Digital Technologies and Value Creation

Dr. Philippe Blaettchen
Bayes Business School (formerly Cass)

www.bayes.city.ac.uk

Overview – subject to change

Overarching theme	Week	
Introduction	1	Introduction to analytics applications and coding basics
Gathering data	2	Scraping web data
Gathering data / descriptive analytics	3	~~More on scraping,~~ data pre-processing and descriptive analytics
Gathering data / descriptive analytics	4	Descriptives in marketing analytics, and using social media APIs
Descriptive analytics	5	Descriptives in people analytics
NO LECTURE	6	NO LECTURE
Predictive analytics	7	Retaining employees and customers
Predictive analytics	8	Valuing a (social media) customer base
Predictive analytics	9	Segmenting customers and positioning products
Prescriptive analytics	10	Optimizing products and organizations
Prescriptive analytics	11	A/B-testing in practice

BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Learning objectives of today

Goals: How to pre-process and feature-engineer data:

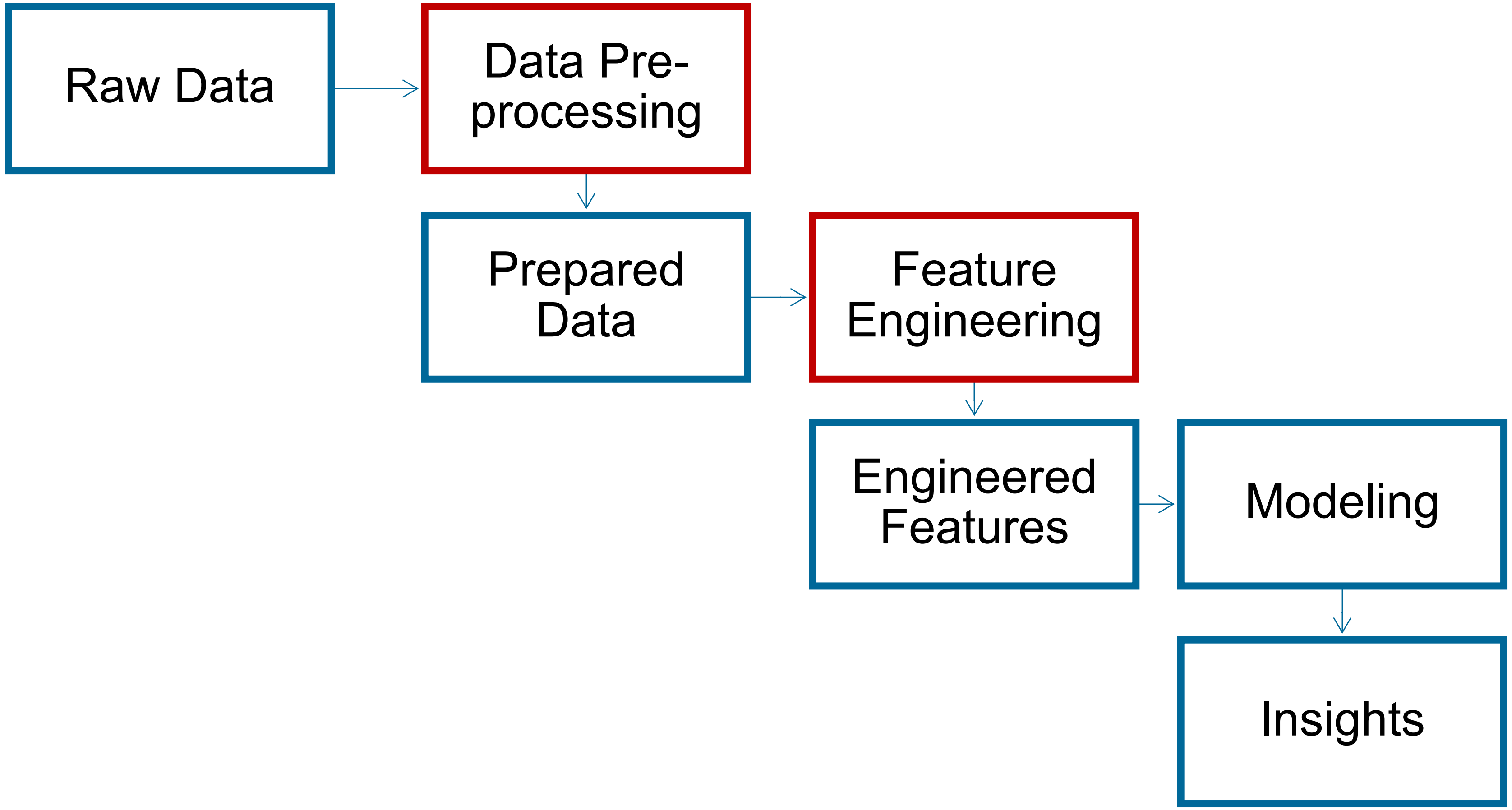
- Understand the key issues that can be faced when considering raw data
- Learn how to identify and deal with them

How will we do this?

- When pre-processing data, appeal to your intuition: if you had to identify these issues how would you go about it? How would you deal with them?
- We consider a simple use case in people analytics, including a comprehensive dataset

We now have some data, so what?

The analytics pipeline



Raw data

What could be some issues encountered in raw data?

Go to pollev.com/dtvc2021

BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Raw data

“Data scientists spend 80% of their time cleaning a dataset and only 20% of their time analyzing it.”

- Many, many techniques for data preprocessing and feature engineering
- For some of them, requires technical or Python knowledge above the scope of this course (e.g., how to correct typing errors in data)
- We focus here on the main issues, give some work-arounds and how to use Python to deal with these issues
- Bear in mind that this list is far from exhaustive

Our use-case

- It has about **18,000** employees

- The attrition rate of employees is quite high (**~14%** compared to industry average of **8-9%**).
  → How to fix it?
  - They have gathered data for calendar year 2020

Data cleansing, scaling, normalization, imputation

Data cleansing

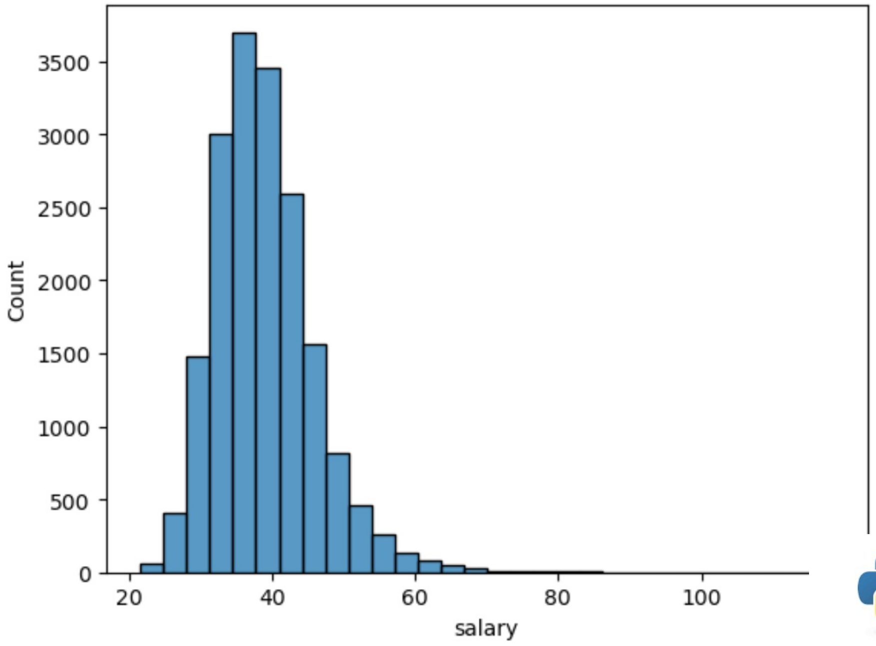
- A dataset may have any of the following **issues**:
- Values that are not in the right **format**
- **Invalid values** (i.e., negative values for a variable that is only positive)
- **Duplicate observations**
- Row records that are basically **empty**
- Columns that contains the **same value for every row**

	admin_support	age	boss_survey	boss_tenure	city_size	clock_in	core	education	gender	half_day_leaves	...
0	0.0	41.0	0.456427	3.0	4.3	0.0	0.0	3.0	0.0	5.0	...
1	2.0	41.0	0.512982	3.0	9.4	1.0	1.0	1.0	1.0	5.0	...
2	0.0	33.0	0.415119	3.0	4.3	1.0	0.0	1.0	0.0	2.0	...
3	0.0	35.0	0.467731	4.0	2.2	1.0	1.0	1.0	0.0	3.0	...
4	0.0	28.0	0.685366	3.0	2.2	0.0	1.0	1.0	0.0	3.0	...

Data cleansing – how to detect corrupt or invalid values in a dataset?

- For **numerical values**: use a **histogram** and analyze the output. Do the values seem reasonable?
- For **categorical values:** use the **function .unique()** to get all the possible values taken on by your variable. Do they seem reasonable?

```python
sns.histplot(df["salary"],kde=False,bins=30)
plt.show()
```




```python
df["education"].unique()
```

```
array([ 3.,  1.,  2., nan])
```

If they are not:
drop the corresponding row or
change the problematic value.

Data cleansing – how to detect duplicate rows?

Python function ready-made

```
dups = df.duplicated() #checks each row of the dataset and returns TRUE or FALSE dependin
print(dups.any()) #returns TRUE if there is any value in dups that is equal to TRUE
print(df[dups]) #returns the problematic row
```

Don't need to run this every time: can simply delete duplicates in an automated way using Python:

```
print(df.shape) #gives current size of dataset
df.drop_duplicates(inplace=True) # delete duplicate rows
print(df.shape)
```

```
(18149, 26)
(18132, 26)
```

Data cleansing – how to detect and delete rows that have empty cells?

```
df.isna().any()
```

```
admin_support           True
age                     True
boss_survey             True
boss_tenure             True
city_size               True
clock_in                True
```

If there are any rows/columns with empty cells:

df_no_na=df.dropna(axis,how,thresh)

- axis = 0 or 1 (depending on row (axis=0) or column (axis=1))
- how = “any” if one NA value is enough; =“all” if all values must be NA for the column to be dropped
- thresh = an integer (number of NA values needed)

python™

BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Data cleansing – how to detect and delete columns that contain the same value for all rows?

```
df.nunique()
```

```
admin_support            3
age                     36
boss_survey          18074
boss_tenure             15
city_size                5
clock_in                 2
core                     2
education                3
gender                   2
half_day_leaves         10
high_potential           2
job_satisfaction     18085
kpi_performance      18048
local                    1
```

If there is a 1 in the list:

df.drop([“Column_Name”])

Scaling and normalizing

- What is **scaling/normalizing** data?

  → Makes sure that your data is on scales that are comparable.

- **Normalizing**: operation that ensures that your data is **between 0 and 1**
- **Scaling:** operation that ensures that the **mean** of your data is **0** and the **std dev** is **1**

```
from sklearn import preprocessing
```

```
X = np.array([[ 1., -1.,  2.],
              [ 2.,  0.,  0.],
              [ 0.,  1., -1.]])
```

```
min_max_scaler = preprocessing.MinMaxScaler()
X_minmax = min_max_scaler.fit_transform(X)
X_minmax
```

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

Normalizing

```
X_scaled = preprocessing.scale(X)
X_scaled
```

```
array([[ 0.        , -1.22474487,  1.33630621],
       [ 1.22474487,  0.        , -0.26726124],
       [-1.22474487,  1.22474487, -1.06904497]])
```

Scaling

Scaling and normalizing – when to use it?

- **Useful for certain algorithms only**:
  - Useful for regressions, PCA. Not for tree algorithms.
  - If your algorithm takes features and multiplies them by numbers etc., then chances are scaling/normalizing could improve it.

- Some use cases:
  - Some columns are **orders of magnitude different** (e.g., column A has values around 1 and column B has values around 10,000,000,000)
  - Your algorithm is returning **warning message** of the type “poor condition number”
  - The **output** you get from your algorithm is **incomprehensible** (e.g., NAs)

BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Data imputation

Consider a dataset:

	Feature 1	Feature 2	Feature 3
Observation 1	221	Small	Blue
Observation 2	157	Large	Green
Observation 3		Medium	Red
Observation 4	50	Extra-Small	Green
Observation 5	122	Large	Red

Activity (in groups): what could be different ways of dealing with the empty cell? What are the pros and cons of your methods?

How would this change if you consider a missing observation in the Feature 2 column?

Data imputation for numerical variables

Numerical variables

Easy

Delete the row(s)/column(s) where values are missing

Replace the value with the **mean/the largest value/the smallest value**

Find the observation that is **“closest”** to it in other observations and use the value there

Find a **couple of observations** that are “close” to it and **randomly pick one of them**

Run a **regression** on rows where all the data is present and infer from it the missing values

Run a regression on rows where all the data is present and infer from it the missing values then **add noise** to the missing values

Hard

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Data imputation for categorical variables

Categorical variables

Easy

Delete the row(s)/column(s) where values are missing

Create a new category: missing values

Replace the value with the **value that appears most/least**

Find the observation that is **"closest"** to it in other observations and use the value there

Find observations that are **close** to it in other observations and randomly pick one

Run a **prediction algorithm** on rows that outputs a categorical variable

Hard

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Different types of missing data

	Sales	Size	Color
1	221	NA	Blue
2	157	Large	NA
3	NA	Medium	Red
4	50	NA	Green
5	122	Large	Red

Missing completely at random
(no pattern to the missing entries)

	Age	Mammography results
1	23	NA
2	55	Negative
3	34	Positive
4	18	NA
5	62	Positive

Missing at random
(absent entries depend on another feature)

python™

	Weight (kgs)	Age	Diabetes
1	80	77	1
2	90	40	0
3	NA	62	1
4	50	18	0
5	NA	54	1

Missing not at random
(entries are absent due to their value or a feature not accounted for)

- **Missing completely at random/missing at random**: easier to deal with, good results for, e.g., regression
- **Missing not at random**: much harder, can be mitigated by adding features

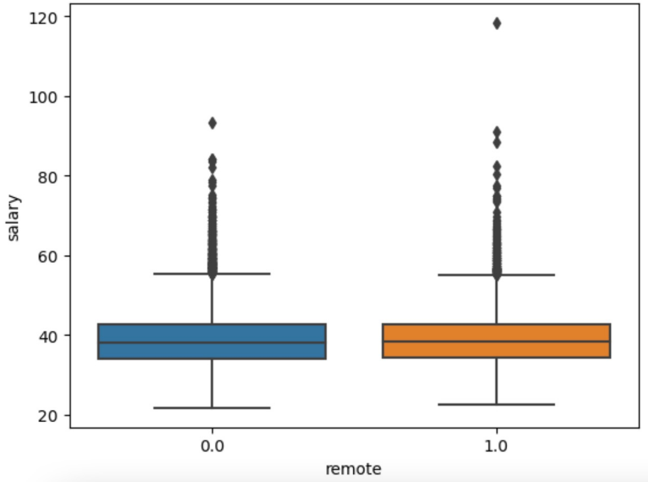
(See sklearn.impute library in Python)

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Outliers

- An outlier is an observation that doesn't "fit" into your dataset.
- This can be due to corrupt data, typos, or real outliers (e.g., Michael Jordan)

- Three main ways of checking for outliers:
  - Boxplots
  - Z-score charts (numbers above 3 or below -3)
  - Anomaly detection (unsupervised learning technique)




```
np.where((Z>3) | (Z<-3))
```

```
(array([ 1093,  2461,  2641,  2708,  3053,  4124,  4514,  5048,  5055,
         5275,  5429,  5757,  6448,  6594,  6728,  6811,  6874,  6966,
         7106,  7132,  7166,  7603,  7844,  8070,  8164,  8243,  9554,
         9806,  9859,  9918, 10728, 11498, 11630, 11816, 12361, 12579,
        12665, 12667, 12744, 13067, 13231, 13329, 13559, 13665, 13787,
        14680, 14709, 16147, 16175, 17025, 17523, 17600, 17731]),)
```

- Serves to flag possible outliers: area-specific knowledge to discard/keep

python™

BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Feature engineering

Numerical ←→ Categorical

- Some algorithms only accept one kind of input (generally numerical, e.g., regression).
- Useful to know how to go from one “type” of data to another

Categorical → Numerical

- **Activity (in groups):** can you see a difference between these two sets of categorical entries? How would you propose to make them numbers?

Small
Large
Medium

Blue
Green
Red

Numerical ← Categorical

Small	1
Large	3
Medium	2

Ordinal variables (these entries can be ranked)

python™

Blue
Green
Red

Nominal variables (these entries cannot be ranked)

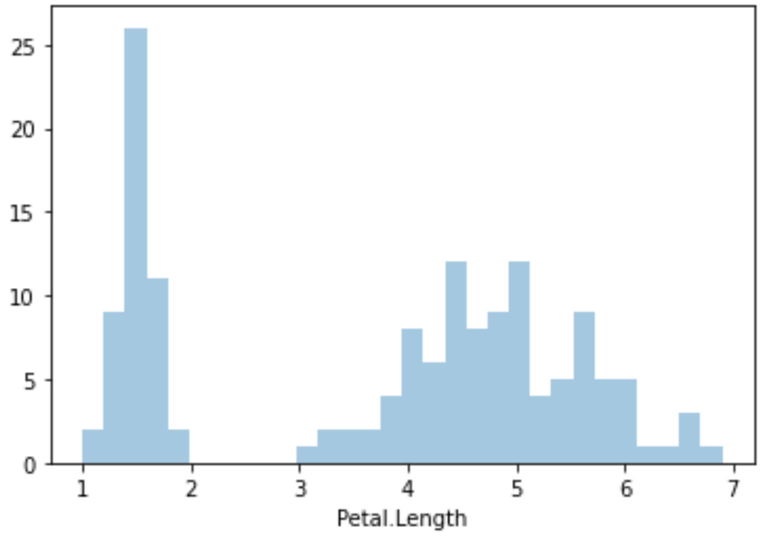
	Blue	Green	Red
Observation 1	1	0	0
Observation 2	0	1	0
Observation 3	0	0	1

One-Hot Encoding

	Blue	Green
Observation 1	1	0
Observation 2	0	1
Observation 3	0	0

Drop redundant column

Numerical → Categorical



- **Idea:** use bucketization or data binning. Take average for each bucket. Number of buckets = number of categories.

- Can also serve to aggregate observations.

Feature selection / dimension reduction

- **Feature selection** involves picking the **“right” features** for the model out of all possible features: will see examples in later lectures.

- **Dimension reduction** involves “**merging**” features together to get as **few features** as possible to explain the variability in the data: covered in predictive analytics.

Transforms and interactions

- Depending on the set-up it may be useful to **transform a feature**:
  - take powers of it
  - subtract/add a constant to it
  - divide/multiply by a constant
  - take an exponential of it or a log

- **Feature interactions** involve adding/multiplying/dividing etc. two features together to obtain a new feature

python™

BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

And off we go: descriptive analytics

Roadmap

- Most of the foundational work on descriptive analytics (statistics, hypothesis testing, etc.) is handled in your methods classes
- The video material of this week recaps on this, and introduces how to use Python for descriptive analytics

- We will soon discuss descriptive analytics in the specific context of marketing – and go back to gathering data
- We will also discuss descriptive analytics in the specific context of organizational design
- In both cases, we take a look at social media data and how it can be used fruitfully

Discussion: coding and scraping

Until next week!