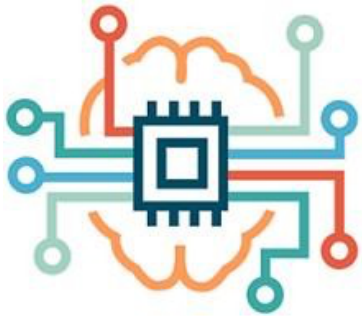
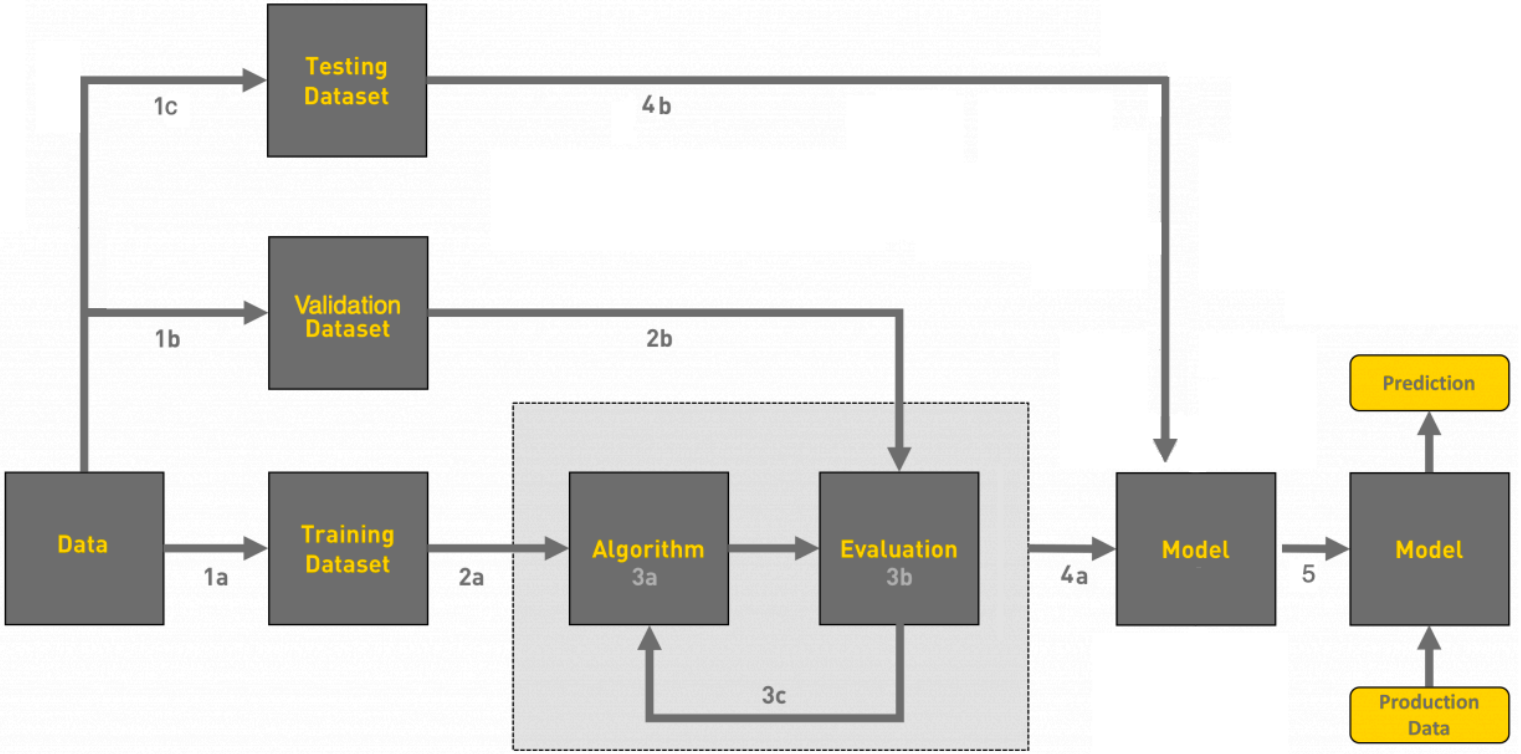
COMP1804
Applied Machine Learning

Lecture 3: Data Pre-Processing

Dr. Dimitrios Kollias



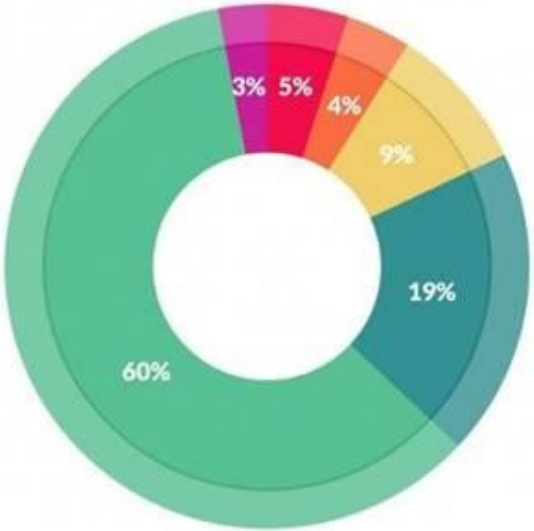
Process of creating a ML system



End-to-end project

1. Look at the big picture
2. Get the data
3. Discover and visualise the data to gain insights
4. Prepare the data for ML algorithms
5. Select a model and train it
6. Fine-tune your model
7. Present your solution
8. Launch, monitor, and maintain your system

Data scientist time spent



What data scientists spend the most time doing

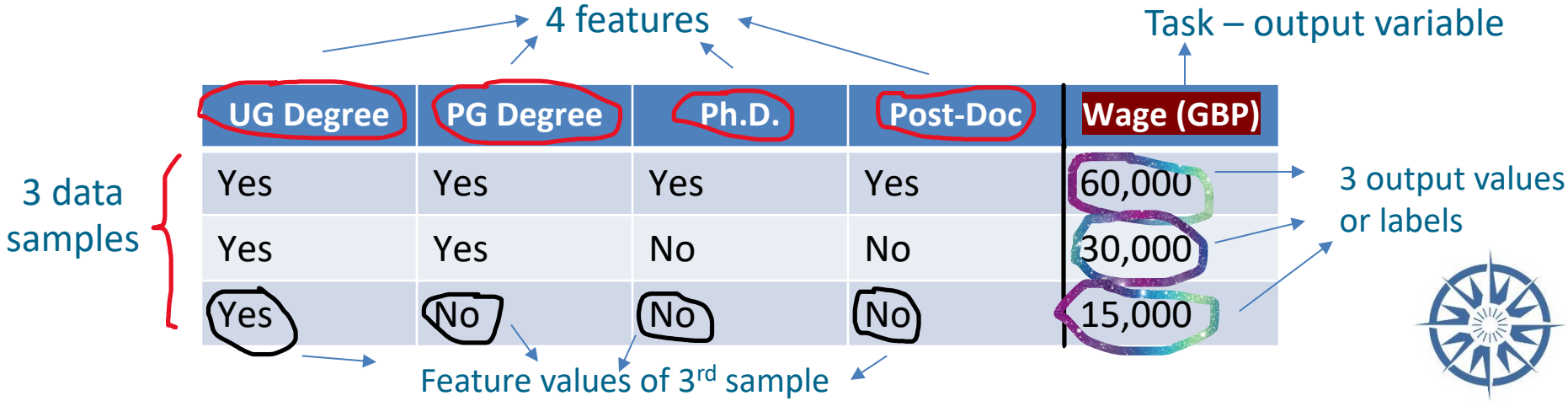
- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

CrowdFlower Survey Results http://omaymas.github.io/Data_Science_Life/

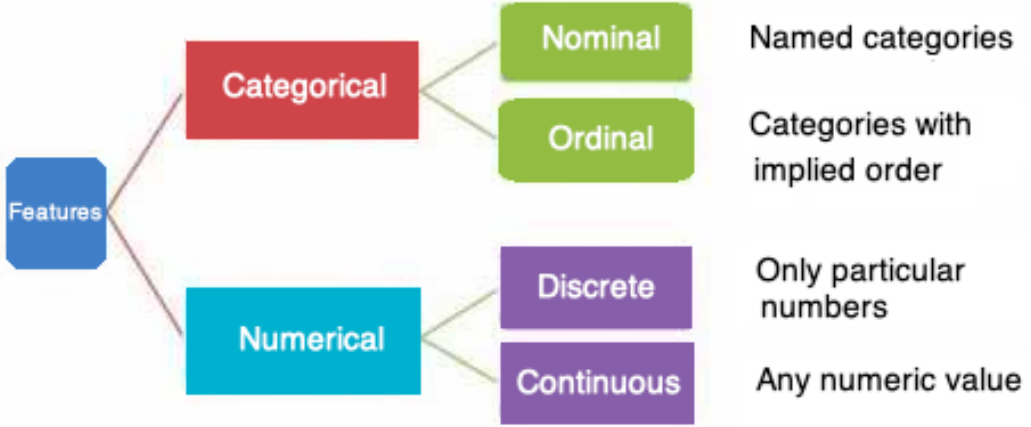
Data

Dataset & Features

A dataset can be viewed as a collection of *data objects*, which are often also called as records, points, vectors, patterns, events, cases, samples, observations, or entities.
Data objects are described by a number of *features*, that capture the basic characteristics of an object, such as the mass of a physical object or the time at which an event occurred, etc..
Features are often called as variables, characteristics, fields, attributes, or dimensions.



Feature Types



Categorical Features

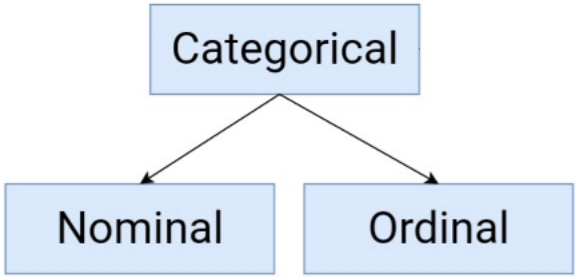
Features whose values are taken from a defined set of values.
Categorical features can only take on a limited, and usually fixed, number of possible values. These features are typically stored as text values which represent various traits of the observations.

For example, if a dataset is about information related to users, then you will typically find features like gender, country, etc.
Gender is described as Male (M) or Female (F), days in a week : {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday} is a category because its value is always taken from this set.

Categorical Feature Types (I)

These type of features where the categories are only labelled without any order of precedence are called nominal features.

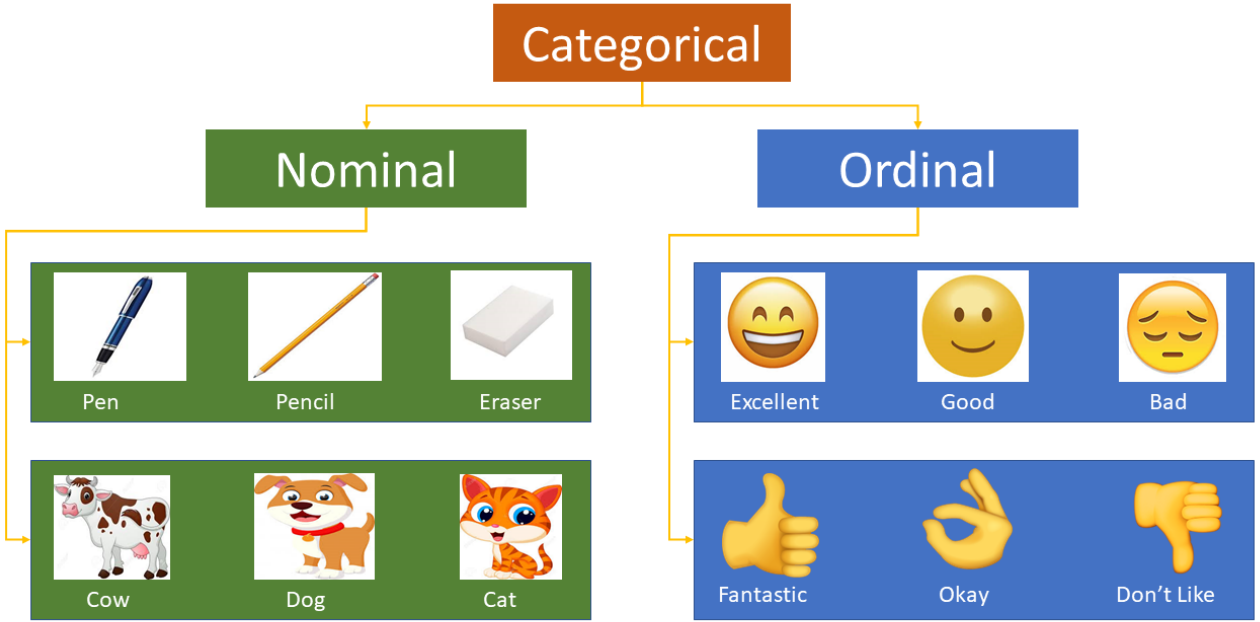
Features which have some order associated with them are called ordinal features.
Example: a feature like economic status, with three categories: low, medium and high, which have an order associated with them.



Categorical Feature Types (II)

Nominal	Ordinal
Categorical variables without any implied order	Categorical variables with a natural implied order but the scale of difference is not defined
Example : A new car model comes in these colors : Black, Blue, White, Silver	Example : Sizes of clothes has a natural order : Extra Small < Small < Medium < Large < Extra Large - But this does not mean Large - Medium = Medium - Small

Categorical Feature Types (IIII)



Numerical Features

Features whose values are continuous or integer-valued. Numerical features are expressed in numbers, rather than natural language description.

For instance:
number of steps you walk in a day, or the speed at which you are driving your car at, or counting the cups of water required to empty an ocean

Data Pre-Processing

What is Data Pre-Processing

Data pre-processing in Machine Learning is a crucial step that helps enhance the quality of data to promote the extraction of meaningful insights from the data.

Data pre-processing in Machine Learning refers to the technique of preparing (cleaning, formatting and organizing) the raw data to make it suitable for building and training Machine Learning models.

Why we need Data Pre-Processing (I)

When it comes to creating a Machine Learning model, data pre-processing is the first step marking the initiation of the process.

Typically, real-world data is:

- incomplete (lacking attribute values, lacking certain attributes of interest, or containing only aggregate data),
- inconsistent (containing discrepancies in codes or names),
- inaccurate/noisy (contains errors or outliers),
- lacking specific attribute values/trends.

Why we need Data Pre-Processing (II)

By pre-processing data, we:

Make our database more accurate. We eliminate the incorrect or missing values that are there as a result of the human factor or bugs.

Boost consistency. When there are inconsistencies in data or duplicates, it affects the accuracy of the results.

Make the database more complete. We can fill in the attributes that are missing if needed.

Smooth the data. This way we make it easier to use and interpret.

Steps of Data Pre-Processing

Remember, not all the steps are applicable for each problem; which steps are chosen is highly dependent on the data we are working with. Generally the steps are :

- Data Quality Assessment
- Exploratory Data Analysis
- Feature Aggregation & Discretization
- Feature Encoding
- Feature Sampling
- Feature Scaling

Data Quality Assessment (I)

Because data is often taken from multiple sources which are normally not too reliable and also in different formats, more than half our time is consumed in dealing with data quality issues when working on a machine learning problem.

It is simply unrealistic to expect that the data will be perfect.
There may be problems due to human error, limitations of measuring devices, or flaws in the data collection process.

Data Quality Assessment (II)

You need to have a good look at the generated database and perform a data quality assessment.
A random collection of data often has irrelevant bits:

- **Mismatching in data types**

Quite often, you might mix together datasets that use different data formats.
Hence, the mismatching: integer vs. float or UTF8 vs ASCII.

- **Different dimensions of data arrays**

When you aggregate data from different datasets, for example, from five different arrays of data for voice recognition, some features that are present in one array can be missing in the rest arrays.

Data Quality Assessment (III)

- **Mixture of data values**

Imagine that you have data, collected from two independent sources. On the one source, the gender field/feature takes the values {male, female}, whereas on the other source it takes the values {man,woman}. We know that the words ‘man’ and ‘male’ describe the same thing (and similarly for the words ‘female’ and ‘woman’).
You have to use only one type of words (e.g., {‘man’, ‘woman’}) in the generated dataset.

- **Inconsistent values**

Data can contain inconsistent values. For instance, the ‘Address’ field may contain a phone number. This may be due to human error or because the information was misread while being scanned from a handwritten form.
It is thus always advised to know what the data type of the features should be and use it to check whether it is the same for all data objects.

Data Quality Assessment (IV)

- **Noisy data:**

A large amount of additional meaningless data is called *noise.*
This can be:

  - **Duplicate values**

    A dataset may include data objects which are duplicates of one another. It may happen when say the same person submits a form more than once. In most cases, the duplicates are removed so as to not give that particular data object an advantage or *bias*, when running machine learning algorithms.

Data Quality Assessment (V)

- **Noisy data:**

A large amount of additional meaningless data is called *noise.*
This can be:

- **Data segments, which have no value for a particular research**

For instance you want to predict brain damage from images of the brain and some data are images of patients' hands.

- **Unnecessary features**

An example is when you need to predict a person salary and the features that you have, include working_experience, qualifications, education and shoe_size. The last one is unnecessary.

Data Quality Assessment (VI)

- **Outliers in the dataset**

The outliers are the singular data points dissimilar to the rest of the domain.
For example, within 200 years of daily temperature observations for New York, there were several days with very low temperatures in summer.
Another example: we are building an algorithm that sorts out different sorts of apples. We can encounter two types of outliers in our dataset:

- The images contain exotic fruits like pineapples and kiwi. They can be found in your data due to a sampling mistake and represent noise in your dataset.

Data Quality Assessment (VII)

- **Outliers in the dataset**
    - There also can be photos of some “weird apples”, for example, that have a strange shape.

When our goal is to teach the machine to recognize the apple sorts, deviation from groups is important. Such outliers will help to teach the ML model to recognize special characters and increase the accuracy of the forecast. It’s important not to substitute the outliers by taking them as noise.

Usually, researchers evaluate the outliers to identify whether each particular record is the result of an error in the data collection or a unique phenomenon which should be taken into consideration for data processing.

Data Quality Assessment (VI)

- **Missing data**

Some important feature values can be missing in the data.
This situation is quite common and is due to human factors, program errors, or other reasons.

How to handle such cases:

- Look for additional datasets or collect more observations.
- Eliminate rows with missing data :

Simple and sometimes effective strategy.
Fails if many objects have missing values.
If a feature has mostly missing values, then that feature itself can also be eliminated.
Removing rows or columns with missing data, can affect the performance of the model, as the size of the data will decrease.

Data Quality Assessment (VI)

- **Missing data**

How to handle such cases:

- Estimate missing values :

If only a reasonable percentage of values are missing, run simple interpolation methods to fill in those values.

Most common method of dealing with missing values is by filling them in with the mean, median or mode value of the respective feature.

Median is preferable to the ***mean*** because it is not affected by outliers.

This solution would preserve the size of the data.

In case of categorical values, missing values can be replaced with the most common categorical value of the column.

Feature Aggregation

Feature Aggregations are performed so as to take the aggregated values in order to put the data in a better perspective.

For example, think of transactional data:
suppose we have day-to-day transactions of a product in five store locations over the year and thus we have 5 x 365 features.
Aggregating the transactions to single store-wide monthly transactions will help us to reduce the number of features to 1 x 12.

This can result in reduction of memory consumption and processing time.
Aggregations provide us with a high-level view of the data as the behaviour of groups or aggregates is more stable than individual data objects.

Discretization

In the case of numerical features (taking continuous values), we can transform their values to discrete.

For example, the ‘age’ feature takes values in e.g., [0,100].

We can transform this to the categories “young”, “middle age”, “senior”.

Discretization helps to improve efficiency.

Exploratory Data Analysis (EDA)

In statistics, Exploratory Data Analysis (EDA) is an approach to analyse datasets, summarizing their main characteristics, often with visual methods.

There are a number of tools that are useful for EDA.

Typical graphical techniques used in EDA are:

- Box Plot
- Histogram
- Scatter Plot

Box Plot (I)

For some distributions/datasets, you will find that you need more information than the measures of central tendency (median, mean, and mode).

You need to have information on the variability or dispersion of the data.

A box plot is a graph that gives you a good indication of how the values in the data are spread out.

Box plots have the advantage of taking up less space, which is useful when comparing distributions between many groups or datasets.

Boxplots are a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”).

Box Plot (II)

- **median (Q2/50th Percentile)**:

The median (middle quartile) marks the mid-point of the data and is shown by the line that divides the box into two parts. Half the scores are greater than or equal to this value and half are less.

- **first quartile (Q1/25th Percentile)**:

The middle number between the smallest number (not the “minimum”) and the median of the dataset. Twenty-five percent of scores fall below the lower quartile.

- **third quartile (Q3/75th Percentile)**:

The middle value between the median and the highest value (not the “maximum”) of the dataset. Seventy-five percent of the scores fall below the upper quartile.

Box Plot (II)

- **interquartile range (IQR)**:

The middle “box” represents the middle 50% of scores for the group. The range of scores from first to third quartile is referred to as the inter-quartile range. The middle 50% of scores fall within the inter-quartile range.

- **“maximum”**: Q3 + 1.5*IQR

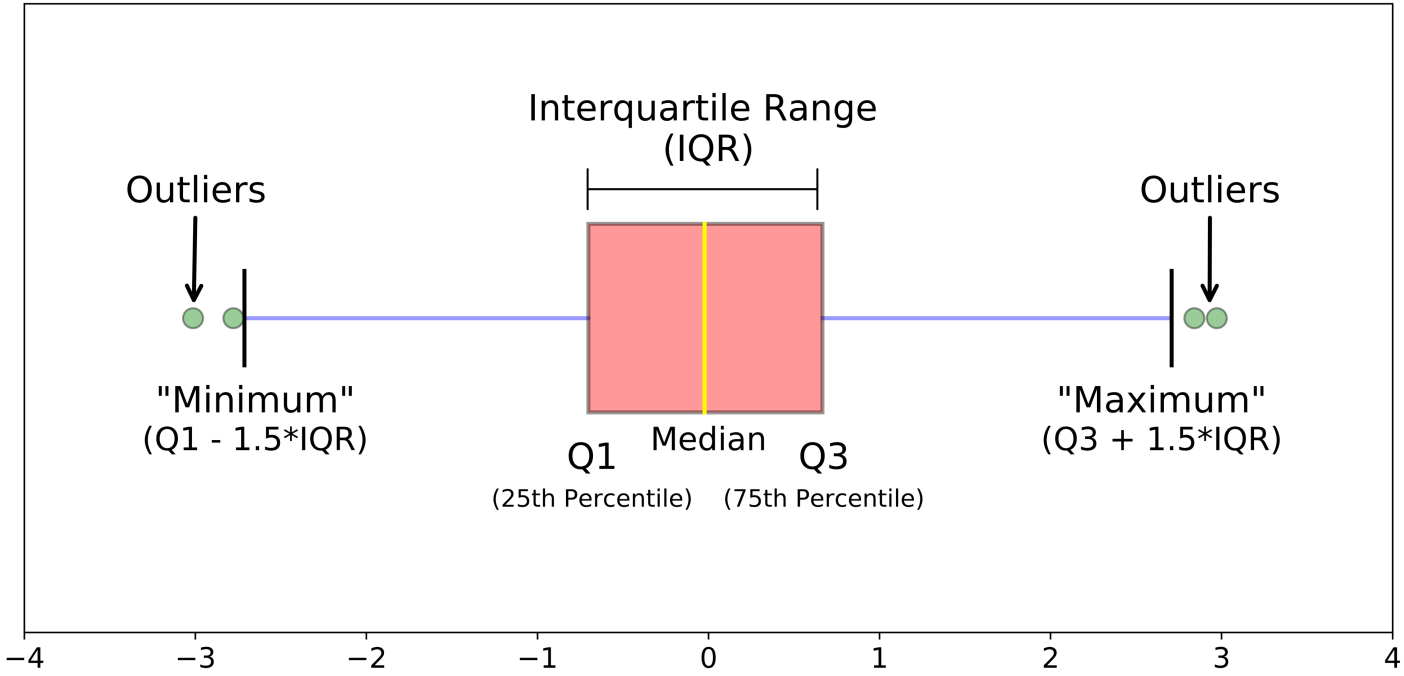
- **“minimum”**: Q1 -1.5*IQR

whiskers (shown in blue):

The upper and lower whiskers represent scores outside the middle 50%. Whiskers often (but not always) stretch over a wider range of scores than the middle quartile groups.

outliers (shown as green circles)

Box Plot (III)


Interquartile Range
(IQR)
Outliers
Outliers
"Minimum"
(Q1 - 1.5*IQR)
Q1
Median
Q3
(25th Percentile)
(75th Percentile)
"Maximum"
(Q3 + 1.5*IQR)
−4
−3
−2
−1
0
1
2
3
4

Histogram (I)

A **histogram** is an approximate representation of the distribution of numerical data.

To construct a histogram, the first step is to "bin" the range of values, i.e., divide the entire range of values into a series of intervals, and then count how many values fall into each interval.

The bins are usually specified as consecutive, non-overlapping intervals of a variable.

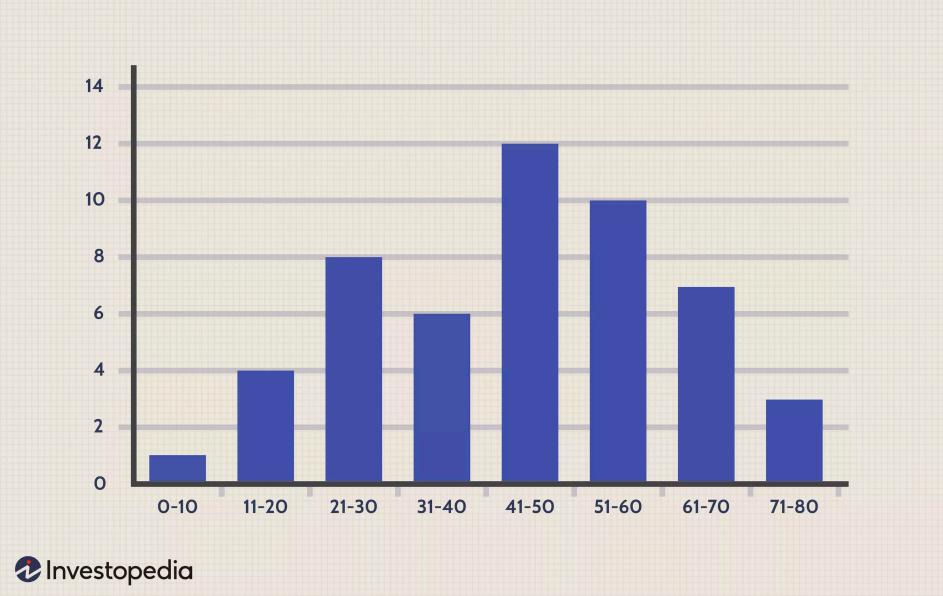
The bins (intervals) must be adjacent and are often (but not required to be) of equal size.

If the bins are of equal size, a rectangle is erected over the bin with height proportional to the frequency (i.e., the number of cases in each bin).

A histogram may also be normalized to display "relative" frequencies.

Histogram (II)

Example: a census focused on the demography of a country may use a histogram to show how many people are between the ages of 0 - 10, 11 - 20, 21 - 30, 31 - 40, 41 - 50, 51 - 60, 61 - 70, 71 - 80.
In the above example, there are 8 bins with an interval of ten.
This could be changed, for example, to 10 bins with an interval of eight instead.



Scatter Plot (I)

A scatter plot uses dots to represent values for two different numeric variables.

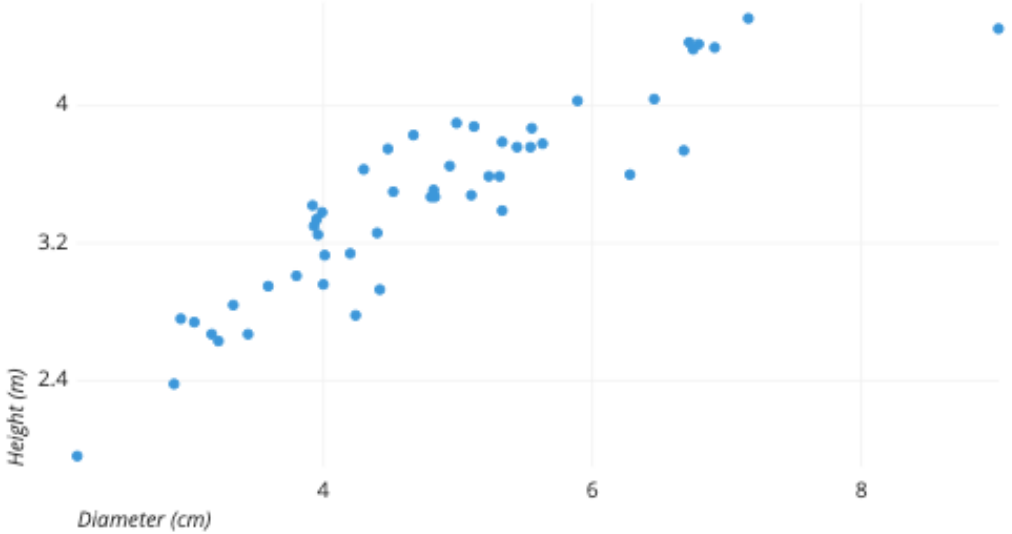
The position of each dot on the horizontal and vertical axis indicates values for an individual data point.

Scatter plots' primary uses are to observe and show relationships between two numeric variables.

The dots in a scatter plot not only report the values of individual data points, but also patterns when the data are taken as a whole.

Scatter Plot (II)

It shows the diameters and heights for a sample of fictional trees.
Each dot represents a single tree; each point's horizontal position indicates that tree's diameter and the vertical position indicates that tree's height.
From the plot, we can see a generally tight positive correlation between a tree's diameter and its height.
We can also observe an outlier point, a tree that has a much larger diameter than the others.



This tree appears fairly short for its girth, which might warrant further investigation.

Feature Encoding: Encoding Categorical Data (I)

Machine learning algorithms require that input and output variables are numbers.
This means that categorical data must be encoded as numbers before we can use it to fit and evaluate a model.

For the machine, categorical data doesn’t contain the same context or information that humans can easily associate and understand.

For example, when looking at a feature called ’City’ with three cities London, Greenwich and Madrid, humans can infer that London is closely related to Greenwich as they are from same country, while London and Madrid are much different.

But for the model, London, Greenwich and Madrid, are just three different levels (possible values) of the same feature ’City’.

If you don’t specify the additional contextual information, it will be impossible for the model to differentiate between highly different levels.

Feature Encoding: Encoding Categorical Data (II)

You are therefore faced with the challenge of figuring out how to turn these text values into numerical values for further processing and unmasking lots of interesting information which these features might hide.

As mentioned before, the whole purpose of data pre-processing is to *encode* the data in order to bring it to such a state that the machine now understands it.
Feature encoding is basically performing transformations on the data such that it can be easily accepted as input for machine learning algorithms while still retaining its original meaning.

Feature Encoding: Encoding Categorical Data (II)

Different techniques that encode the categorical features to numeric quantities are:

- Replacing values
- Label encoding
- One-Hot encoding
- Frequency Encoding
- Miscellaneous features

Feature Encoding: Replacing Values (I)

It is the most basic method: just replace the categories with desired numbers.

The idea is that you have the liberty to choose whatever numbers you want to assign to the categories according to the business use case.

Feature Encoding: Replacing Values (II)

City	Country
London	England
Bath	Germany
Bristol	France

London → 5; Bath → 1; Bristol → 2

England → 10; Germany → 5; France → 1

City	Country
5	10
1	5
2	1

Feature Encoding: Label Encoding (I)

This approach allows you to convert each value of a feature to a number.

The numbers that you choose are always between 0 and number_of_categories - 1.

Feature Encoding: Label Encoding (II)

City	Country
London	England
Bath	Germany
Bristol	France

Bath → 0; Bristol → 1; London → 2

France → 0; Germany → 1; England → 2;

City	Country
2	2
0	1
1	0

Feature Encoding: One-hot Encoding (I)

This strategy converts each feature value into a new feature and assign a 1 or 0 (True/False) value to the feature.

This has the benefit of not weighting a value improperly.

Feature Encoding: One-hot Encoding (II)

City	Country
London	England
Bath	Germany
Bristol	France

One-hot encoding

London	Bath	Bristol	England	Germany	France
1	0	0	1	0	0
0	1	0	0	1	0
0	0	1	0	0	1

Feature Encoding: Frequency Encoding

This technique substitutes a feature value with its frequency in the data.

A	0.44 (4 out of 9)
B	0.33 (3 out of 9)
C	0.22 (2 out of 9)

Feature	Encoded Feature
A	0.44
A	0.44
A	0.44
A	0.44
B	0.33
B	0.33
B	0.33
C	0.22
C	0.22

Feature Encoding: Miscellaneous features

Sometimes you may encounter features which specify the ranges of values for observation points/data samples, for example, the ’age’ feature might be described in the form of categories like 0-20, 20-40 and so on.

While there can be a lot of ways to deal with such features, the most common ones are:

- split these ranges into separate features (as in one-hot encoding)

or

- replace them with some measure like the mean of that range

Feature Sampling

Sampling is a very common method for selecting a subset of the dataset that we are analysing. In most cases, working with the complete dataset can turn out to be too expensive considering the memory and time constraints.
Using a sampling algorithm can help us reduce the size of the dataset to a point where we can use a better, but more *expensive*, ML algorithm.

The key principle here is that the sampling should be done in such a manner that the sample generated should have approximately the same properties as the original dataset, meaning that the sample is *representative*.
This involves choosing the correct sample size and sampling strategy.

Feature Sampling: Simple Random Sampling

Simple Random Sampling dictates that there is an equal probability of selecting any particular entity. It has two main variations as well:

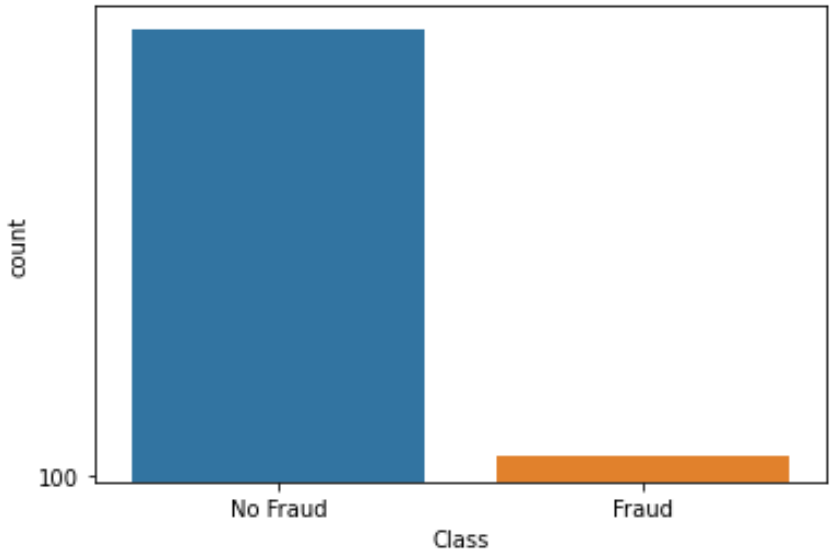
Sampling without Replacement : As each item is selected, it is removed from the set of all the objects that form the total dataset.

Sampling with Replacement : Items are not removed from the total dataset after getting selected. This means they can get selected more than once.

Although Simple Random Sampling provides two great sampling techniques, it can fail to output a representative sample when the dataset includes object types which vary drastically in ratio. This can cause problems when the sample needs to have a proper representation of all object types, for example, when we have an *imbalanced* dataset.

Feature Sampling: Data Imbalance

An Imbalanced dataset is one where the number of instances of a class(es) are significantly higher than another class(es), thus leading to an imbalance and creating rarer class(es).



Feature Sampling: Stratified Sampling

It is critical that the rarer classes be adequately represented in the sample.

In these cases, there is another sampling technique which we can use, called *Stratified Sampling*, which begins with predefined groups of objects.
There are different versions of Stratified Sampling too, with the simplest version suggesting equal number of objects be drawn from all the groups even though the groups are of different sizes.

Feature Scaling (I)

The goal of scaling is to change the values of numeric features in the dataset to a common scale, without distorting differences in the ranges of values.
For machine learning, every dataset does not require scaling.
It is required only when features have different ranges.

For example, consider a data set containing two features, ‘age’, and ‘income’.
‘Age’ ranges from 0–100, while income ranges from 0–100,000 and higher.
Income is about 1,000 times larger than age.
So, these two features are in very different ranges.
When we do further analysis, the attributed income will intrinsically influence the result more due to its larger value.
But this doesn’t necessarily mean it is more important as a predictor.
So we normalize the data to bring all the variables to the same range.

Feature Scaling (II)

Features that are measured at different scales do not contribute equally to the analysis and might end up creating a bias.

Some techniques to scale values on a dataset are ***Normalization*** and ***Standardization***.

Feature Scaling: Normalisation

Normalization scales all values in the range 0 and 1.
Each value of the variable is subtracted by the min value and divided by the difference between the max value and the min value.

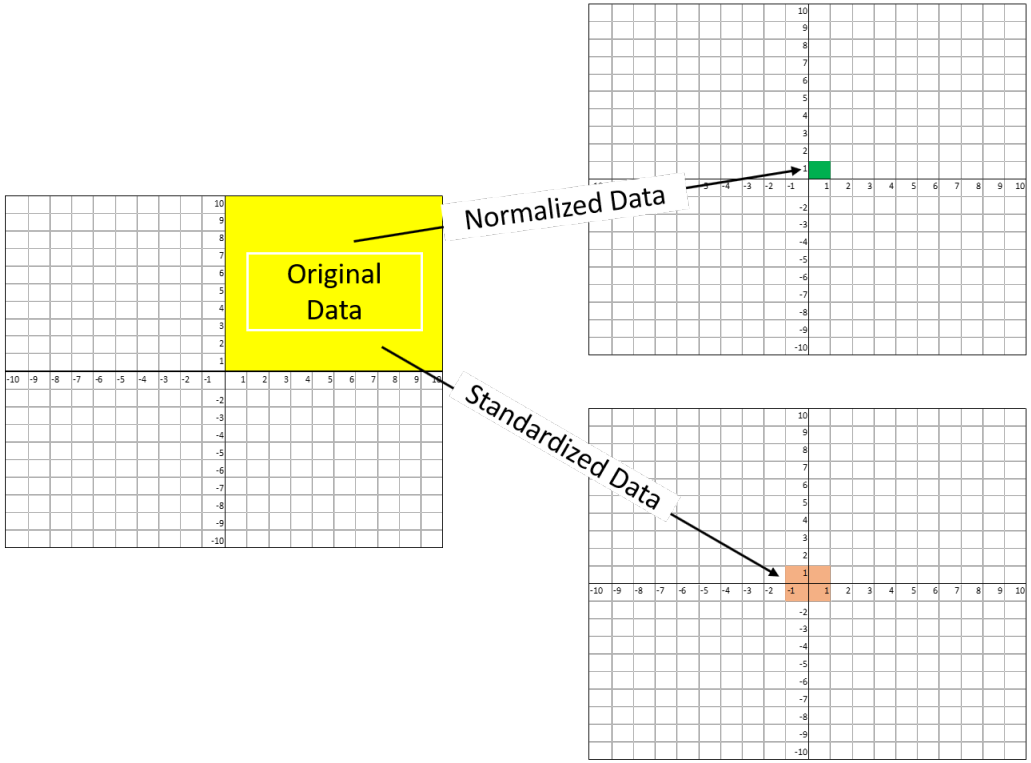
$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Feature Scaling: Standardisation

This method is also called the z-score and scale each of the values of features by removing them from their mean and dividing by the standard deviation.
Also, unlike normalization, standardization does not have a bounding range.

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

Feature Scaling

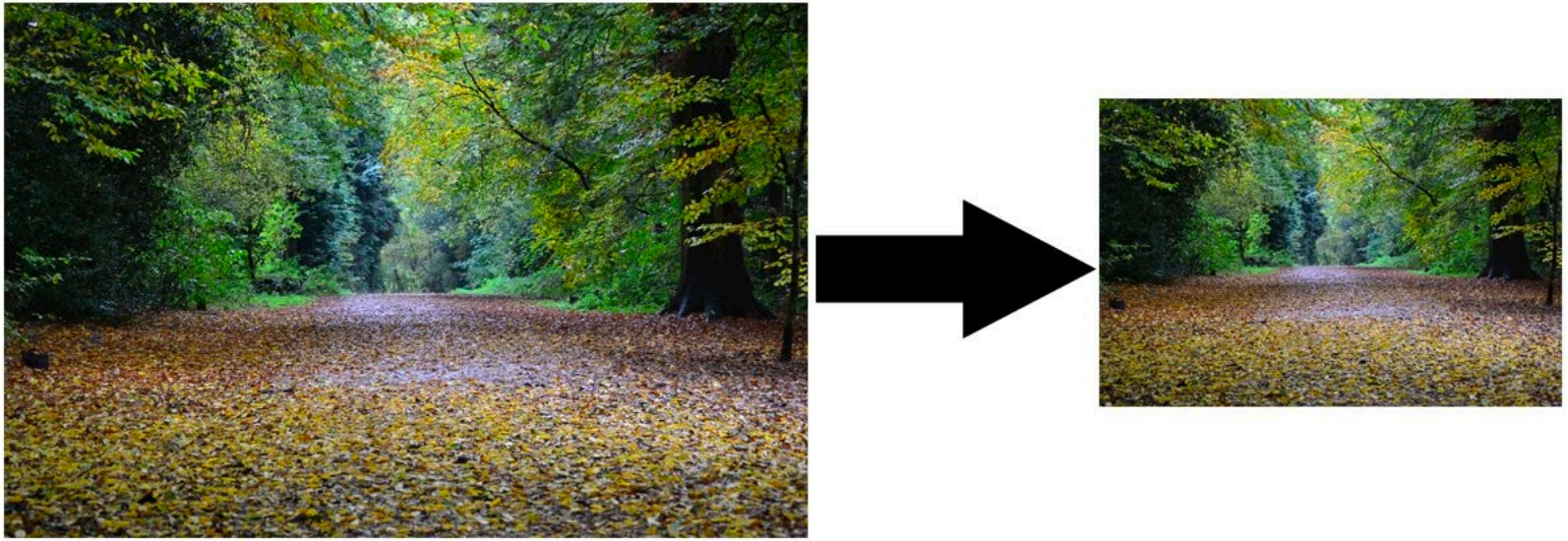


Data Pre-Processing: Which technique to use and when

What to do for data pre-processing is tailored to the specific application and depends on both the dataset and the machine learning algorithm utilised.

For example a standard pre-processing step performed when we are dealing with images is image resize.
Generally images vary in size; to feed them to our ML algorithm, we should establish a base size for all images.

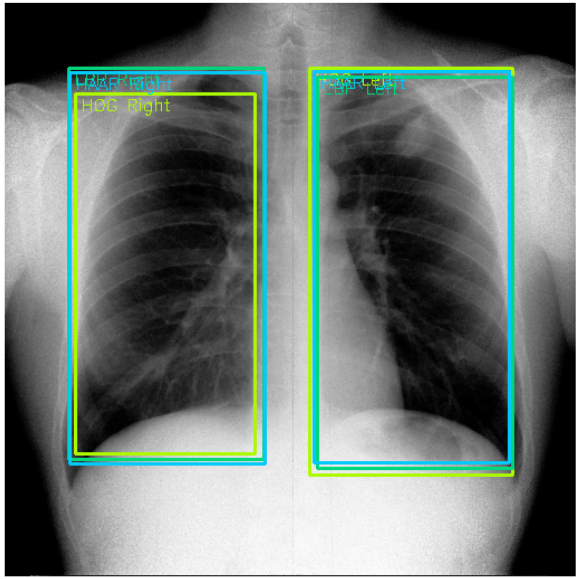
Data Pre-Processing Example: Image Resize

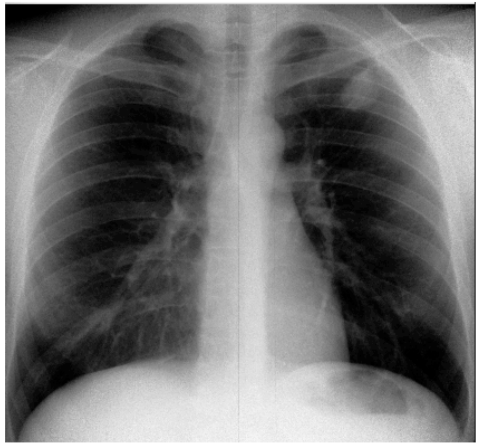


Data Pre-Processing Example: Lung Detection + Extraction

For lung pneumonia detection from xrays:

Pre-processing step: extract the lungs

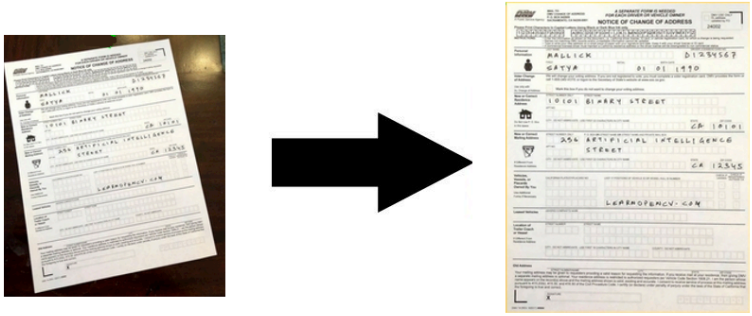




Data Pre-Processing Example: Document Detection, Extraction & Alignment

For optical character recognition from documents:

Pre-processing step:
extract and align the documents



Thank you for the attention

- **Structure**

One hour weekly lecture: pre-recorded, 09:00-10:00 Fridays

Half hour weekly Q&A: online, 10:00-10.30 Fridays

One and half hour weekly lab: online, 10:30-12:00 Fridays

from next week (12/2) there will be 2 parallel lab sessions

- **Assessment**

No Logbook

Only Practical Coursework (100%)