INT 303 BIG DATA ANALYTICS

Lecture8: Representing Data and Engineering Features

Jia WANG

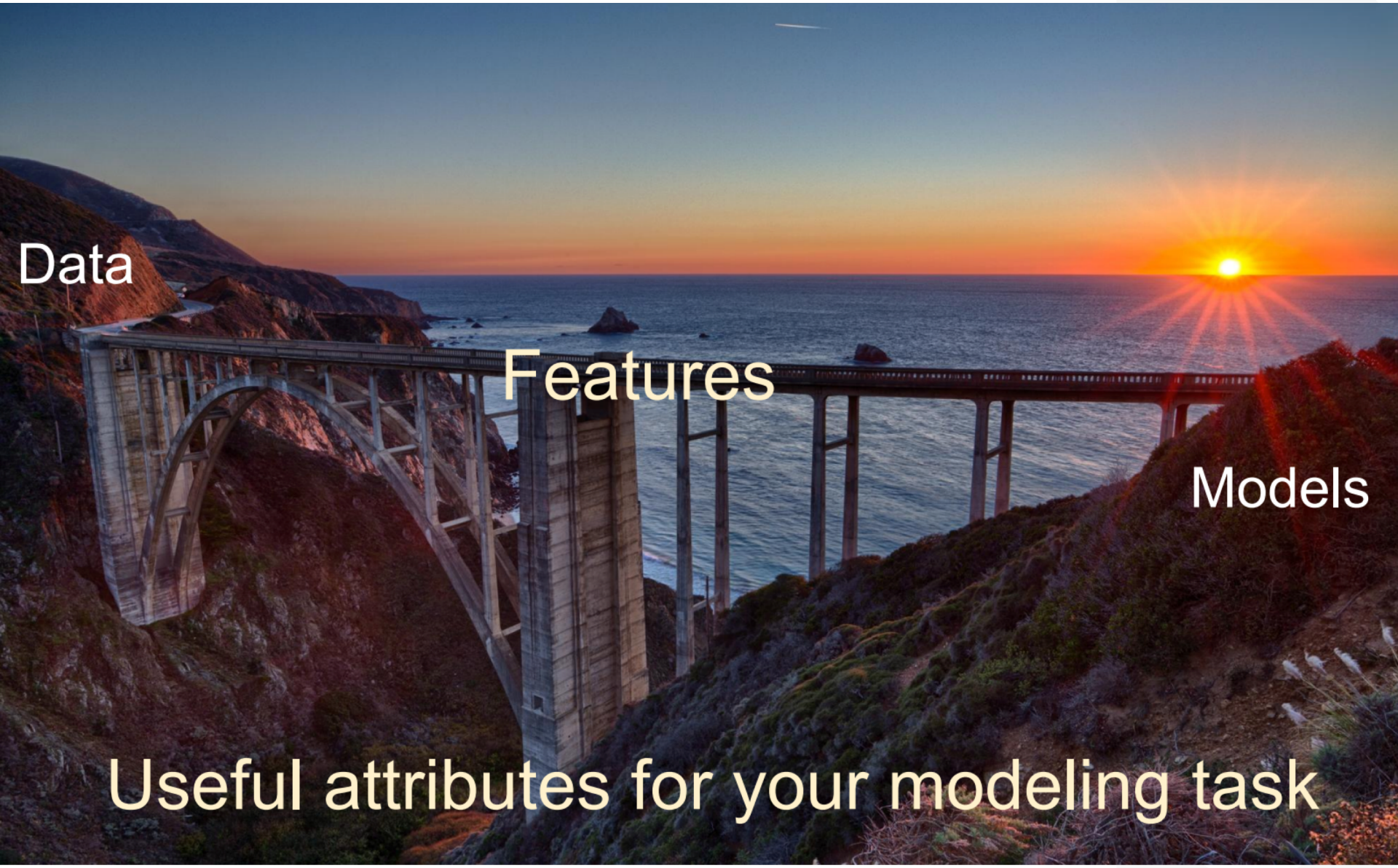
Jia.wang02@xjtlu.edu.cn

Xi'an Jiaotong-Liverpool University

西交利物浦大學

OUTLINE

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


Data
Features
Models
Useful attributes for your modeling task

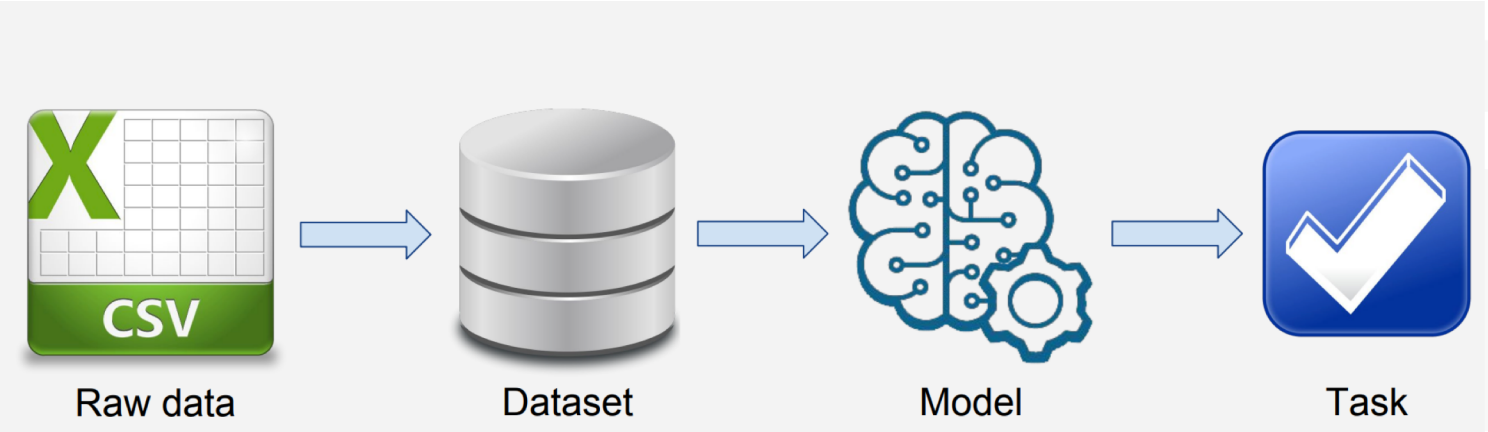
- "Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data."

- – Jason Brownlee

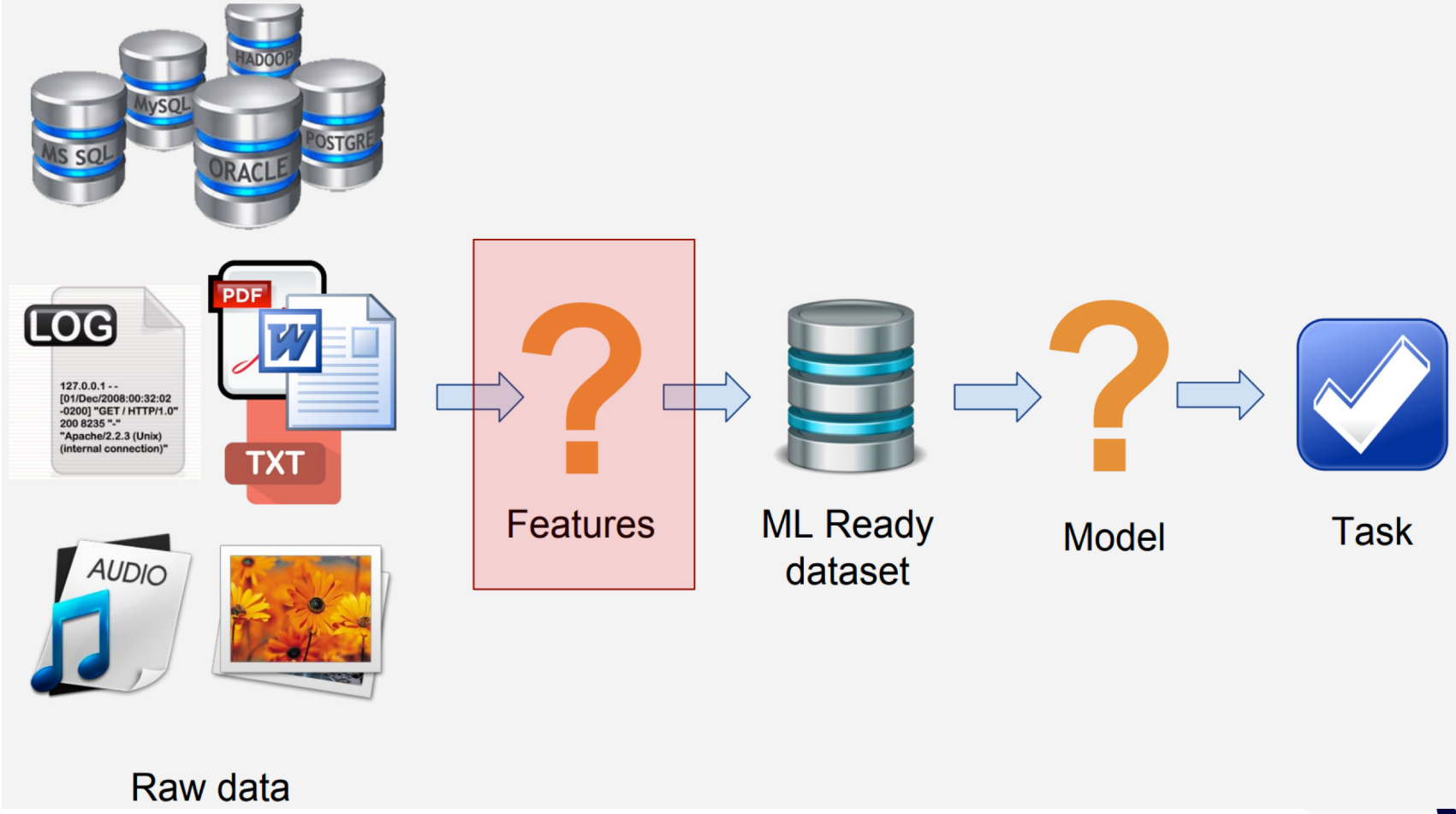
- “Coming up with features is difficult, time-consuming, requires expert knowledge. 'Applied machine learning' is basically feature engineering.”

- – Andrew Ng

THE DREAM...



… THE REALITY

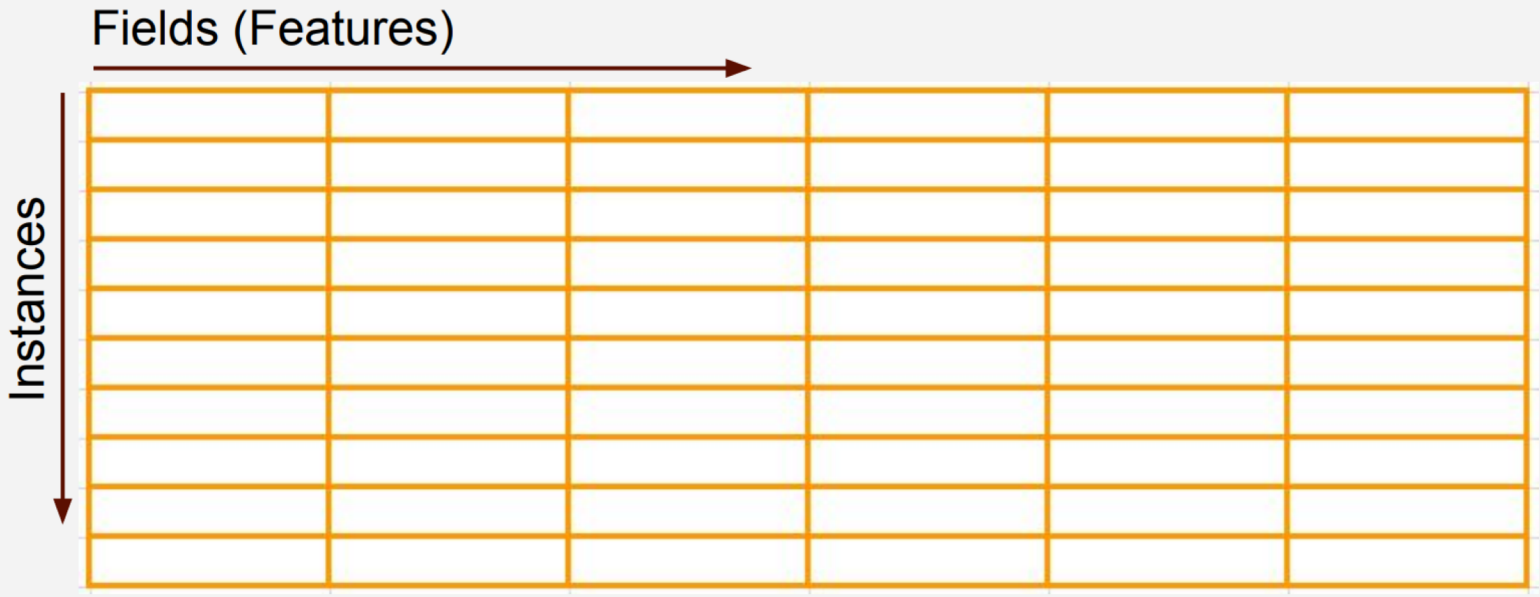

HADOOP
MySQL
MS SQL
ORACLE
POSTGRE
LOG
127.0.0.1 - -
[01/Dec/2008:00:32:02
-0200] "GET / HTTP/1.0"
200 8235 "-"
"Apache/2.2.3 (Unix)
(internal connection)"
PDF
TXT
AUDIO
Features
ML Ready
dataset
Model
Task
Raw data

HERE ARE SOME FEATURE ENGINEERING TECHNIQUES FOR YOUR DATA SCIENCE TOOLBOX...

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)

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

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

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

- Binning is to group data according to specific rules
  - Achieve discretization of data
  - enhance data stability
  - reduce the risk of overfitting

- Binning is very necessary in logistic regression
- Tree models do not need to be binned

BINNING

Age	After Binning
2	Under 18
10	Under 18
99	Order than 60
49	30-60
23	18-30
1	Under 18

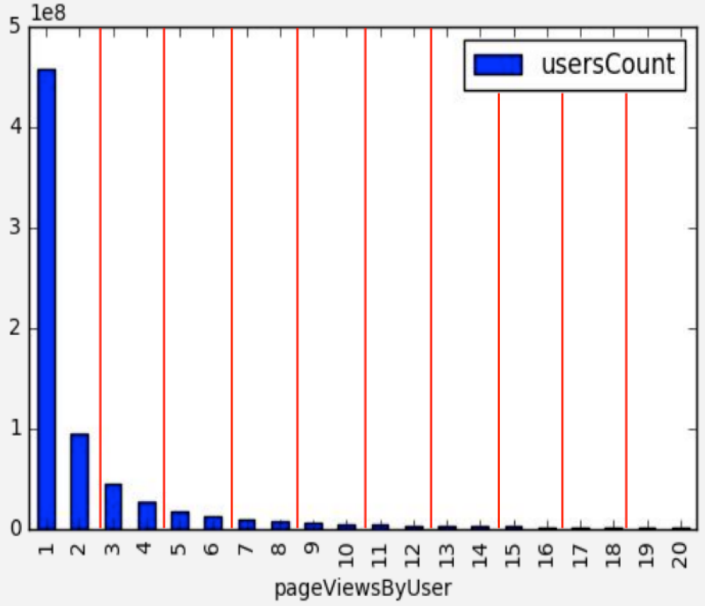
BINNING

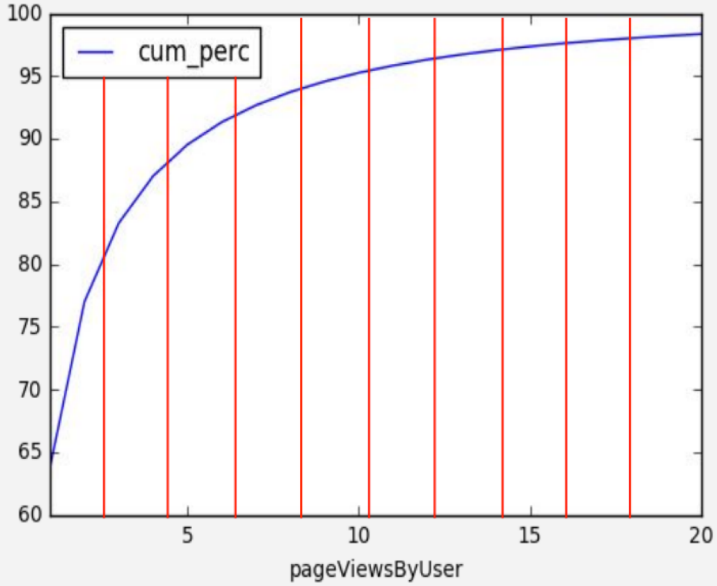
- From the theoretical stand-point, there are several possible methods of disretization (binning) a continuous variable as follows
  - Equal width discretization
  - Equal Frequency discretization
  - Discretization using decision trees

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?





EQUAL WIDTH DISCRETIZATION

- The bins or interval limits are determined so that each interval is of the same width.
  - Dividing that range into the amount of bins desired, (let's say 10)
  - *Note:* if the distribution is skewed, this technique does not improve the spread of the values.

EQUAL FREQUENCY DISCRETIZATION

- The boundaries of the intervals are determined so that each bin contains the same number of observations

- This is a better solution
  - We want to spread the values evenly across all bins.
- The usual approach is:
  - The percentiles
  - Quartiles to determine the intervals

DISCRETIZATION WITH DECISION TREES

- Sorting the observations into the tree end leaves, after training a decision tree.
  - Different leaves will contain different number of observations
  - It does not preserve frequency like equal frequency discretization.
- For some datasets, discretization with decision trees can improve model performance
  - Creating monotonic relationships

  (already capture some of the predictive power of the variable)

PYTHON

- So just look at the distribution stats (percentiles, quartiles) to determine the boundaries of the bins (categories). In such a way, you will address the fact of your data set to have more observations for people of Under 18.

- **Equal width discretization**: pd.cut()
- **Equal frequency discretization**: pd.qcut()

PYTHON

```python
import pandas as pd

# 导入一列数据
df = pd.DataFrame({'年龄': [29,7,49,12,50,34,36,75,61,20,3,11]})

df['等距分箱'] = pd.cut(df['年龄'] ,4)     # 实现等距分箱，分为4个箱
df['等频分箱'] = pd.qcut(df['年龄'],4)     # 实现等频分箱，分为4个箱

df
```

(Age)

(Equal width discretization)

(Equal frequency discretization)

RESULT

	Age	Equal width discretization	Equal frequency discretization
0	29	(21.0, 39.0]	(11.75, 31.5]
1	7	(2.928, 21.0]	(2.999, 11.75]
2	49	(39.0, 57.0]	(31.5, 49.25]
3	12	(2.928, 21.0]	(11.75, 31.5]
4	50	(39.0, 57.0]	(49.25, 75.0]
5	34	(21.0, 39.0]	(31.5, 49.25]
6	36	(21.0, 39.0]	(31.5, 49.25]
7	75	(57.0, 75.0]	(49.25, 75.0]
8	61	(57.0, 75.0]	(49.25, 75.0]
9	20	(2.928, 21.0]	(11.75, 31.5]
10	3	(2.928, 21.0]	(2.999, 11.75]
11	11	(2.928, 21.0]	(2.999, 11.75]

PYTHON

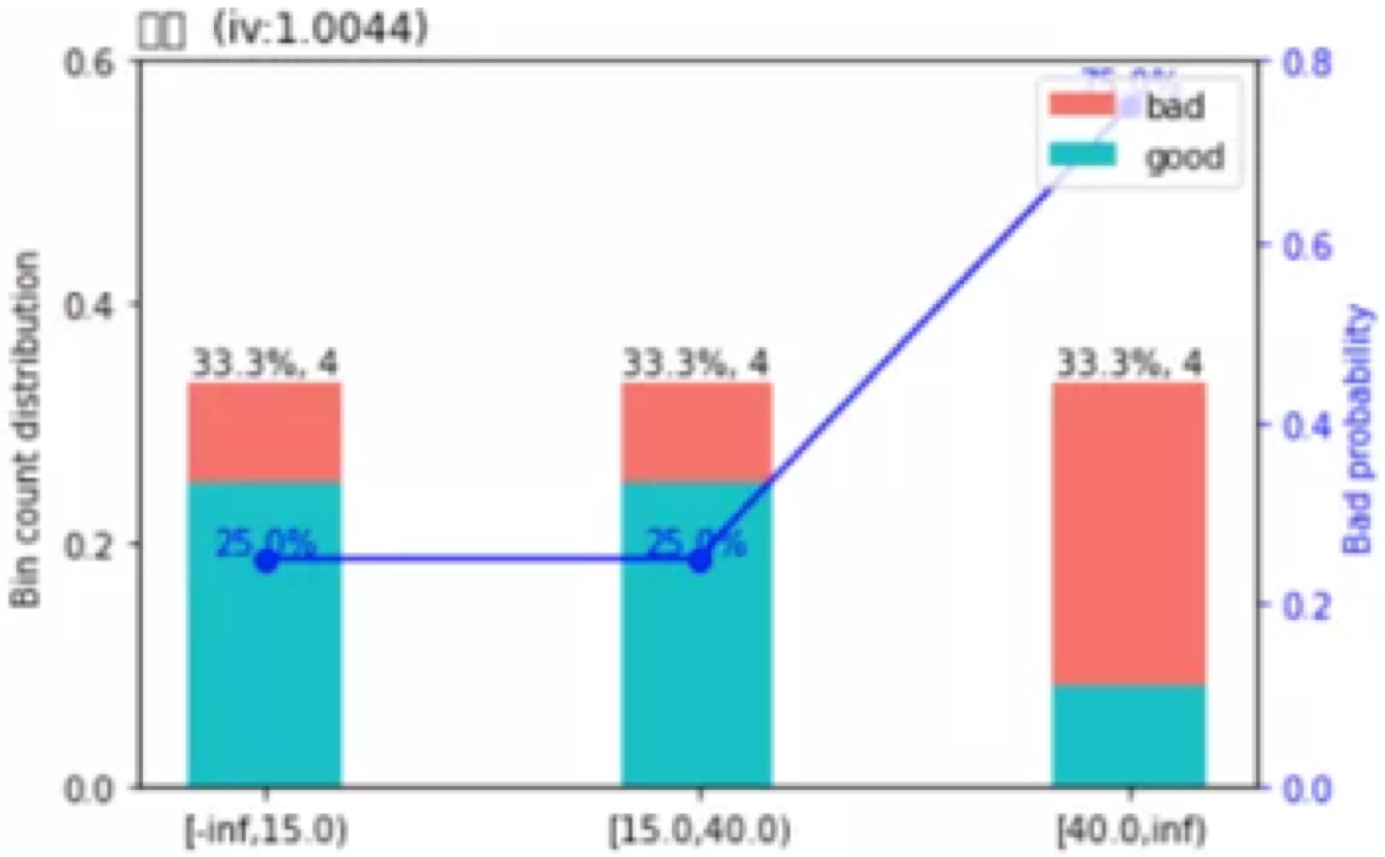
```python
import pandas as pd
import scorecardpy as sc

# 导入两列数据
df = pd.DataFrame({'年龄': [29,7,49,12,50,34,36,75,61,20,3,11],
                   'Y'   : [0,0,1,1,0,1,0,1,1,0,0,0]})

bins = sc.woebin(df, y='Y', method='tree')  # 决策树分箱
sc.woebin_plot(bins)
```

(Age)

RESULT


(iv:1.0044)
0.6
0.4
0.2
0.0
0.8
0.6
0.4
0.2
0.0
Bin count distribution
Bad probability
bad
good
33.3%, 4
33.3%, 4
33.3%, 4
25.0%
25.0%
[-inf,15.0)
[15.0,40.0)
[40.0,inf)

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

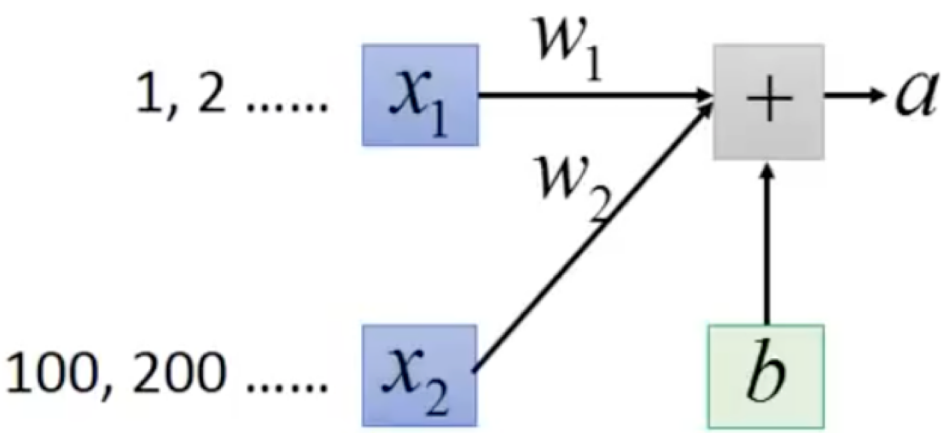
log(1+views_count)
6.91
6.22
5.71
5.30
5.02
4.62
4.26
3.93
3.43
3.04
2.40
1.79
0.69

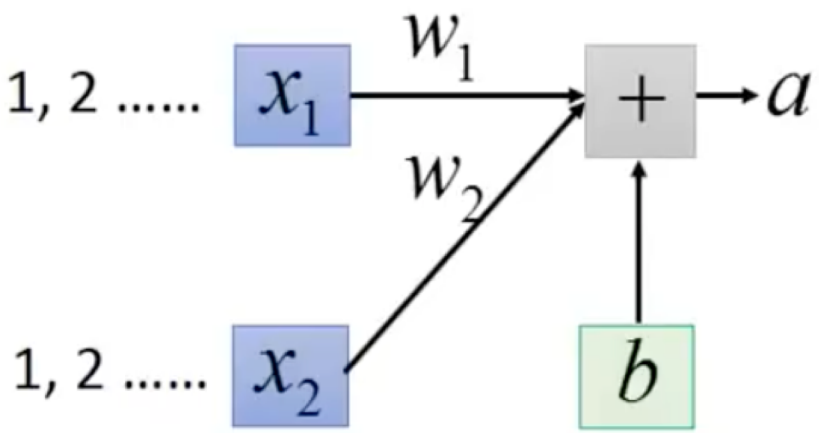
SCALING

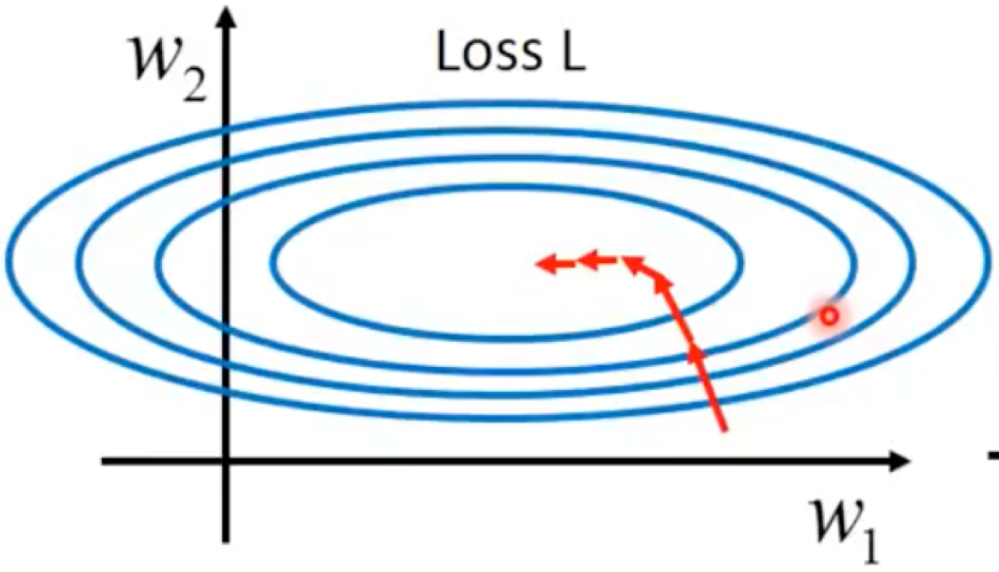
- Models that are smooth functions of input features are sensitive to the scale of the input (eg. Linear Regression)
- Scale numerical variables into a certain range, dividing values by a normalization constant (no changes in single-feature distribution)
- Popular techniques
  - MinMax Scaling
  - Standard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

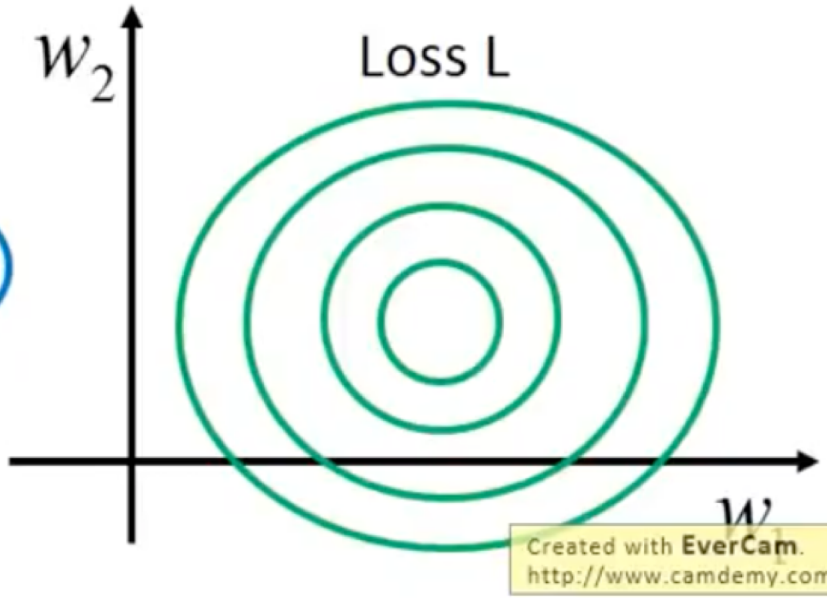
Feature Scaling

Make different features have the same scaling



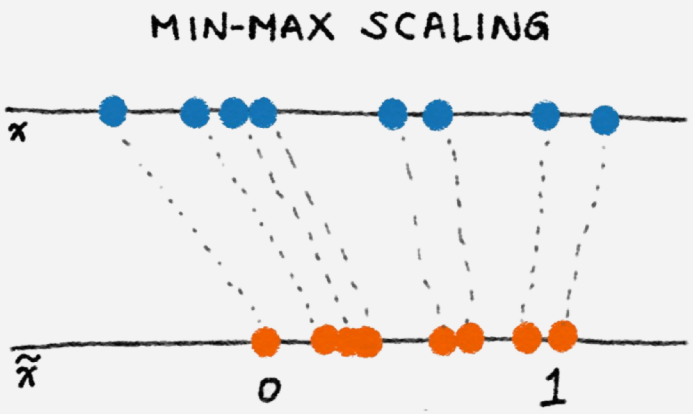






MIN-MAX SCALING

- 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.



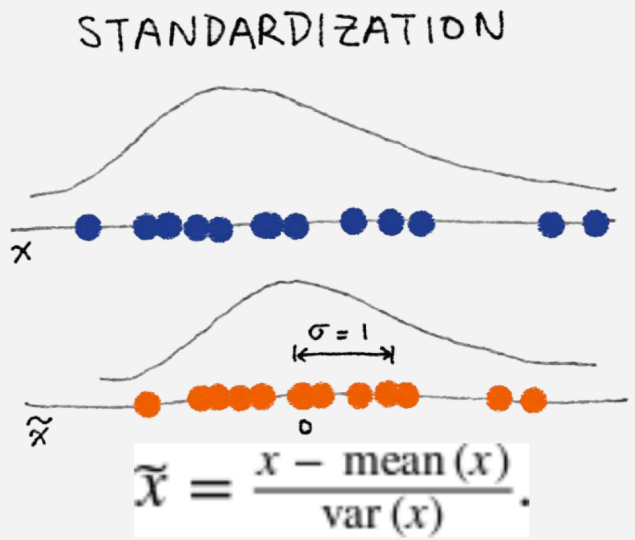
$$\tilde{x} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

```
>>> 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.])
```

INTERACTION FEATURES

- Simple linear models use a linear combination of the individual input features, x1 , x2 , ... xn to predict the outcome y.
- y = w1 x1 + w2 x2 + ... + w n x 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 = (x1,x2)

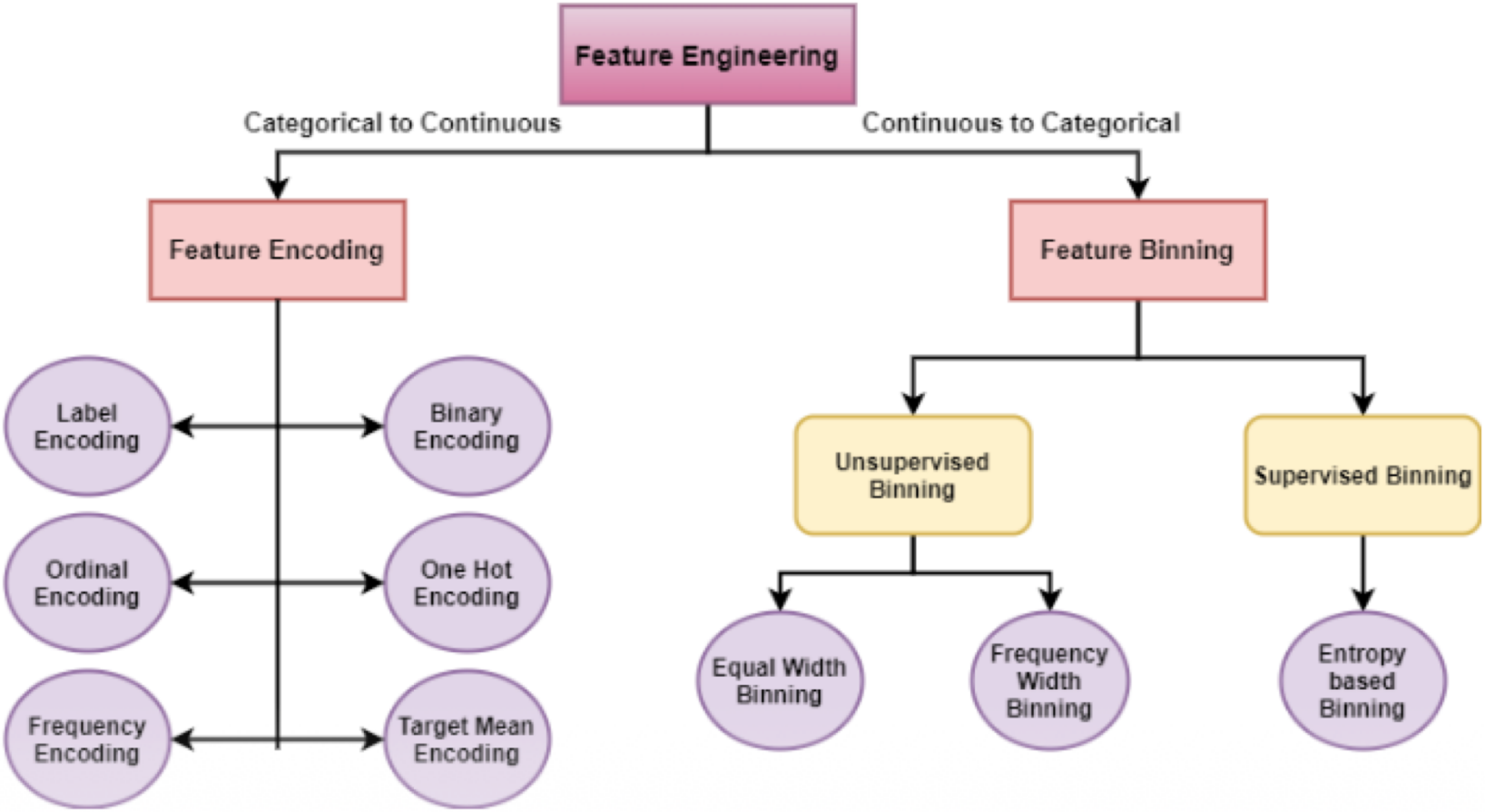
$$y = w_1x_1 + w_2x_2 + w_3x_1x_2 + w_4x_1^2 + w_4x_2^2$$

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.]])
```

SUMMARY


Feature Engineering
Categorical to Continuous
Continuous to Categorical
Feature Encoding
Feature Binning
Label Encoding
Binary Encoding
Ordinal Encoding
One Hot Encoding
Frequency Encoding
Target Mean Encoding
Unsupervised Binning
Supervised Binning
Equal Width Binning
Frequency Width Binning
Entropy based Binning

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”

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

country	
brazil	
chile	
venezuela	
colombia	
... 222 countries	

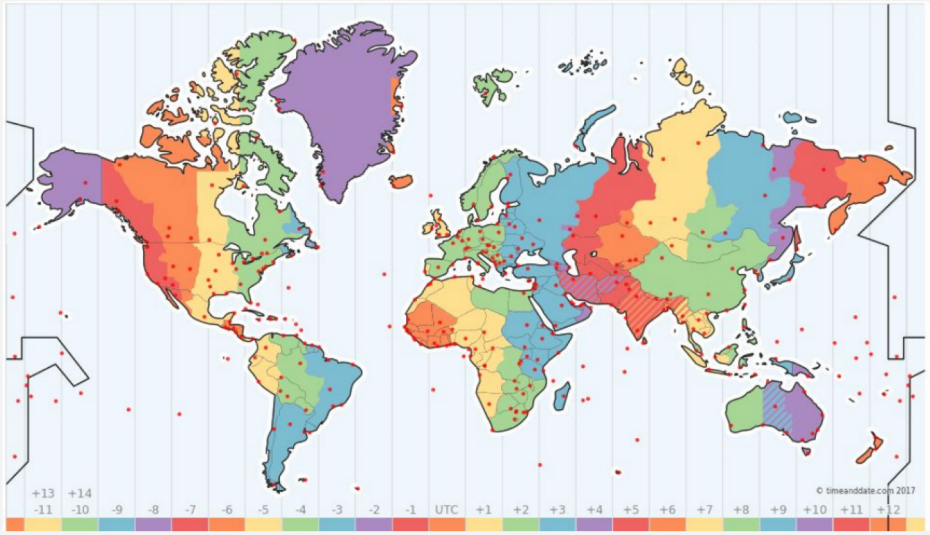
100 hashed columns

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	...
...	...	...	...	...

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 & TRENDLINES

- Time binning
  - Apply binning on time data to make it categorial and more general.
- 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 FEATURE

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

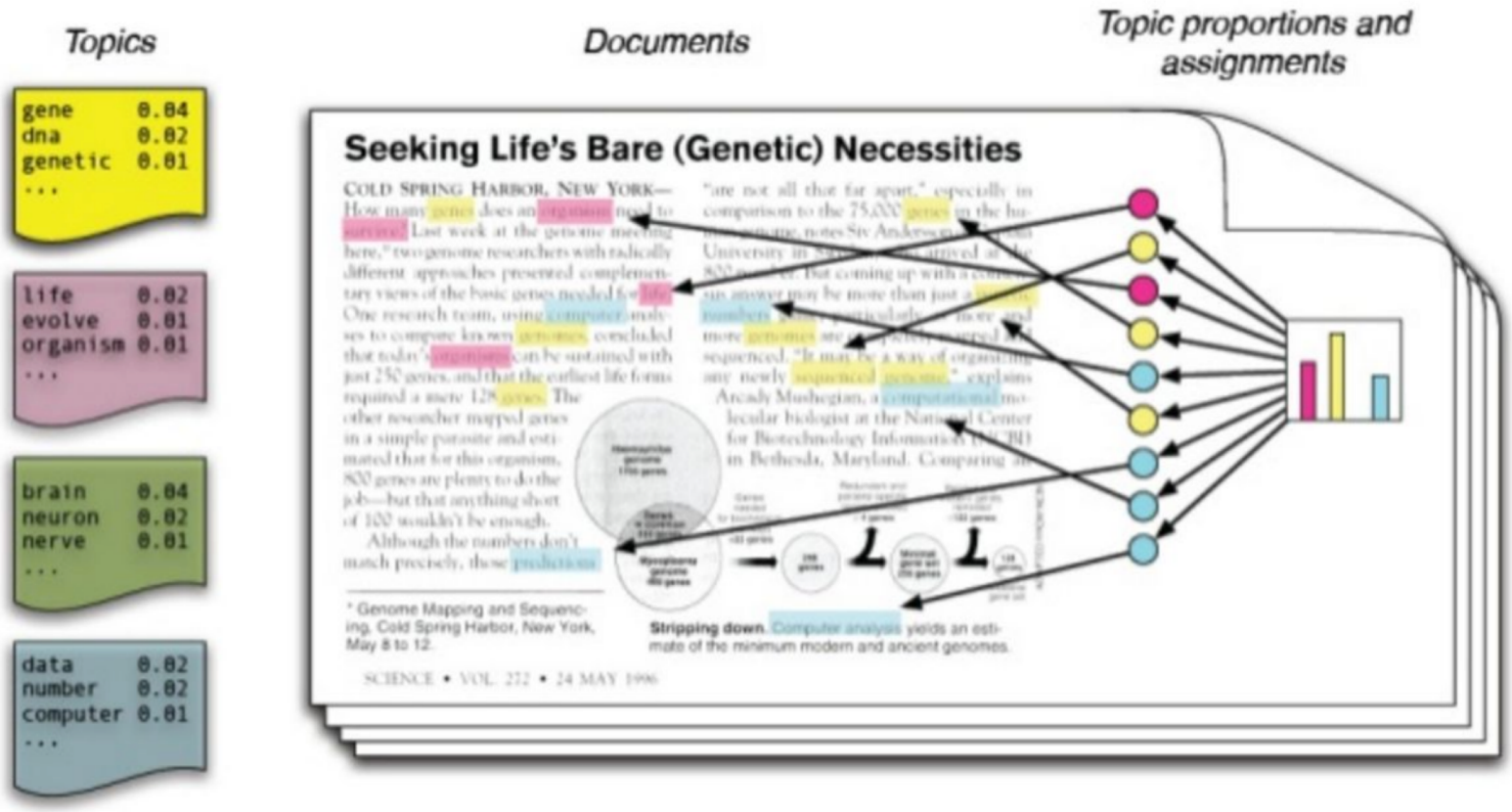
TEXT VECTORIZATION

- Represent each document as a feature vector in the vector space, where each position represents a word (token) and the contained value is its presence 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

TOPIC MODELING



FEATURE SELECTION

FEATURE SELECTION

- Reduces model complexity and training time
  - Filtering - 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)

RANDOM FOREST TREE

- Each tree of the random forest can calculate the importance of a feature according to its ability to increase the pureness of the leaves.

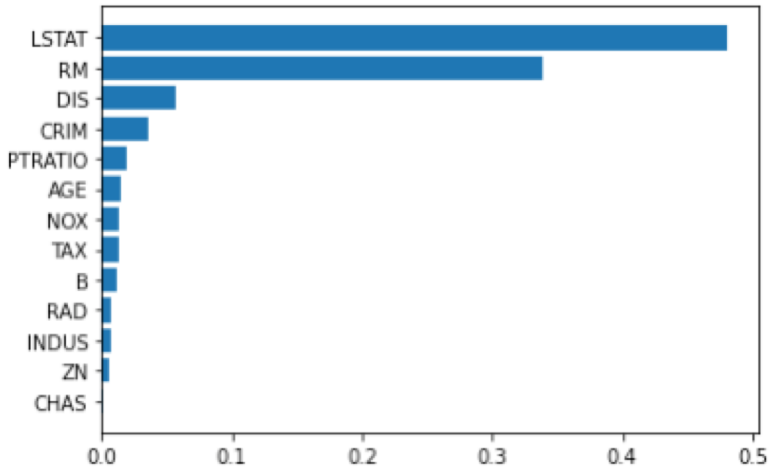
- It’s a topic related to how Classification And Regression Trees (CART) work.

```
rf = RandomForestRegressor(random_state=0)

rf.fit(X_train,y_train)
```

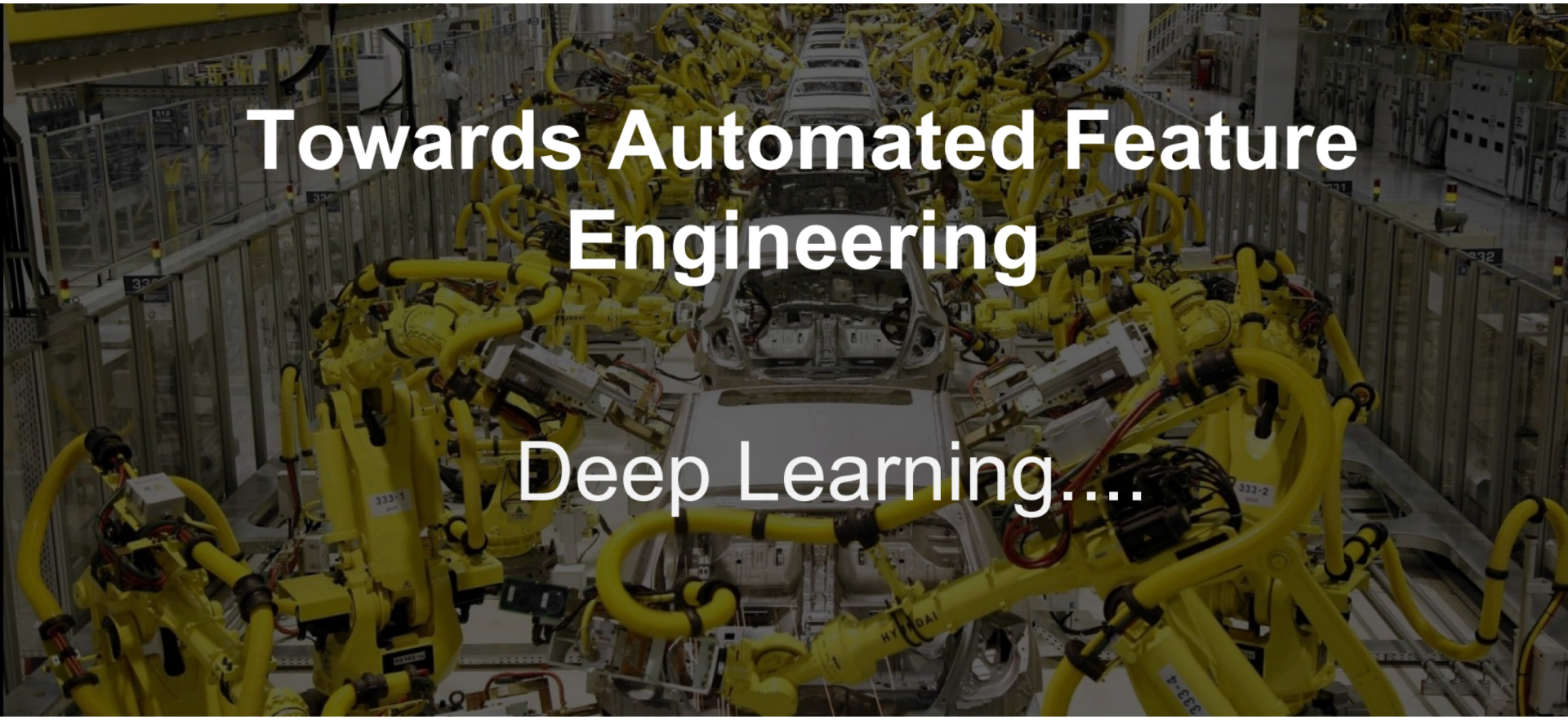
```
f_i = list(zip(features,rf.feature_importances_))
f_i.sort(key = lambda x : x[1])
plt.barh([x[0] for x in f_i],[x[1] for x in f_i])

plt.show()
```



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

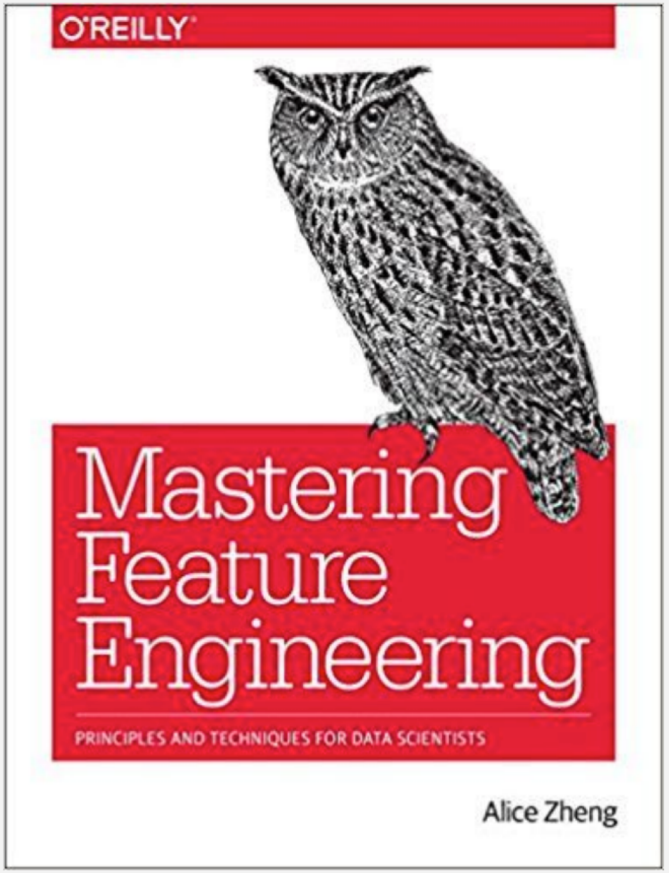
– Peter Norvig


Towards Automated Feature Engineering
Deep Learning....

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

– *Pedro Domingos*

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
西交利物浦大学