Machine Learning Practice (Part 1)

Zhirong Yang

Outline

- (Today) Part 1 focuses on steps before predictions
  - Prepare tabular data
  - Manipulate tabular data
  - Exploratory data analysis
  - Data cleaning
- (Later) Part 2 focuses on steps after predictions
  - Feature engineering
  - Model interpretation
  - Automatic machine learning

Prepare tabular data

Tabular data: example

ID	Color	Quality	Order Date	Location	Units	Amount
a1011	red	good	11/06/2016	BJ	11	563.19
a1013	blue	best	03/01/2017	GZ	23	1286.21
a1017	white	ok	13/03/2016	DG	12	345.62
a1022	red	bad	22/01/2016	SA	8	190.95
a1025	red	best	31/01/2017	BJ	98	3290.87
a1029	green	good	10/02/2017	GZ	30	1090.87
...	...	...	...	...	...	...
a3362	white	ok	28/02/2017	FS	20	600.28

Tabular data: example

ID	Color	Quality	Order Date	Location	Units	Amount
a1011	red	good	11/06/2016	BJ	11	563.19
a1013	blue	best	03/01/2017	GZ	23	1286.21
a1017	white	ok	13/03/2016	DG	12	345.62
a1022	red	bad	22/01/2016	SA	8	190.95
a1025	red	best	31/01/2017	BJ	98	3290.87
a1029	green	good	10/02/2017	GZ	30	1090.87
...	...	...	...	...	...	...
a3362	white	ok	28/02/2017	FS	20	600.28

Why tabular data?

- Excel is widely used
- Raw data is often stored in SQL tables
- Easy to add/remove fields

Tabular data: example

ID	Color	Quality	Order Date	Location	Units	Amount
a1011	red	good	11/06/2016	BJ	11	563.19
a1013	blue	best	03/01/2017	GZ	23	1286.21
a1017	white	ok	13/03/2016	DG	12	345.62
a1022	red	bad	22/01/2016	SA	8	190.95
a1025	red	best	31/01/2017	BJ	98	3290.87
a1029	green	good	10/02/2017	GZ	30	1090.87
...	...	...	...	...	...	...
a3362	white	ok	28/02/2017	FS	20	600.28

Why tabular data?

- Excel is widely used
- Raw data is often stored in SQL tables
- Easy to add/remove fields
- Need to convert the tabular data to a numeric matrix

Scalar data types

Scalar means result is invariant by changing the order these columns

- nominal
  - Examples: ID numbers, eye color, zip codes
- ordinal
  - Examples: rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in tall, medium, short
- interval
  - Examples: calendar dates, temperatures in Celsius or Fahrenheit.
- ratio
  - Examples: temperature in Kelvin, length, time, counts

One-Hot encoding of categorical data

- Ordinal, interval, and ratio data are usually treated as numeric
- For nominal
  - ID and other unique attributes do no contribute to ML
  - Reduce number of values
    - e.g. zip codes to areas
- One-hot encoding (e.g. Color)

ID	red	blue	white	red	green
a1011	1	0	0	0	0
a1013	0	1	0	0	0
a1017	0	0	1	0	0
a1022	0	0	0	1	0
a1025	0	0	0	1	0
a1029	0	0	0	0	1
...	...	...	...	...	...
a3362	0	0	1	0	0

- In software
  - `pandas.get_dummies`
  - `sklearn.preprocessing.OneHotEncoder`

One-Hot encoding of categorical data: demo

Structured data

Structured = result can alter by changing element order

- text
- speech
- other sequences
- image
- links

Bag-of-Words (BoW) representation

- Term Frequency (TF)
  - $\mathrm{TF}(d, t)$ is the count of the tth term in the dth document
  - Remove stop-list and weird words

Bag-of-Words (BoW) representation

- Term Frequency (TF)
  - $\mathrm{TF}(d, t)$ is the count of the tth term in the dth document
  - Remove stop-list and weird words
- Term frequency-Inverse document frequency (TFIDF)
  - Frequent terms dominate TF
  - TFIDF weights TF by IDF:

$$\mathrm{TFIDF}(d, t) = \mathrm{TF}(d, t) \cdot \log(1 + \frac{N}{n_t})$$

N: total number of documents
n_t: number of documents where the term t appears

Bag-of-Words (BoW) representation

- Term Frequency (TF)
  - $\mathrm{TF}(d, t)$ is the count of the tth term in the dth document
  - Remove stop-list and weird words
- Term frequency-Inverse document frequency (TFIDF)
  - Frequent terms dominate TF
  - TFIDF weights TF by IDF:

$$\mathrm{TFIDF}(d, t) = \mathrm{TF}(d, t) \cdot \log(1 + \frac{N}{n_t})$$

N: total number of documents
n_t: number of documents where the term t appears

- See `NLTK` or `TfidfVectorizer` in `sklearn`

Bag-of-Words (BoW) representation

- Term Frequency (TF)
  - $\mathrm{TF}(d, t)$ is the count of the tth term in the dth document
  - Remove stop-list and weird words
- Term frequency-Inverse document frequency (TFIDF)
  - Frequent terms dominate TF
  - TFIDF weights TF by IDF:

$$\mathrm{TFIDF}(d, t) = \mathrm{TF}(d, t) \cdot \log(1 + \frac{N}{n_t})$$

N: total number of documents
n_t: number of documents where the term t appears

- See `NLTK` or `TfidfVectorizer` in `sklearn`
- Pros and cons
  - \+ easy to compute at scale
  - \+ reasonably good result with TFIDF
  - − term order is not encoded
  - − need dimensionality reduction
- Applicable for other count data and sequence of symbols

Mel-frequency cepstrum (MFCC) for acoustic data

- MFCC represents the short-term power spectrum of a sound
  - It uses a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency
  - Tutorial available at http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/
- MFCC is the most widely used method for speech
- In python

```
from python_speech_features import mfcc
freq_samp, audio = wavfile.read("sample.wav")
features_mfcc = mfcc(audio, freq_samp)
```

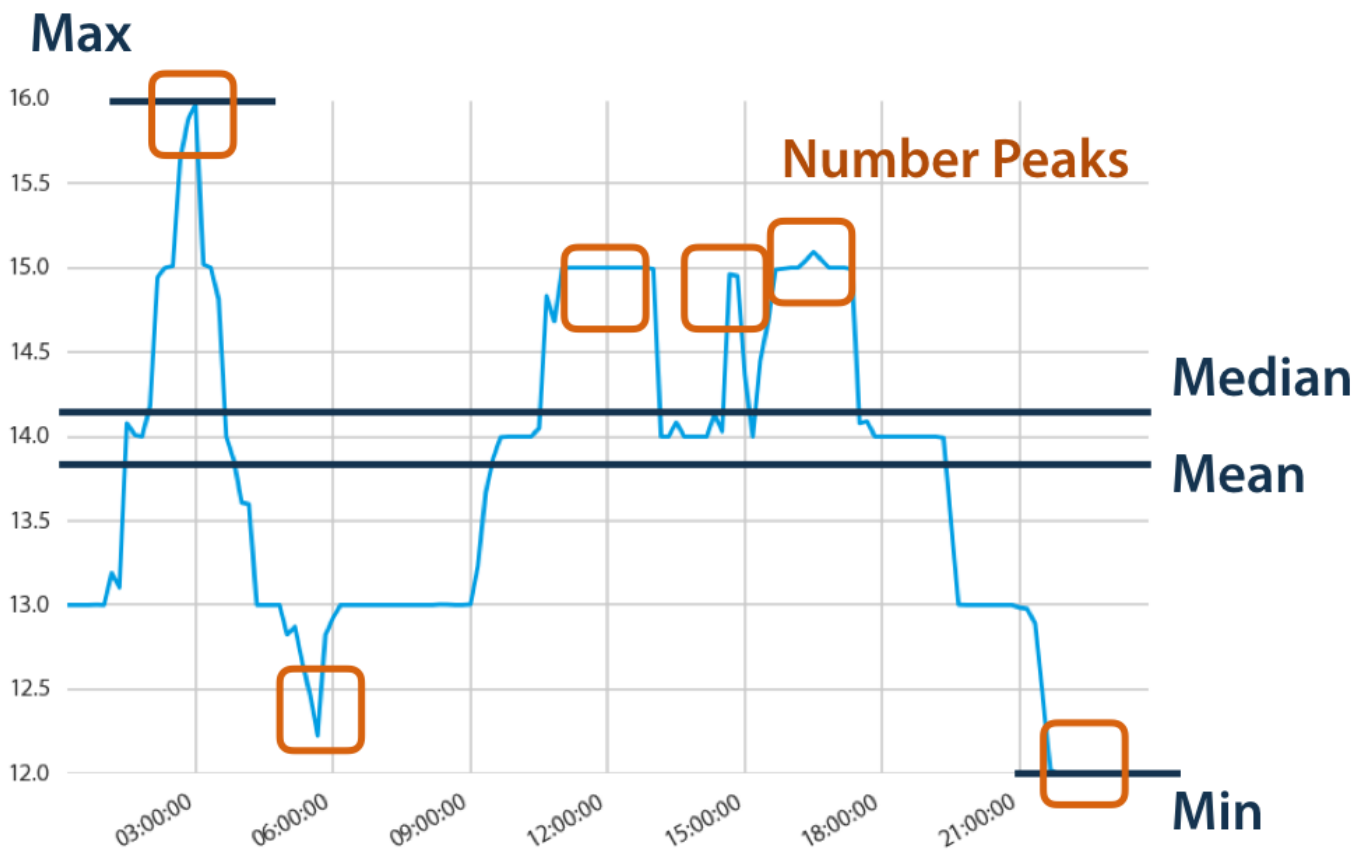
  - Outputs a matrix sized #frames $\times$ #cepstrum (default 13)
  - Adjusts parameters to change #frames (e.g. `winlen`, `samplerate`)
- Other signal processing methods (e.g. `logf` and `scattering`)

TSFRESH for time series data

- Can we directly use curves (if lengths are the same)?

TSFRESH for time series data

- Can we directly use curves (if lengths are the same)?
  - No, it is sensitive to small variation
- TSFRESH automatically extracts 100s of features from time series, e.g.

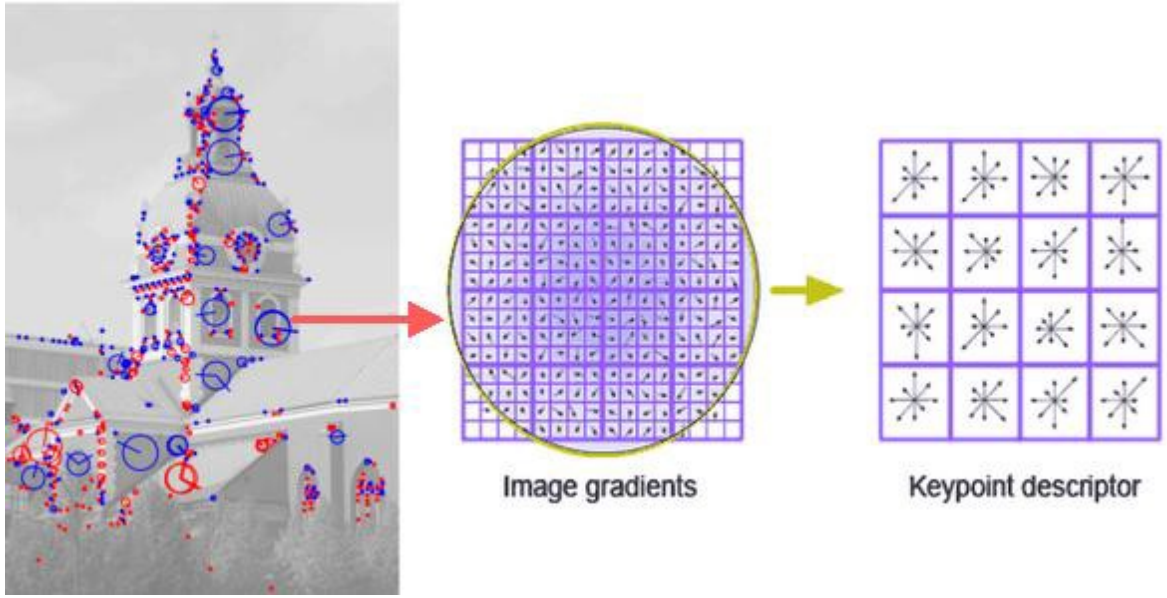


TSFRESH demo

Feature extraction for images

- Computer vision is far beyond this course; We only touch a tip of the iceberg
- OpenCV package (`cv2`) provides rich functionality
- Low-level feature extraction
  - fast; mainly based on signal processing
  - e.g. color, texture, shape, edges/corners
  - `skimage.feature` provides handy wrappers
  - See e.g.
    - `cv2.calcHist`
    - `skimage.feature.local_binary_pattern`
    - `skimage.feature.shape_index`

Scale Invariant Feature Transform (SIFT)



- A classical and widely used method
- Each image is described by a set of keypoint descriptors
- Can be treated as "visual words" after discretization
- See also
  - SURF and FAST (for faster extraction)
  - BRIEF (for shorter representation)
  - ORB (for free use)

(Additional reading) Pretrained models

Large-scale pretrained models using self-supervised learning

- BERT or GPT for text
- ViT/DINO/iBOT for images
- Other X2vec

HuggingFace is a hub of many pretrained models

Supervised feature extraction for structured data

- The state-of-the-art for structured data is deep learning
- Use supervised deep neural networks for individual structured fields
- Then use the network output as representation
- Actually is one of the stacking approaches

Supervised feature extraction for structured data

- The state-of-the-art for structured data is deep learning
- Use supervised deep neural networks for individual structured fields
- Then use the network output as representation
- Actually is one of the stacking approaches
- Pros and cons
  - \+ high accuracy if well tuned
  - \+ from end to end
  - − expensive in training time, tuning effort and computing resource
  - − hard to explain so far

Supervised feature extraction for structured data

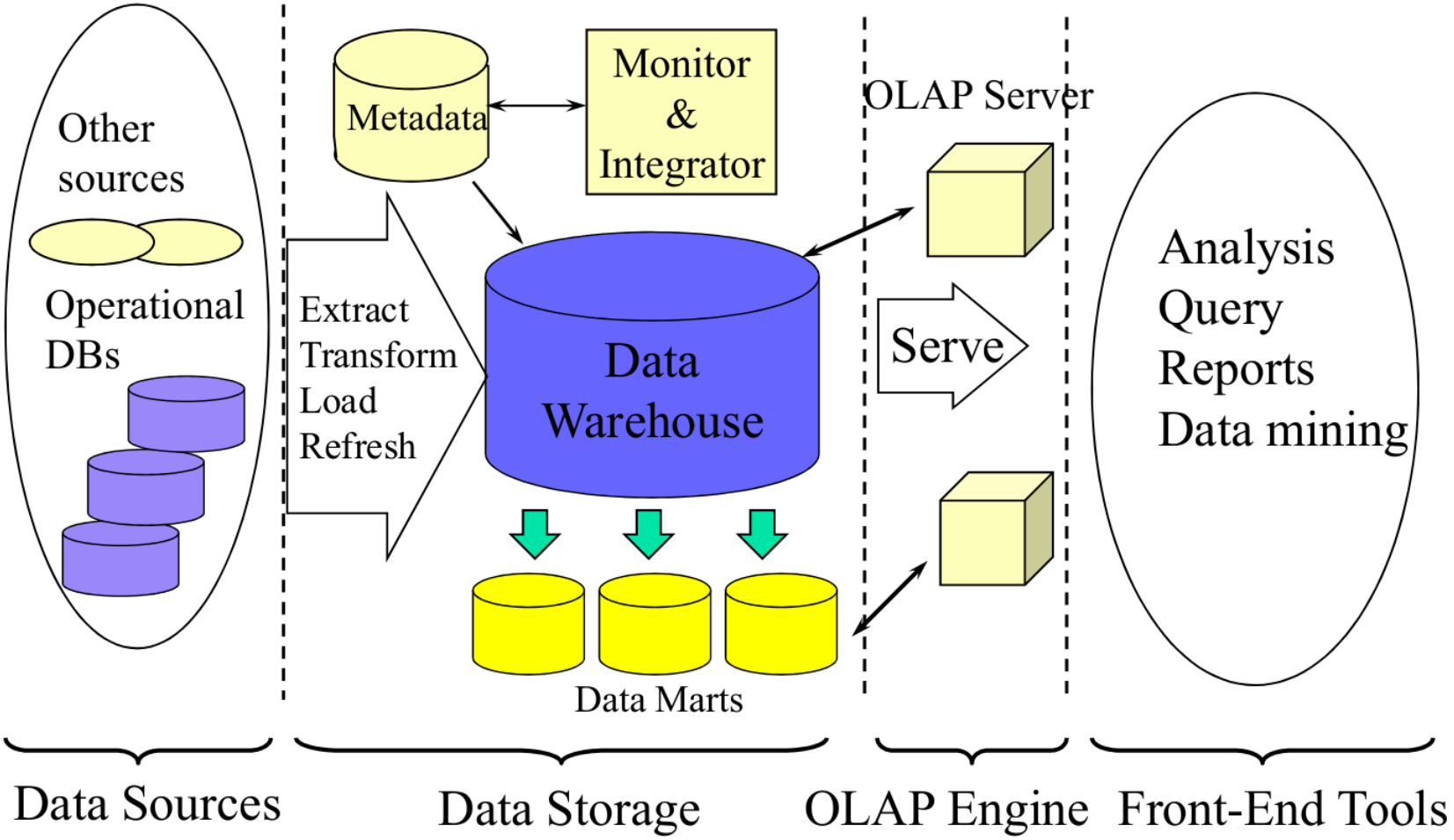
- The state-of-the-art for structured data is deep learning
- Use supervised deep neural networks for individual structured fields
- Then use the network output as representation
- Actually is one of the stacking approaches
- Pros and cons
  - \+ high accuracy if well tuned
  - \+ from end to end
  - – expensive in training time, tuning effort and computing resource
  - – hard to explain so far
- NTNU IDI provides the following courses
  - IT3030 Deep Learning
  - TDT4310 Intelligent Text Analytics and Language Understanding
  - TDT4265 Computer Vision and Deep Learning

(Additional reading) Links

- The data samples can link to each other, e.g.
  - Web pages have hyperlinks
  - User have friend circles
  - Articles have citations
- Represent the network nodes (data samples) as vectors
  - A: network adjacency matrix; $D_{ii} = \sum_j A_{ij}$
  - Laplacian Eigenmap: $[V, \Lambda] = \text{eig}\left(I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right)$
    - `sklearn.manifold.SpectralEmbedding`
  - Student t-distributed Stochastic Neighbor Embedding (t-SNE)
    - Good for clustering
  - Large-scale Information Network Embedding (LINE)
    - https://github.com/tangjianpku/LINE
  - Use Multi-dimensional Scaling (MDS) if the links are distances
    - `sklearn.manifold.MDS`

Manipulate tabular data

Data Warehouse



Pandas

- Pandas is the Python tool for data warehousing
- The name Pandas comes from the word Panel Data — an Econometrics from Multidimensional data.

Multidimensional data

- Data warehouse uses multidimensional data
- $v = f(x_1, x_2, \ldots, x_d)$
  - $d = 1$: vector
  - $d = 2$: matrix (if all numeric) or tabular
  - $d = 3$: tensor (if all numeric)
- Most common format for *unstructured data*
  - Good mathematical tools
  - Computer friendly
- Pandas supports $d <= 3$
- Tabular data is widely used in ML

Pandas data structure

- **Series** $\approx$ vector or 1D array

10	23	56	17	52	61	73	90	26	72

- **DataFrame** $\approx$ matrix, table, or 2D array, container of Series

Name	Age	Gender	Rating
Steve	32	Male	3.45
Lia	28	Female	4.6
Vin	45	Male	3.9
Katie	38	Female	2.78

- **Panel** $\approx$ cube or 3D array, container of DataFrame

Data interface

Extraction, Transformation, and Loading (ETL)

- Data extraction
  - get data from multiple, heterogeneous, and external sources
- Data cleaning
  - detect errors in the data and rectify them when possible
- Data transformation
  - convert data from legacy or host format to warehouse format
- Load
  - sort, summarize, consolidate, compute views, check integrity, and build indicies and partitions
- Refresh
  - propagate the updates from the data sources to the warehouse

Data interface examples

Handy import from or export to various formats

- `pd.read_sql(sql,con)`: Read SQL query or database table into a DataFrame
  Example

```
import pyodbc
conn = pyodbc.connect(r'Driver={Microsoft Access Driver (*.mdb,*.accdb)};'+
                      r'DBQ=C:\Users\Zhirong\Desktop\sales.accdb;')
df = pd.read_sql('SELECT * FROM sales', self.conn)
```

- `pd.read_csv(filepath_or_buffer)`: Read a comma-separated values (csv) file into DataFrame.
  - accepts URL, including http, ftp, s3, and file
  - e.g. `pd.read_csv('data.csv')`
- `df.to_numpy()`: convert DataFrame to Numpy ndarray

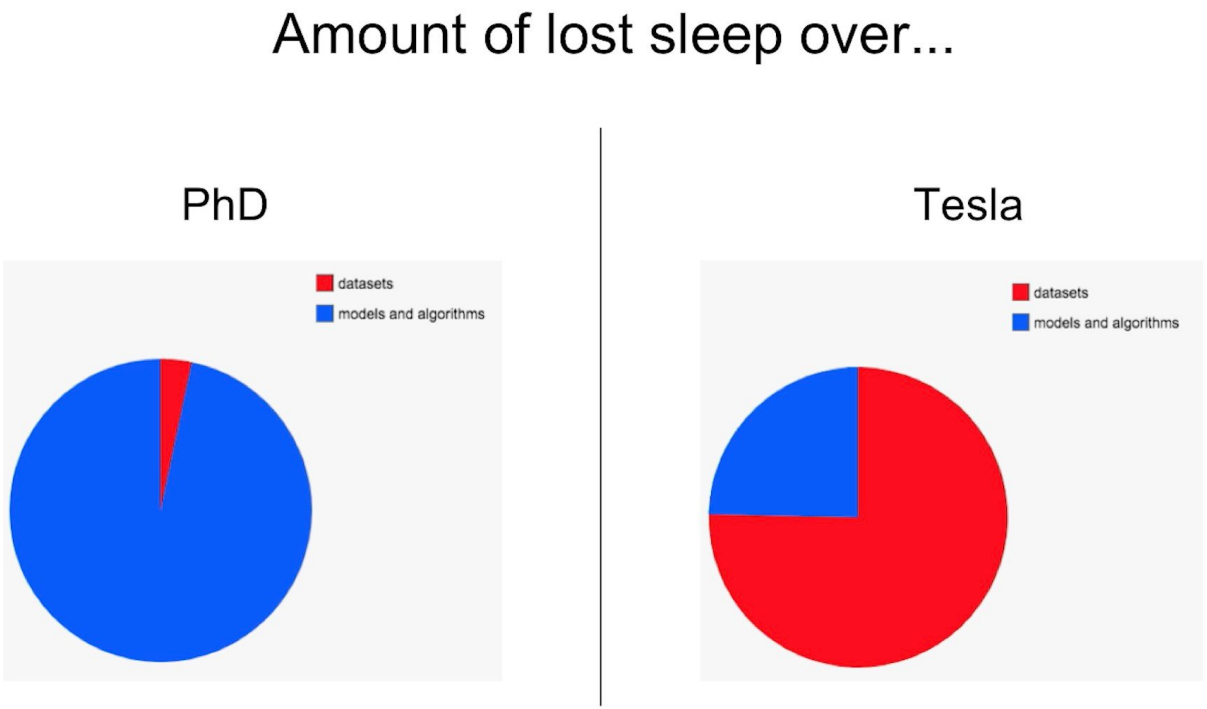
A simple Pandas example

Pandas further reading

- 10 minutes to Pandas
  https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html
- Tutorials point
  https://www.tutorialspoint.com/python_pandas/
- Cheat sheet
  https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf
- More tutorials
  https://pandas.pydata.org/pandas-docs/stable/getting_started/tutorials.html

Exploratory Data Analysis (EDA)

Become one with the data

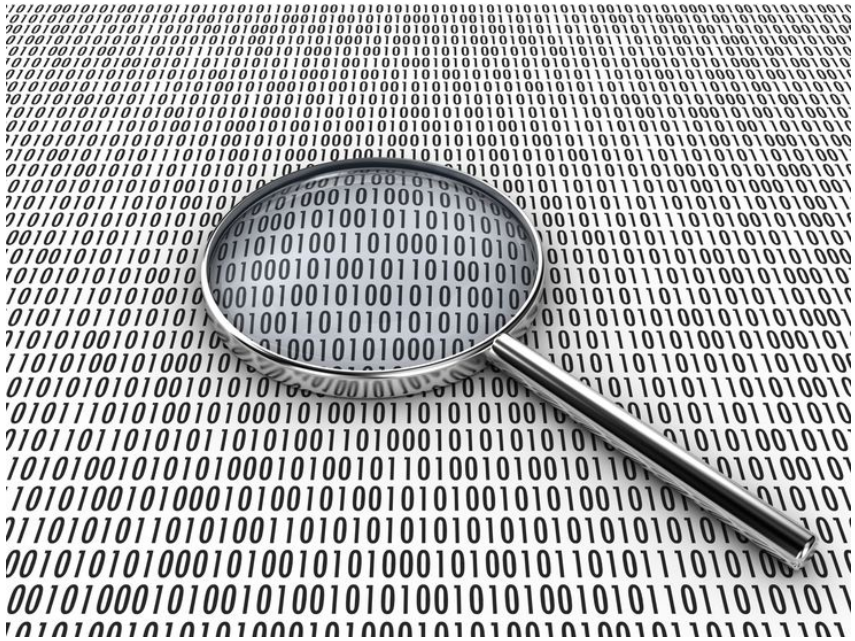


"I like to spend copious amount of time (measured in units of hours) scanning through thousands of examples, understanding their distribution and looking for patterns. Luckily, your brain is pretty good at this."

—Andrej Karpathy

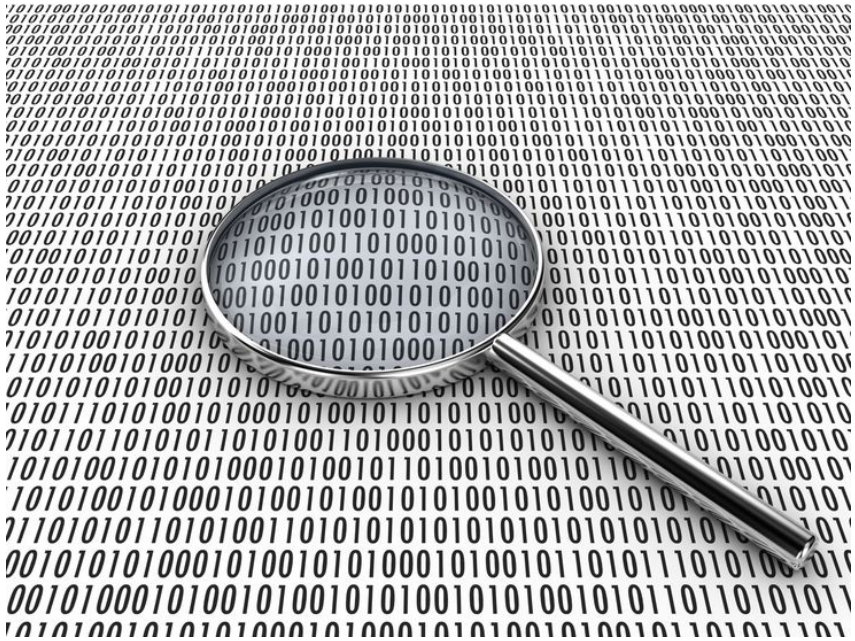
EDA allows to

- Better understand the data
- Build an intuition about the data
- Generate hypothesizes
- Find insights



EDA allows to

- Better understand the data
- Build an intuition about the data
- Generate hypothesizes
- Find insights



Do EDA first. Do not immediately dig into modeling.

Motivating example

person id	person info	promo info	#promos sent	# promos used	used this promo?
14	...	...	13	4	1
3	...	...	43	35	0
0	...	...	6	0	1
32	...	...	15	13	1

Motivating example

person id	...	#promos sent	# promos used	diff	used this promo?
13	...	0	0	1	1
13	...	1	1	0	0
13	...	2	1	1	1
13	...	4	2	1	1
13	...	5	3	1	1
13	...	6	3	NaN	1

Building intuition about the data

1. Getting domain knowledge
   - It helps to deeper understand the problem
2. Checking if the data is intuitive
   - agrees with domain knowledge
3. Understanding how the data was generated
   - As it is crucial to set up a proper validation

Check if the data is intuitive (example 1)

...	**Age**	...
...	21	...
...	45	...
...	**336**	...
...	19	...
...	...	...

Check if the data is intuitive (example 2)

Predict advertiser's cost

AdGroupId	AdNetwork Type2	MaxCpc	Slot	Clicks	Impressions	...
78db044136	s	0.28	s_2	3	0	...
68a0110c33	s	1	s_2	1	13	...
2r39fw11w3	p	1.2	p_1	3	419	...
...	...	...	...	...	...	...

Check if the data is intuitive (example 2)

Predict advertiser's cost

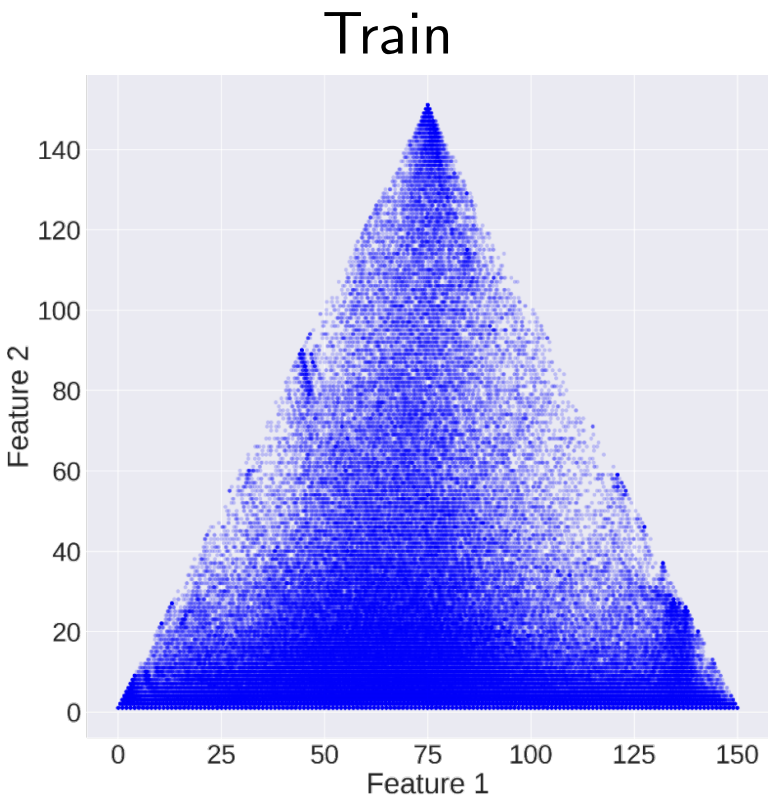
AdGroupId	AdNetwork Type2	MaxCpc	Slot	Clicks	Impressions	...
78db044136	s	0.28	s_2	3	0	...
68a0110c33	s	1	s_2	1	13	...
2r39fw11w3	p	1.2	p_1	3	419	...
...	...	...	...	...	...	...

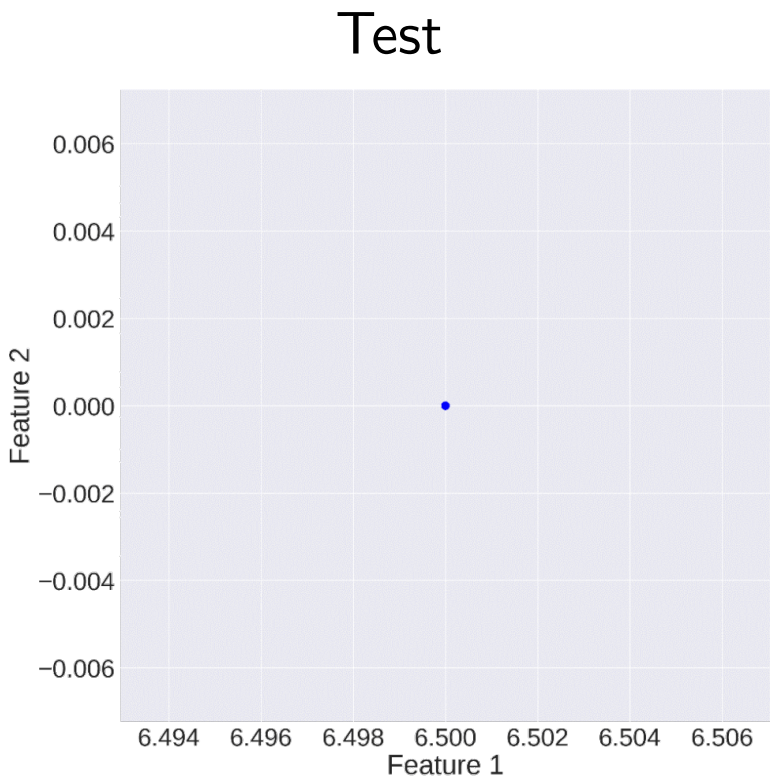
Check if the data is intuitive (example 2)

Predict advertiser's cost

AdGroupId	AdNetwork Type2	MaxCpc	Slot	Clicks	Impressions	is_incorrect
78db044136	s	0.28	s_2	3	0	True
68a0110c33	s	1	s_2	1	13	False
2r39fw11w3	p	1.2	p_1	3	419	False
...	...	...	...	...	...	...

Understand how the data was generated





Anonymized data

Original text	Encoded text
I want this table	7ugy 972h 98ww hj34
Table is what I want	hj34 4f08 rtte 7ugy 972h
This table is red	98ww hj34 4f08 4rj9
And this is me	jk8r 98ww 4f08 9jo4

Anonymized data

Original text	Encoded text
I want this table	7ugy 972h 98ww hj34
Table is what I want	hj34 4f08 rtte 7ugy 972h
This table is red	98ww hj34 4f08 4rj9
And this is me	jk8r 98ww 4f08 9jo4

id	x1	x2	x3	x4	x5	x6
1	m268i97y	0	NO	105.4	14	
2	j0gheu6	1	YES	25.631	12	
3	26fmsp6u	1	NO	12.0	12	m268i97y
4	13e5dpzp	0	NO	140.12	14	m268i97y

Anonymized data

Original text	Encoded text
I want this table	7ugy 972h 98ww hj34
Table is what I want	hj34 4f08 rtte 7ugy 972h
This table is red	98ww hj34 4f08 4rj9
And this is me	jk8r 98ww 4f08 9jo4

id	x1	x2	x3	x4	x5	x6
1	m268i97y	0	NO	105.4	14	
2	j0gheu6	1	YES	25.631	12	
3	26fmsp6u	1	NO	12.0	12	m268i97y
4	13e5dpzp	0	NO	140.12	14	m268i97y

- Often appears in data competitions
- Sometimes happens for sensitive clients
- Reverse feature engineering (i.e. guessing) is needed

Visualization

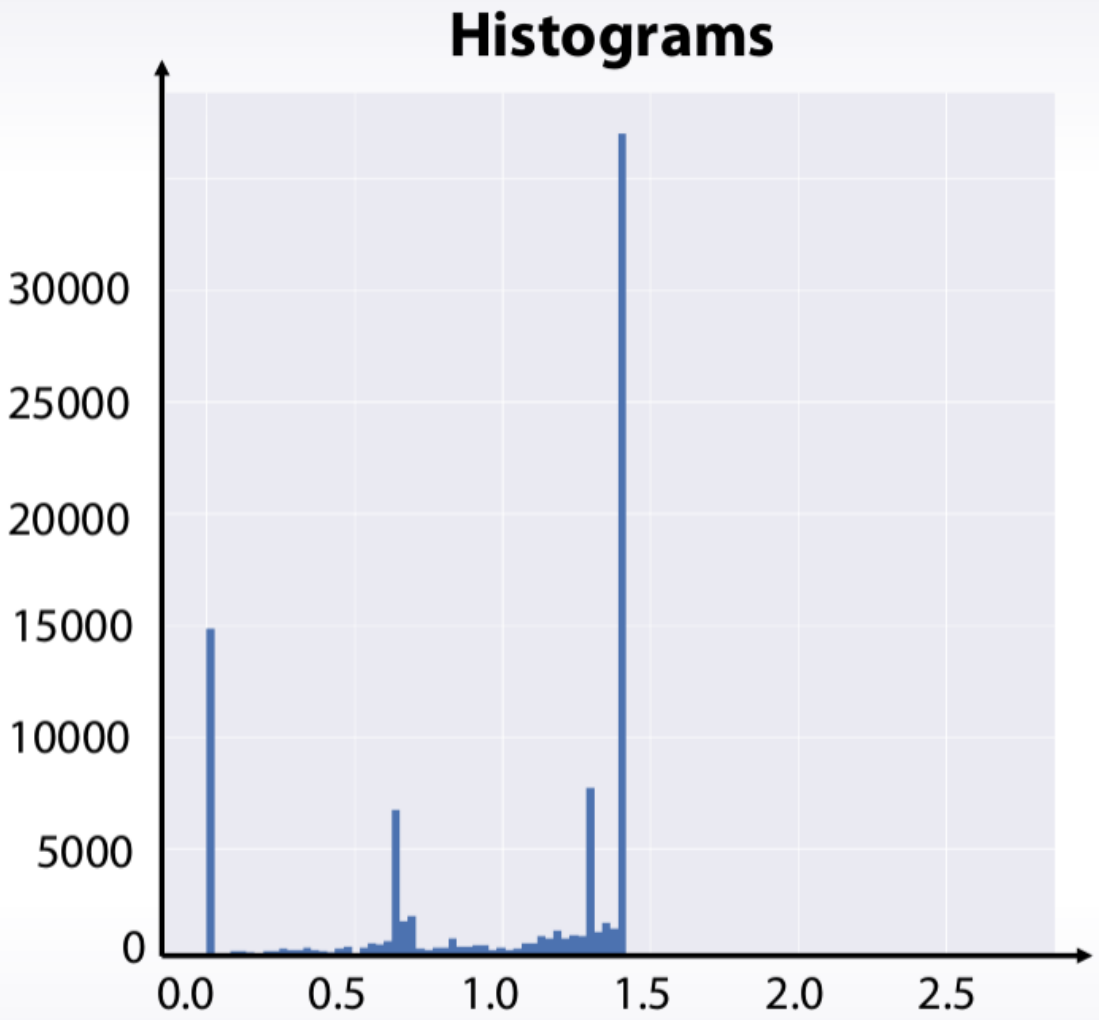
- EDA is an art!

Visualization

- EDA is an art!
- And visualizations are our art tools
- Visualization tools to
  - Explore individual features
    - Histograms
    - Plots
    - Statistics
  - Explore feature relations
    - Scatter plots
    - Correlation plots
    - Plot (index vs feature statistics)
    - etc.

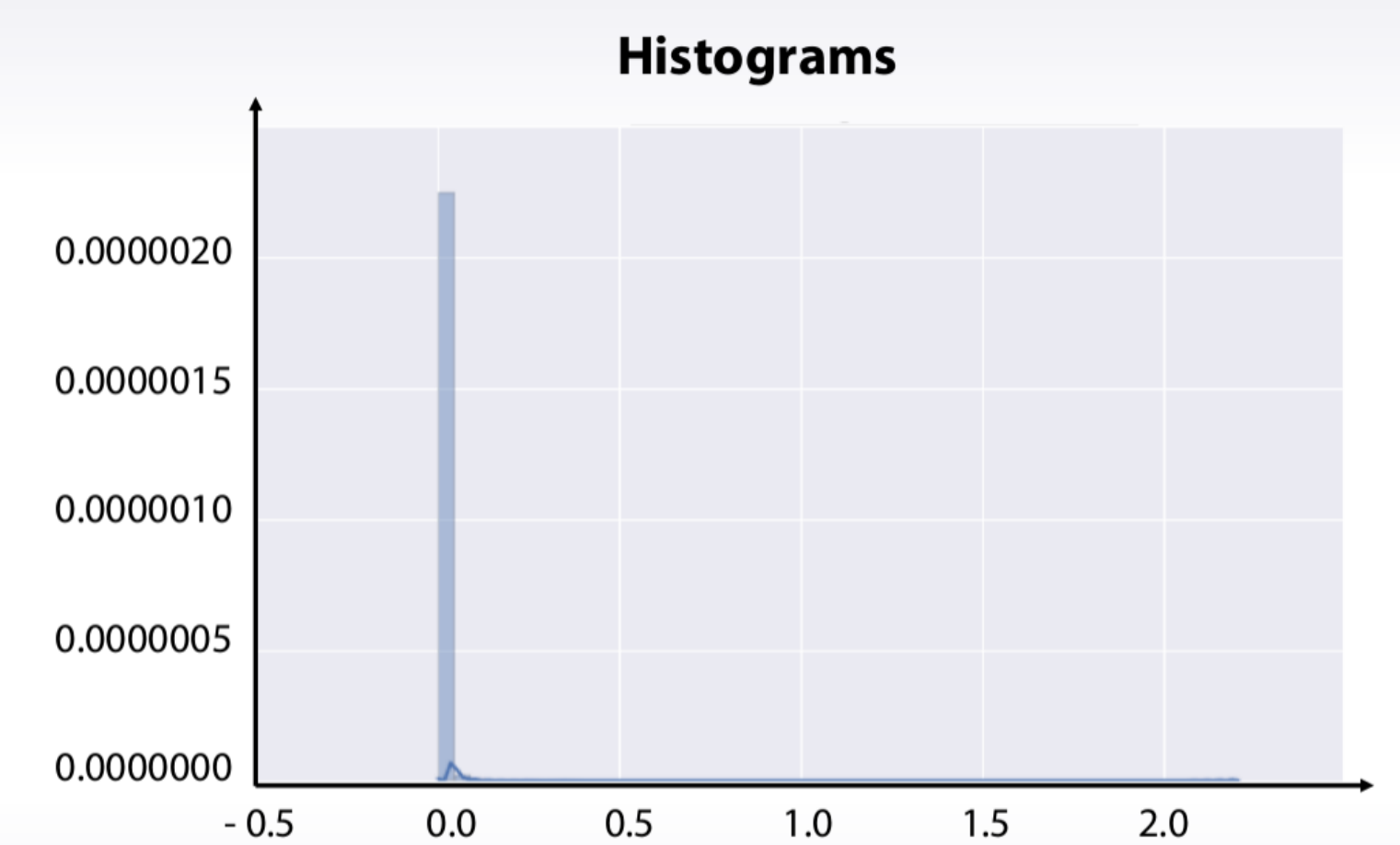
Tools for individual features exploration

Histogram



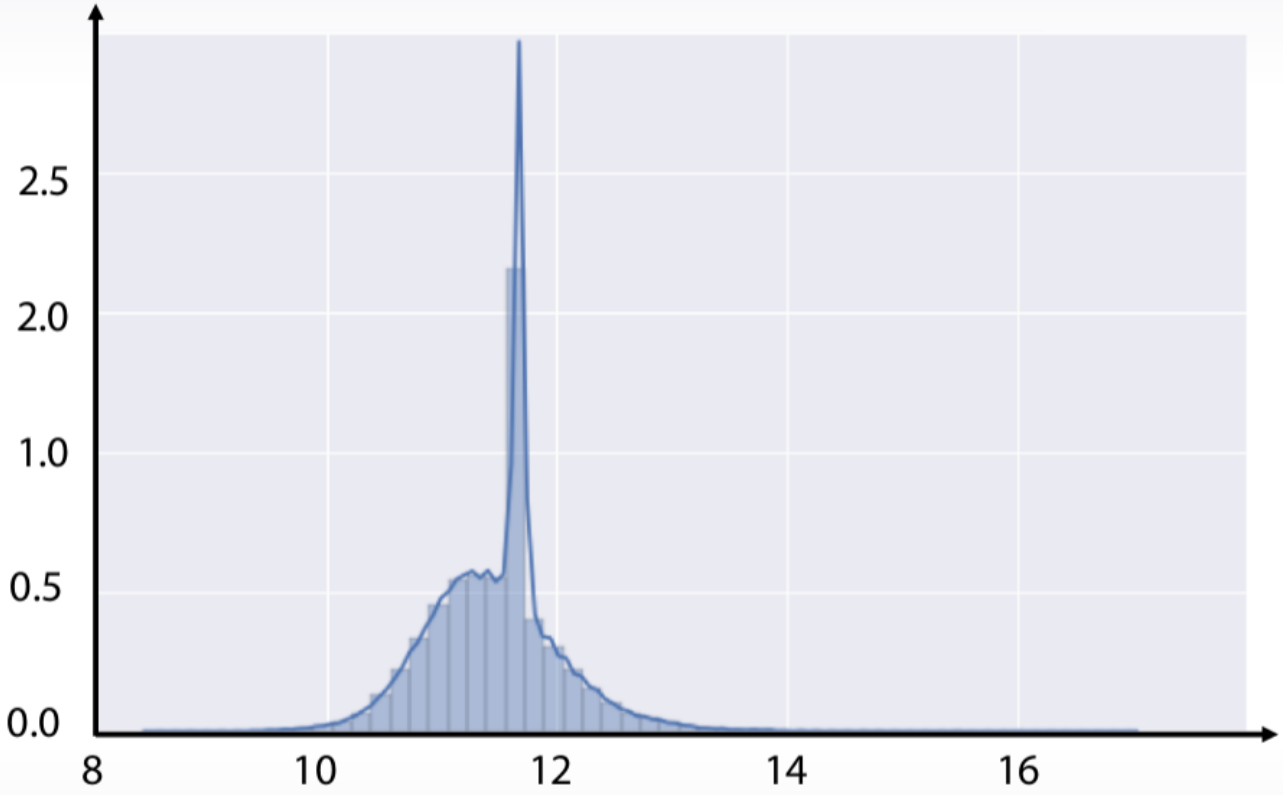

```
plt.hist(x)
```

Histogram



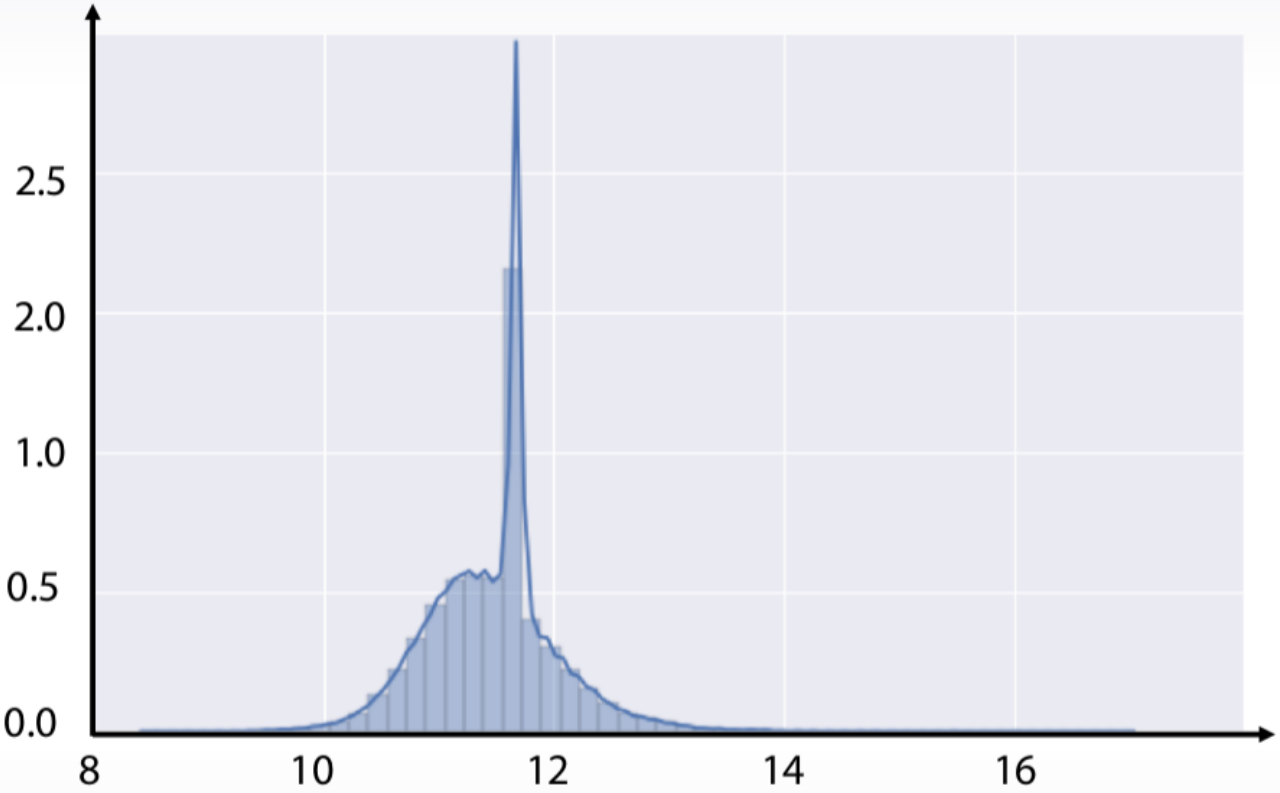

```
plt.hist(x)
```

Histogram




```
plt.hist(x)
```

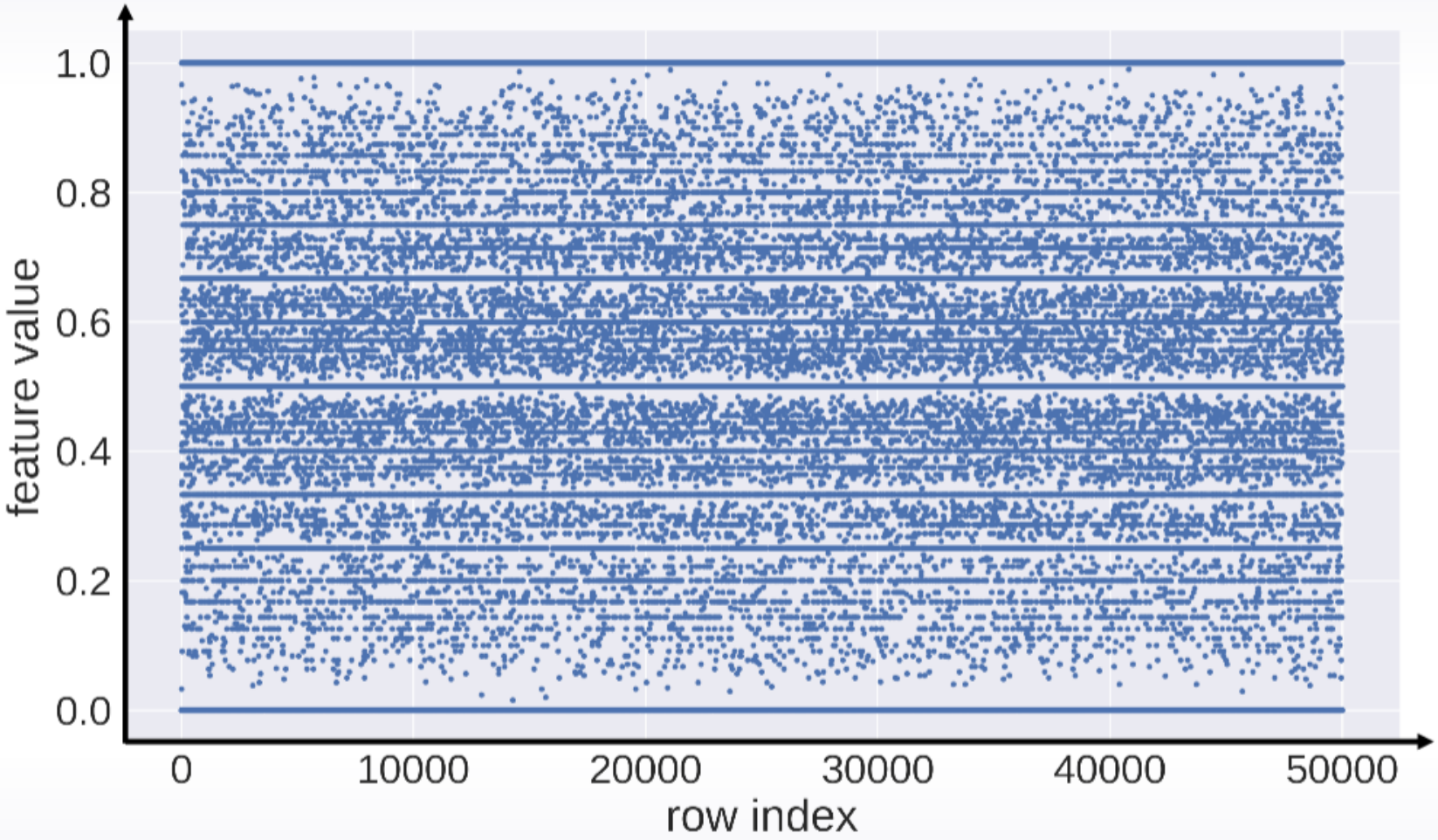
Histogram




```
plt.hist(x)
```

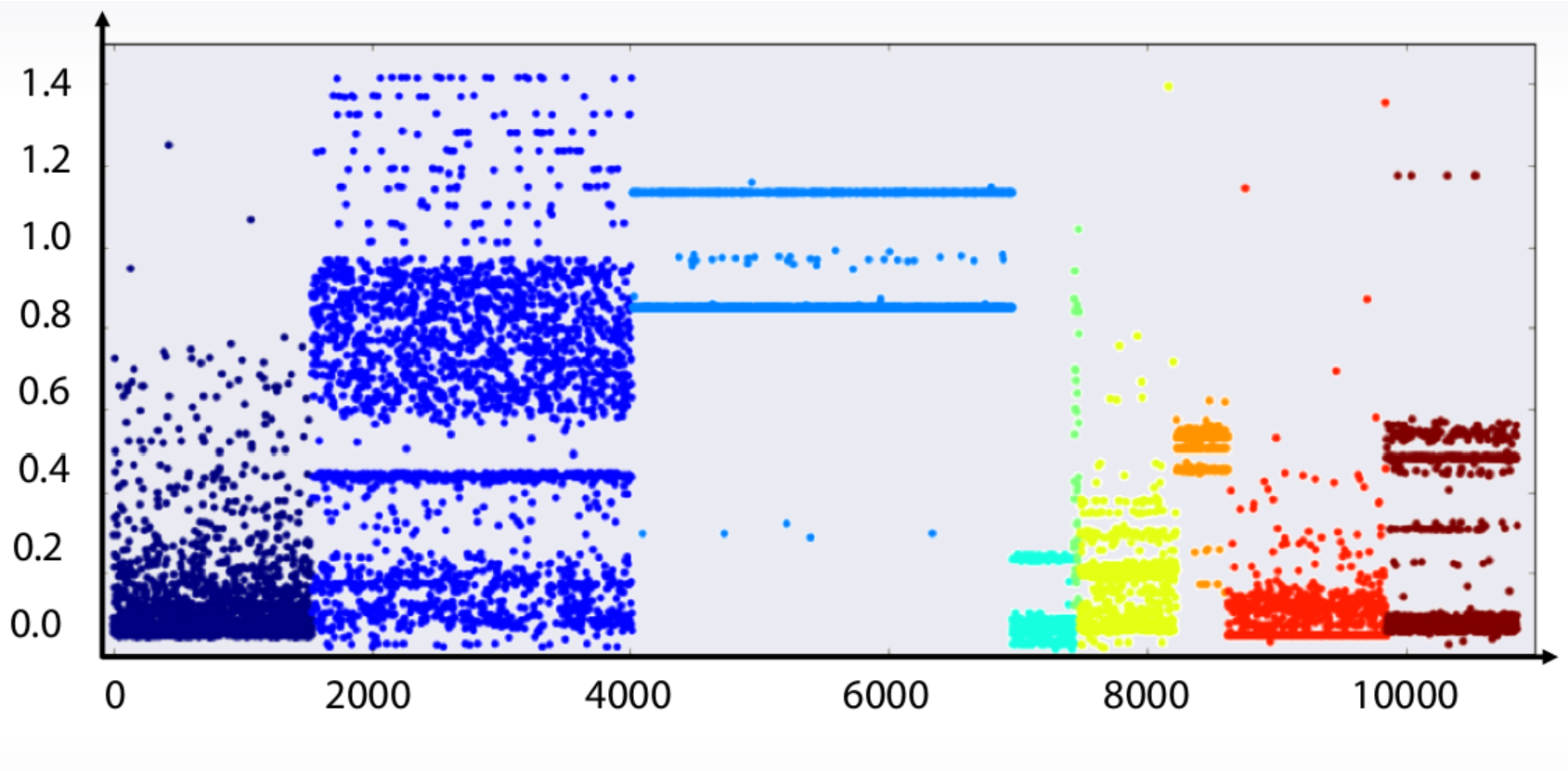
$$\ln(99999) = 11.5129$$

Plot (index versus value)




```
plt.plot(x,'.')
```

Plot (index versus value)



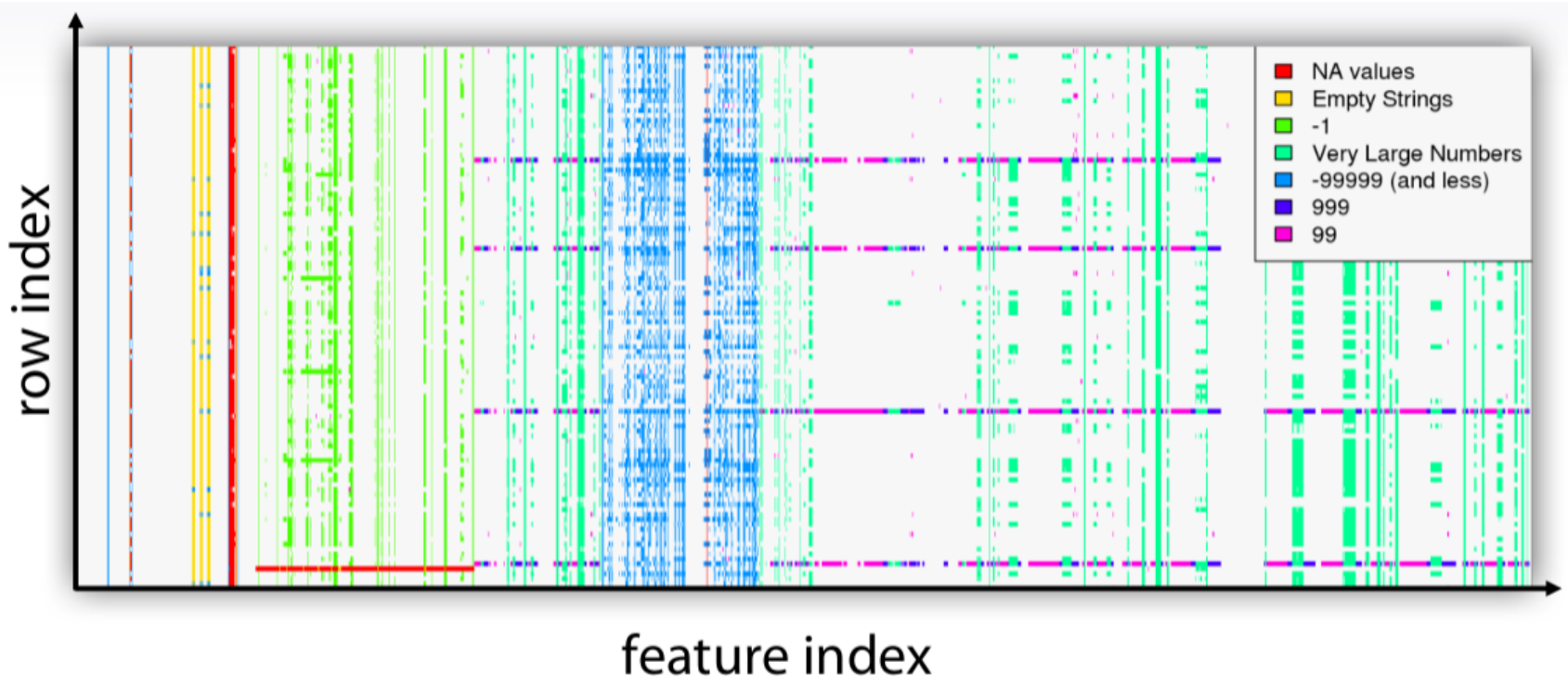

```
plt.scatter(range(len(x)), x, c=y)
```

Feature statistics

	x6	x7	x8	x13
count	50000.00000	50000.000000	48793.000000	45512.000000
mean	0.99296	0.975860	-0.000252	4428.915253
std	0.08361	0.153485	1.023282	10943.884658
min	0.00000	0.000000	-85.252444	-99.000000
25%	1.00000	1.000000	-0.255490	0.000000
50%	1.00000	1.000000	0.221047	1817.000000
75%	1.00000	1.000000	0.567620	5582.000000
max	1.00000	1.000000	3.426844	776759.000000

```
df.describe()
x.mean()
x.var()
```

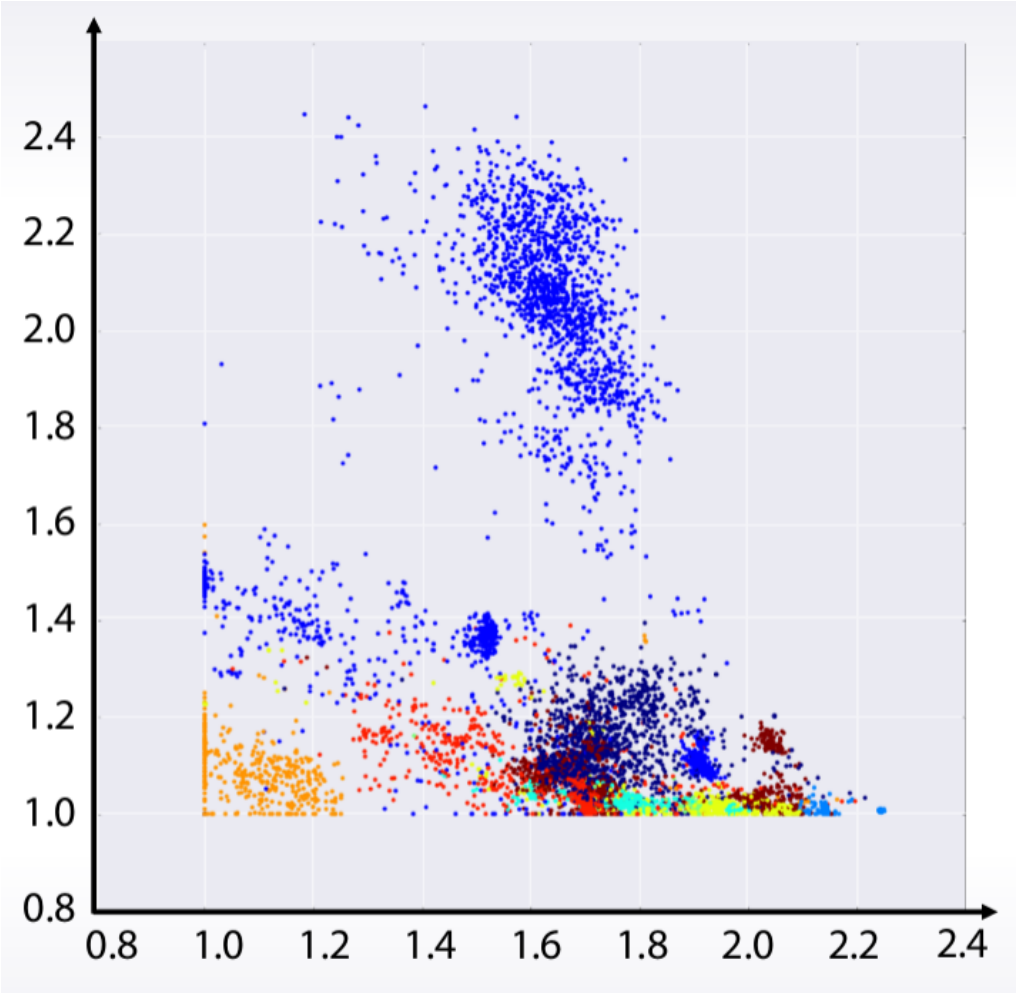
Other tools



- `x.value_counts()`
- `x.isnull()`

Tools for exploring feature relations

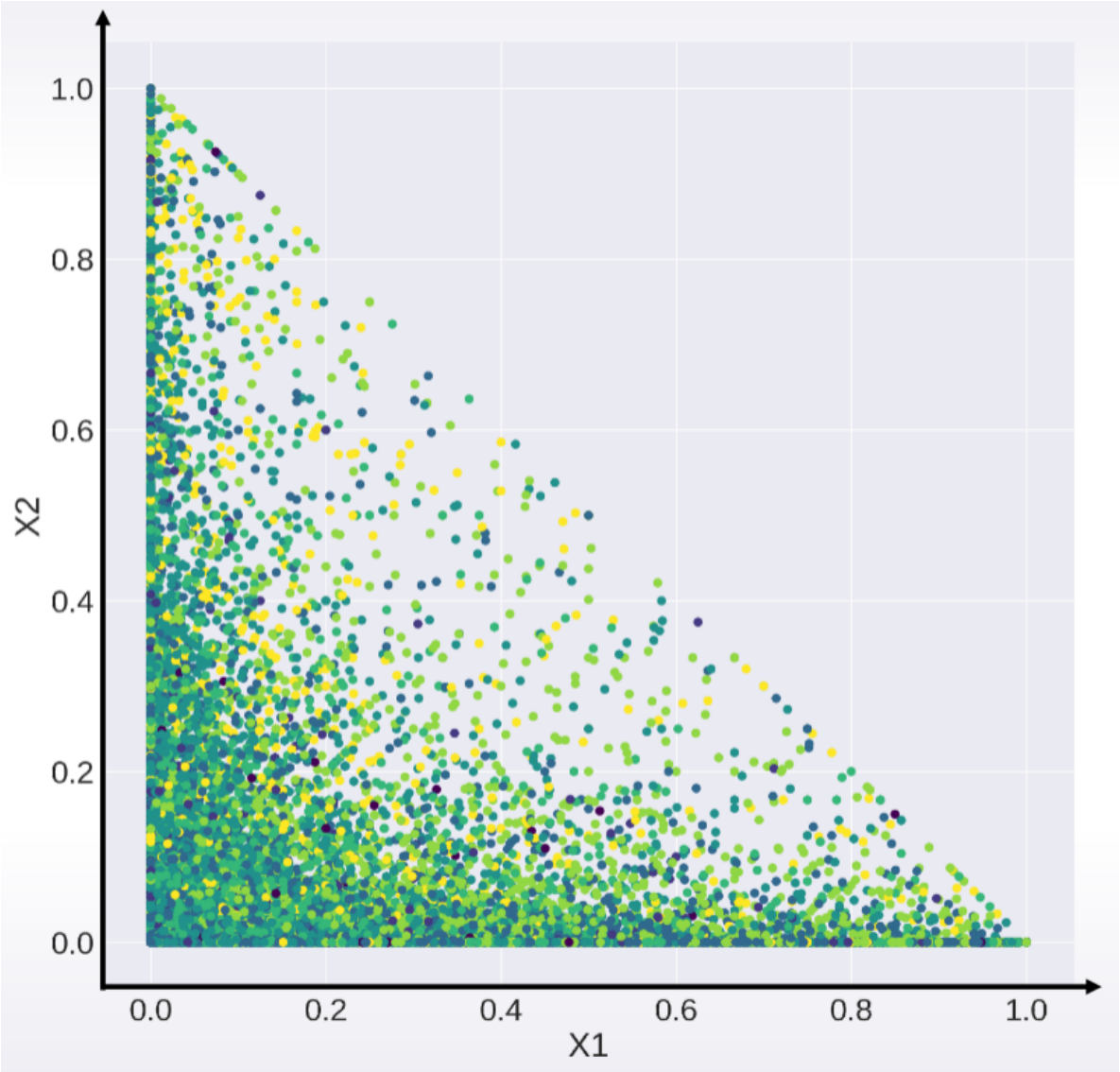
Exploring feature pairs (scatterplot)



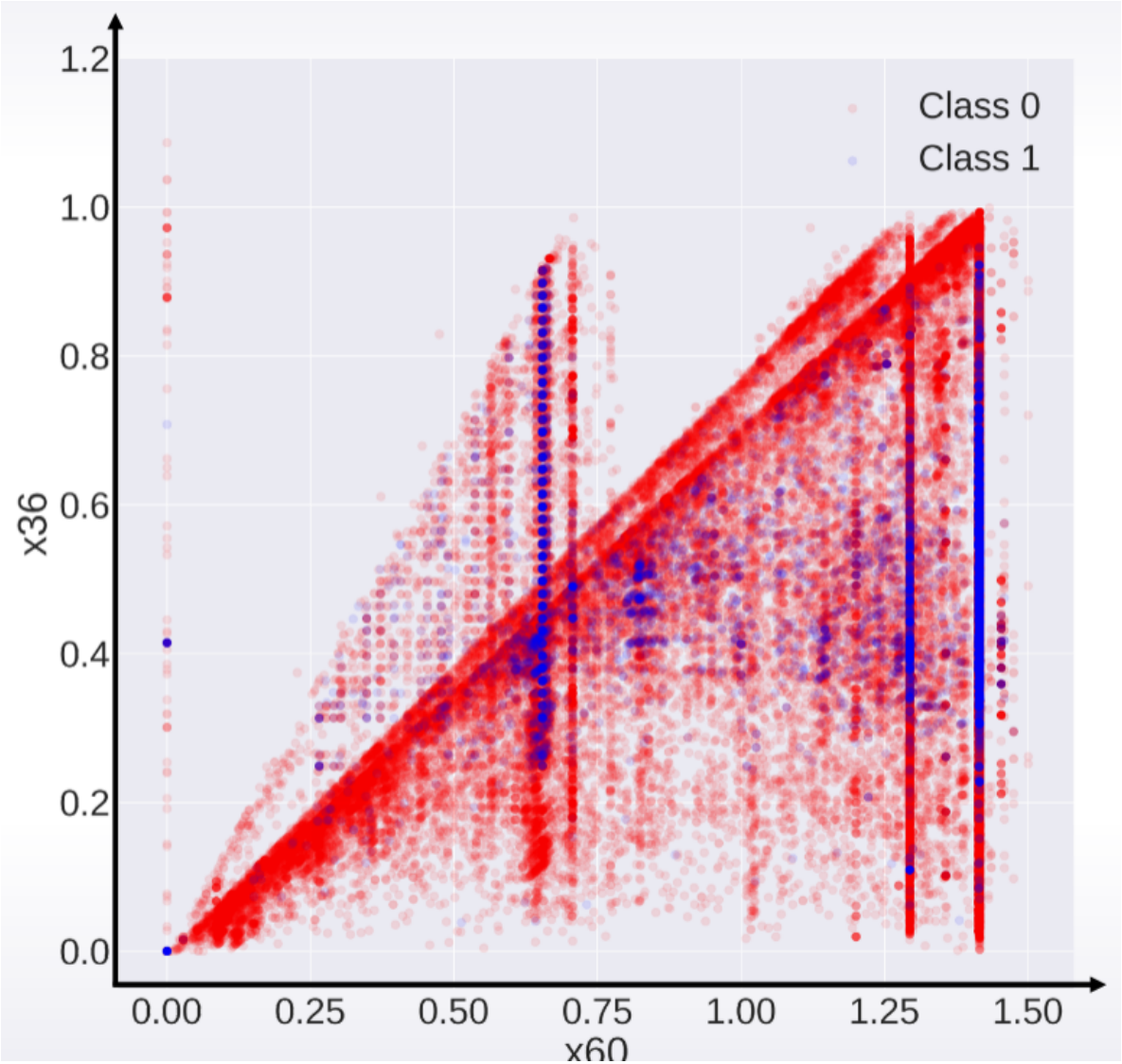

```
plt.scatter(x1, x2)
```

Exploring feature pairs (scatterplot)

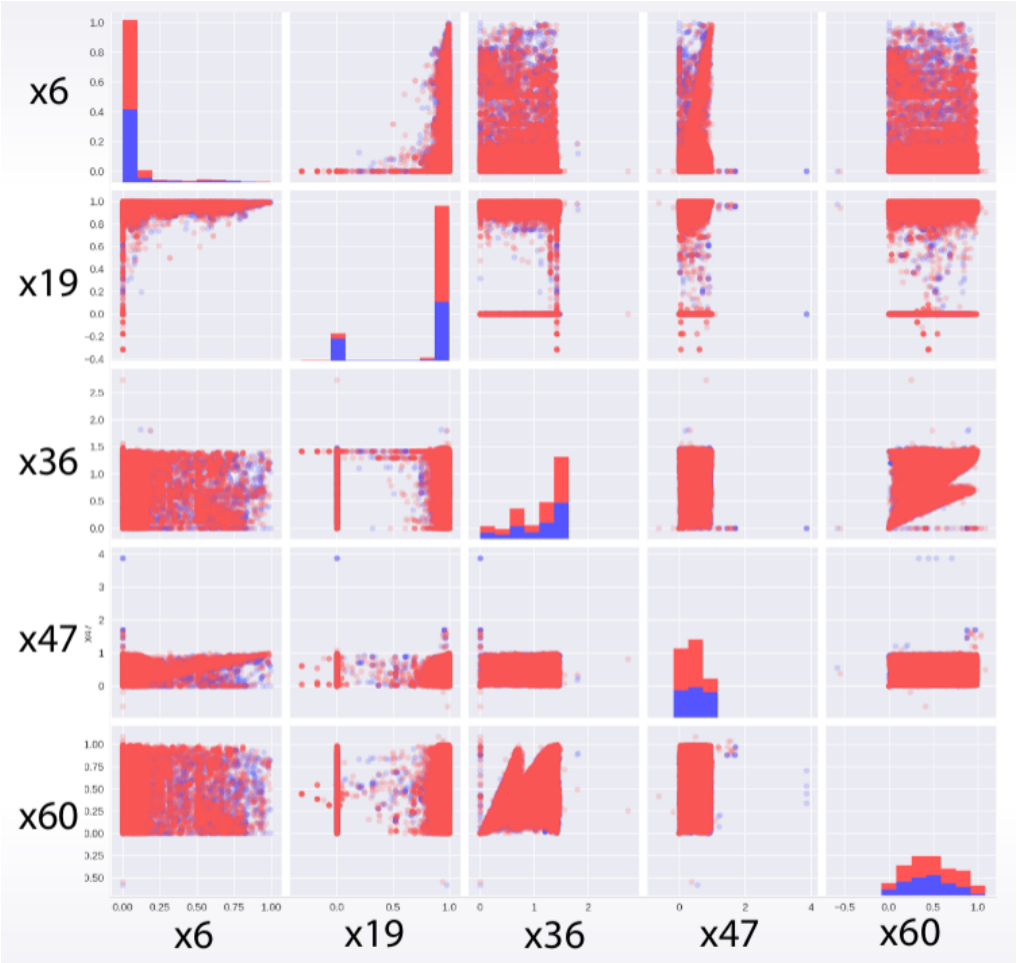
Exploring feature pairs (scatterplot)



Exploring feature pairs (scatterplot)

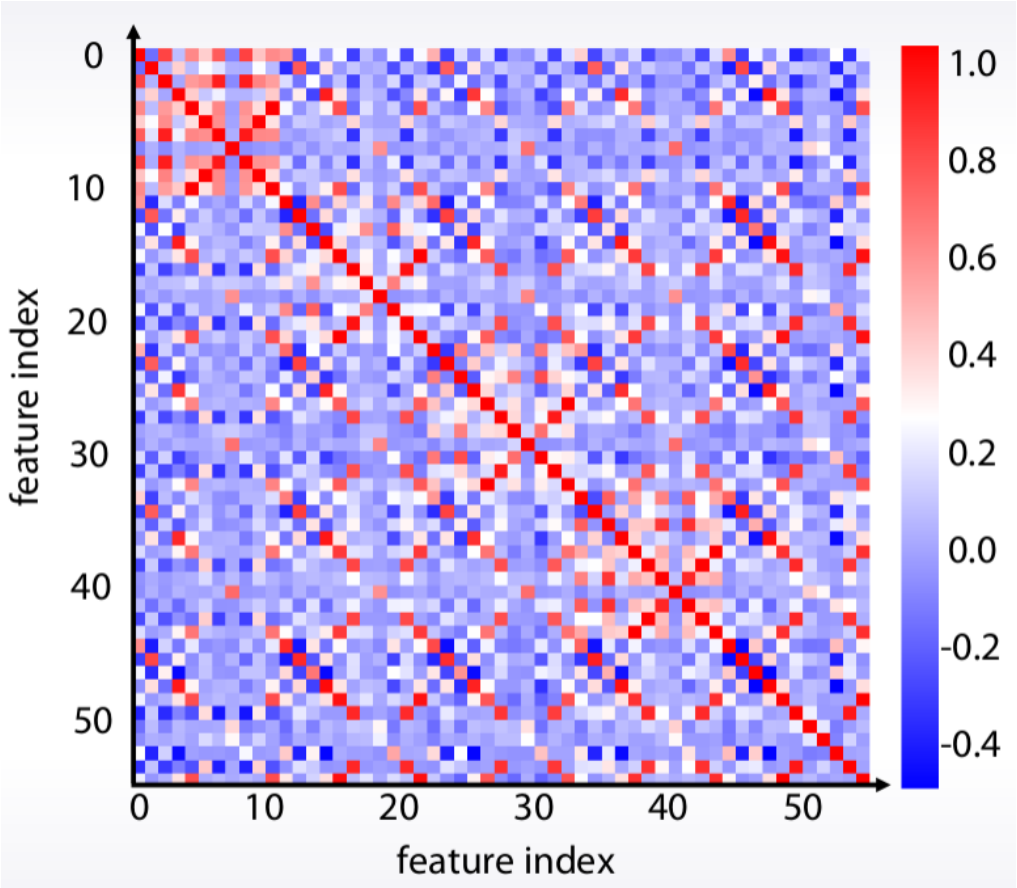


Exploring feature pairs (scatter matrix)



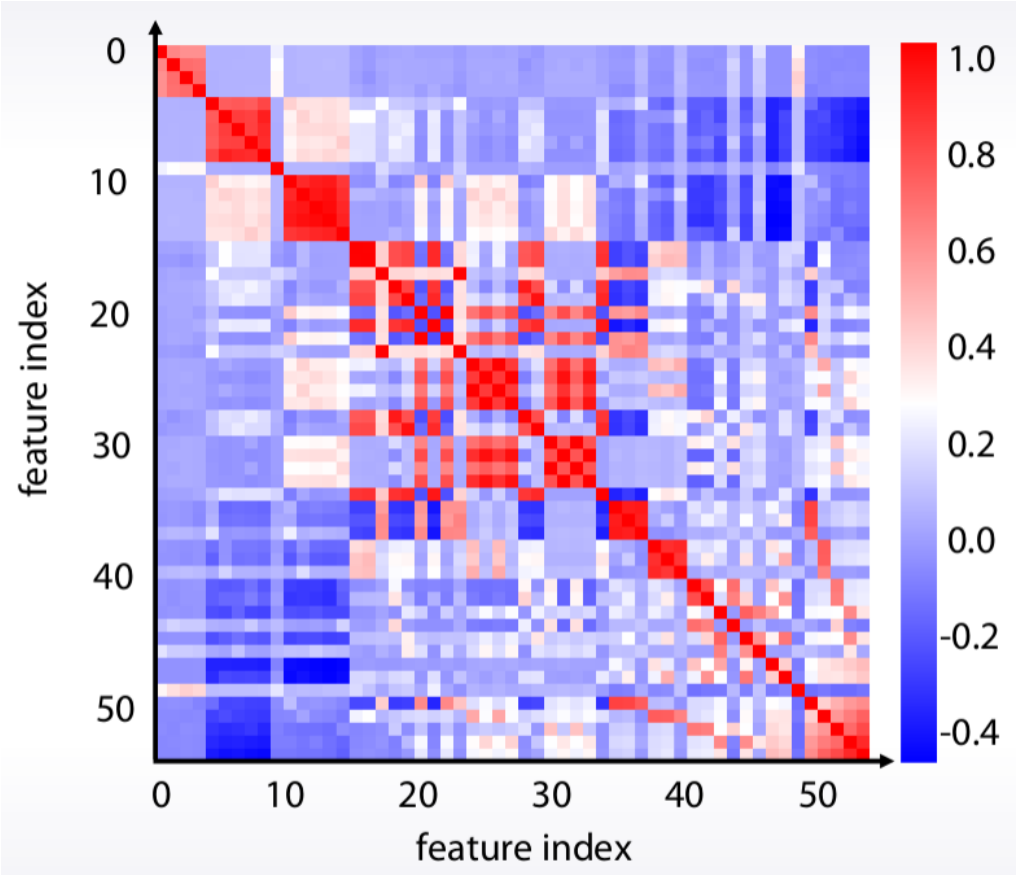

```
pd.scatter_matrix(df)
```

Exploring feature pairs (correlation matrix)



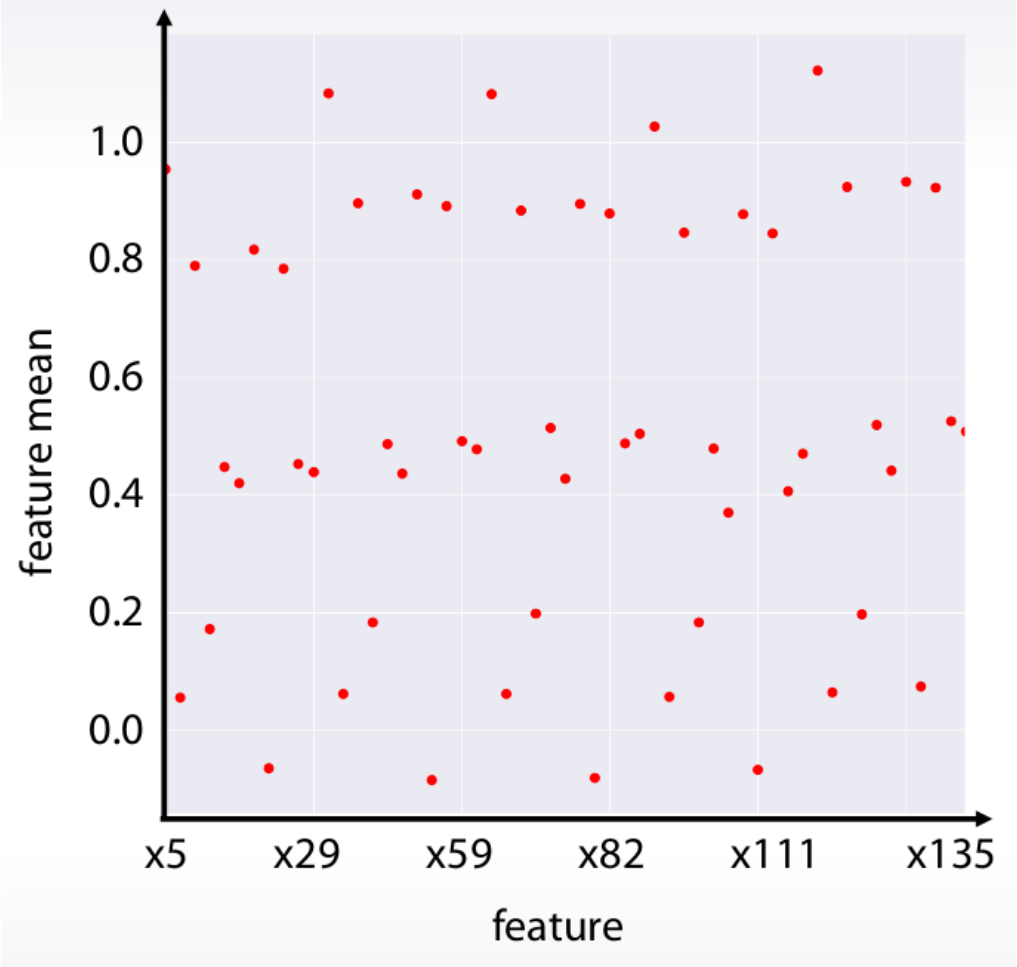

```
df.corr(), plt.matshow( ...  )
```

Exploring feature pairs (correlation matrix)



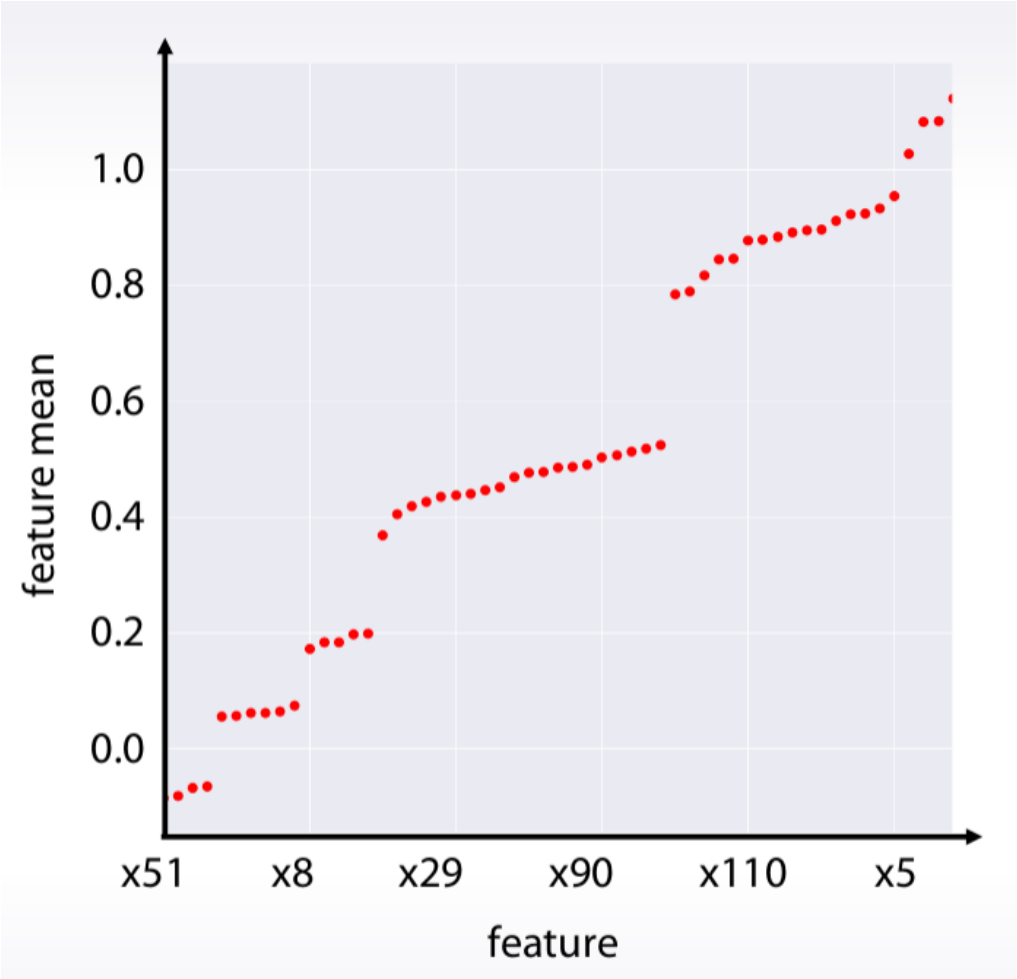

```
df.corr(), plt.matshow( ...  )
```

Exploring feature groups




```
df.mean().plot(style='.')
```

Exploring feature groups




```
df.mean().sort_values().plot(style='.')
```

Data cleaning

Constant duplicates

is_train	f0	f1	f2	f3	f4	f5
True	13	H	1.2	1.2	A	C
True	13	H	36.6	36.6	B	A
False	13	H	0	0	A	C
False	13	G	-14	-14	C	B

```
traintest.nunique(axis=1) == 1
```

Constant features

is_train	f0	f1	f2	f3	f4	f5
True	13	H	1.2	1.2	A	C
True	13	H	36.6	36.6	B	A
False	13	H	0	0	A	C
False	13	G	-14	-14	C	B

```
train.nunique(axis=1) == 1
```

Duplilcated features

is_train	f0	f1	f2	f3	f4	f5
True	13	H	1.2	1.2	A	C
True	13	H	36.6	36.6	B	A
False	13	H	0	0	A	C
False	13	G	-14	-14	C	B

```
traintest.T.drop_duplicates()
```

Duplilcated features

is_train	f0	f1	f2	f3	f4	f5
True	13	H	1.2	1.2	A	C
True	13	H	36.6	36.6	B	A
False	13	H	0	0	A	C
False	13	G	-14	-14	C	B

```
for f in categorical_feats:
    traintest[f] =raintest[f].factorize()
traintest.T.drop_duplicates()
```

Duplicated rows

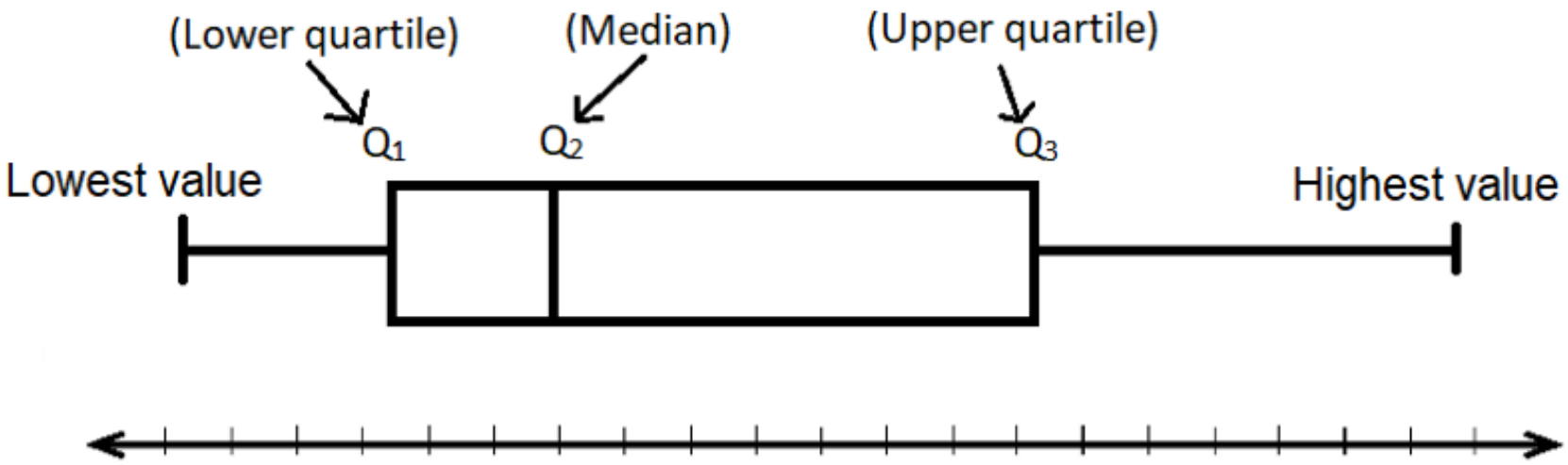
f1	f2	f3	y
13	34r9	A	0
13	34r9	A	1
13	34r9	A	1

- Check if same rows have same label
- Find duplicated rows, understand why they are duplicated

Show outliers with box plot

Box plot (or box-and-whisker plot)

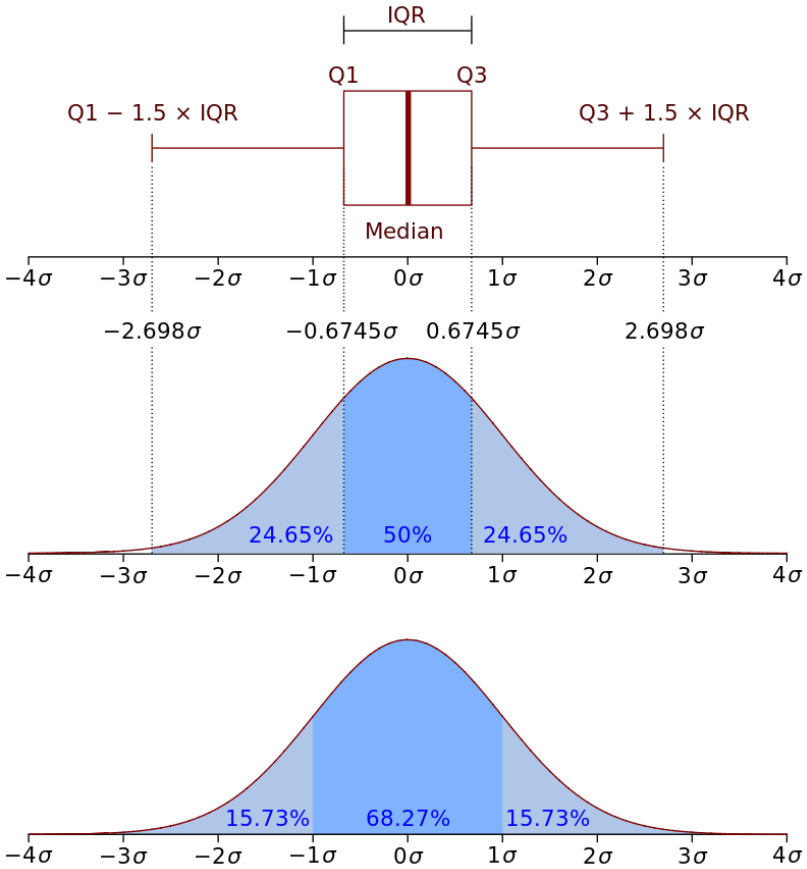
- middle line shows median
- box shows the majority
- whiskers shows the extremes
- beyond extremes are outliers



Example of box plot

Show outliers with statistics

- Z-score: $z = \frac{x-\mu}{\sigma}$
- Interquartile range (IQR)
  - also called the midspread or middle 50%, or technically H-spread



Example of Z-score and IQR

Missing values

- Missing values can happen when, e.g.,
  - User forgot to fill in a field.
  - Data was lost while transferring manually from a legacy database.
  - There was a programming error.
  - Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted
- Patterns of missing values
  - Missing values in many real-world data are not at random!
  - Do not assume uniform or another simple pattern for the missing positions.
  - The missing value often depends on the reason it's missing:
    - People with high salaries generally do not want to reveal their incomes in surveys
    - Some females don't want to reveal their ages

How can we impute the missing values?

ID	age	bmi	hyp	chl
1	1	NA	NA	NA
2	2	22.7	1	187
3	1	NA	1	187
4	3	NA	NA	NA
5	1	20.4	1	113
6	3	NA	NA	184
7	1	22.5	1	118
8	1	30.1	1	187
9	2	22	1	238
10	2	NA	NA	NA
11	1	NA	NA	NA
12	2	NA	NA	NA
13	3	21.7	1	206
14	2	28.7	2	204
15	1	29.6	1	NA
16	1	NA	NA	NA
17	3	27.2	2	284
18	2	26.3	2	199
19	1	35.3	1	218
20	3	25.5	2	NA
21	1	NA	NA	NA
22	1	33.2	1	229
23	1	27.5	1	131
24	3	24.9	1	NA
25	2	27.4	1	186

MICE: Multiple Imputation by Chained Equations

State-of-the-art imputation method for handling missing values

1. Preliminary imputation ("place holders")
2. Loop for several iterations:
   - Loop for each variable with missing value(s)
     - treat the current variable as y and the other variables as x
     - use rows where y is observed as training data
     - use rows where y is missing as test data
     - fit $p(y|x)$ and draw imputed $y \sim p(y|x)$
3. The above gives an imputation. Using different draws can give multiple imputations.

MICE illustration (original data)

`nhanes` data set: age=age group, bmi=body mass index, hyp=hypertensive, chl=total serum cholesterol

ID	age	bmi	hyp	chl
1	1	NA	NA	NA
2	2	22.7	1	187
3	1	NA	1	187
4	3	NA	NA	NA
5	1	20.4	1	113
6	3	NA	NA	184
7	1	22.5	1	118
8	1	30.1	1	187
9	2	22	1	238
10	2	NA	NA	NA
11	1	NA	NA	NA
12	2	NA	NA	NA
13	3	21.7	1	206
14	2	28.7	2	204
15	1	29.6	1	NA
16	1	NA	NA	NA
17	3	27.2	2	284
18	2	26.3	2	199
19	1	35.3	1	218
20	3	25.5	2	NA
21	1	NA	NA	NA
22	1	33.2	1	229
23	1	27.5	1	131
24	3	24.9	1	NA
25	2	27.4	1	186

MICE illustration (after preliminary imputation)

ID	age	bmi	hyp	chl
1	1	26.6	1	191.4
2	2	22.7	1	187
3	1	26.6	1	187
4	3	26.6	2	191.4
5	1	20.4	1	113
6	3	26.6	1	184
7	1	22.5	1	118
8	1	30.1	1	187
9	2	22	1	238
10	2	26.6	2	191.4
11	1	26.6	1	191.4
12	2	26.6	2	191.4
13	3	21.7	1	206
14	2	28.7	2	204
15	1	29.6	1	191.4
16	1	26.6	2	191.4
17	3	27.2	2	284
18	2	26.3	2	199
19	1	35.3	1	218
20	3	25.5	2	191.4
21	1	26.6	2	191.4
22	1	33.2	1	229
23	1	27.5	1	131
24	3	24.9	1	191.4
25	2	27.4	1	186

Loop for each column: e.g., to impute bmi,

- fit prediction model p(bmi|age, hyp, chl)
- using rows with black bmi's as training set and rows with red bmi's as test set (to predict)

MICE illustration (after one MICE iteration)

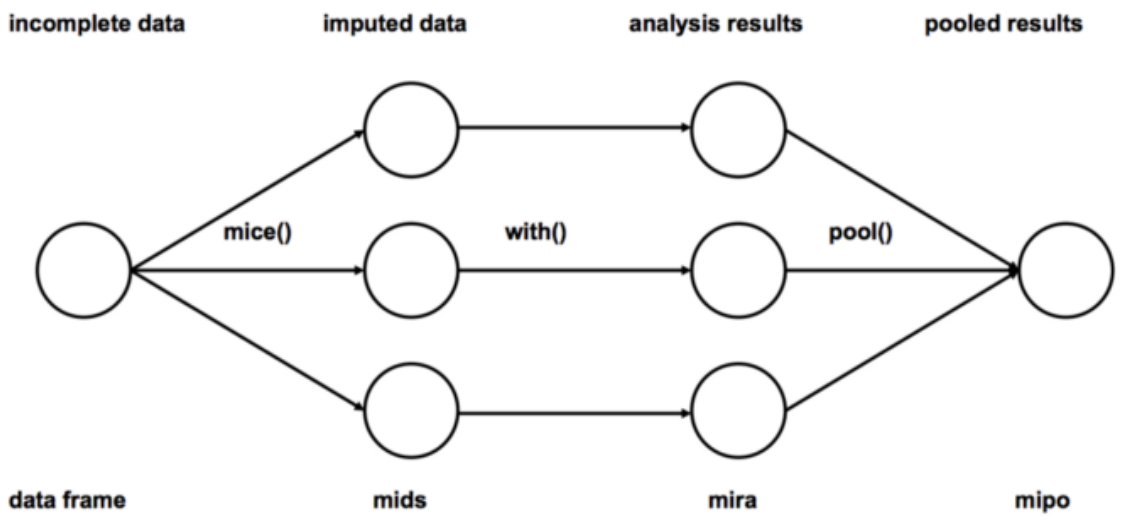
ID	age	bmi	hyp	chl
1	1	30.1	1	187
2	2	22.7	1	187
3	1	29.6	1	187
4	3	27.4	2	184
5	1	20.4	1	113
6	3	27.4	2	184
7	1	22.5	1	118
8	1	30.1	1	187
9	2	22	1	238
10	2	28.7	2	187
11	1	22.7	1	187
12	2	21.7	1	206
13	3	21.7	1	206
14	2	28.7	2	204
15	1	29.6	1	187
16	1	27.2	1	131
17	3	27.2	2	284
18	2	26.3	2	199
19	1	35.3	1	218
20	3	25.5	2	187
21	1	35.3	1	229
22	1	33.2	1	229
23	1	27.5	1	131
24	3	24.9	1	206
25	2	27.4	1	186

MICE illustration (after five MICE iterations)

ID	age	bmi	hyp	chl
1	1	30.1	2	131
2	2	22.7	1	187
3	1	28.7	1	187
4	3	22.5	1	184
5	1	20.4	1	113
6	3	20.4	1	184
7	1	22.5	1	118
8	1	30.1	1	187
9	2	22	1	238
10	2	33.2	2	284
11	1	26.3	1	187
12	2	28.7	1	218
13	3	21.7	1	206
14	2	28.7	2	204
15	1	29.6	1	187
16	1	27.2	1	187
17	3	27.2	2	284
18	2	26.3	2	199
19	1	35.3	1	218
20	3	25.5	2	218
21	1	35.3	2	184
22	1	33.2	1	229
23	1	27.5	1	131
24	3	24.9	1	186
25	2	27.4	1	186

Multiple Imputation

1. **Imputation**: generate multiple samples of missing values
2. **Analysis**: fit a set of base learners, each with an imputation
3. **Pooling**: ensemble the base learners



MICE in R

- So far there is no good python MICE package
- Could be integrated to `scikit-learn`
- A simple example

```
library(mice)
data_with_na <- nhanes
imp <- mice(data_with_na)
data <- complete(imp)
```

(Additional reading) Call MICE from Python

Two methods to use MICE in Python

- exchange with disk file
  1. Python writes data with missing values to input disk file and calls R script by system command line
  2. R script reads input disk file, call MICE, and writes imputed data to output disk file
  3. Python reads output disk file and gets imputed data
- `rpy2` package
  1. import `numpy2ri` to convert np.array for R
  2. use `importr` to load the R MICE library
  3. use np.array to convert the R MICE result

Note: Numpy doesn't have NA; use NaN instead

Additional reading

Examples of using the two ways to call MICE from Python

IterativeImputer in Scikit-Learn

- A plausible equivalent purely in Python
- Basic usage: https://scikit-learn.org/stable/modules/impute.html
- More examples:
  - https://scikit-learn.org/stable/auto_examples/impute/plot_iterative_imputer_variants_comparison.html
  - https://scikit-learn.org/stable/auto_examples/impute/plot_missing_values.html

Additional readings

- Credit Card Fraud Detection Case Study
- *Exploratory Data Analysis with Python Cookbook* by Ayodele Oluleye