Introduction to Data Science

with Python

Alexis Bogroff

May 22, 2022

Presenter

Alexis Bogroff
Lecturer and Mentor in Data Science
at Paris 1 Panthéon-Sorbonne, ESILV,
Openclassrooms, EM-Lyon

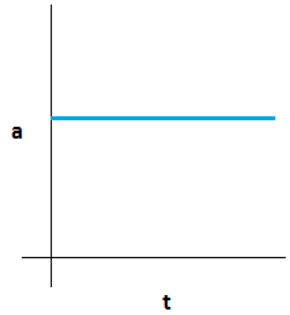
Presenter

Alexis Bogroff
Lecturer and Mentor in Data Science
at Paris 1 Panthéon-Sorbonne, ESILV,
Openclassrooms, EM-Lyon

- 4 years Teaching Assistant and lecturer in VBA, Python for finance, SQL, Data Analysis and Data Science
- 9 months Researcher Assistant at Paris 1 Panthéon-Sorbonne within H2020 European Project
- 1 year Data Scientist at Pléiade Asset Management

Data Cleaning

- Select variables (features) drop others
  - Drop constant features



	name	age	time_since_birth	group	country	europe	salary	patrimony	weight	size	comment	voted
0	ju	25.0	25.0	2.0	fr	True	1800.0	3000.0	70.0	175.0	NaN	True
1	ma	36.0	36.0	2.0	gb	True	3000.0	7000.0	NaN	180.0	NaN	NaN
2	lo	18.0	18.0	1.0	gb	True	1500.0	2000.0	NaN	150.0	NaN	True
3	fi	18.0	18.0	1.0	fr	True	1500.0	60000.0	NaN	NaN	NaN	NaN
4	xa	25.0	25.0	2.0	fr	True	1800.0	8000000.0	NaN	NaN	NaN	True
5	pa	38.0	38.0	2.0	es	True	7500.0	6000.0	NaN	170.0	NaN	NaN
6	pe	40.0	40.0	2.0	it	True	8000.0	4000.0	90.0	170.0	NaN	True
7	pe	40.0	40.0	2.0	it	True	8000.0	4000.0	90.0	170.0	NaN	True
8	jb	18.0	18.0	NaN	gb	True	NaN	NaN	60.0	NaN	NaN	NaN
9	mp	40.0	40.0	NaN	fr	True	NaN	NaN	NaN	NaN	NaN	NaN
10	ka	NaN	NaN	NaN	es	True	NaN	NaN	NaN	NaN	NaN	True
11	te	180.0	NaN	1.0	gb	True	4000.0	4000.0	NaN	150.0	NaN	True
12	ko	60.0	60.0	2.0	it	True	20000.0	7000.0	NaN	180.0	NaN	True

Data Cleaning

- Select variables (features) drop others
  - Drop constant features

```
# Detect columns with a constant value
const_col = df.nunique(dropna=False) == 1
const_col
```

✓ 0.5s

```
name                False
age                 False
time_since_birth    False
group               False
country             False
europe               True
salary              False
patrimony           False
weight              False
size                False
comment              True
voted               False
dtype: bool
```

```
# Drop these constant columns
# To drop all:
#df = df.T[~const_col].T
df = df.drop(columns='europe')
df
```

✓ 0.4s

	name	age	time_since_birth	group	country	salary	patrimony	weight	size	comment	voted
0	ju	25.0	25.0	2.0	fr	1800.0	3000.0	70.0	175.0	NaN	True
1	ma	36.0	36.0	2.0	gb	3000.0	7000.0	NaN	180.0	NaN	NaN
2	lo	18.0	18.0	1.0	gb	1500.0	2000.0	NaN	150.0	NaN	True
3	fi	18.0	18.0	1.0	fr	1500.0	60000.0	NaN	NaN	NaN	NaN
4	xa	25.0	25.0	2.0	fr	1800.0	8000000.0	NaN	NaN	NaN	True
5	pa	38.0	38.0	2.0	es	7500.0	6000.0	NaN	170.0	NaN	NaN
6	pe	40.0	40.0	2.0	it	8000.0	4000.0	90.0	170.0	NaN	True
7	pe	40.0	40.0	2.0	it	8000.0	4000.0	90.0	170.0	NaN	True
8	jb	18.0	18.0	NaN	gb	NaN	NaN	60.0	NaN	NaN	NaN
9	mp	40.0	40.0	NaN	fr	NaN	NaN	NaN	NaN	NaN	NaN
10	ka	NaN	NaN	NaN	es	NaN	NaN	NaN	NaN	NaN	True
11	te	180.0	NaN	1.0	gb	4000.0	4000.0	NaN	150.0	NaN	True
12	ko	60.0	60.0	2.0	it	20000.0	7000.0	NaN	180.0	NaN	True

Data Cleaning

- Select variables (features) drop others
  - Duplicated columns

```
# See which column values are duplicated
dup_cols = df.T.duplicated()
dup_cols
```

✓ 0.5s

```
name                False
age                 False
time_since_birth    False
group               False
country             False
salary              False
patrimony           False
weight              False
size                False
voted               False
dtype: bool
```

```
# Drop these duplicated columns by transposing the cleaned DataFrame
df = df.T.drop_duplicates().T
df
```

✓ 0.1s

	name	age	time_since_birth	group	country	salary	patrimony	weight	size	voted
0	ju	25.0	25.0	2.0	fr	1800.0	3000.0	70.0	175.0	True
1	ma	36.0	36.0	2.0	gb	3000.0	7000.0	NaN	180.0	NaN
2	lo	18.0	18.0	1.0	gb	1500.0	2000.0	NaN	150.0	True
3	fi	18.0	18.0	1.0	fr	1500.0	60000.0	NaN	NaN	NaN
4	xa	25.0	25.0	2.0	fr	1800.0	8000000.0	NaN	NaN	True
5	pa	38.0	38.0	2.0	es	7500.0	6000.0	NaN	170.0	NaN
6	pe	40.0	40.0	2.0	it	8000.0	4000.0	90.0	170.0	True
7	pe	40.0	40.0	2.0	it	8000.0	4000.0	90.0	170.0	True
8	jb	18.0	18.0	NaN	gb	NaN	NaN	60.0	NaN	NaN
9	mp	40.0	40.0	NaN	fr	NaN	NaN	NaN	NaN	NaN
10	ka	NaN	NaN	NaN	es	NaN	NaN	NaN	NaN	True
11	te	180.0	NaN	1.0	gb	4000.0	4000.0	NaN	150.0	True
12	ko	60.0	60.0	2.0	it	20000.0	7000.0	NaN	180.0	True

Data Cleaning

- Select variables (features) drop others
  - Drop columns full missing values (NA)

```
# See which columns have only NA values
df.isna().all()
```

✓ 0.4s

```
name                False
age                 False
time_since_birth    False
group               False
country             False
salary              False
patrimony           False
weight              False
size                False
comment              True
voted               False
dtype: bool
```

```
# Drop columns with only NA values
df.dropna(axis=1, how='all', inplace=True)
df
```

✓ 0.5s

	name	age	time_since_birth	group	country	salary	patrimony	weight	size	voted
0	ju	25.0	25.0	2.0	fr	1800.0	3000.0	70.0	175.0	True
1	ma	36.0	36.0	2.0	gb	3000.0	7000.0	NaN	180.0	NaN
2	lo	18.0	18.0	1.0	gb	1500.0	2000.0	NaN	150.0	True
3	fi	18.0	18.0	1.0	fr	1500.0	60000.0	NaN	NaN	NaN
4	xa	25.0	25.0	2.0	fr	1800.0	8000000.0	NaN	NaN	True
5	pa	38.0	38.0	2.0	es	7500.0	6000.0	NaN	170.0	NaN
6	pe	40.0	40.0	2.0	it	8000.0	4000.0	90.0	170.0	True
7	pe	40.0	40.0	2.0	it	8000.0	4000.0	90.0	170.0	True
8	jb	18.0	18.0	NaN	gb	NaN	NaN	60.0	NaN	NaN
9	mp	40.0	40.0	NaN	fr	NaN	NaN	NaN	NaN	NaN
10	ka	NaN	NaN	NaN	es	NaN	NaN	NaN	NaN	True
11	te	180.0	NaN	1.0	gb	4000.0	4000.0	NaN	150.0	True
12	ko	60.0	60.0	2.0	it	20000.0	7000.0	NaN	180.0	True

Data Cleaning

- Select variables (features) drop others
  - Excessive correlation between features

Data Cleaning

- Drop poor rows
  - Drop duplicated rows
  - Drop rows with excessing NA proportion

Data Cleaning

- Impute NAs (NaNs, missing values)
  - Missing is the information
  - Reconstruct (impute)
    - Constant (average, median)
    - Ffil, bfill
    - Group by
    - Interpolate

Data Cleaning

- Outliers
  - Extreme values too keep
  - Abberations to delete
  - Variables to transform

Data Management

- Merge tables
  - Concatenation on rows
  - Merge on unique key column
    - Outer (indicator)
    - Left
    - Right, inner

Feature Engineering

- Quantitative variables (numbers representing quantities): create groups
- Qualitative variables (categories): one-hot encode
- Filter

Why using vizualizations

- Quick understanding simple patterns (trend line plot, groups scatter plot)
- Better intuition on complex patterns (CNN weights maps)
- Reporting

Graphs types

- Univariate Analysis
  - Histograms (distributions)
  - Line plots (Time series)
  - Lorentz Curve (inegalities)
- Multivariate Analysis
  - Scatter plots
  - Heatmaps
    - Correlations
    - Confusion matrices

Libraries

- Matplotlib (.pyplot)
- Seaborn for nice default graphs
- Plotly (Dash) for interactive graphs