1495 UNIVERSITY OF ABERDEEN | CELEBRATING 525 YEARS 1495 – 2020

ABERDEEN 2040

Exploratory Data Analysis in Practice

Data Mining & Visualisation

Lecture 4

2025

Today…

- Exploring your data
- Cleaning and pre-processing
- Dealing with missing data
- Class imbalance
- Split–Apply–Combine

Exploring Your Data

The first step in data analysis is often to understand what our dataset ‘looks like’.

Pandas has lots of tools to help explore a given dataset.

Let’s say we load in the Titanic dataset…

```
# Load the 'titanic' example dataset
titanic = sns.load_dataset("titanic")
```

Exploring Your Data – head()

What do the first few rows of your data look like?

```
# Show the first five rows of the titanic dataset
titanic.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

Exploring Your Data – tail()

What do the last few rows of your data look like?

```
# Show the last ten rows
titanic.tail(n=10)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
881	0	3	male	33.0	0	0	7.8958	S	Third	man	True	NaN	Southampton	no	True
882	0	3	female	22.0	0	0	10.5167	S	Third	woman	False	NaN	Southampton	no	True
883	0	2	male	28.0	0	0	10.5000	S	Second	man	True	NaN	Southampton	no	True
884	0	3	male	25.0	0	0	7.0500	S	Third	man	True	NaN	Southampton	no	True
885	0	3	female	39.0	0	5	29.1250	Q	Third	woman	False	NaN	Queenstown	no	False
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

Note that for both head() and tail(), you can change the number of rows returned with *n*.

Exploring Your Data – shape

How many variables (columns) does your dataset have? How many rows?

```
titanic.shape
```

```
(891, 15)
```

Exploring Your Data – info()

What are the data types of the variables?

```
titanic.info()
✓ 0.0s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

Think about their levels of measurement.

Think about what these variables represent!

Exploring Your Data – describe()

What are the central tendencies of the variables? What are their variability?

```
titanic.describe()
✓ 0.0s
```

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Think about how each of the variables are distributed.

What does that tell us about the data?

Cleaning Data

Cleaning Data

In AI lectures and tutorials, there's often an assumption that your data is 'clean'– i.e. free from issues and ready for analysis.

In that case, you can fetch data from a Python library, Kaggle, or GitHub repository and you're ready to go!

Cleaning Data

In practice, however, this is often not the case!

Data can be 'messy', and the process of **cleaning data** seeks to address any issues in the dataset.

By cleaning the data properly before you analyse it, your analysis can be more accurate and robust.

Cleaning Data – Examples

- Some examples of messy data:
  - Erroneous values (e.g. spelling mistakes)
  - Duplicated rows
  - Partial data (e.g. abandoned questionnaires)
  - Missing values
  - Extreme outliers

Erroneous Values

Let's say we want to check how many people embarked the Titanic from each town.

```
# Count the number of rows with each embark town
titanic['embark_town'].value_counts()
```

[3] ✓ 0.0s

```
Southampton    644
Cherbourg      167
Queenstown      77
Cherbuorg        1
Name: embark_town, dtype: int64
```

Erroneous Values

Let's say we want to check how many people embarked the Titanic from each town.

```
# Count the number of rows with each embark town
titanic['embark_town'].value_counts()
```

[3] ✓ 0.0s

```
Southampton    644
Cherbourg      167
Queenstown      77
Cherbuorg        1
Name: embark_town, dtype: int64
```

Probably a spelling mistake in the data!

This is particularly an issue worth checking when dealing with user-input data.

Missing Data

Is_purchaser	time_on_site	pages_visited	location	was_referred
True	0-9 minutes	4	UK	True
True	10-20 mins	13	UK	False
False	0-9 minutes	?	US	True
False	10-20 minutes	7	UK	False
True	0-9 minutes	3	US	False

What can we do if we have some missing data within the dataset?

Missing Data

We could delete the row(s) where data is missing.

But this might result in lots of missing data, and in practice, that column might not be that interesting or useful anyway!

Missing Data

Instead, we could insert the mean, median, or mode of the variable.

This is a reasonable solution, but it feels a little primitive.

We can do better!

Missing Data

We could copy across the value held within a similar row, e.g. using its ‘nearest neighbour’, or use other methods to find similar rows.

This method is more sophisticated, but it may have pros and cons depending on the context, and the data at hand.

Missing Data – Overview

In short, there are many ways of addressing missing data, and there are no real ‘right’ or ‘wrong’ approaches.

Missing data can appear in many ways. Could be “**None**”; Could be “**NaN**”; Could be “”; Could be “ ”; Could be “**{}**” etc.

Many approaches will exist in popular packages (e.g. pandas).

Outliers

Outliers are data points that differ significantly from other observations.

Sometimes, a value might be so large or small that it severely distorts some of your analysis.

E.g. think of the average annual salary for a group of 100 people (but one of them happens to be a billionaire).

Dealing With Outliers

We could opt to remove the row of data, or impute some other value to the data (e.g. 2 std devs above the mean).

However, removing or changing data can be a contentious issue (particularly within research contexts).

By removing or changing data, you are at risk of modifying your dataset in order to ‘look for the right answer’.

Dealing With Outliers – Risks

Removing or changing data to get more favourable results is often referred to as ‘p-hacking’ your data, or ‘data dredging’.

It is a significant problem in science as a whole. And as such, removing outliers is not something to be taken lightly!

Again, whether and how you deal with outliers will often be contextual, but your decision should be **justifiable**!

Data Pre-Processing

Data Pre-Processing

Say you've cleaned your data, addressing common issues.

Next, we consider how we might want to pre-process the data, to get it ready for analysis.

We'll outline a few examples, but there are many options for pre-processing and 'feature engineering'.

Data Pre-Processing – Dummy Coding

Dummy coding refers to converting a categorical variable to a set of numerical (Boolean) values, representing the same data.

This can be particularly useful when dealing with nominal or ordinal variables.

Data Pre-Processing – Dummy Coding Example

Let's say we have a categorical variable *height*, which can contain either {short, medium, tall}.

animal	height
giraffe	tall
cat	medium
hamster	short
dog	medium
elephant	tall

Data Pre-Processing – Dummy Coding Example

Dummy coding this variable would give us three new variables: *height_short*; *height_medium*; *height_tall.*

animal	height	height_short	height_medium	height_tall
giraffe	tall	0	0	1
cat	medium	0	1	0
hamster	short	1	0	0
dog	medium	0	1	0
elephant	tall	0	0	1

For each row of data, only the variable corresponding to the correct categorical value would be 1; the rest would be 0.

Data Pre-Processing – Bucketing

In contrast, there may be times when we want to convert numerical (interval or scale) data into ordinal categories.

This can be useful when we want to deal with groups which each share some similar numerical properties.

It is sometimes referred to as binning, because we are placing our numerical data into ‘bins’, or ‘buckets’.

Data Pre-Processing – Bucketing Example

Let's say we have an *age* variable, containing quantitative values: [15, 17, 24, 28, 33, 45]

We might choose to convert these values into: [11-19, 20-29, 30-39, 40+]

Or, instead we might choose to use: [Adolescent, Adolescent, Adult, Adult, Adult, Adult]

It is our choice how we engineer and pre-process our data!

Class Imbalance

Class Imbalance – Fraud Example

Let's say you want to predict fraud in purchasing records.

However, you have very few examples of fraudulent transactions within your dataset (~2%).

You build a predictive model, which ends up having 98% accuracy. Great! ...right?

Class Imbalance

You may want to predict rare events, with very few examples within your dataset.

You want to make sure that the model doesn't 'cheat’.

One way to do this is by addressing the underlying class imbalance. There are two main approaches for doing so.

Data Downsampling

Option one – Downsampling: We **sample** the **majority class**, such that its n = n of the minority class.

We can do this via random sampling, though other (more advanced) approaches exist (we won't cover these).

The upside is balanced classes. The downside is we now have way less data!

Data Upsampling

Option two – Upsampling: We duplicate instances of the **minority class**, such that its n = n of the majority class.

This gets us way more data overall, and we avoid throwing any of our data away!

However, this can lead to models ‘overfitting’ the data.

Split–Apply–Combine

The Split–Apply–Combine Paradigm

’Split-apply-combine’ is a very common, and very powerful paradigm for data analysis:

- [**Split**] a dataset into groups based on some criteria
- [**Apply**] some functions to each group
- [**Combine**] the results into a newly transformed dataset

Split–Apply–Combine – Example

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

- [**Split**] based on class
- [**Apply**] mean('age')
- [**Combine**] (happens automatically)

```
class
First     38.233441
Second    29.877630
Third     25.140620
Name: age, dtype: float64
```

Split–Apply–Combine – Example

Note: If this sounds very similar to querying a database, that's because it is!

'SELECT sex, mean(age) FROM titanic GROUP BY sex'

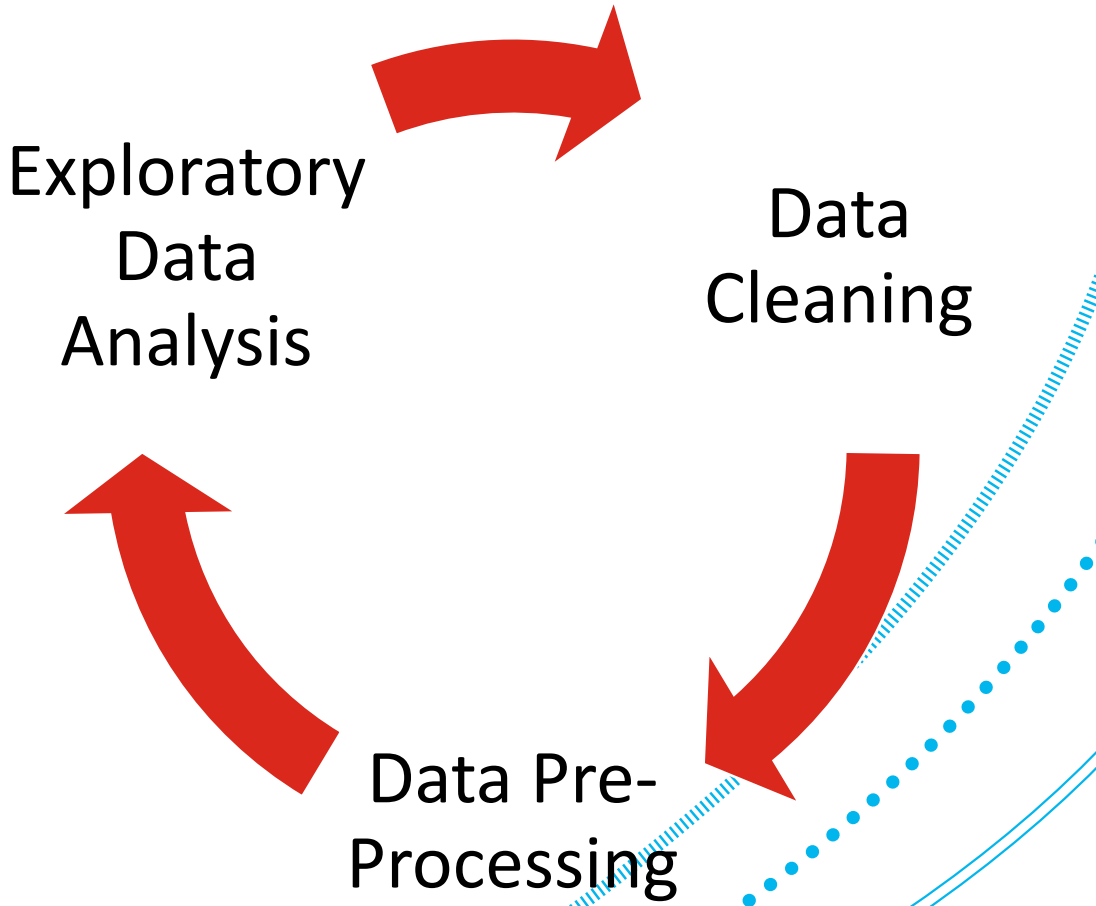
Many data analysis methods and libraries take heavy inspiration from SQL.

Summary

Summary

Some think of EDA as the bit before you analyse your data.

In reality, it is an ongoing part of the analysis process…



Summary

There are many ways to clean and process your data.

Importantly, these will often bring with them implications and consequences!

Appropriate use of these methods will be contextual.