ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Feature engineering

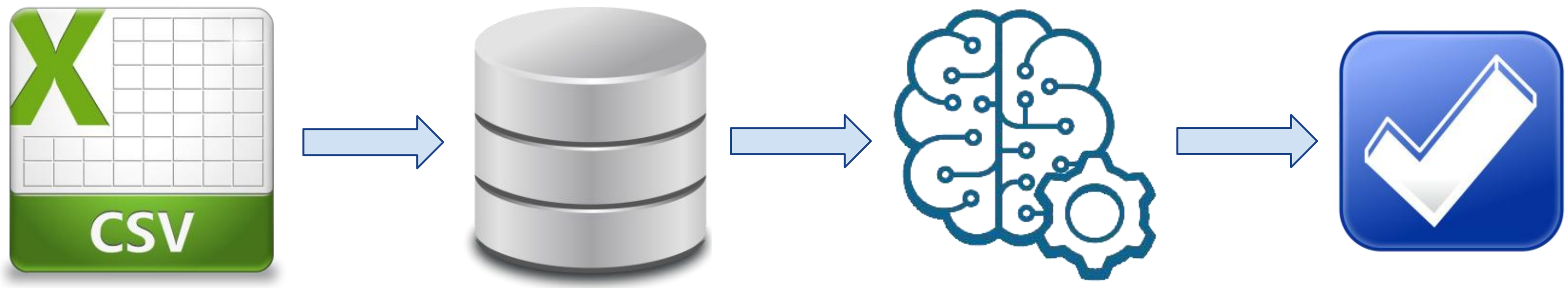
Feature engineering

- "Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data." – Jason Brownlee

Feature engineering

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

The dream ...



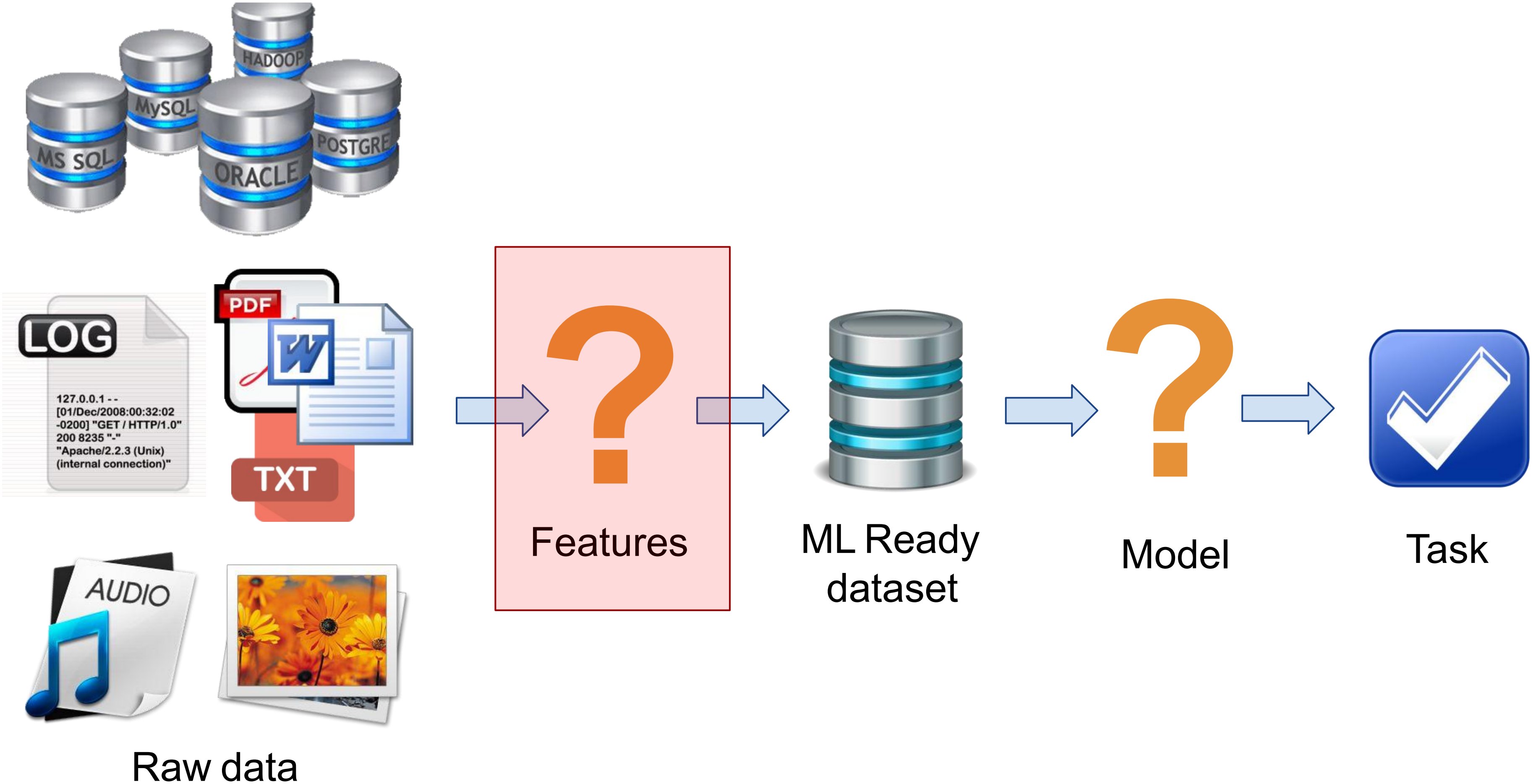
Raw
data

Datas
et

Mod
el

Tas
k

… The Reality

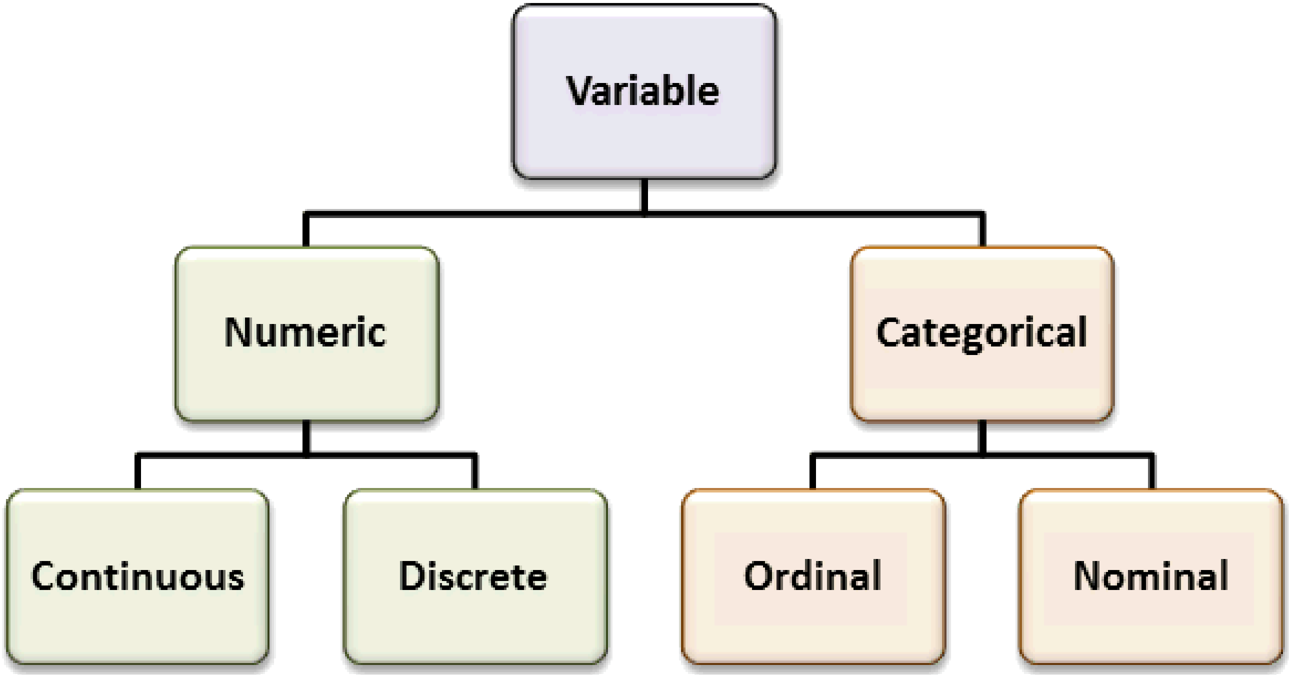


HADOOP
MySQL
MS SQL
ORACLE
POSTGRE
LOG
127.0.0.1 - - [01/Dec/2008:00:32:02 -0200] "GET / HTTP/1.0" 200 8235 "-" "Apache/2.2.3 (Unix) (internal connection)"
PDF
TXT
AUDIO
Raw data
Features
ML Ready dataset
Model
Task

Feature engineering toolbox

- Just kidding :)

Variable data types



Number variables

Binarization

- Counts can quickly accumulate without bound
- convert them into binary values (0, 1) to indicate presence

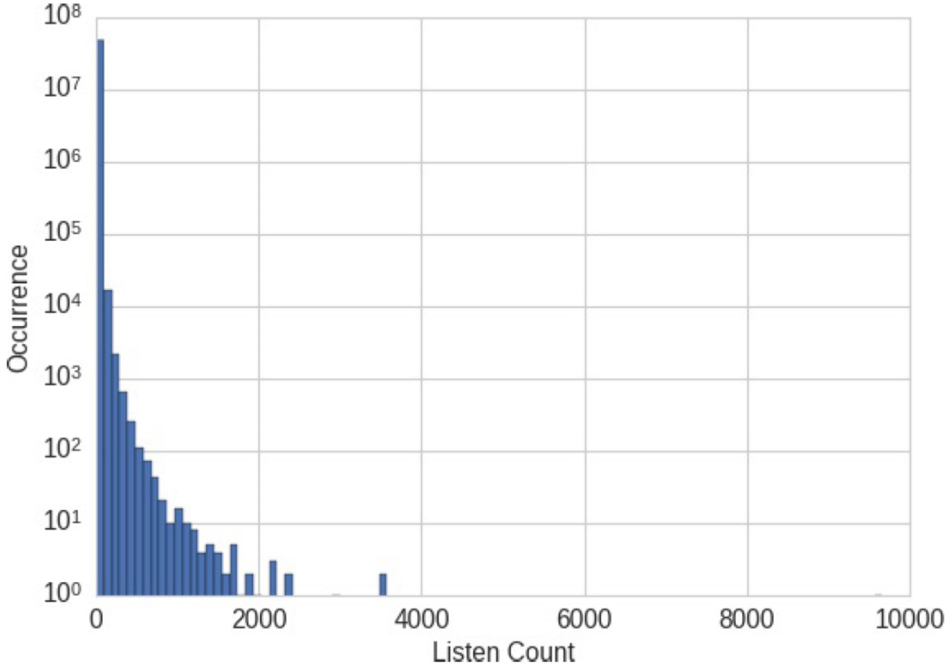
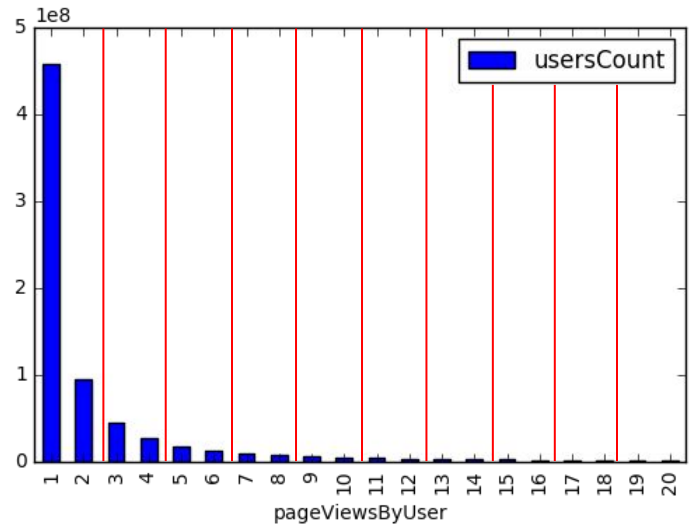


Figure 2-3. Histogram of listen counts in the Taste Profile subset of the Million Song Dataset—note that the y-axis is on a log scale

Quantization or Binning

- Group the counts into bins
- Maps a continuous number to a discrete one
- Bin size
  - Fixed-width binning
    - Eg.
      - 0–12 years old
      - 12–17 years old
      - 18–24 years old
      - 25–34 years old
  - Adaptive-width binning



Equal Width Binning

- divides the continuous variable into several categories having bins or range of the same width

$$\textbf{Categories} : [min, min + w - 1], [min + w, min + 2 * w - 1], [min + 2 * w, min + 3 * w - 1] \cdots [min + (x - 1) * w, max]$$

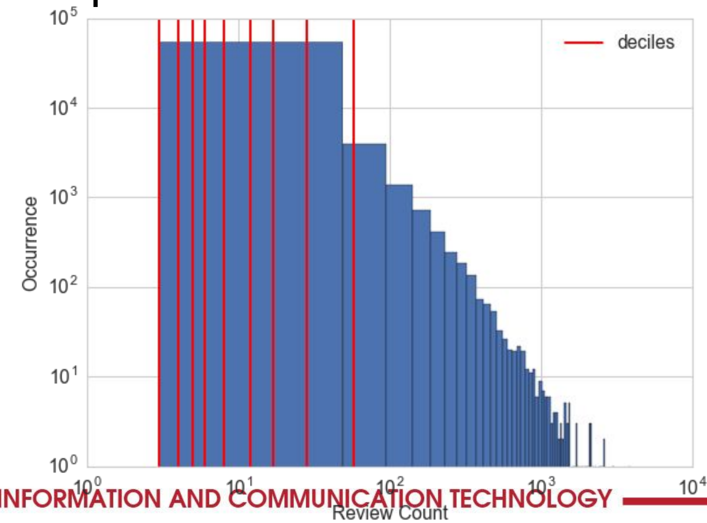
$$w = \left\lfloor \frac{max - min}{x} \right\rfloor$$

- Pros
  - easy to compute
- Cons
  - large gaps in the counts
  - many empty bins with no data

AGE	AGE_bins
10	[10, 21]
15	[10, 21]
16	[10, 21]
18	[10, 21]
20	[10, 21]
30	[22, 33]
35	[34, 45]
42	[34, 45]
48	[46, 55]
50	[46, 55]
52	[46, 55]
55	[46, 55]

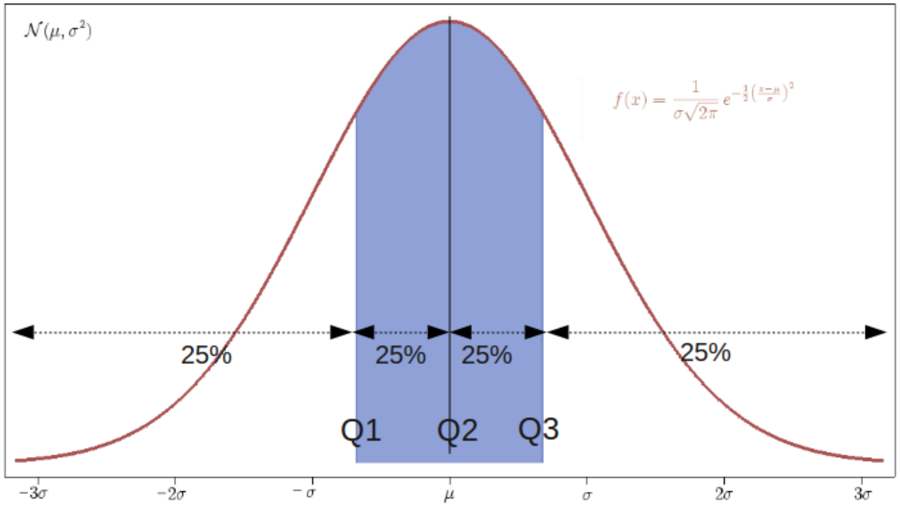
Adaptive-width binning

- Equal frequency binning
  - Quantiles: values that divide the data into equal portions (continuous intervals with equal probabilities)
  - Some q-quantiles have special names
    - The only 2-quantile is called the median
    - The 4-quantiles are called quartiles → Q
    - The 6-quantiles are called sextiles → S
    - The 8-quantiles are called octiles
    - The 10-quantiles are called deciles → D



AGE	AGE_bins
10	[10, 16]
15	[10, 16]
16	[10, 16]
18	[17, 30]
20	[17, 30]
30	[17, 30]
35	[31, 48]
42	[31, 48]
48	[31, 48]
50	[49, 55]
52	[49, 55]
55	[49, 55]

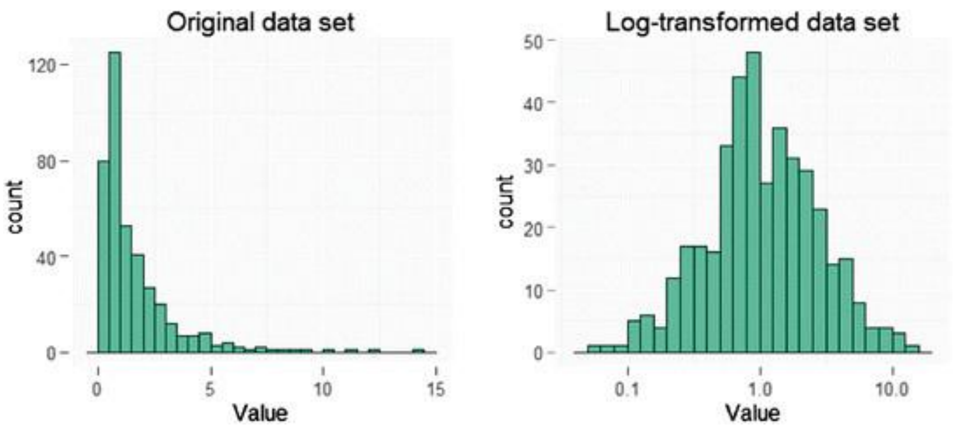
Example: quartiles

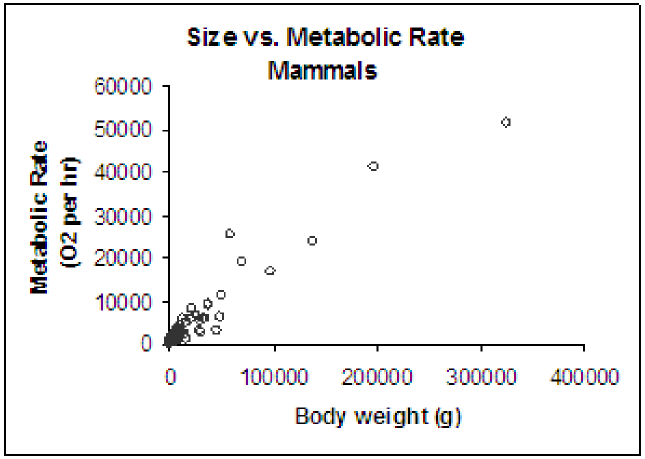



```
>>> import pandas as pd

# Map the counts to quartiles
>>> pd.qcut(large_counts, 4, labels=False)
array([1, 2, 3, 0, 0, 1, 1, 2, 2, 3, 3, 0, 0, 2, 1, 0, 3], dtype=int64)

# Compute the quantiles themselves
>>> large_counts_series = pd.Series(large_counts)
>>> large_counts_series.quantile([0.25, 0.5, 0.75])
0.25     122.0
0.50     926.0
0.75    8286.0
dtype: float64
```

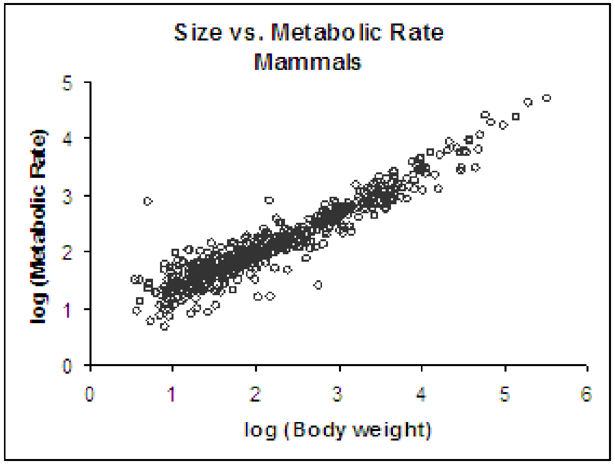
Log Transformation

- Original number = x
- Transformed number $x'=\log_{10}(x)$
- Backtransformed number = $10^{x'}$





Log
Trans-
form



Box-Cox transformation

$$\tilde{x} = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(x) & \text{if } \lambda = 0. \end{cases}$$

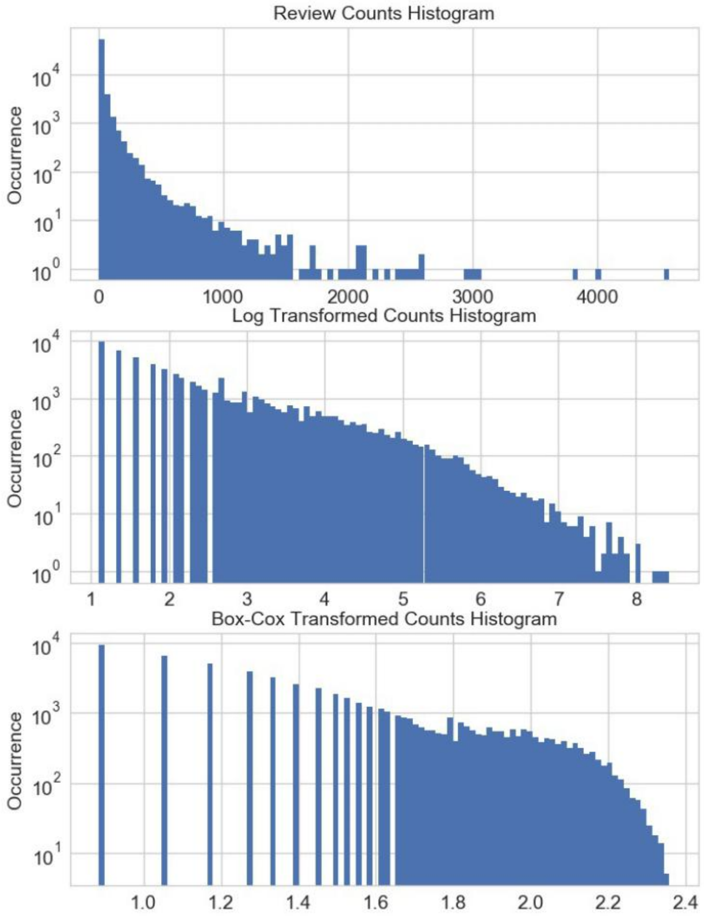


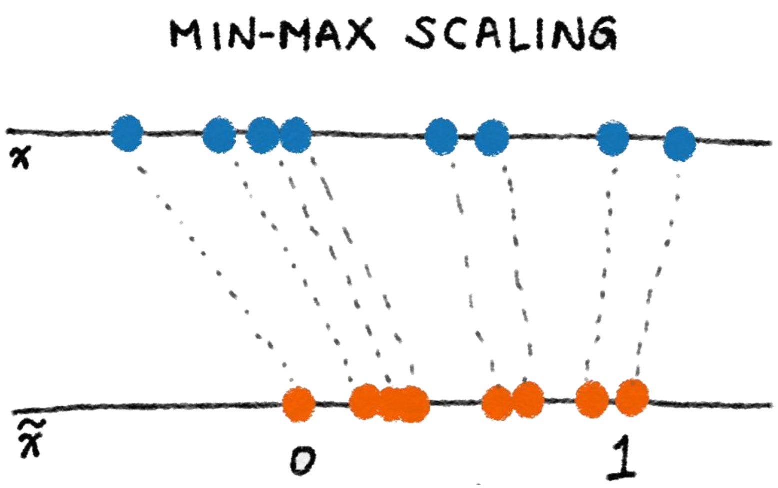
Figure 2-13. Box-Cox transformation of Yelp business review counts (bottom), compared to original (top) and log transformed (middle) histograms

Feature Scaling (Normalization)

- Models that are smooth functions of the input, such as linear regression, logistic regression are affected by the scale of the input
- Feature scaling or normalization changes the scale of the features

Min-max scaling

- Squeezes (or stretches) all values within the range of [0, 1] to add robustness to very small standard deviations and preserving zeros for sparse data.

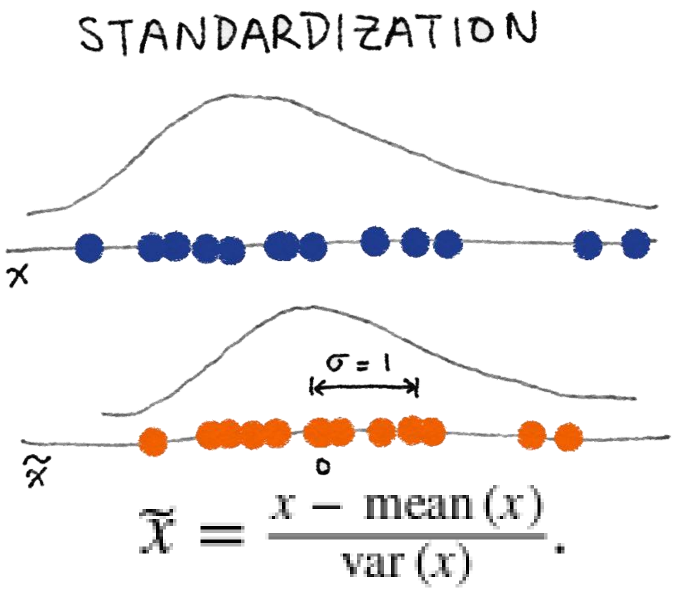


$$\tilde{x} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

```
>>> from sklearn import preprocessing
>>> X_train = np.array([[ 1., -1.,  2.],
...                     [ 2.,  0.,  0.],
...                     [ 0.,  1., -1.]])
...
>>> min_max_scaler = preprocessing.MinMaxScaler()
>>> X_train_minmax = min_max_scaler.fit_transform(X_train)
array([[ 0.5       ,  0.        ,  1.        ],
       [ 1.        ,  0.5       ,  0.33333333],
       [ 0.        ,  1.        ,  0.        ]])
```

Standard (Z) Scaling

After Standardization, a feature has mean of 0 and variance of 1 (assumption of many learning algorithms)




```
>>> from sklearn import preprocessing
>>> import numpy as np
>>> X = np.array([[ 1., -1.,  2.],
...               [ 2.,  0.,  0.],
...               [ 0.,  1., -1.]])
>>> X_scaled = preprocessing.scale(X)
>>> X_scaled
array([[ 0.  ..., -1.22...,  1.33...],
       [ 1.22...,  0.  ..., -0.26...],
       [-1.22...,  1.22..., -1.06...]])
>> X_scaled.mean(axis=0)
array([ 0., 0.,  0.])
>>> X_scaled.std(axis=0)
array([ 1., 1.,  1.])
```

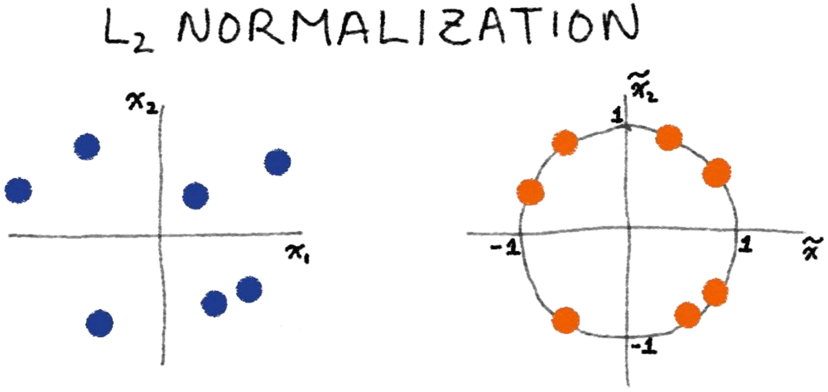
Standardization with scikit-learn

l2 Normalization

- also known as the Euclidean norm
- measures the length of the vector in coordinate space
- scale the values so that if they were all squared and summed, the value would be 1

$$\tilde{x} = \frac{x}{\|x\|_2} \qquad \|x\|_2 = \sqrt{x_1^2 + x_2^2 + \ldots + x_m^2}$$

```
from pandas import read_csv
from numpy import set_printoptions
from sklearn.preprocessing import Normalizer
path = r'./pima-indians-diabetes.csv'
names = ['preg', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv (path, names=names)
array = dataframe.values
Data_normalizer = Normalizer(norm='l2').fit(array)
Data_normalized = Data_normalizer.transform(array)
```



Categorical Variables

Categorical Features

- Nearly always need some treatment to be suitable for models
- High cardinality can create very sparse data
- Difficult to impute missing
- Examples
  - Platform: [“desktop”, “tablet”, “mobile”]
  - Document_ID or User_ID: [121545, 64845, 121545]

Label Encoding

- transform categorical variables into numerical variables by assigning a numerical value to each of the categories

[male, female]	[0, 1]
[blue, green, red, black]	[0, 1, 2, 3]
[10, 21], [22, 33], [34, 45], [46, 55]	[0, 1, 2, 3]

LabelCount encoding

- Rank categorical variables by count in train set
- Useful for both linear and non-linear algorithms (eg: decision trees)
- Not sensitive to outliers
- Won't give same encoding to different variables

ad_id	clicks	ad_rank
54345	35387	1
423654	18339	2
98799	12352	3
68655	9430	4
123646	8232	5

Ordinal encoding

- transform an original categorical variable to a numerical variable by **ensuring the ordinal nature of the variables is sustained**

[male, female]	[0, 1]
[10, 21], [22, 33], [34, 45], [46, 55]	[0, 1, 2, 3]
[cold, warm, hot]	[0, 1, 2]
[poor, fair, good, very good, excellent]	[0, 1, 2, 3, 4]

Frequency encoding

- transform an original categorical variable to a numerical variable by considering the frequency distribution of the data

Column	Freq_Encoding
red	5
green	3
red	5
green	3
blue	4
red	5
red	5
blue	4
red	5
blue	4
blue	4
green	3

One hot encoding

- creates k different columns each for a category and replaces one column with 1 rest of the columns is 0

Column	red	green	blue
red	1	0	0
green	0	1	0
red	1	0	0
green	0	1	0
blue	0	0	1
red	1	0	0
red	1	0	0
blue	0	0	1
red	1	0	0
blue	0	0	1
blue	0	0	1
green	0	1	0

Target Mean encoding

- one of the best techniques
- replace the categorical variable with the mean of its corresponding target variable
- **Steps for mean encoding**
  - For each category
  - Calculate aggregated sum (= a)
  - Calculate aggregated total count (= b)
  - Numerical value for that category = a/b

Column	Target	Target Mean	Target Mean (numerical value)
red	1	3/5	0.6
green	1	2/3	0.67
red	0	3/5	0.6
green	0	2/3	0.67
blue	1	2/4	0.5
red	0	3/5	0.6
red	1	3/5	0.6
blue	0	2/4	0.5
red	1	3/5	0.6
blue	0	2/4	0.5
blue	1	2/4	0.5
green	1	2/3	0.67

Feature Hashing

- Dealing with Large Categorical Variables

categorical_feature	unique_values
landing_page_document_id	636482
ad_id	418295
ad_document_id	143856
content_entities	52439
advertiser	2052
publisher	830
country_state	1892

Some large categorical features from Outbrain Click Prediction competition

Feature hashing [2]

- Hashes categorical values into vectors with fixed-length.
- Lower sparsity and higher compression compared to one hot encoding
- Deals with new and rare categorical values (eg: new user-agents)
- May introduce collisions

100 hashed columns

country
brazil
chile
venezuela
colombia
... 222 countries

→

country_hashed_1	country_hashed_2	country_hashed_3	country_hashed_4	...
1	0	0	0	...
0	0	0	1	...
0	0	1	0	...
0	0	1	0	...
...	...	...	...	...

Bin-counting

- Instead of using the value of the categorical variable as the feature, **we compute the association statistics between that value and the target that we wish to predict**
- Useful for both linear and non-linear algorithms
- May give collisions (same encoding for different categories)
- Be careful about leakage
- Strategies
  - Count
  - Average CTR

ad_id
423654
123646
68655
54345

ad_views
18339
335
1244
35387

or

ad_clicks
1355
12
132
1244

or

ad_CTR
0.074
0.036
0.106
0.035

Counts

Click-Through Rate

P(click | ad) = ad_clicks / ad_views

Text Data

Natural Language Processing

- **Cleaning**
- Lowercasing
- Convert accented characters
- Removing non-alphanumeric
- Repairing
- **Tokenizing**
- Encode punctuation marks
- Tokenize
- N-Grams
- Skip-grams
- Char-grams
- Affixes
- **Removing**
- Stopwords
- Rare words
- Common words
- **Roots**
- Spelling correction
- Chop
- Stem
- Lemmatize
- **Enrich**
- Entity Insertion / Extraction
- Parse Trees
- Reading Level

Text vectorization

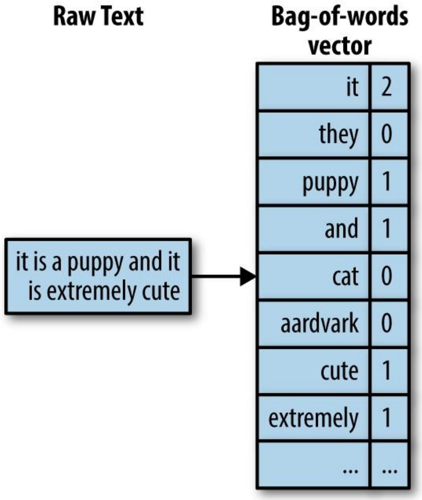
- Represent each document as a feature vector in the vector space, where each position represents a word (token) and the contained value is its relevance in the document.
  - BoW (Bag of words)
  - TF-IDF (Term Frequency - Inverse Document Frequency)
  - Embeddings (eg. Word2Vec, Glove)
  - Topic models (e.g LDA)

	linux	modern	the	system	steering	petrol
D1	3	4	3	0	2	0
D2	4	3	4	1	0	1
D3	1	0	4	1	0	1
D4	0	1	3	3	3	4

Document Term Matrix - Bag of Words

Bag-of-Words

- Input
  - “Customer reviews build something known as social proof, a phenomenon that states people are influenced by those around them. This might include friends and family, industry experts and influencers, or even internet strangers.”
- Output
  - a text document is converted into a “flat” vector of counts
  - doesn’t contain any of the original textual structures
  - *John is quicker than Mary* and *Mary is quicker than John* have the same vectors

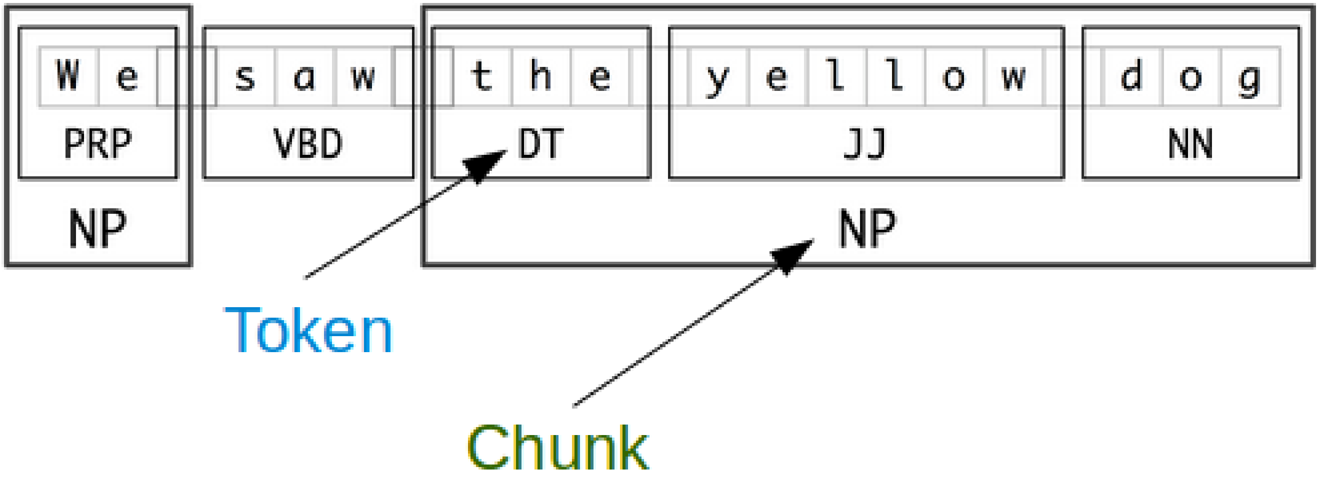


Bag-of-n-Grams

- a natural extension of bag-of-words (a word is essentially an unigram)
- bag-of-n-grams representation can be more informative
  - n-grams retain more of the original sequence structure
- Cons
  - bag-of-n-grams is a much bigger and sparser feature space

From Words to n-Grams to Phrases

- **Tokenization** is the process of **tokenizing** or splitting a string, text into a list of tokens.
- **Chunking** a **sentences** refers to breaking/dividing a **sentence** into parts of words such as word groups and verb groups.



Document frequency

- Rare terms are more informative than frequent terms
  - Recall stop words
- Consider a term in the query that is rare in the collection (e.g., *arachnocentric*)
- A document containing this term is very likely to be relevant to the query *arachnocentric*
  - → We want a high weight for rare terms like *arachnocentric*.

Tf-idf

- The term frequency $\mathrm{tf}_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.
- df_t is the document frequency of *t*: the number of documents that contain *t*
  - df_t is an inverse measure of the informativeness of *t*
  - $\mathrm{df}_t \le N$
- We define the idf (inverse document frequency) of *t* by

$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

  - We use log (N/df_t) instead of N/df_t to "dampen" the effect of idf.

Tf-idf

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = \log(1 + \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

- Best known weighting scheme in information retrieval
  - Note: the “-” in tf-idf is a hyphen, not a minus sign!
  - Alternative names: tf.idf, tf x idf
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

Filtering for Cleaner Features

- **Stopwords**
  - weeding out common words that make for vacuous features
- **Frequency-Based Filtering**
  - filtering out corpus-specific common words as well as general-purpose stopwords
- **Rare words**
  - Depending on the task, one might also need to filter out rare words.
  - These might be truly obscure words, or misspellings of common words.
- **Stemming**
  - An NLP task that tries to chop each word down to its basic linguistic word stem form

Word representation: embedding the context

- Attempt to encode similarity inside the word vectors
- Built ontop of the following great idea
  - "You shall know a word by the company it keeps" (J. R. Firth 1957)

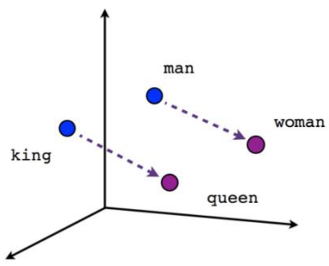
During his presidency, **Trump** ordered a travel ban on citizens
controversial or false. **Trump** was elected president in a surprise victory over
1971, renamed it to The **Trump** Organization, and expanded it into Manhattan.
coordination between the **Trump** campaign and the Russian government in its election interference.

These words describe the meaning of Trump

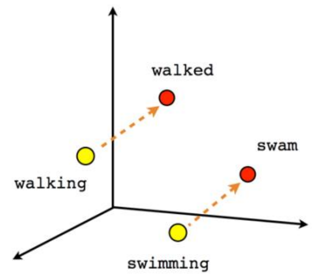
Word embedding

- Each word is encoded in a dense vector (Low dimension)
- Able to capture the semantics
    - Similar words ~ Similar vectors

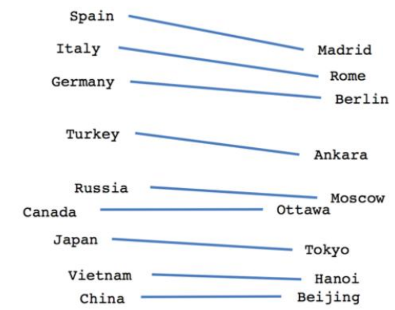
$$\text{University} = \begin{bmatrix} 0.13 \\ 0.67 \\ -0.34 \\ 0.76 \\ -0.21 \\ -0.11 \\ -0.45 \\ 0.87 \\ 0.44 \end{bmatrix}$$



Male-Female



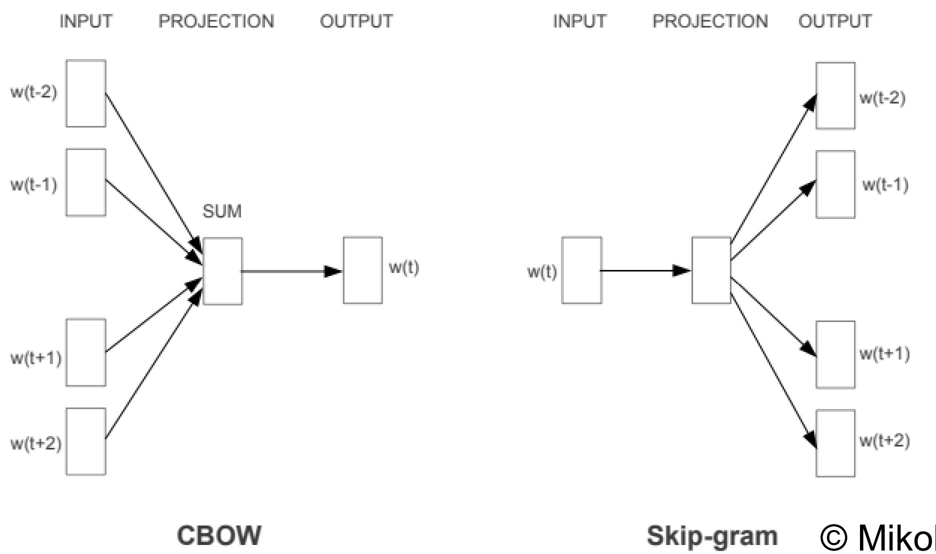
Verb tense



Country-Capital

How to learn word embeddings

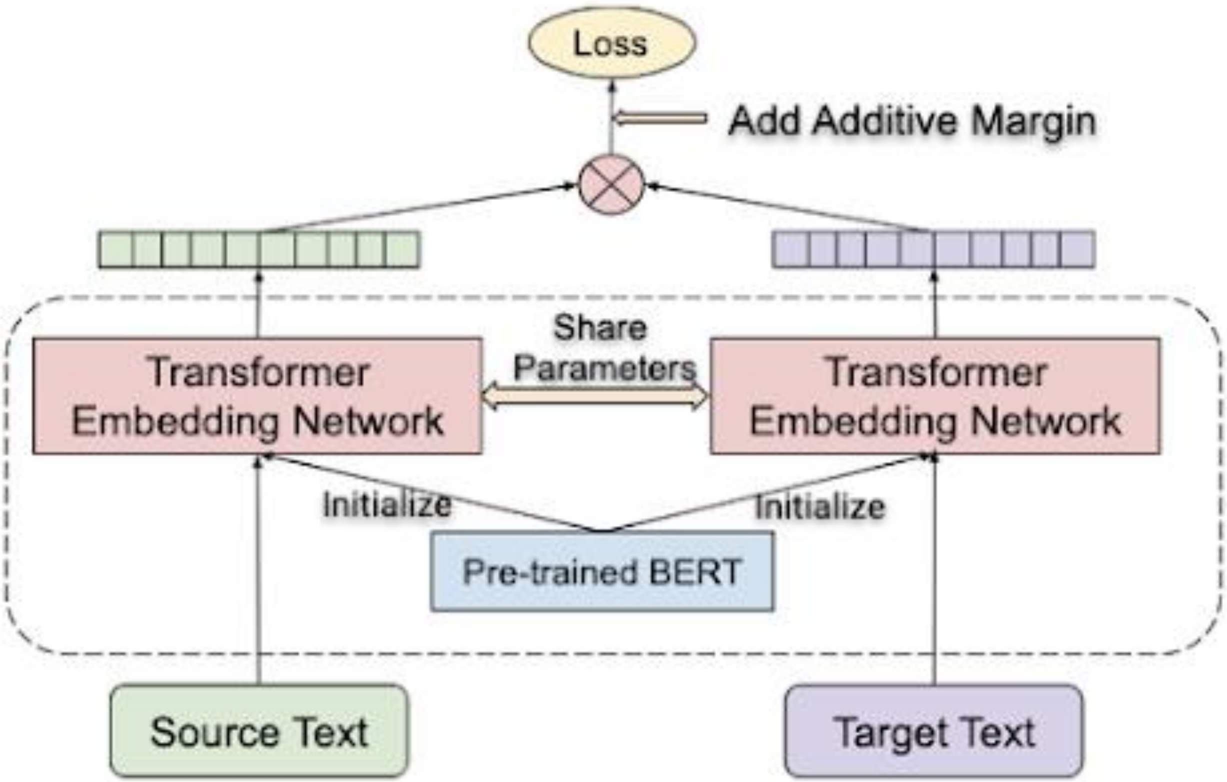
- The famous approach: Word2vec (Mikolov et. al. 2013)
- Unsupervised learning
- Large-scale dataset
- Lower computation cost
- High quality word vectors



© Mikolov et. Al. 2013

BERT sentence embedding

- Feng, Fangxiaoyu, et al. "Language-agnostic BERT Sentence Embedding." *arXiv preprint arXiv:2007.01852* (2020).



Feature selection

Interaction Features

- A simple pairwise *interaction feature* is the product of two features
- A simple linear model
  - $y=w_1x_1 +w_2x_2 +...+w_nx_n$
- An easy way to extend the linear model is to include combinations of pairs of input features
  - $y = w_1x_1 + w_2x_2 + ... + w_nx_n + w_{1,1}x_1x_1 + w_{1,2}x_1x_2 + w_{1,3}x_1x_3 + ...$

Polynomial Features

$$(X_1, X_2) \Longrightarrow (1, X_1, X_2, X_1^2, X_1X_2, X_2^2)$$

```
>>> import numpy as np
>>> from sklearn.preprocessing import PolynomialFeatures
>>> X = np.arange(6).reshape(3, 2)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5]])
>>> poly = poly = PolynomialFeatures(degree=2, interaction_only=False,
include_bias=True)
>>> poly.fit_transform(X)
array([[  1.,   0.,   1.,   0.,   0.,   1.],
       [  1.,   2.,   3.,   4.,   6.,   9.],
       [  1.,   4.,   5.,  16.,  20.,  25.]])
```

Polynomial features with scikit-learn

Feature Selection

- Objective
  - prune away nonuseful features in order to reduce the complexity of the resulting model
- Advantages
  - **Training** a machine learning algorithm **faster**.
  - Reducing the **complexity** of a model and making it easier to **interpret**.
  - Building a **sensible model** with **better prediction power**.
  - **Reducing overfitting** by selecting the right set of features.

Wrapper methods

- The *feature selection* process is based on a specific machine learning algorithm
- Exhaustive search follows a *greedy search approach* by evaluating all the possible combinations of features against the *evaluation criterion*
- Random search methods randomly generate a subset of features
- Computationally intensive since for each subset a new model needs to be trained

Initial set of all features

SC

Embedded Methods

- Perform feature selection during the model training
- Decision tree
  - select a feature in each recursive step of the tree growth process and divide the sample set into smaller subsets
  - The more child nodes in a subset are in the same class, the more informative the features are

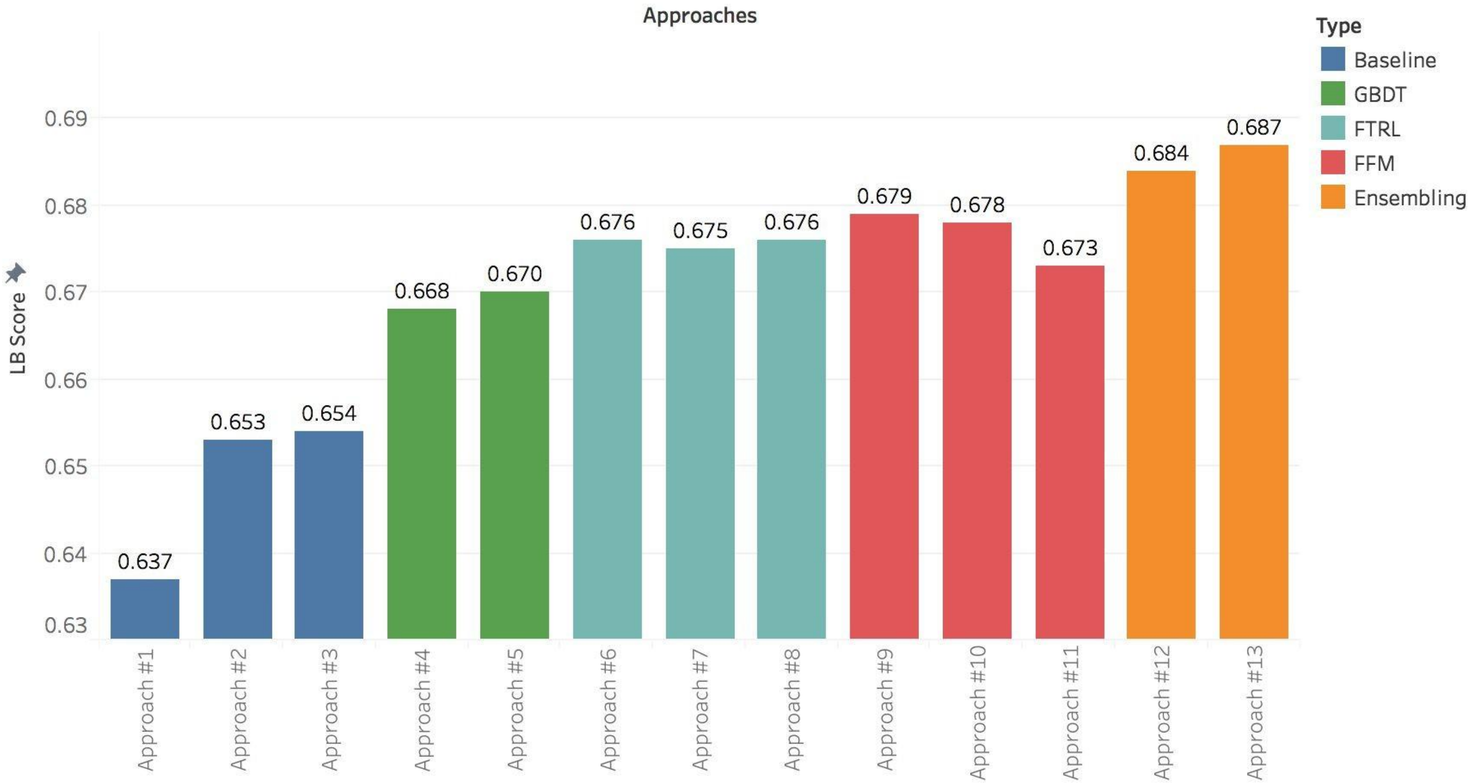
Method comparision

Filter methods	Wrapper methods	Embedded methods
Generic set of methods which do not incorporate a **specific machine learning algorithm.**	Evaluates on a **specific machine learning algorithm** to find optimal features.	Embeds (fix) features during **model building process.** Feature selection is done by observing each iteration of model training phase.
Much **faster** compared to Wrapper methods in terms of time complexity	**High computation time** for a dataset with many features	Sits **between Filter methods and Wrapper methods** in terms of time complexity
Less prone to **over-fitting**	High chances of **over-fitting** because it involves training of machine learning models with different combination of features	Generally used to reduce **over-fitting** by **penalizing** the coefficients of a model being too large.
Examples – **Correlation, Chi-Square test, ANOVA, Information gain** etc.	Examples - **Forward Selection, Backward elimination, Stepwise selection** etc.	Examples - **LASSO, Elastic Net, Ridge Regression** etc.

"More data beats clever algorithms, but better data beats more data."

– Peter Norvig

Diverse set of features and models leads to different results!



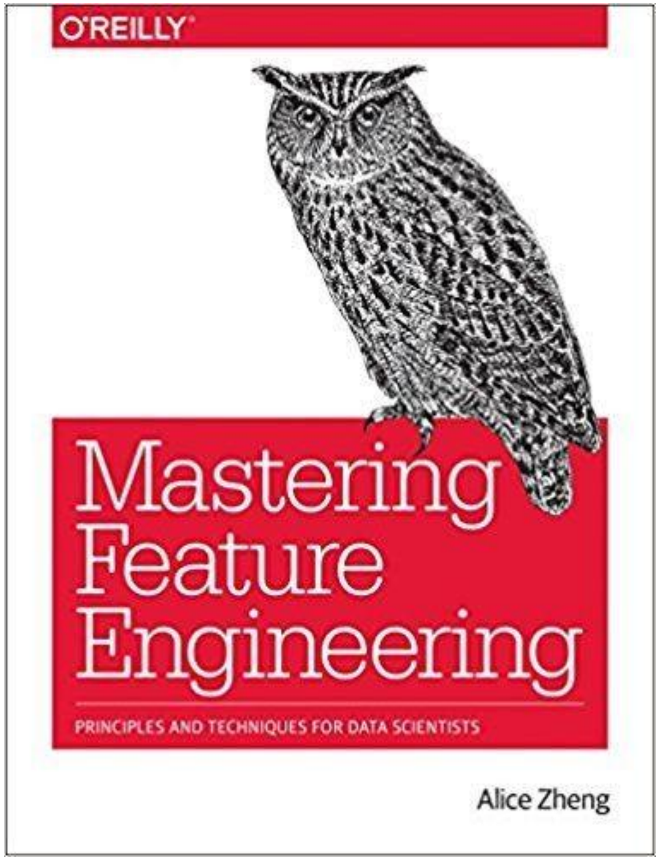
Outbrain Click Prediction

Towards Automated Feature Engineering

Deep Learning....

Thank you
for your
attention!!!

References

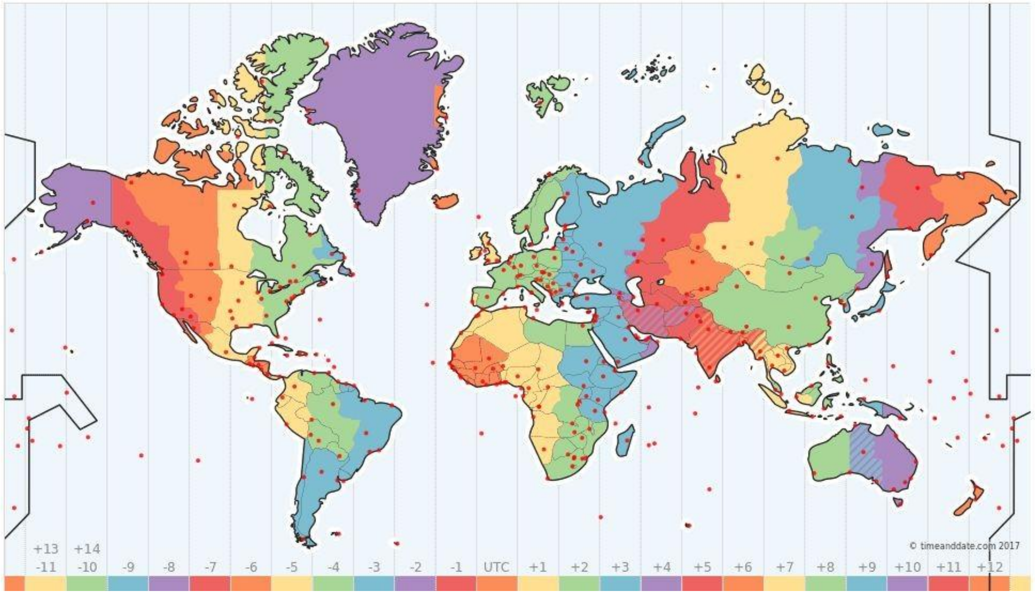


- Scikit-learn - Preprocessing data
- Spark ML - Feature extraction

Temporal Features

Time Zone conversion

- Factors to consider:
  - Multiple time zones in some countries
  - Daylight Saving Time (DST)
    - Start and end DST dates



	country_name	utc_time_offset	dst_time_offset
0	Afghanistan	+04:30	-
1	Aaland Islands	+02:00	+03:00
2	Albania	+01:00	+02:00
3	Algeria	+01:00	-
4	Samoa (American)	−11:00	-
5	Andorra	+01:00	+02:00
6	Angola	+01:00	-
7	Anguilla (UK)	−04:00	-
8	Antigua & Barbuda	−04:00	-
9	Argentina	−03:00	-

Time binning

- Apply binning on time data to make it categorial and more general.
- Binning a time in hours or periods of day, like below.

Hour range	Bin ID	Bin Description
[5, 8)	1	Early Morning
[8, 11)	2	Morning
[11, 14)	3	Midday
[14, 19)	4	Afternoon
[19, 22)	5	Evening
[22-24) and (00-05]	6	Night

- Extraction: weekday/weekend, weeks, months, quarters, years...

Closeness to major events

- Hardcode categorical features from dates
- Example: Factors that might have major influence on spending behavior
- Proximity to major events (holidays, major sports events)
  - Eg. date_X_days_before_holidays
- Proximity to wages payment date (monthly seasonality)
  - Eg. first_saturday_of_the_month

Time differences

- Differences between dates might be relevant
- Examples:
  - user_interaction_date - published_doc_date
- To model how recent was the ad when the user viewed it. Hypothesis: user interests on a topic may decay over time
  - last_user_interaction_date - user_interaction_date
- To model how old was a given user interaction compared to his last interaction

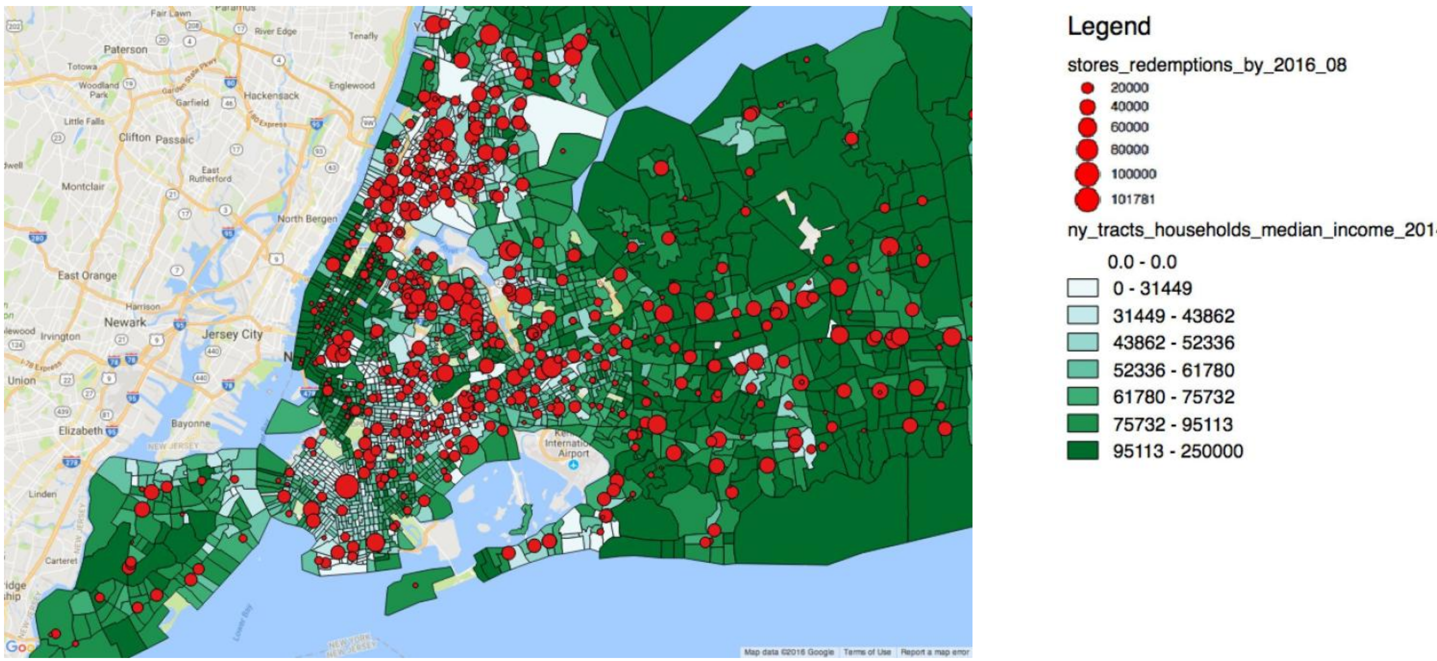
Spatial Features

Spatial Variables

- Spatial variables encode a location in space, like:
  - GPS-coordinates (lat. / long.) - sometimes require projection to a different coordinate system
  - Street Addresses - require geocoding
  - ZipCodes, Cities, States, Countries - usually enriched with the centroid coordinate of the polygon (from external GIS data)
- Derived features
  - Distance between a user location and searched hotels (Expedia competition)
  - Impossible travel speed (fraud detection)

Spatial Enrichment

- Usually useful to enrich with external geographic data (eg. Census demographics)



Beverage Containers Redemption Fraud Detection: Usage of # containers redeemed (red circles) by store and Census households median income by Census Tracts