Data Preprocessing

CSMODEL

Data Cleaning

Data Cleaning

Raw data is inherently dirty. Some issues may include:

- Spelling/encoding errors
- Incorrect data types
- Non-ideal formats
- Missing values

In data cleaning, these inconsistencies are addressed to prevent problems in analysis.

Data Cleaning

> In Data Science, 80% of time spent prepare data, 20% of time spent complain about need for prepare data.
>
> 10:47 AM · Feb 27, 2013 · Twitter Web Client

Data Cleaning

Separate Files

Multiple Representations

Incorrect Datatype

Default Values

Missing Data

Duplicate Data

Inconsistent Format

Data Cleaning

Separate Files

Data comes from multiple sources

- Collected at different times
- Completely different datasets

Use various **pandas** functions such as **concat()** and **merge()** to combine separate files

Data Cleaning

Multiple Representations

- Different representations of text may occur in the dataset.
- Use the **unique()** function to ensure that categorical variables are correctly represented.

Sex
M
Female
F
Male
m
mALe

Data Cleaning

Multiple Representations

- Modify the values by assigning the same value for each categorical level.
- The **map()** function of pandas might be useful for this problem

Sex
M
F
F
M
M
M

Data Cleaning

Incorrect Datatype

- Numerical Values in the dataset may be represented as text or string.
- Use the info() function to check the datatype of each column in the dataset.

Count
1
3
'4'
2
'5'
1

Data Cleaning

Incorrect Datatype

- Convert all variables to their appropriate numerical type.
- Use the **apply()** function to the appropriate column and use **lambda** function to convert each element to a numerical data type.

Count
1
3
4
2
5
1

Data Cleaning

Default Values

- Datasets may use a default value for when the data is missing or not applicable.
- Handle these cases appropriately depending on the context.
- In pandas, missing values are represented as **NaN** or **None**

Age
19
18
N/A
20
9999
20

Data Cleaning

Default Values

- Make sure that all missing values are represented as **None** or **NaN**
- For other kinds of default values, handle them depending on the context.

Age
19
18
NaN
20
9999
20

Data Cleaning

Missing Data

- Elements in the dataset may be missing because the data is unavailable.
- Missing values are represented as **NaN** or **None** in pandas.

Grade
90
86
85
91
NaN
72

Data Cleaning

Missing Data

Handling missing data depends on the domain. Use your own judgement.

- Option: Delete all rows that contain missing data.
- Option: Replace the missing data with the mean of the variable
- And other schemes…

Grade
90
86
85
91
72

Grade
90
86
85
91
85
72

Data Cleaning

Duplicate Data

A dataset may contain duplicate data.

- In some cases, there really is a duplicate.
- In others, it may be caused by an error.

Name	Age
Robb	14
Sansa	11
Bran	10
Sansa	11
Robb	14
Arya	9

Data Cleaning

Duplicate Data

- If there are duplicate rows, consider removing them from the dataset.
- It depends whether this is caused by an error or really part of the dataset.
- Use the drop_duplicates() function to delete duplicates in the dataset.

Name	Age
Robb	14
Sansa	11
Bran	10
Arya	9

Data Cleaning

DOH notes that seventy-nine (79) duplicates were removed from the total case count. The total cases reported may be subject to change as these numbers undergo constant cleaning and validation. | @gmanews

5:08 PM · Jul 11, 2020 · Twitter for Android

Data Cleaning

Inconsistent Format

Date and time may be recorded in inconsistent formats

- Human error
- Inconsistencies in data collection and encoding

Datetime
Jun 20 2018
6/20/2018
20 June 18
20/6/2018 18:00
Jun 20, 2018
20/06/2018

Data Cleaning

Inconsistent Format

Date and time may be recorded in inconsistent formats

- Human error
- Inconsistencies in data collection and encoding

Datetime
20/6/2018 00:00
20/6/2018 00:00
20/6/2018 00:00
20/6/2018 18:00
20/6/2018 00:00
20/6/2018 00:00

Data Preprocessing

Data Preprocessing

- Querying
- Imputation
- Binning
- Outlier Detection
- One Hot Encoding
- Log Transformation
- Aggregation
- Column Transformation
- Feature Scaling
- Feature Engineering

Data Preprocessing

Querying

- Use the **query()** function of pandas to write complex conditions for selecting appropriate info from the dataset.
- The **query()** function queries the columns of a DataFrame with a Boolean expression
- Alternative for using the bracket operator discussed during data representation

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.query.html

Data Preprocessing

Querying

Ex1. Select rows based on numerical value

#	Section	CCPROG1	CCPROG2
0	S17	3.0	1.0
1	S18	1.0	None
2	S17	3.5	4.0
3	S17	2.5	1.0
4	S18	4.0	4.0

```
df.query('CCPROG1 >= 3.5')
```

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.query.html

Data Preprocessing

Querying

Ex2. Select rows based on string value

#	Section	CCPROG1	CCPROG2
0	S17	3.0	1.0
1	S18	1.0	None
2	S17	3.5	4.0
3	S17	2.5	1.0
4	S18	4.0	4.0

```
df.query('Section == "S17"')
```

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.query.html

Data Preprocessing

Querying

Ex3. Select rows based on multiple conditions

#	Section	CCPROG1	CCPROG2
0	S17	3.0	1.0
1	S18	1.0	None
2	S17	3.5	4.0
3	S17	2.5	1.0
4	S18	4.0	4.0

```
df.query('CCPROG1 >= 3.0 and Section == "S17"')
```

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.query.html

Data Preprocessing

Querying

Ex4. Select rows based on index

#	Section	CCPROG1	CCPROG2
0	S17	3.0	1.0
1	S18	1.0	None
2	S17	3.5	4.0
3	S17	2.5	1.0
4	S18	4.0	4.0

```
df.query('index > 2')
```

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.query.html

Data Preprocessing

Querying

Ex5. Select rows by comparing multiple columns

#	Section	CCPROG1	CCPROG2
0	S17	3.0	1.0
1	S18	1.0	None
2	S17	3.5	4.0
3	S17	2.5	1.0
4	S18	4.0	4.0

```
df.query('CCPROG1 > CCPROG2')
```

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.query.html

Data Preprocessing

Looping

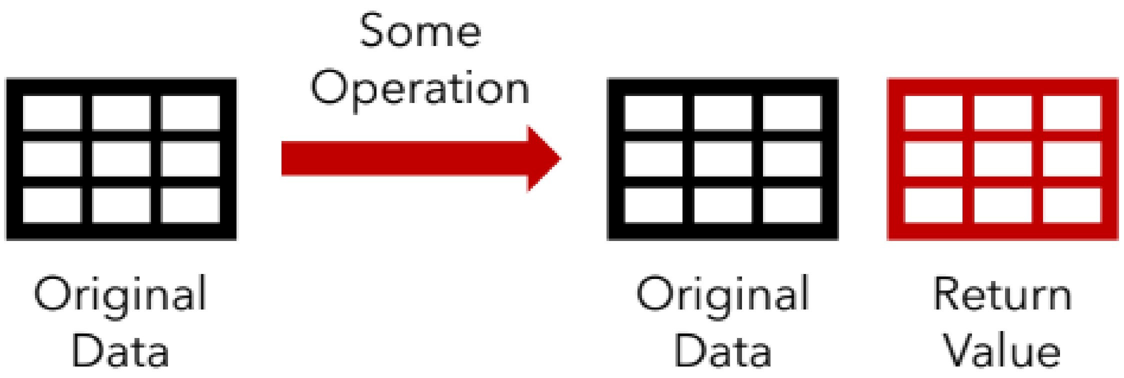
Use `iterrows()` to iterate over all rows and apply some operation over each row

```
for cur_idx, cur_row in df.iterrows():
      # perform some operations
```

In this example, cur_idx is the index of the current row and cur_row is the actual row.

Data Preprocessing

Inplace Parameter




```
Inplace = False
```

Data Preprocessing

Inplace Parameter


```
Inplace = True
```

Data Preprocessing

Dealing with missing values (again)

Datasets with missing values could be a result of:

- Error in the data collection process
- Privacy/refusal to give information
- Interruptions/failure in sensors

Rows with missing values might be dropped from the dataset. However, this might not be the best approach in all cases.

Data Preprocessing

Dropping entire rows/columns through a threshold

Use a threshold value to decide whether to drop or now.

Drop <u>columns</u> with missing value less than the threshold t:

```
df = df[df.columns[df.isnull().mean() < t]]
```

Drop <u>rows</u> with missing values less than the threshold t:

```
df = df.loc[df.isnull().mean(axis=1) < t]]
```

Data Preprocessing

Imputation

- Another approach to dealing with missing values is called **imputation**, where missing values are estimated based on existing values.
- There are two types of imputation: (a) numerical imputation and (b) categorical imputation

Data Preprocessing

Imputation

- In numerical imputation, use the **median** or **average** of the other values as an estimate of the missing values.
- For example, the age of one person is missing. Then, just use the median age of all other people in the dataset to estimate the age of that person – the missing age is likely the same as the most common age in the dataset.
- **Note:** Check the dataset whether this is a good idea or not.

Data Preprocessing

Imputation

- In categorical imputation, use the **mode** as the estimate for the missing values.
- For example, a person’s smoking information is missing. However, 95% of the people in the dataset are non-smokers. Assume that the missing value is “non-smoker.”
- **Note:** This might not be the best approach if there is no dominant value in the dataset, and the values are approximately uniformly distributed.

Data Preprocessing

Binning

- Binning refers to a technique where data is **grouped together** into a series of categories.
- There are two types of binning – binning numerical data and binning categorical data

Data Preprocessing

Binning

Name	Age
Robb	14
Sansa	11
Bran	10
Arya	9
Rickon	4

➡

Name	Age
Robb	13-15
Sansa	10-12
Bran	10-12
Arya	7-9
Rickon	4-6

Numerical data are grouped in different ranges of values.

Data Preprocessing

Binning

District
Malate
Cubao
Ermita
BGC
Quiapo

City
Manila
QC
Manila
Taguig
Manila

Categorical data are grouped by related values.

Data Preprocessing

Binning

- Binning is useful in cases wherein the dataset is too specific for the research questions.
- It could reveal relationships and insights in a clearer way that could not easily be seen if the original values were used.
- In machine learning, binning helps prevent overfitting, where models are tailor-fitted to specific examples instead of being able to recognize general cases.

Data Preprocessing

Outlier Detection

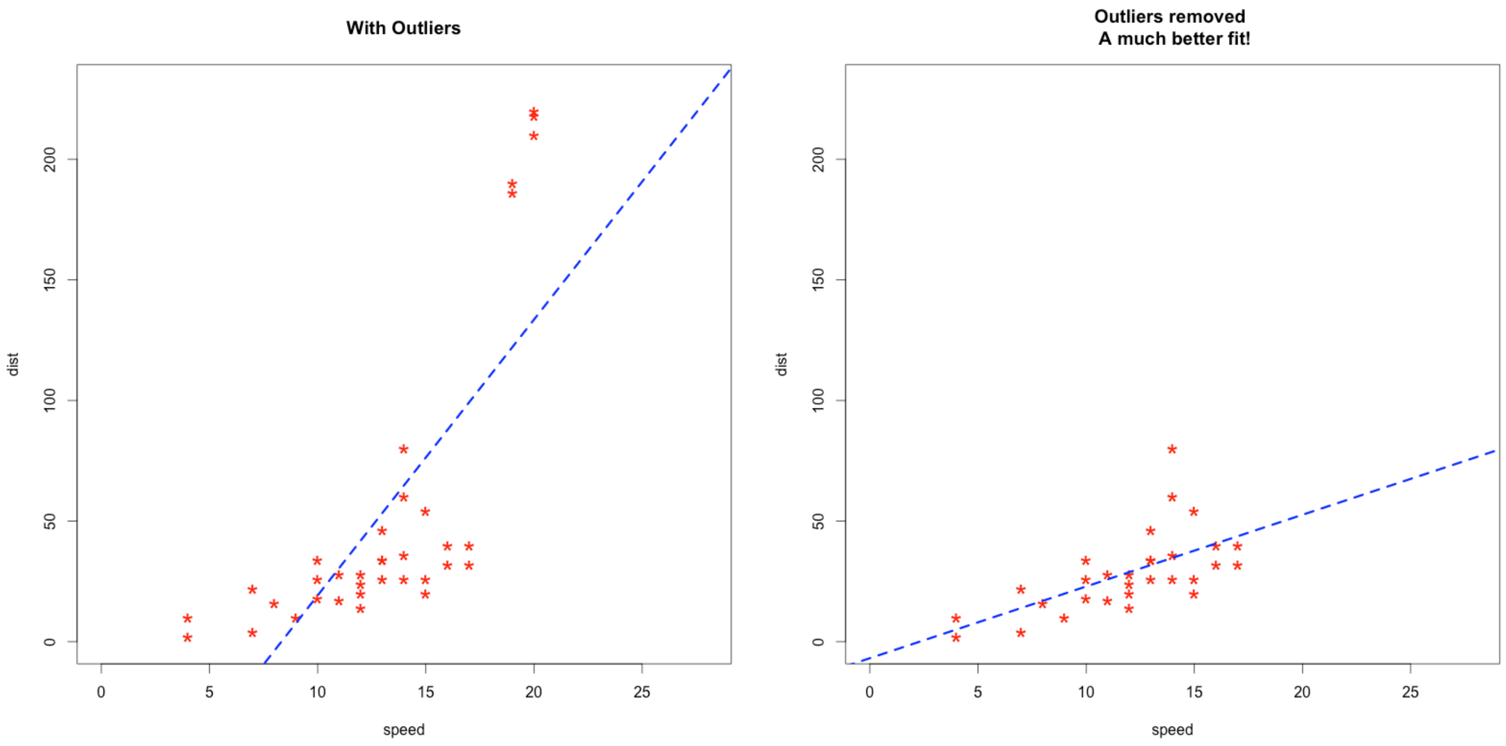
Outliers are extreme values in the dataset. These might be detected using different approaches:

- Visualization techniques
- Standard deviation/Z-score
- Percentiles

Data Preprocessing

Outlier Detection

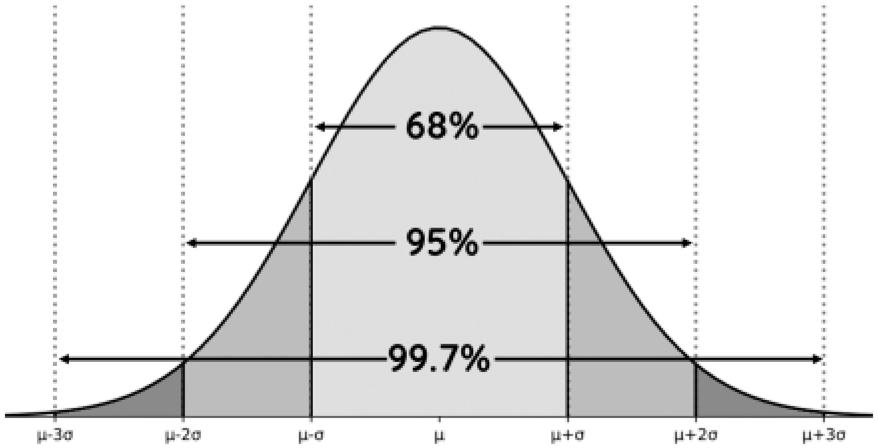
Detect outliers by plotting data points in a graph



https://cdn-0.r-statistics.co/screenshots/outliers_effect.png

Data Preprocessing

Outlier Detection



Get the middle x percent of the data. Anything outside the middle x percent is an outlier.

Data Preprocessing

Outlier Detection

- Drop – Remove the rows with outliers to exclude them from analysis since these might be extreme cases that do not happen normally.
- Use a cap – Instead of reducing data size, simply clamp the values within the desired range. This can help prevent adverse effects on algorithms and models that are sensitive to outliers but may affect the actual distribution of the data.

Data Preprocessing

Outlier Detection

Consideration:

- There are many causes of outliers, from errors in encoding to real rare occurrences in the dataset.
- By removing or disregarding them, some important insights might be removed from our data.
- Thus, be careful in handling outliers.

Data Preprocessing

One Hot Encoding

- Some data modelling techniques, including machine learning algorithms, require different data representation.
- In the Boolean representation, each property is represented as a column where the value is 1 if it is true and 0 otherwise.
- For example, in content-based recommender systems, the item profile was represented as a Boolean vector.

Data Preprocessing

One Hot Encoding

Sometimes, data must be represented in this format.

Item	Color
1	Red
2	Blue
3	Red
4	Green
5	Blue

➡

Red	Green	Blue
1	0	0
0	0	1
1	0	0
0	1	0
0	0	1

Data Preprocessing

One Hot Encoding

```
boolean_cols = pd.get_dummies(df['Color'])
```

Item	Color
1	Red
2	Blue
3	Red
4	Green
5	Blue

➡

Red	Green	Blue
1	0	0
0	0	1
1	0	0
0	1	0
0	0	1

Data Preprocessing

One Hot Encoding

```
df = df.join(boolean_cols).drop('Color', axis=1)
```

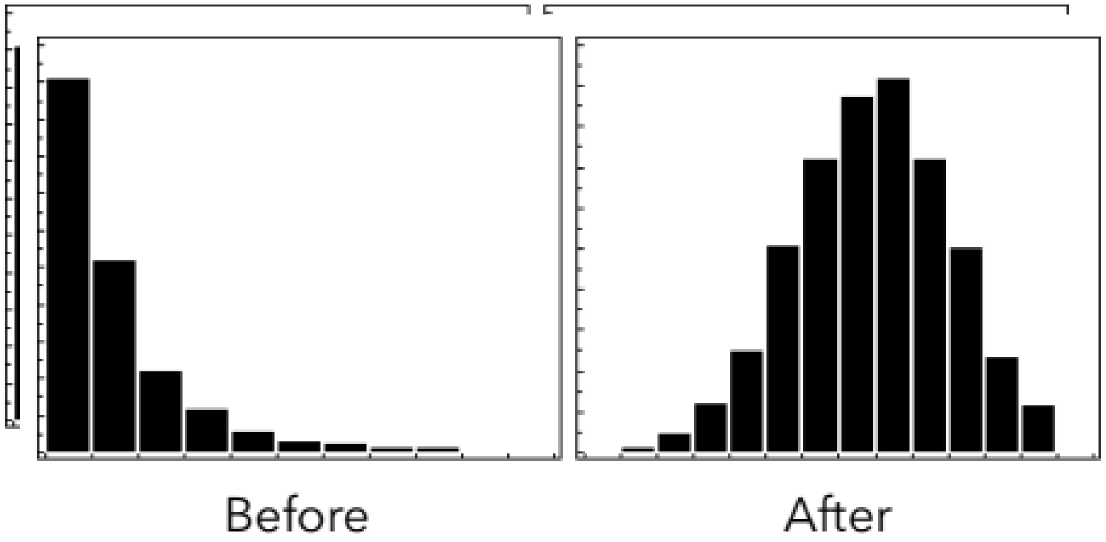
Item	Red	Green	Blue
1	1	0	0
2	0	0	1
3	1	0	0
4	0	1	0
5	0	0	1

df

Data Preprocessing

Log Transformation

Log transformation may be applied on non-normal data to make them more normally distributed.

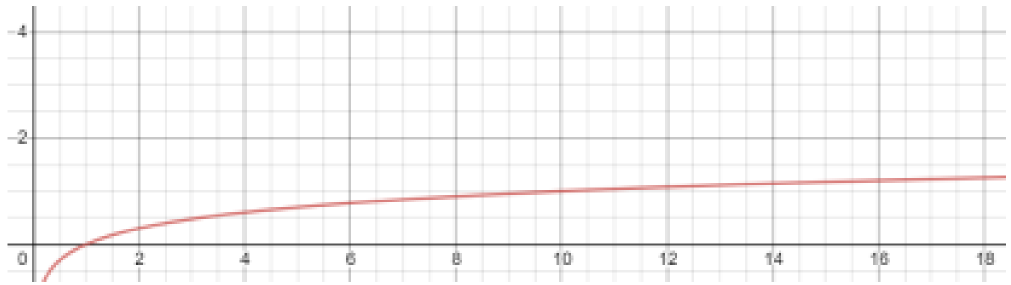


Data Preprocessing

Log Transformation

- The log operation log(n) evaluates to x, where 10x = n
- Note that relationship between n and log(n).

n	log(n)
1	0
10	1
100	2
1000	3



Data Preprocessing

Log Transformation

- By applying the log function to every data point, the values are transformed by reducing the effect of extremely high values while preserving the order.
- **Note:** The order of the data points are preserved in log transformation, but the scale of the distances between each data point might be distorted.

Data Preprocessing

Aggregation

- Apply aggregation to summarize data.
- Aggregation is generally performed when we have several rows that belong to the same group or same instance.
- There are two types of aggregation – aggregating numerical data and aggregating categorical data

Data Preprocessing

Aggregation

Section	Score
S17	5
S17	1
S18	2
S18	5
S17	4

➡

Section	Mean
S17	3.3
S18	3.5

Sum or mean might be extracted from numerical data.

Data Preprocessing

Aggregation

Group	City
A	Manila
A	Manila
B	Manila
B	Makati
B	Makati

Group	City
A	Manila
B	Makati

Mode within a group can be used for categorical data.

Data Preprocessing

Column Transformation

Name
Ted Mosby
Pedro P. Perez
Maria Gomez
Robb Stark
Tom Nook

➡

First	Last
Ted	Mosby
Pedro	Perez
Maria	Gomez
Robb	Stark
Tom	Nook

Transform columns to extract better information from the dataset.

Data Preprocessing

Column Transformation

- Transform dates into better representations since computers cannot understand order given dates such as “2019-06-04”.
- Extract the month, day, and year, and put them in separate columns as numerical values.
- Use a single numerical representation for each date (e.g., the number of seconds since a constant date).

Data Preprocessing

Feature Scaling

- Different variables may have different ranges.
- Some algorithms will be affected if one variable “overpowers” others. (e.g., scatterplot)
- Example, Euclidean distance will be affected if the scale of the variables are not equal.

Height	Income
163	29,000
164.5	35,000
162	60,000
162.6	19,000
180.2	15,000

Data Preprocessing

Feature Scaling

Normalization scales all features into a range of 0 to 1.

$$\frac{X - X_{min}}{X_{max} - X_{min}}$$

Note: This is susceptible to outliers.

Height	Income
0.05	0.31
0.14	0.44
0	1
0.03	0.09
1	0

Data Preprocessing

Feature Scaling

Standardization (z-score) measures the distance of each point from the mean.

$$\frac{x - \mu}{\sigma}$$

Note: Less susceptible to outliers, since it considers the standard deviation.

Height	Income
-0.45	-0.15
-0.25	0.19
-0.58	1.60
-0.50	-0.71
1.78	-0.94

Data Preprocessing

Feature Engineering

- Feature engineering refers to the use of domain knowledge to extract new features from data.
- This can help generate better insights and improve performance of machine learning algorithms.

Data Preprocessing

Feature Engineering

Examples

- From the height and weight, extract the body mass index
- From the date, determine whether it is a holiday or not.
- From the location, determine the nearest hospital/school/etc.
- From the geo-coordinates (latitude), estimate the climate.

Further Reading

- StackOverflow: Why use query instead of bracket operator?
  - https://stackoverflow.com/questions/67341369/pandas-why-query-instead-of-bracket-operator

Data Preprocessing

CSMODEL