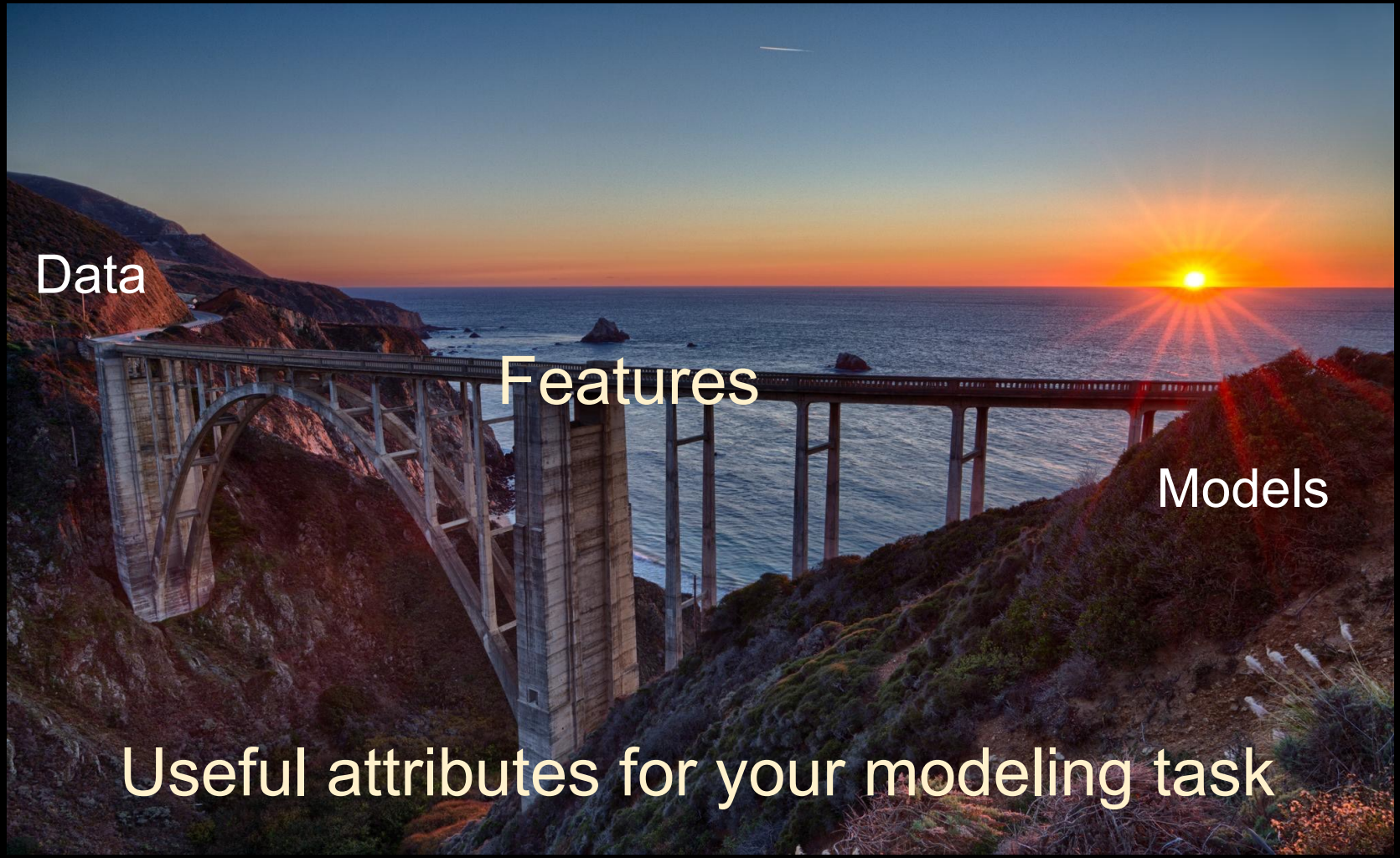
INT 303 BIG DATA ANALYTICS

Lecture7: Feature Engineering

Pengfei FAN
pengfei.fan@xjtlu.edu.cn

Xi'an Jiaotong-Liverpool University
西交利物浦大学


Data
Features
Models
Useful attributes for your modeling task

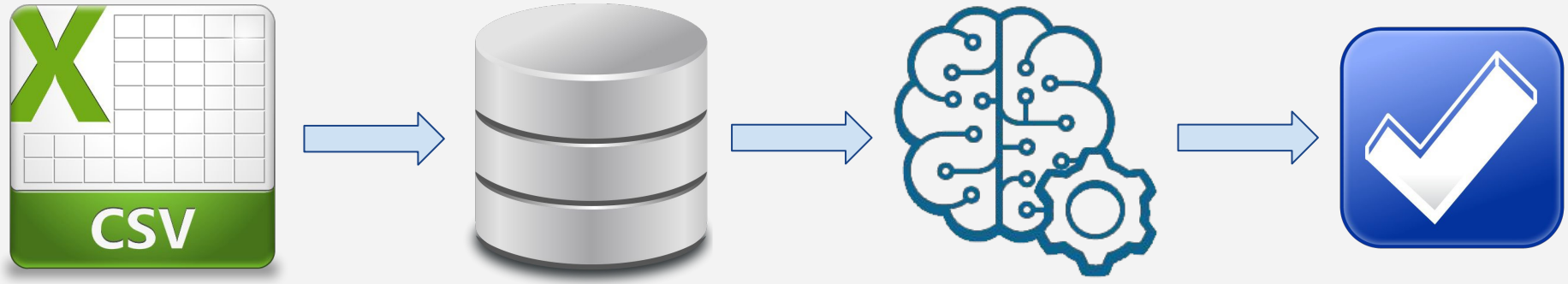
"Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data."

– Jason Brownlee

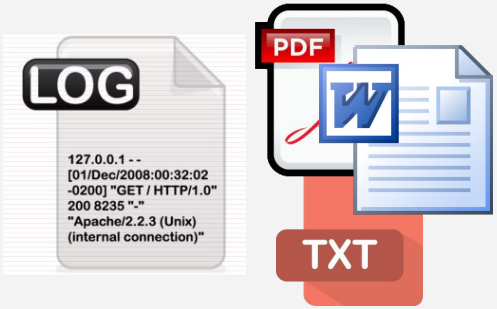
“Coming up with features is difficult,
time-consuming,
requires expert knowledge.
'Applied machine learning' is basically
feature engineering.”

– Andrew Ng

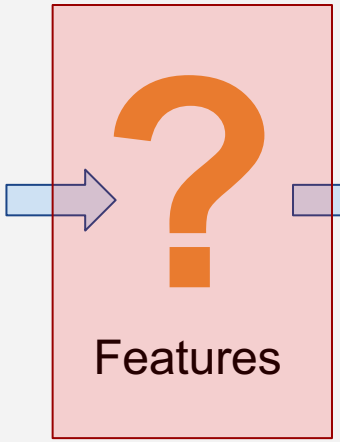
The Dream...



…The Reality



Raw data



Features

ML Ready dataset

Model

Task

Here are some Feature Engineering techniques for your Data Science toolbox...

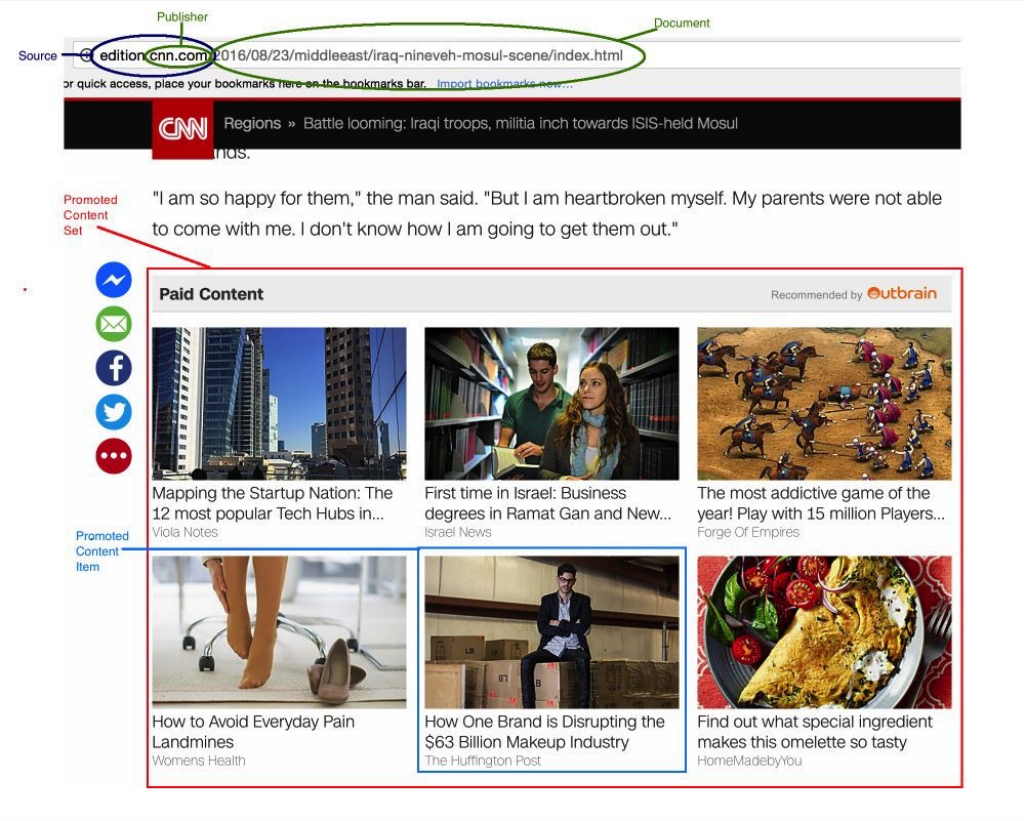
Outline

- Case study
- Numerical Features
- Categorical Features
- Temporal Features
- Spatial Features
- Textual Features
- Feature Selection

Case Study

Outbrain Click Prediction - Kaggle competition

Can you predict which recommended content each user will click?



Dataset

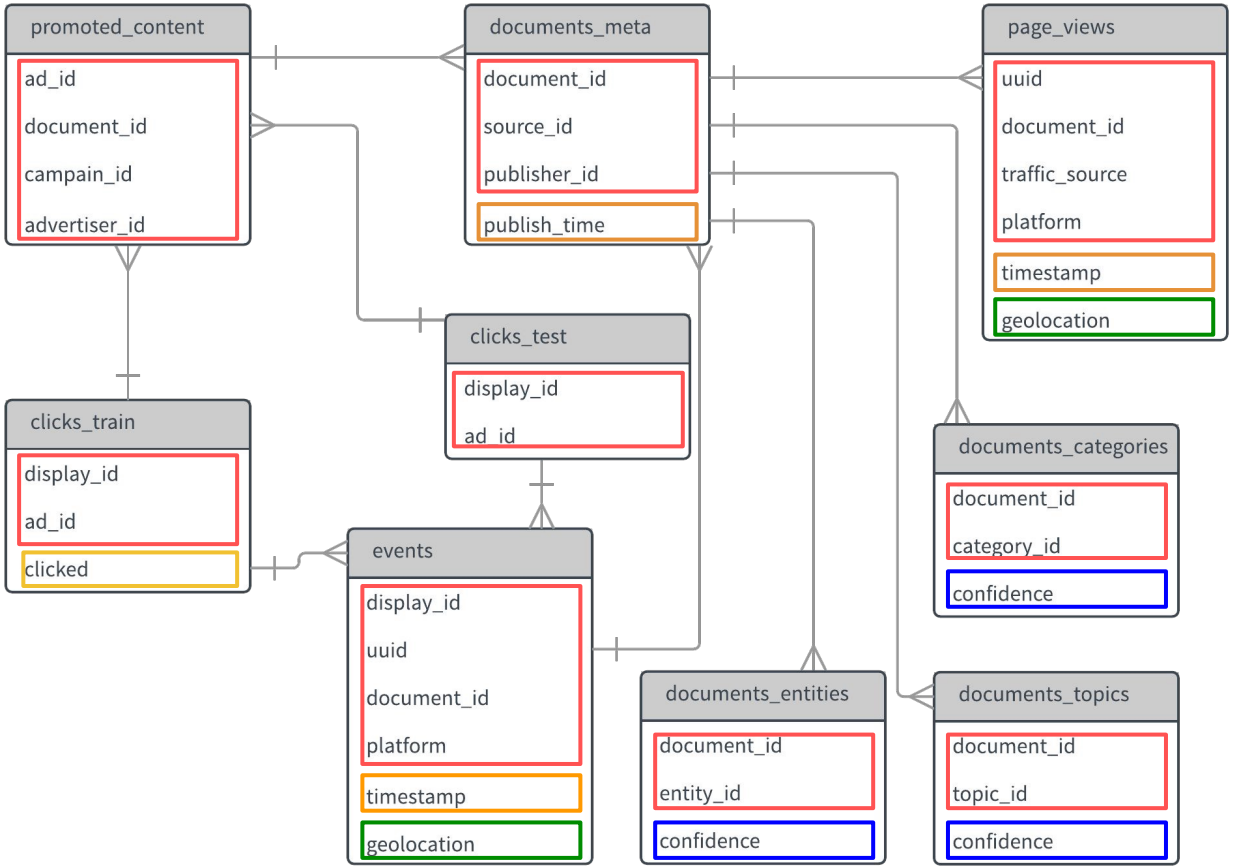
- Sample of users page views and clicks during 14 days on June, 2016
- 2 Billion page views
- 17 million click records
- 700 Million unique users
- 560 sites

First at all …a closer look at your data

- What does the data model look like?
- What is the features distribution?
- What are the features with missing or inconsistent values?
- What are the most predictive features?

- Conduct a Exploratory Data Analysis (EDA)

Outbrain Click Prediction - Data Model



Numerical

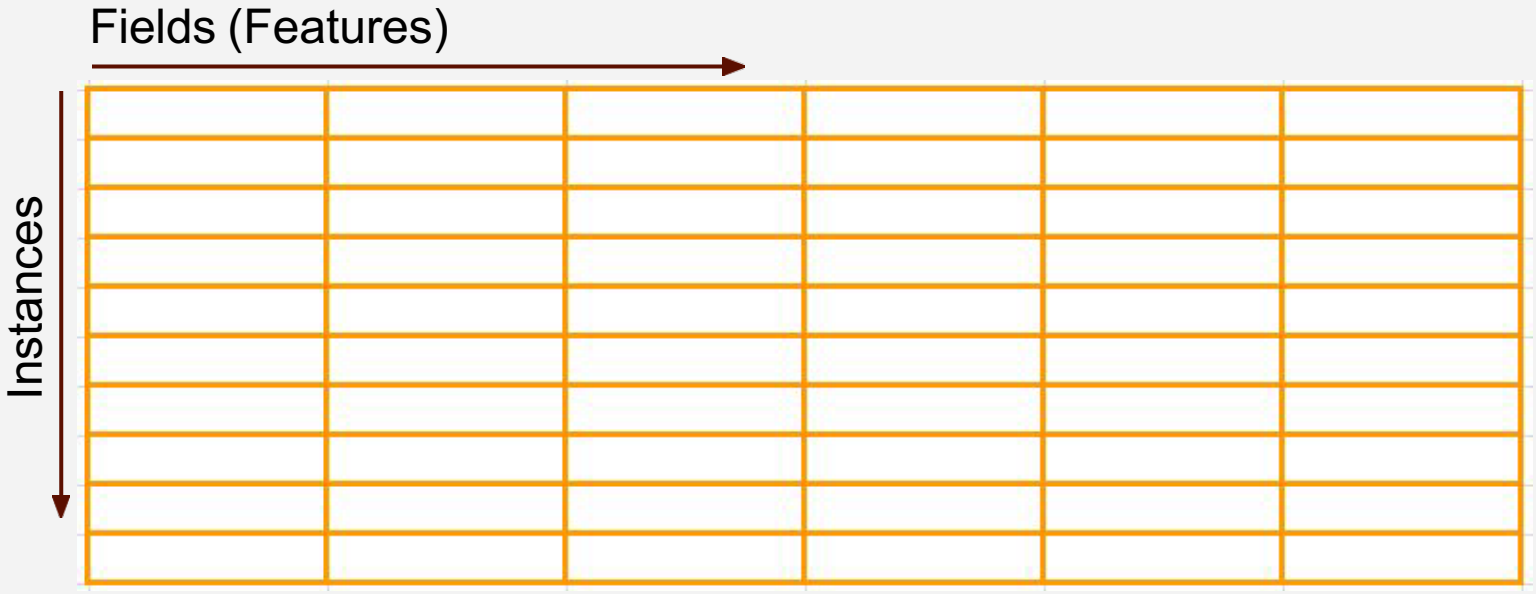
Temporal

Spatial

Categorical

Target

ML-Ready Dataset



Tabular data (rows and columns)

- Usually denormalized in a single file/dataset
- Each row contains information about one instance
- Each column is a feature that describes a property of the instance

Data Cleansing

Homogenize missing values and different types in the same feature, fix input errors, types, etc.

Name	Date	Duration (s)	Genre	Plays
Highway star	1984-05-24	-	Rock	139
Blues alive	1990/03/01	281	Blues	239
Lonely planet	2002-11-19	5:32s	Techno	42
Dance, dance	02/23/1983	312	Disco	N/A
The wall	1943-01-20	218	Reagge	83
Offside down	1965-02-19	4 minutes	Techno	895
The alchemist	2001-11-21	418	Bluesss	178
Bring me down	18-10-98	328	Classic	21
The scarecrow	1994-10-12	269	Rock	734

Original data

Cleaned data

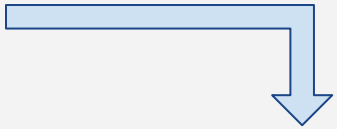
Name	Date	Duration (s)	Genre	Plays
Highway star	1984-05-24		Rock	139
Blues alive	1990-03-01	281	Blues	239
Lonely planet	2002-11-19	332	Techno	42
Dance, dance	1983-02-23	312	Disco	
The wall	1943-01-20	218	Reagge	83
Offside down	1965-02-19	240	Techno	895
The alchemist	2001-11-21	418	Blues	178
Bring me down	1998-10-18	328	Classic	21
The scarecrow	1994-10-12	269	Rock	734

Aggregating

Necessary when the entity to model is an aggregation from the provided data.

Original data (list of playbacks)

Content	Genre	Duration	Play Time	User	Device
Highway star	Rock	190	2015-05-12 16:29:33	User001	TV
Blues alive	Blues	281	2015-05-13 12:31:21	User005	Tablet
Lonely planet	Techno	332	2015-05-13 14:26:04	User003	TV
Dance, dance	Disco	312	2015-05-13 18:12:45	User001	Tablet
The wall	Reagge	218	2015-05-14 09:02:55	User002	Smartphone
Offside down	Techno	240	2015-05-14 11:26:32	User005	Tablet
The alchemist	Blues	418	2015-05-14 21:44:15	User003	TV
Bring me down	Classic	328	2015-05-15 06:59:56	User001	Tablet
The scarecrow	Rock	269	2015-05-15 12:37:05	User003	Smartphone



Aggregated data (list of users)

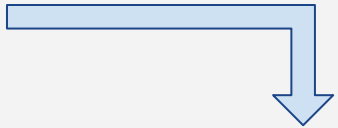
User	Num.Playbacks	Total Time	Pref.Device
User001	3	830	Tablet
User002	1	218	Smartphone
User003	3	1019	TV
User005	2	521	Tablet

Pivoting

Necessary when the entity to model is an aggregation from the provided data.

Original data

Content	Genre	Duration	Play Time	User	Device
Highway star	Rock	190	2015-05-12 16:29:33	User001	TV
Blues alive	Blues	281	2015-05-13 12:31:21	User005	Tablet
Lonely planet	Techno	332	2015-05-13 14:26:04	User003	TV
Dance, dance	Disco	312	2015-05-13 18:12:45	User001	Tablet
The wall	Reagge	218	2015-05-14 09:02:55	User002	Smartphone
Offside down	Techno	240	2015-05-14 11:26:32	User005	Tablet
The alchemist	Blues	418	2015-05-14 21:44:15	User003	TV
Bring me down	Classic	328	2015-05-15 06:59:56	User001	Tablet
The scarecrow	Rock	269	2015-05-15 12:37:05	User003	Smartphone



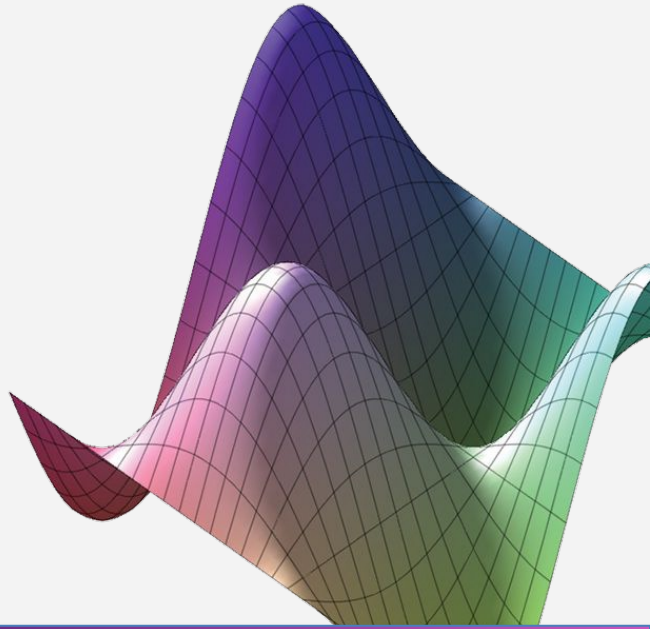
Aggregated data with pivoted columns

playbacks by device

Play duration by device

User	Num.Playback	Total Time	Pref.Device	NP_TV	NP_Tablet	NP_Smartphone	TT_TV	TT_Tablet	TT_Smartphone
User001	3	830	Tablet	1	2	0	190	640	0
User002	1	218	Smartphone	0	0	1	0	0	218
User003	3	1019	TV	2	0	1	750	0	269
User005	2	521	Tablet	0	2	0	0	521	0

Numerical Features



Numerical features

- Usually easy to ingest by mathematical models, but feature engineering is indeed necessary.
- Can be floats, counts, ...
- Easier to impute missing data
- Distribution and scale matters to some models

Imputation for missing values

- Datasets contain missing values, often encoded as blanks, NaNs or other placeholders
- Ignoring rows and/or columns with missing values is possible, but at the price of losing data which might be valuable
- Better strategy is to infer them from the known part of data
- Strategies
  - **Mean**: Basic approach
  - **Median**: More robust to outliers
  - **Mode**: Most frequent value
  - **Using a model**: Can expose algorithmic bias

Binarization

- Transform discrete or continuous numeric features in binary features

  Example: Number of user views of the same document

document_id	uuid	views_count
25792	6d82e412aa0f0d	8
25792	571016386ffee7	6
25792	6a91157d820e37	6
25792	ad45fc764587b0	6
25792	a743b03f2b8ddc	3

document_id	uuid	viewed
25792	6d82e412aa0f0d	1
25792	571016386ffee7	1
25792	6a91157d820e37	1
25792	ad45fc764587b0	1
25792	8d87becfb35857	1
25792	abcdefg1234567	0

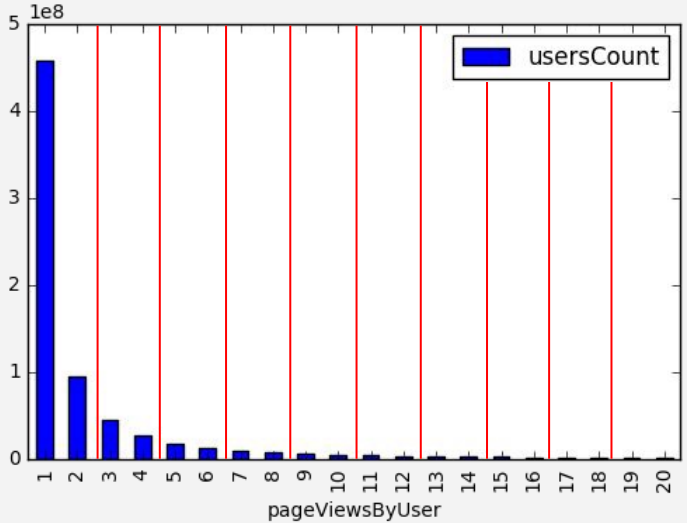
```
>>> from sklearn import preprocessing
>>> X = [[ 1., -1.,  2.],
...      [ 2.,  0.,  0.],
...      [ 0.,  1., -1.]]

>>> binarizer =
preprocessing.Binarizer(threshold=1.0)
>>> binarizer.transform(X)
array([[ 1.,  0.,  1.],
       [ 1.,  0.,  0.],
       [ 0.,  1.,  0.]])
```

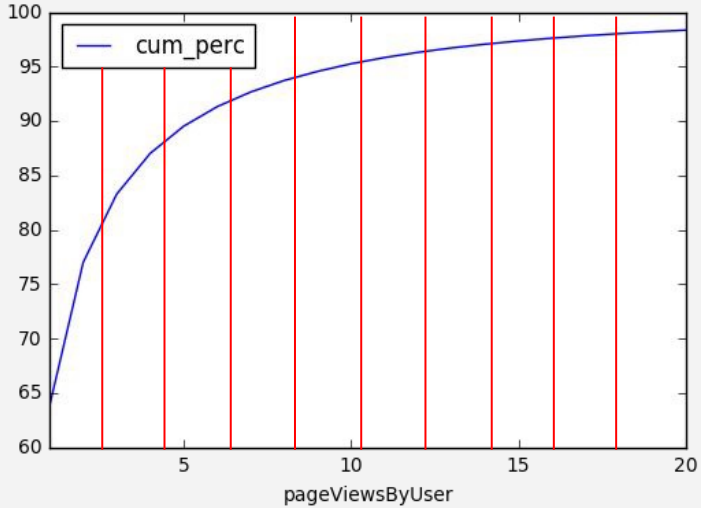
Binarization with scikit-learn

Binning

- Split numerical values into bins and encode with a bin ID
- Can be set arbitrarily or based on distribution
- **Fixed-width binning**

Does fixed-width binning make sense for this long-tailed distribution?



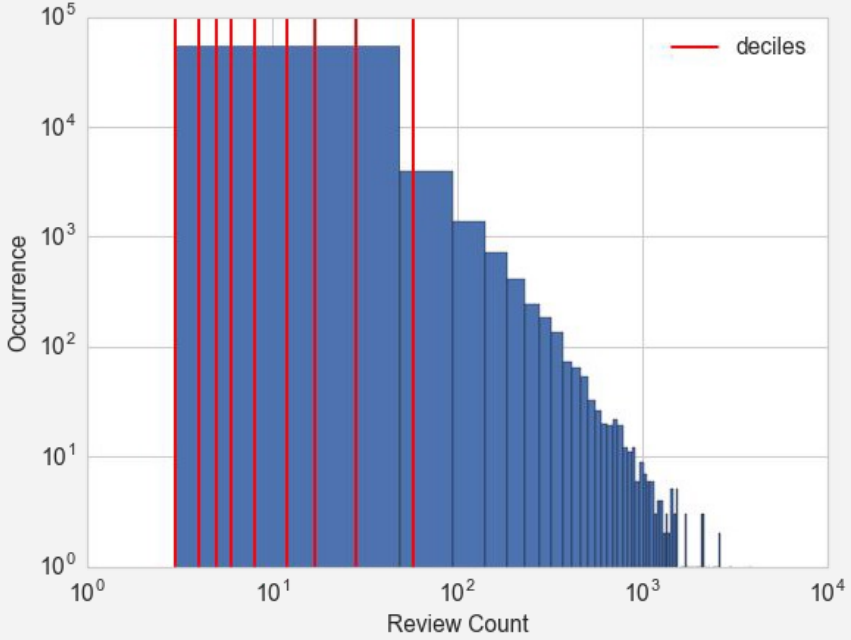


Most users (458,234,809 ~ 5*10^8) had only 1 pageview during the period.

Binning

- **Adaptative or Quantile binning**

  Divides data into equal portions (eg. by median, quartiles, deciles)




```
>>> deciles = dataframe['review_count'].quantile([.1, .2, .3, .4, .5, .6, .7, .8, .9])
>>> deciles
0.1     3.0
0.2     4.0
0.3     5.0
0.4     6.0
0.5     8.0
0.6    12.0
0.7    17.0
0.8    28.0
0.9    58.0
```

Quantile binning with Pandas

Log transformation

Compresses the range of large numbers and expand the range of small numbers.

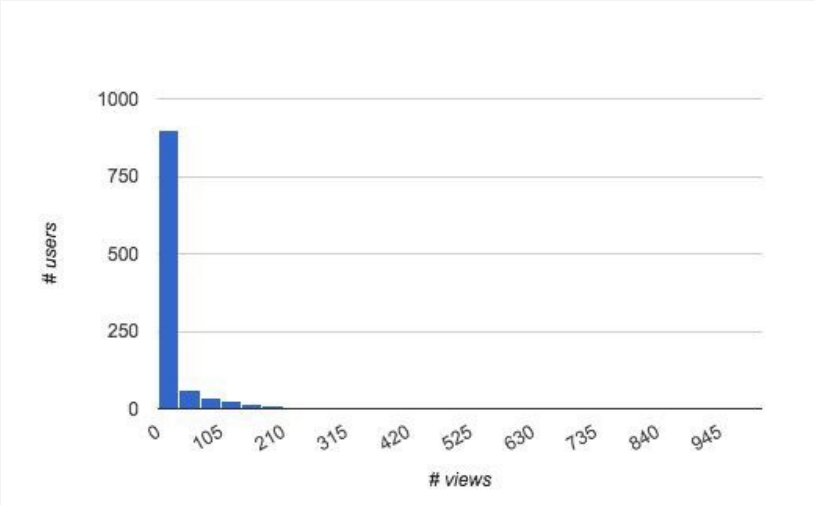
Eg. The larger x is, the slower log(x) increments.

user_id	views_count
a	1000
b	500
c	300
d	200
e	150
f	100
g	70
h	50
i	30
j	20
k	10
l	5
m	1

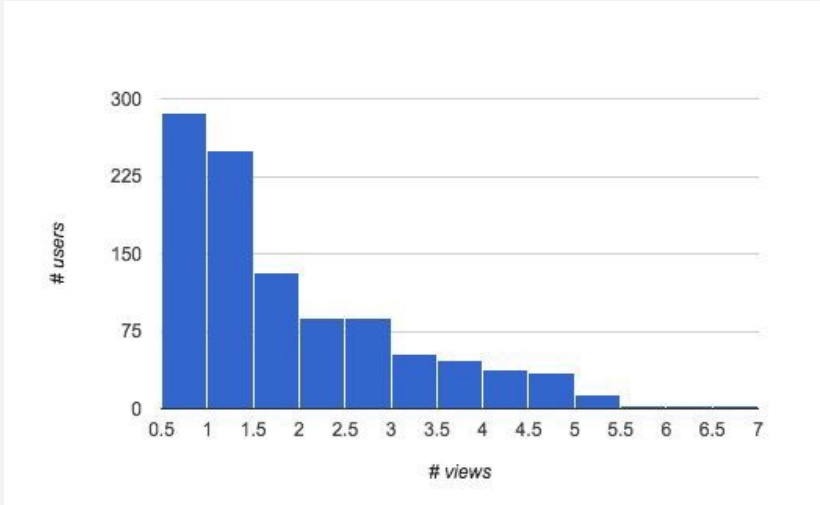
log(1+views_count)
6.91
6.22
5.71
5.30
5.02
4.62
4.26
3.93
3.43
3.04
2.40
1.79
0.69

Log transformation

Smoothing long-tailed data with log



Histogram of # views by user



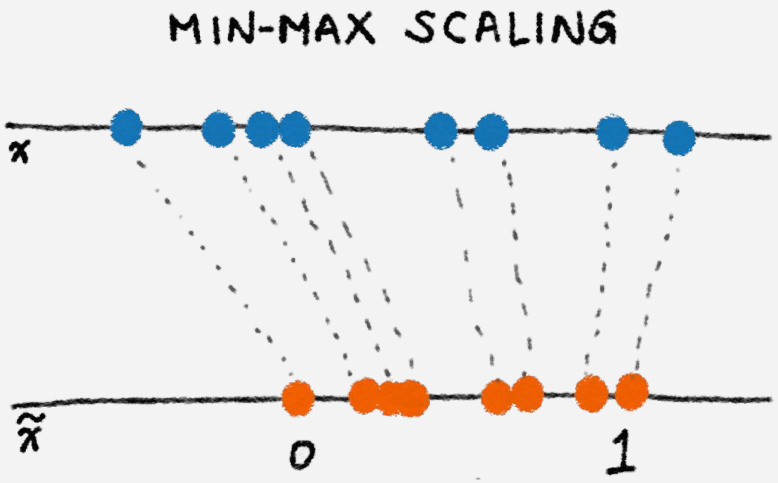
Histogram of # views by user
smoothed by log(1+x)

Scaling

- Models that are smooth functions of input features are sensitive to the scale of the input (eg. Linear Regression)
- Scale numerical variables into a certain range, dividing values by a normalization constant (no changes in single-feature distribution)
- Popular techniques
  - MinMax Scaling
  - Standard (Z) Scaling

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.



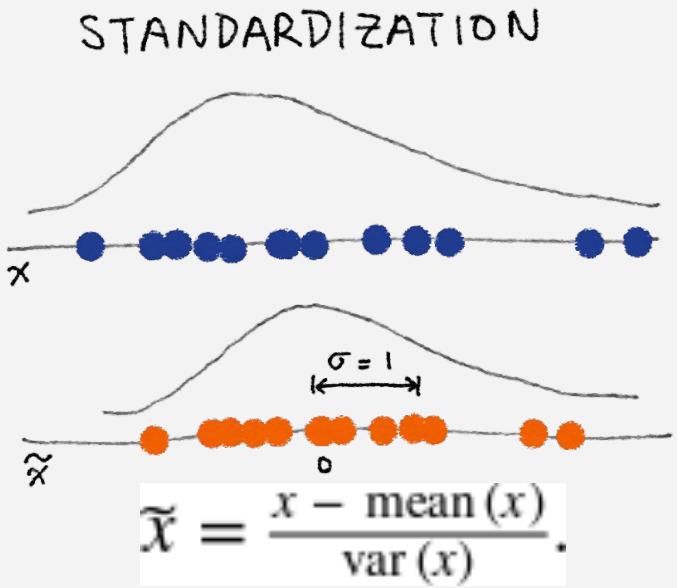
$$\tilde{x} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

```
>>> 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)
>>> X_train_minmax
 array([[ 0.5       ,  0.        ,  1.        ],
        [ 1.        ,  0.5       ,  0.33333333],
        [ 0.        ,  1.        ,  0.        ]])
```

Min-max scaling with scikit-learn

Standard (Z) Scaling

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



$$\tilde{x} = \frac{x - \text{mean}(x)}{\text{var}(x)}.$$

```
>>> 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

Interaction Features

- Simple linear models use a linear combination of the individual input features, x_1, x_2, ... x_n to predict the outcome y.

$$y = w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

- An easy way to increase the complexity of the linear model is to create feature combinations (nonlinear features).

- Example:

Degree 2 interaction features for vector $x = (x_1, x_2)$

$$y = w_1x_1 + w_2x_2 + w_3x_1x_2 + w_4x_1^2 + w_5x_2^2$$

Interaction 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

Categorical Features

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]

One-Hot Encoding (OHE)

- Transform a categorical feature with *m* possible values into *m* binary features.
- If the variable cannot be multiple categories at once, then only one bit in the group can be on.

platform
desktop
mobile
tablet

platform=desktop	platform=mobile	platform=tablet
1	0	0
0	1	0
0	0	1

- Sparse format is memory-friendly
- Example: "platform=tablet" can be sparsely encoded as "2:1"

Large Categorical Variables

- Common in applications like targeted advertising and fraud detection
- Example:

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

- Hashes categorical values into vectors with fixed-length.
- Lower sparsity and higher compression compared to OHE
- 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 actual categorical value, use a global statistic of this category on historical data.
- Useful for both linear and non-linear algorithms
- May give collisions (same encoding for different categories)
- Be careful about leakage
- Strategies
  - Count
  - Average CTR

Bin-counting

ad_id
423654
123646
68655
54345

ad_views	ad_clicks	ad_CTR
18339	1355	0.074
335	12	0.036
1244	132	0.106
35387	1244	0.035

or or

Counts

Click-Through Rate

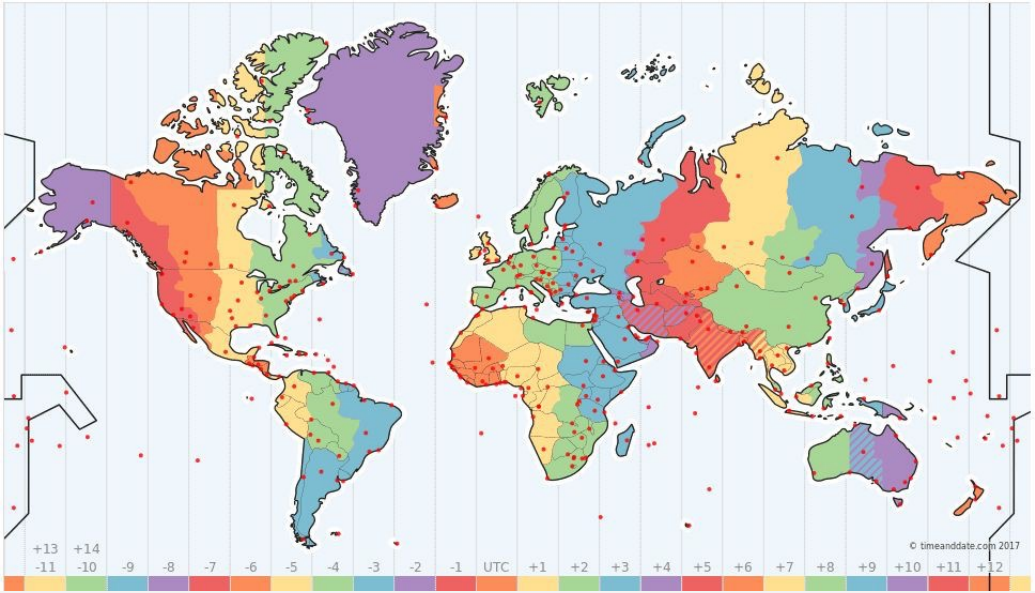
P(click | ad) = ad_clicks / ad_views

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...

Trendlines

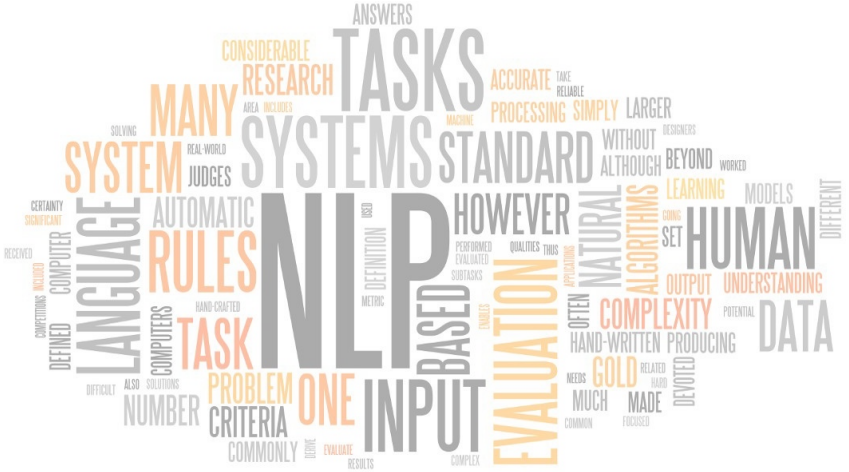
- Instead of encoding: total spend, encode things like: Spend in last week, spend in last month, spend in last year.
- Gives a trend to the algorithm: two customers with equal spend, can have wildly different behavior — one customer may be starting to spend more, while the other is starting to decline spending.

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)

Textual 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

Text vectorization - TF-IDF

Xi'an Jiaotong-Liverpool University 西交利物浦大学

tokens

documents	face	person	guide	lock	cat	dog	sleep	micro	pool	gym	...
	0	1	2	3	4	5	6	7	8	9	
D1		0.05					0.25				
D2	0.02			0.32					0.45		
...											

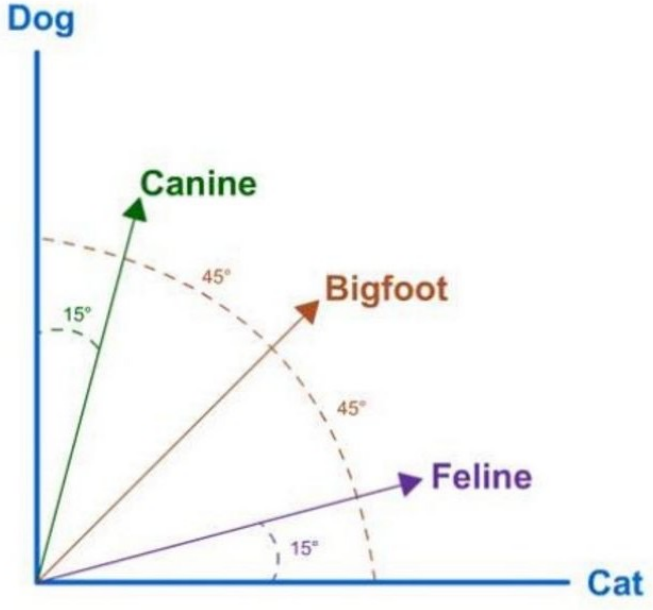
TF-IDF sparse matrix example

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(max_df=0.5, max_features=1000,
                             min_df=2, stop_words='english')
tfidf_corpus = vectorizer.fit_transform(text_corpus)
```

TF-IDF with scikit-learn

Cosine Similarity

Similarity metric between two vectors is cosine among the angle between them



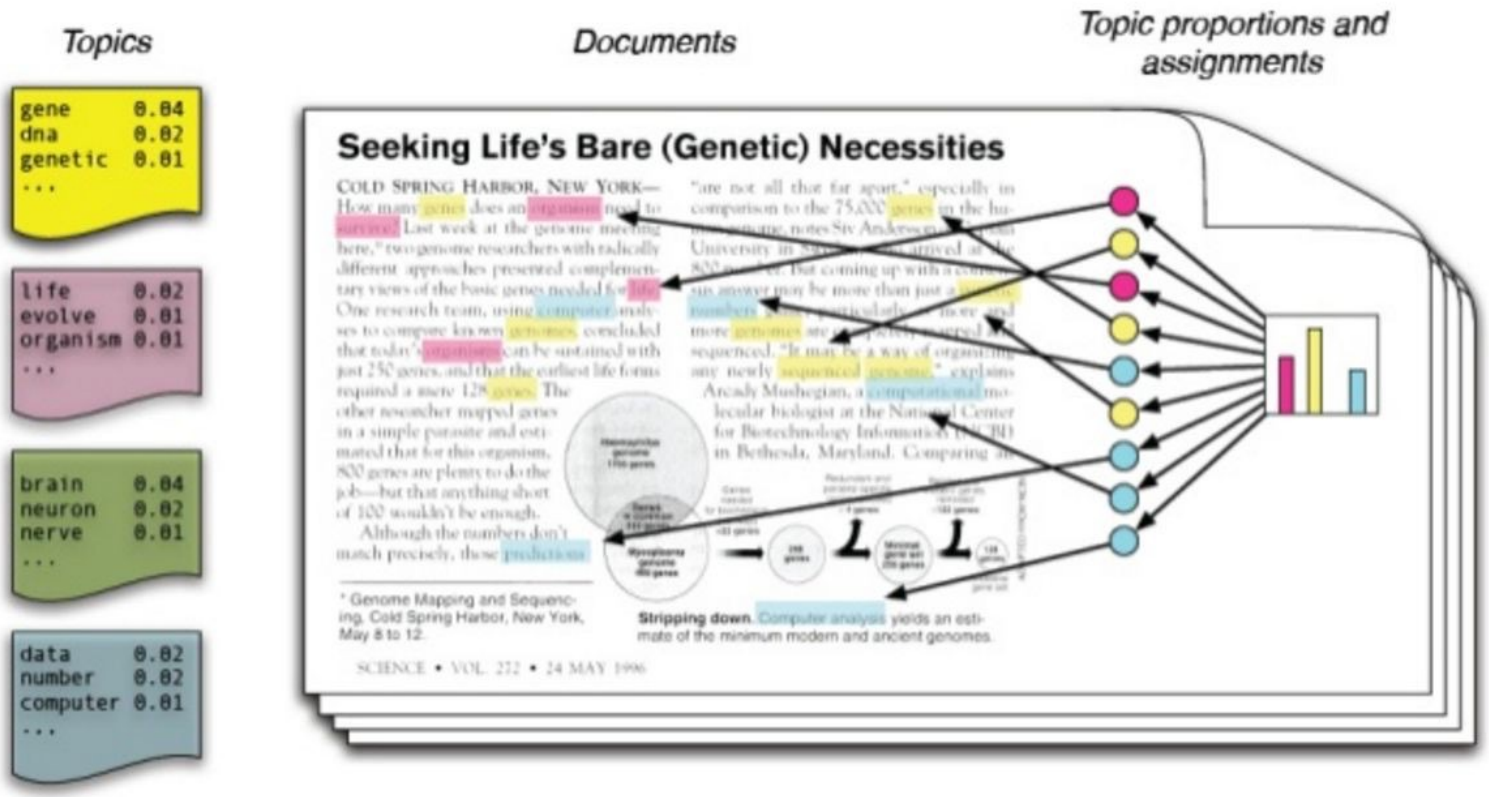
$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

```
from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)
```

Cosine Similarity with scikit-learn

Topic Modeling



Feature Selection

Feature Selection

Reduces model complexity and training time

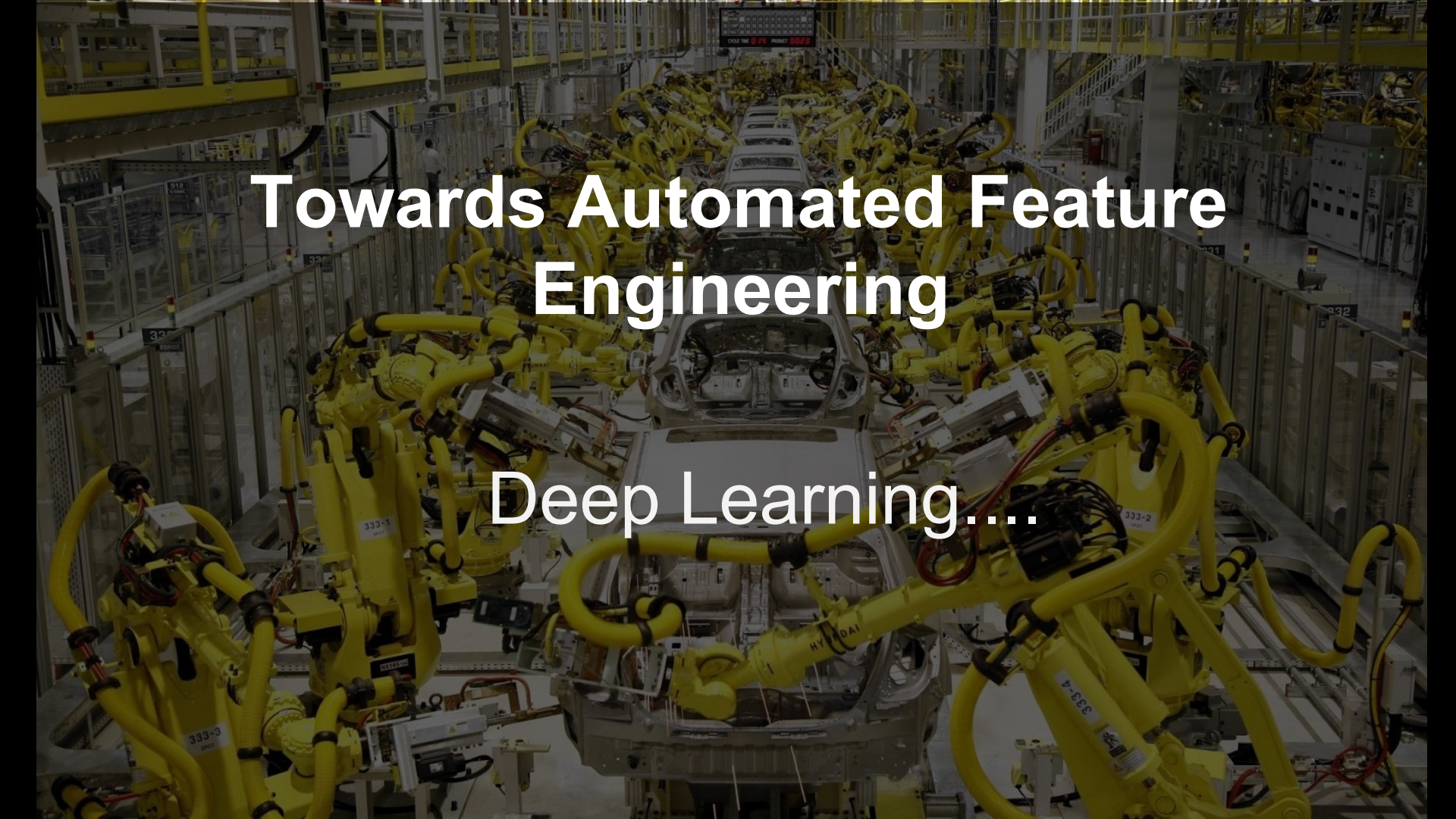
- **Filter methods** - Eg. Correlation our Mutual Information between each feature and the response variable
- **Wrapper methods** - Expensive, trying to optimize the best subset of features (eg. Stepwise Regression)
- **Embedded methods** - Feature selection as part of model training process (eg. Feature Importances of Decision Trees or Trees Ensembles)

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

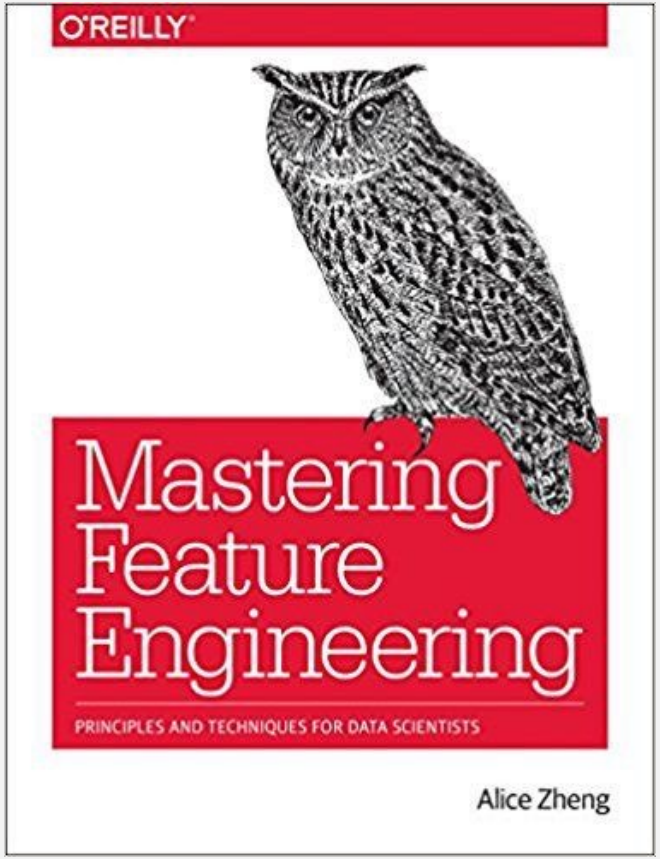
– Peter Norvig

"...some machine learning projects succeed and some fail.
Where is the difference?
Easily the most important factor is the features used."

– *Pedro Domingos*


Towards Automated Feature Engineering
Deep Learning....

References



Scikit-learn - Preprocessing data

Spark ML - Feature extraction

Discover Feature Engineering...

THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

@XJTLU

Xi'an Jiaotong-Liverpool University
西交利物浦大學