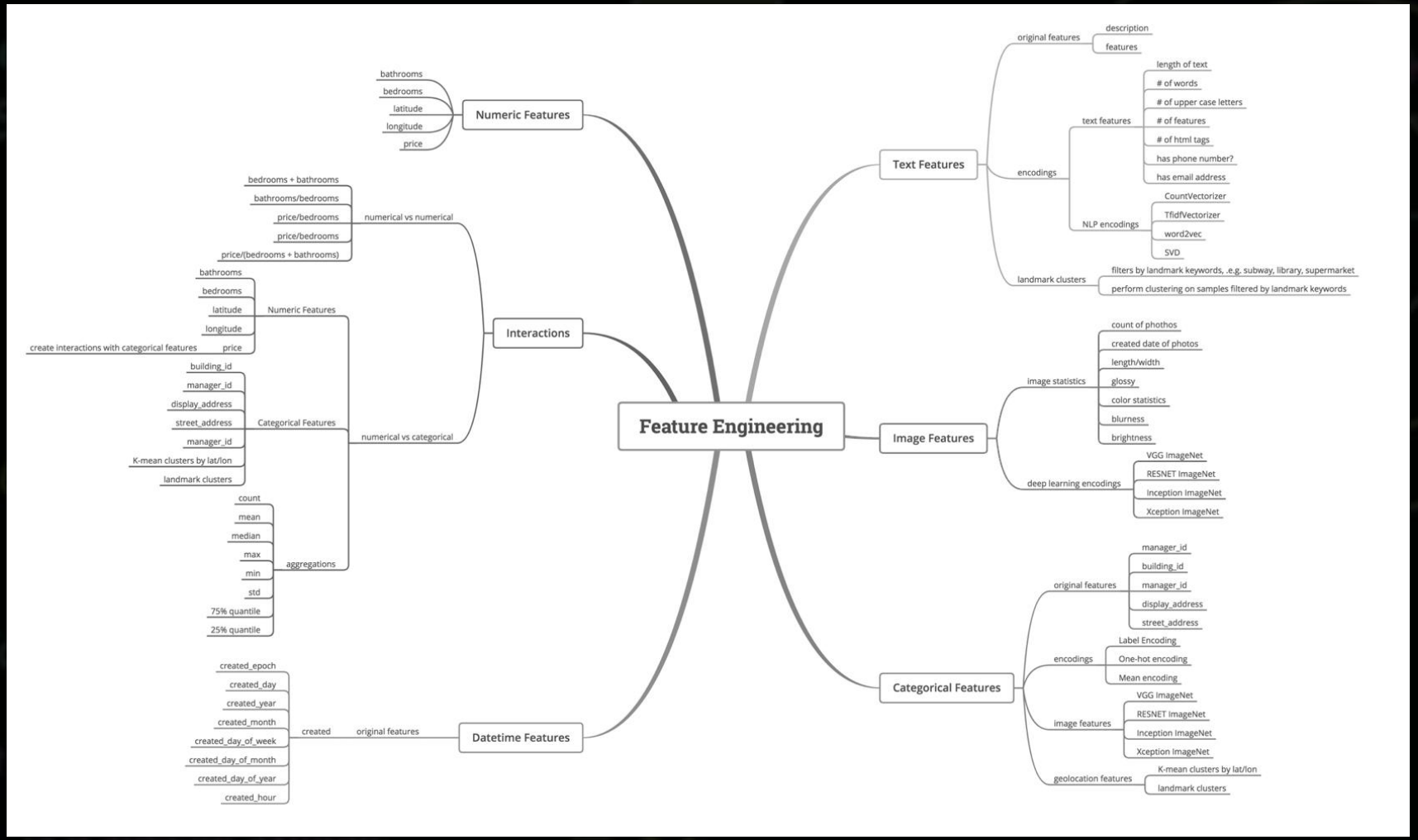
“Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering”

Andrew Ng

Feature Engineering Tasks

- Cleaning up data
- Converting data into consumable formats
- Creating more informative features
- Selecting most representative features
- Note: any transformations applied on the training data have to be applied on the validation and test consistently

Overview of feature engineering for TwoSigma



Dealing with missing values

- Remove the feature when missing ratio is high
- Imputation:
  - Mean imputation for normally distributed features.
  - Median imputation for skewed distribution (reduce the impacts of outliers).
  - Interpolation for time series or sequence data.
  - Majority imputation for categorical features.
  - Model-based imputation
    - Bayesian imputation
    - Supervised-learning for imputation
- Using particular algorithms, e.g. XGBoost, LightGBM
  - Question: why using -99999 for missing value works well for Random Forest?
  - Missing values are gracefully treated by considering missing value as candidate for tree splitting
- Understanding the root cause for missing value is CRITICAL!

One-hot encoding for categorical features

One Hot Encoding

- Encode categorical features using a one-hot aka one-of-K scheme
- The most adaptive encoding approach. Works for both linear models and tree models
- sklearn.preprocessing.OneHotEncoder
- Categorical text features need to be transformed to integers prior to OHE.
- Sparse matrix is the preferable format for outputs

Sample	Category	Numerical
1	Human	1
2	Human	1
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4

Sample	Human	Penguin	Octopus	Alien
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	0	0	1	0
7	0	0	0	1

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
enc.fit([[0, 0, 3], [1, 1, 0], [0, 2, 1], [1, 0, 2]])
```

Label encoding for categorical features

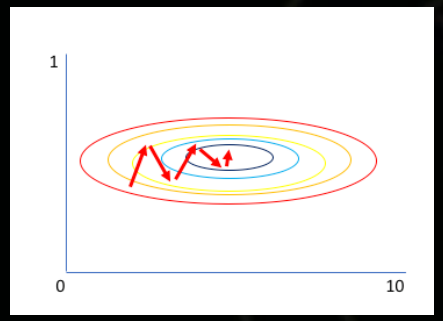
Label Encoding

- Encode categorical features with integers.
- Works for tree models such as decision tree, Random Forest, GBDT, GBM, XGBoost and LightGBM
- Definitely not work for linear models (linear regression, logistic regression, GLM, linear kernel SVM, neural networks etc.)
- sklearn.preprocessing.OneHotEncoder
- Question: why label encoding performs better than OHE for features with large number of categorical (e.g. postal code) for tree-based models?

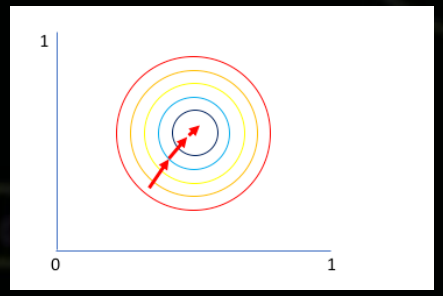
Sample	Category	Numerical
1	Human	1
2	Human	1
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4

Scaling numeric features

- Helps speed up model training.
- Can improve model accuracy in most cases.
- Normalization
  - Scale individual samples to have unit norm (e.g. 0-1)
  - sklearn.preprocessing.Normalizer
- Standardization
  - To standardize features by removing the mean and scaling to unit variance
  - sklearn.preprocessing.StandardScaler
- Not applicable for tree models



Unnormalized: converges slowly as larger parameter dominates the updates

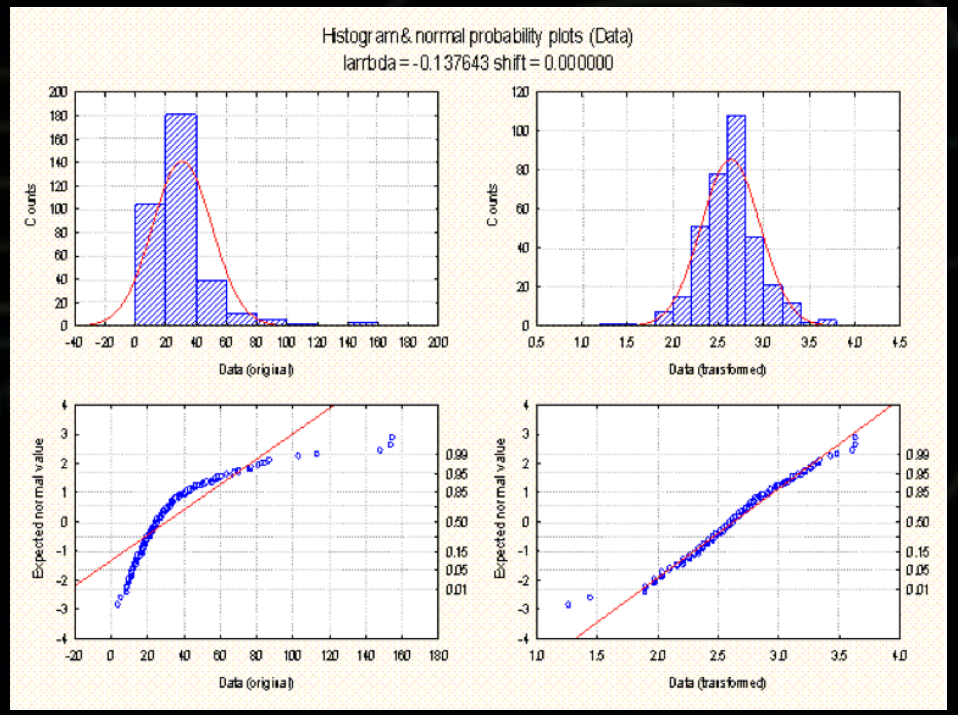


Normalized: both parameters can be updated in similar proportions.

Image retrieved from https://stats.stackexchange.com/questions/322822/how-normalizing-helps-to-increase-the-speed-of-the-learning

Adjusting the skewness for numeric features

- Box-Cox transformation
  - Reduce data skewness by shifting the distribution.
  - scipy.stats.boxcox()
  - Not applicable for tree models



Discretizing numeric features

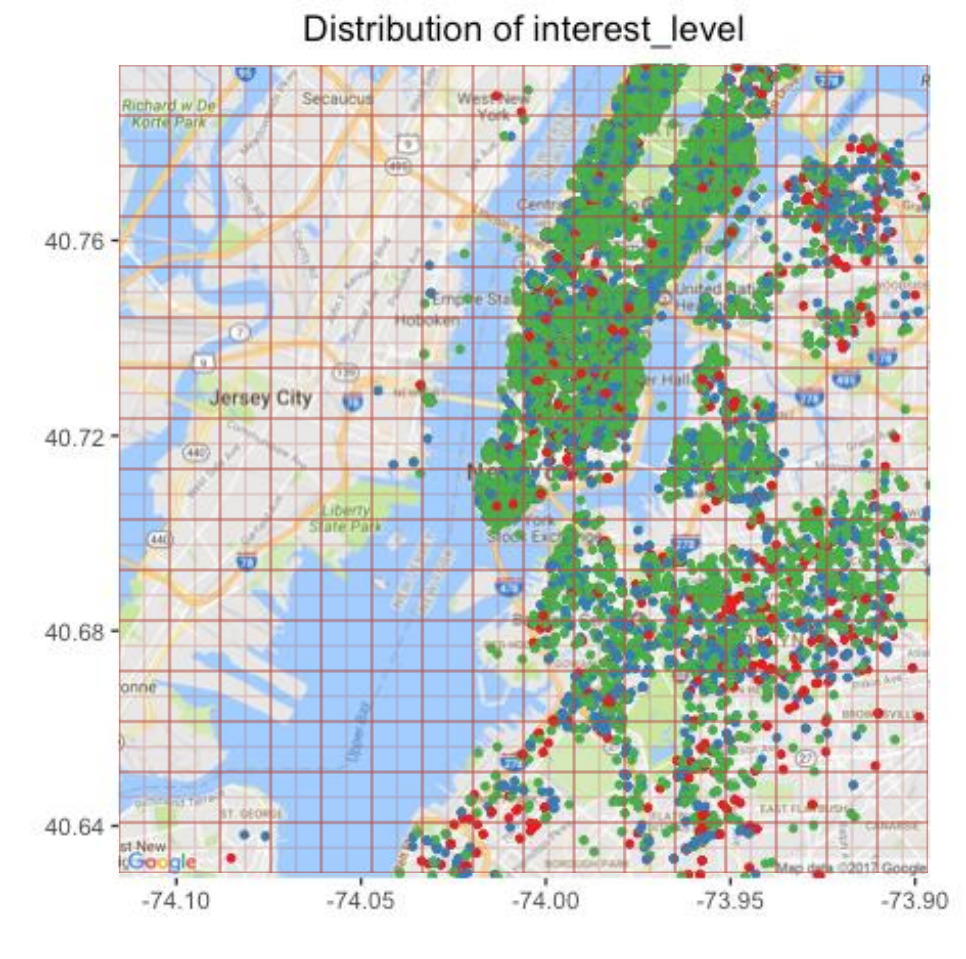
- Convert continuous values into discrete values, then apply one-hot encoding. E.g. convert age to age groups (0-10, 10-20, 20-30 etc.).
- Easy for interpretation.
- Robust to missing values and outliers.
- Mingle non-linearity to linear models.
- Sparse features are computationally easier for matrix product operation hence can be used to generate feature interactions more efficiently, e.g. age vs. income.
- Widely used by industry for linear models.
- Question: can discretized numeric features be used along with the raw ones?

Decomposing date/time features

Enumerate all the properties of a time-stamp, such as:

- Seconds elapsed from Unix epoch
- Minutes elapsed for the day.
- Hour of day.
- Business hours or not.
- Weekend or not.
- Season of the year.
- Holidays.
- Some of the properties, e.g. hour, can also be treated as categorical features.

Geolocation features



- Latitude and longitude can be considered as numeric features.
- Geo areas at different levels are very useful and can be used as categorical features. E.g. state, city, community, postal code etc.
- Similarly, grids based on various scales are helpful.
- Distance to landmarks and/or POIs.

Representing text by word frequencies

Token count encoding

- Convert a collection of text documents to a matrix of token (word) counts
- sklearn.feature_extraction.text.CountVectorizer()

	text
0	I have a friend whose name is Kevin
1	Kevin's best friend is Joe
2	Joe is not a friend of mine
3	Kevin and Joe both know another Kevin

	best	friend	joe	kevin	know
0	0	1	0	1	0
1	1	1	1	1	0
2	0	1	1	0	0
3	0	0	1	2	1

```
from sklearn.feature_extraction.text import CountVectorizer
documents = ["I have a friend whose name is Kevin",
             "Kevin's best friend is Joe",
             "Joe is not a friend of mine",
             "Kevin and Joe both know another Kevin"
            ]
encoder = CountVectorizer(stop_words='english', max_features=200)
text_features =encoder.fit_transform(documents)
text_features = pd.DataFrame(text_features.todense(), columns=encoder.vocabulary_.keys())
```

TF-IDF: penalizing highly frequent tokens

TF-IDF encoding

- Transform a count matrix to a normalized tf or tf-idf representation
- Term Frequency
- Raw count of a term in a document, i.e. the number of times that term t occurs in document dInverse document frequency
- Inverse document frequency
- How much information the word provides, that is, whether the term is common or rare across all documents
- sklearn.feature_extraction.text.TfidfVectorizer()

	text
0	I have a friend whose name is Kevin
1	Kevin's best friend is Joe
2	Joe is not a friend of mine
3	Kevin and Joe both know another Kevin

	best	friend	joe	kevin	know
0	0.000000	0.707107	0.000000	0.707107	0.000000
1	0.670819	0.428175	0.428175	0.428175	0.000000
2	0.000000	0.707107	0.707107	0.000000	0.000000
3	0.000000	0.000000	0.366260	0.732521	0.573818

```
from sklearn.feature_extraction.text import CountVectorizer
documents = ["I have a friend whose name is Kevin",
             "Kevin's best friend is Joe",
             "Joe is not a friend of mine",
             "Kevin and Joe both know another Kevin"
            ]
encoder = CountVectorizer(stop_words='english', max_features=200)
text_features =encoder.fit_transform(documents)
text_features = pd.DataFrame(text_features.todense(), columns=encoder.vocabulary_.keys())
```