esade

Intermediate Python & a Glimpse into AI applications

Class 2

Pepe Bonet Giner

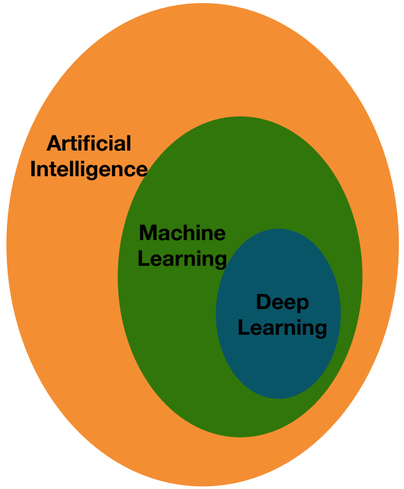
10th January 2024

Index Class 2

esade

Topic 1: Recap

Recap Class 1



- AI: Effort to automate intellectual tasks normally performed by humans
- AI is changing the world.
- Will we adapt?
- Writing code is at the heart of this revolution, i.e. programming computers
- Python is the preferred programming language nowadays

Recap Class 1

- Jupyter Notebooks
- Variables
- Lists
- Dictionaries
- If-else
- For loops
- While Loops
- Functions

Notebook lessons

1.- Load a csv/tsv/excel file and explore it

2.- Basic Data Exploration

3.- Column & row operations

Comments Assignments

Comment your code

Substract 2 from list numbers

```python
# Create a list of numbers
numbers = [1, 2, 3, 4, 5]

# Create a new list to store the results
results = []

# Iterate over the numbers in the list
for number in numbers:
    # Subtract 2 from each number and append the result to the results list
    results.append(number - 2)
```

Break up the JN into sections
Use bold levels

```
# Hello
## I
### am
#### Pepe
```

Hello

I

am

Pepe

Class 1.2. Intermediate Python & AI

Pandas

Table of contents

1. Load dataset
2. DataFrame exploration
3. Operations with Rows & Columns
4. Groupby
5. Apply
6. Lambda

Comments Assignments

- Do df.head() and not df to show fewer rows. Make your Jupyter clean
- Comment code on top of the line of code
- Show the results of the code. Especially if you send an HTML (run cells)
- Incentives Example. It is not all about the exercises.

esade

Topic 2: Pandas

Groupby

- The groupby operation can be used to group large amounts of data and compute operations on these groups separately.

```
gk = df.groupby('Day_Of_Week')
gk.get_group('Saturday')
```

	Rank	Name	Age	End_Of_Watch	Day_Of_Week
1	Sheriff	Cornelius Hogeboom	53.0	1791-10-22	Saturday
3	Marshal	Robert Forsyth	40.0	1794-01-11	Saturday
9	Deputy Sheriff	John A. Gooch	NaN	1807-03-07	Saturday

```
for group, df_group in df.groupby(['Day_Of_Week']):
    print(group, round(df_group['Age'].mean(), 2))
```

```
Friday 40.33
Monday 40.32
Saturday 39.89
Sunday 39.79
Thursday 40.36
Tuesday 40.53
Wednesday 40.89
```

Exercise

- What is the day of the week when more crimes happen? Can you put the number in percentage?

Apply

- Apply allows you to implement a function along an axis (rows or columns) of the DataFrame.
- Functions can be custom or already defined (mean, max, min, etc…)

Defined Functions

```
import numpy as np
```

```
df['Age'].apply(np.sqrt)
```

```
0         NaN
1    7.280110
2         NaN
3    6.324555
4         NaN
```

Custom Functions

```
def age_ranges(x):
    Fill this in!
```

```
df['Age Range'] = df['Age'].apply(age_ranges)
```

Exercise

- Build the *age ranges* function and apply it to the age column creating a new column.

Lambda

- An anonymous function that we can pass in instantly without defining a name or anything like a traditional function.
- We easily create columns with df.assign
- Lambda allows us to make quick operations with columns

Getting the corrected age column in one go

```
df = df.assign(Corrected_age_2 = lambda x: x['Age'] - 2)
```

Exercise

- Build a new column called life_percentage that puts the age column into a life percentage in this way: df[‘Age’] / 80 * 100

Groupby + apply + Lambda Function

- Combining all of them gives us an easy way to go through the data
- "split-apply-combine" operation:
    - Splitting the data into groups based on some criteria
    - Applying a function to each group independently
    - Combining the results into a data structure

```
df.groupby('Day_Of_Week').apply(lambda x: x['Age Range'].value_counts())
```

Day_Of_Week	Age Range	Older than 24 but younger than 45	Older than 44	Younger than 25
Friday		2009	1550	181
Monday		1840	1482	193
Saturday		2188	1610	204
Sunday		2042	1507	216
Thursday		1959	1483	178
Tuesday		1882	1460	175
Wednesday		1856	1424	184

Exercise

- How many different ranks are there if we group by day of the week and age range?

Concat

- The concat() function is used to concatenate two or more DataFrames along a particular axis.

```python
import pandas as pd

# Creating two sample DataFrames
df1 = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
df2 = pd.DataFrame({'A': [4, 5, 6], 'B': [7, 8, 9]})

# Concatenating the DataFrames vertically
result = pd.concat([df1, df2])

# Printing the concatenated DataFrame
print(result)
```

	A	B
0	1	4
1	2	5
2	3	6
0	4	7
1	5	8
2	6	9

```python
import pandas as pd

# Creating two sample DataFrames
df1 = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
df2 = pd.DataFrame({'C': [7, 8, 9], 'D': [10, 11, 12]})

# Concatenating the DataFrames horizontally
result = pd.concat([df1, df2], axis=1)

# Printing the concatenated DataFrame
print(result)
```

	A	B	C	D
0	1	4	7	10
1	2	5	8	11
2	3	6	9	12

Merge

- The merge() function is used to merge two or more DataFrames based on a common column (or index) between them.

```
df1 = pd.DataFrame({'key': ['foo', 'bar', 'baz'], 'value': [1, 2, 3]})
df1
```

✓ 0.5s

	key	value
0	foo	1
1	bar	2
2	baz	3

```
df2 = pd.DataFrame({'key': ['foo', 'bar', 'qux'], 'value': [4, 5, 6]})
df2
```

✓ 0.5s

	key	value
0	foo	4
1	bar	5
2	qux	6

```
# Merging the DataFrames based on the 'key' column
result = pd.merge(df1, df2, on='key')

# Printing the merged DataFrame
print(result)
```

✓ 0.3s

	key	value_x	value_y
0	foo	1	4
1	bar	2	5

Exercise

- Start two dataframes and play around with them using concat and merge operations.

Save your DataFrame

- At some point when you finish writing part of the code you may want to save your changes to the dataframe as a new file.
- Having a gigantic JN file that does all the work is not a good practice
- Is better to start a new JN to keep doing other stuff to the latest version of your data

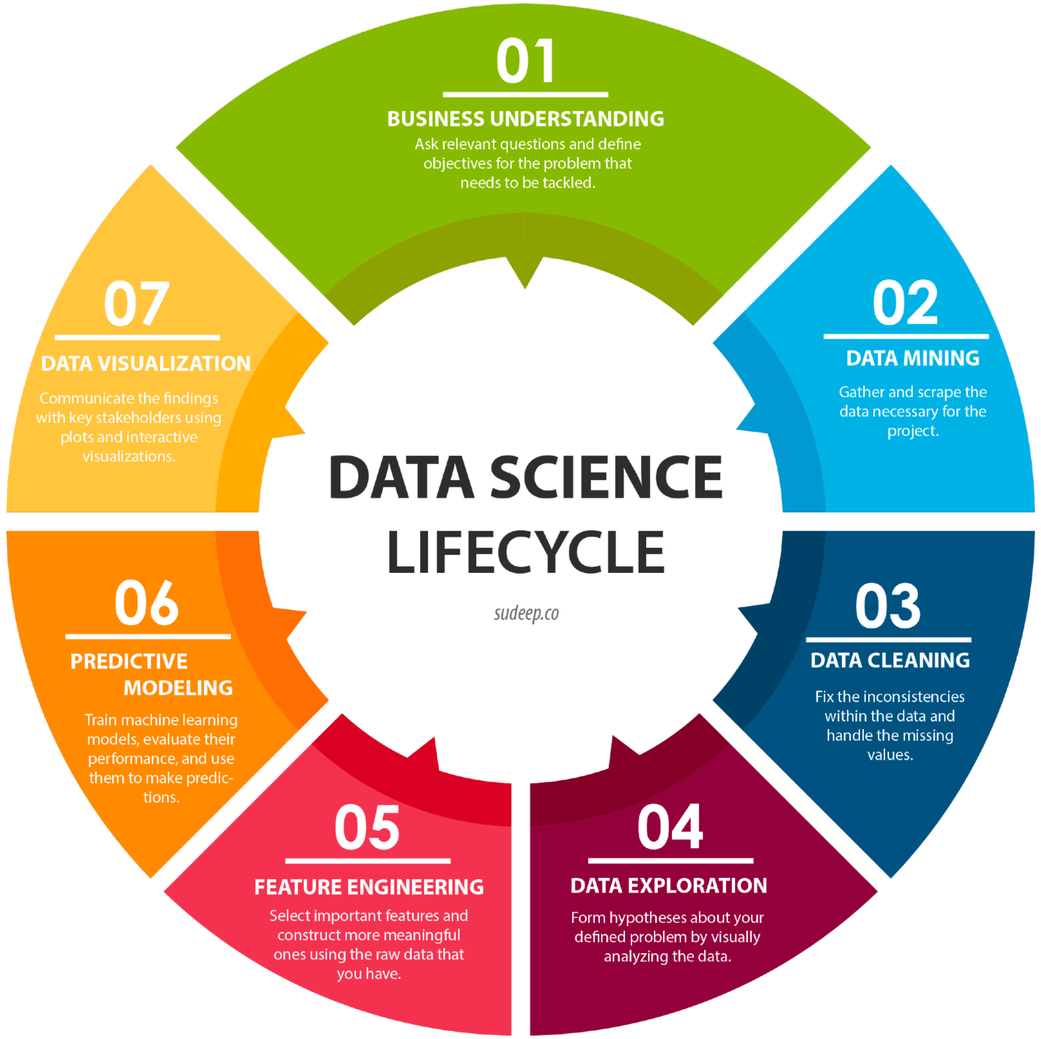
```
# Creating a sample DataFrame
data = {'name': ['Alice', 'Bob', 'Charlie', 'David'],
        'age': [25, 30, 35, 40],
        'city': ['New York', 'London', 'Paris', 'Tokyo']}
df = pd.DataFrame(data)

# Saving the DataFrame to a CSV file
df.to_csv('output.csv', index=None, sep=',')
```

esade

Topic 3: The Data Project Cycle

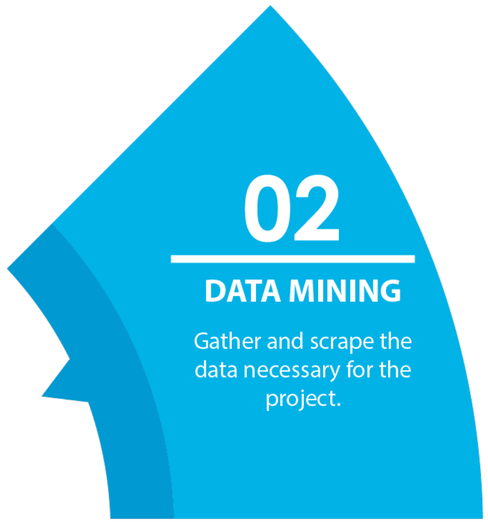
The Data Science Lifecycle



Business Understanding

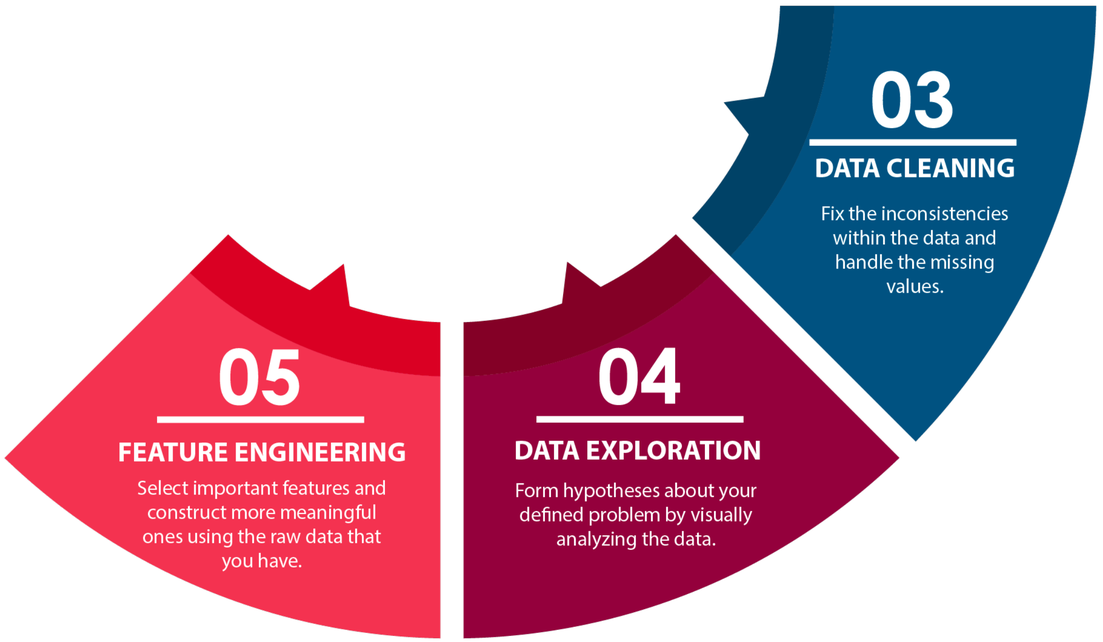
- Everything starts with the Business / Data understanding:
  - Understand what you want to do
  - Why do you want to do it
  - The outputs. Do they make sense?

Data Mining



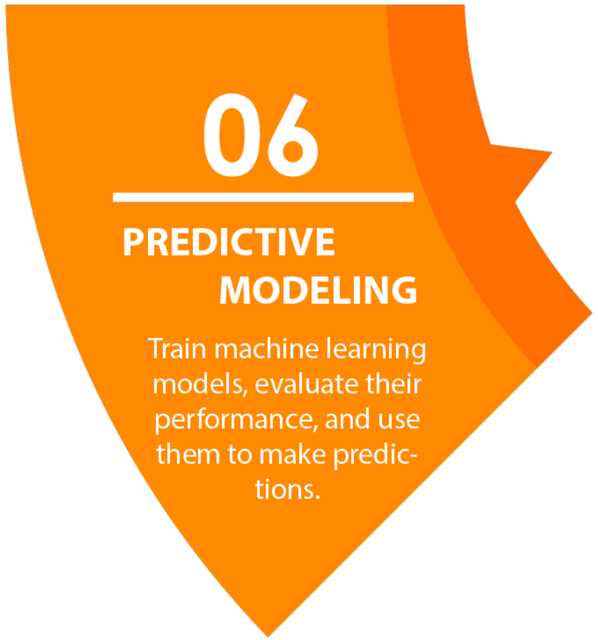
- How do you acquire your data?
- What is the data origin:
  - Files
  - Data Bases
  - API Services
  - Others

Data Preparation: Lectures 2, 3, 4, 5



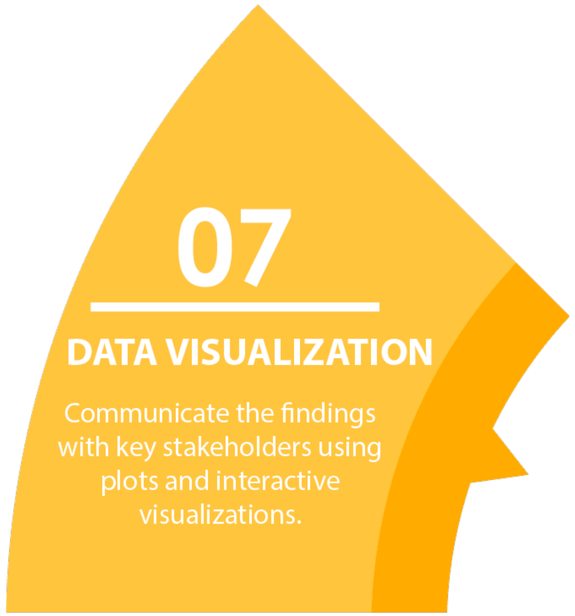
- It implies up to 80 % of the whole process
- Key to extracting value from your data
- Correct errors, remove duplicates and handle missing values (Data Cleaning)
- Exploratory Data Analysis (EDA). Data Understanding (Data Exploration)
- Transform data (Feature Engineering)

Predictive Modeling: Lecture 4, 5



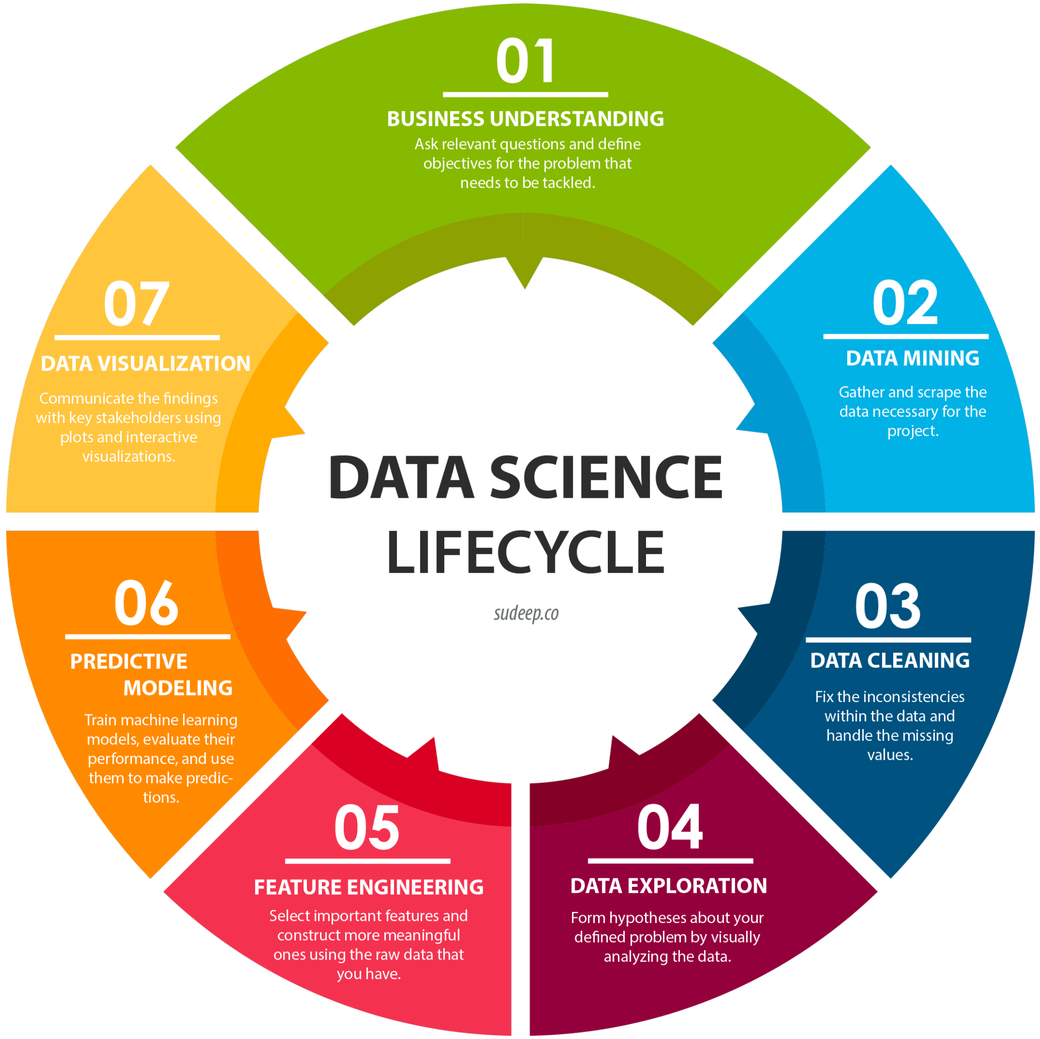
- Selection, training, evaluation, and testing of AI or machine learning models
- Build models that are trained on data from the past to make predictions about the future
- We will go slightly over it and we will try to get an overall understanding with Python

Data Visualization: Lecture 2, 3, 4



- Numbers on their own are difficult to understand
- Correctly visualizing the results of any data cycle is crucial
- Plots need to tell a story of your understanding on the data

The Data Science Lifecycle



- Iterative and experimental process which can be repeated many times until meaningful results are obtained

esade

Topic 4: Data Preparation - Data Cleaning

Data Cleaning

- Step 1: Remove irrelevant data
- Step 2: Deduplicate your data
- Step 3: Fix structural errors
- Step 4: Deal with missing data
- Step 5: Filter out data outliers
- Step 6: Validate your data

Exercise

- Load exams dataset in Jupyter Notebook

Remove Irrelevant Data

- What questions do you want to answer or problems do you want to solve?
- What is relevant and what you may not need?
- Filter out data or observations that aren't relevant to your downstream needs.

Look at the columns Year:

```
df['Year'].value_counts()
```

```
2022    953
1970     47
Name: Year, dtype: int64
```

The entries on year 1970 were introduced by mistake

and we need to take them out of the dataset

```
new_df = df[df['Year'] != 1970]
```

```
new_df.shape
```

```
(953, 12)
```

Exercise

- What if race/ethnicity group A would be irrelevant to the study? Could you build a new dataframe?

Deduplicate your data

- Duplicate records slow down analysis and require more storage
- If you train a machine learning model on a dataset with duplicate results, the model will likely give more weight to the duplicates

```
df.shape
```

```
(963, 12)
```

```
df.drop_duplicates(inplace=True)
```

```
df.shape
```

```
(953, 12)
```

Exercise

- Deduplicate your data as shown in the previous slide

Fix Structural Errors

- We are searching for misspellings, incongruent naming conventions, improper capitalization, incorrect word use…
- Computers can not detect these errors as easy as humans

```
df['gender'].value_counts()
```

```
male      360
female    317
FEMALE     99
MALE       91
Male       52
Female     44
Name: gender, dtype: int64
```

```
df['gender'] = df['gender'].str.lower()
df['gender'].value_counts()
```

```
male      503
female    460
Name: gender, dtype: int64
```

Exercise

- The parental education level seems to have repeated degrees for high school. Can you make only one name for some high school and high school?

Deal with Missing values

- You need to determine whether everything connected to this missing data – an entire column or row, etc. – should be:
    - Discarded
    - Individual cells entered manually (Imputation)
    - Left as it is.
- As a personal rule: If a column has more than 60 % of missing values, it has the potential to be deleted

```
df.isna().sum()
```

```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
id_student                     0
address                      963
Year                           0
Age                          102
dtype: int64
```

```
df.drop(['address'], axis=1, inplace=True)
```

Missing value imputation

- Filling in the missing values is not a trivial task. You can fill them in with:
  - The mean
  - The median
  - Drawing from a probability distribution
- At this point, it is the understanding of the data and the following analysis that matters

```
df.drop(['address'], axis=1, inplace=True)
```

```
df['Age'].mean()
```

```
17.0
```

```
df['Age'].fillna(df['Age'].mean(), inplace=True)
```

```
df.isna().sum()
```

```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
id_student                     0
Year                           0
Age                            0
dtype: int64
```

Exercise

- Delete the address column and fill in the missing values of the column age with the mean of the column.

Deal with data outliers

- Outliers are data points that fall far outside of the norm and may skew your analysis too far in a certain direction
- However, the fact that outliers exist, doesn't mean that they shouldn't be considered

```
df.describe()
```

	math score	reading score	writing score	id_student	Year	Age
count	963.000000	963.000000	963.000000	963.000000	963.0	963.0
mean	66.481828	68.962617	67.681205	1505.058152	2022.0	17.0
std	15.485236	14.772377	15.573150	292.310206	0.0	0.0
min	13.000000	27.000000	23.000000	1000.000000	2022.0	17.0
25%	55.500000	59.000000	58.000000	1254.500000	2022.0	17.0
50%	67.000000	70.000000	68.000000	1505.000000	2022.0	17.0
75%	77.000000	79.000000	79.000000	1758.500000	2022.0	17.0
max	120.000000	100.000000	100.000000	2009.000000	2022.0	17.0

```
df[df['math score'] > 100]
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score
11	female	group D	high school	standard	completed	120

Deal with data outliers. Outlier Negative Effect Example

SQUARE METERS	PRICE
60	280.000 €
150	450.000 €
55	1.000.000 €
80	300.000 €
90	290.000 €
110	430.000 €
120	420.000 €

SQUARE METERS	PRICE
60	280.000 €
55	1.000.000 €
80	300.000 €
90	290.000 €
71,25	**467.500 €**

SQUARE METERS	PRICE
150	450.000 €
110	430.000 €
120	420.000 €
126,67	**433.334 €**

Deal with data outliers. Outlier Negative Effect Example

SQUARE METERS	PRICE
60	280.000 €
150	450.000 €
55	1.000.000 €
80	300.000 €
90	290.000 €
110	430.000 €
120	420.000 €

SQUARE METERS	PRICE	SQUARE METERS	PRICE
60	280.000 €	150	450.000 €
55	1.000.000 €	110	430.000 €
80	300.000 €	120	420.000 €
90	290.000 €		
Mean	**71,25**	**467.500 €**	**126,67**	**433.334 €**

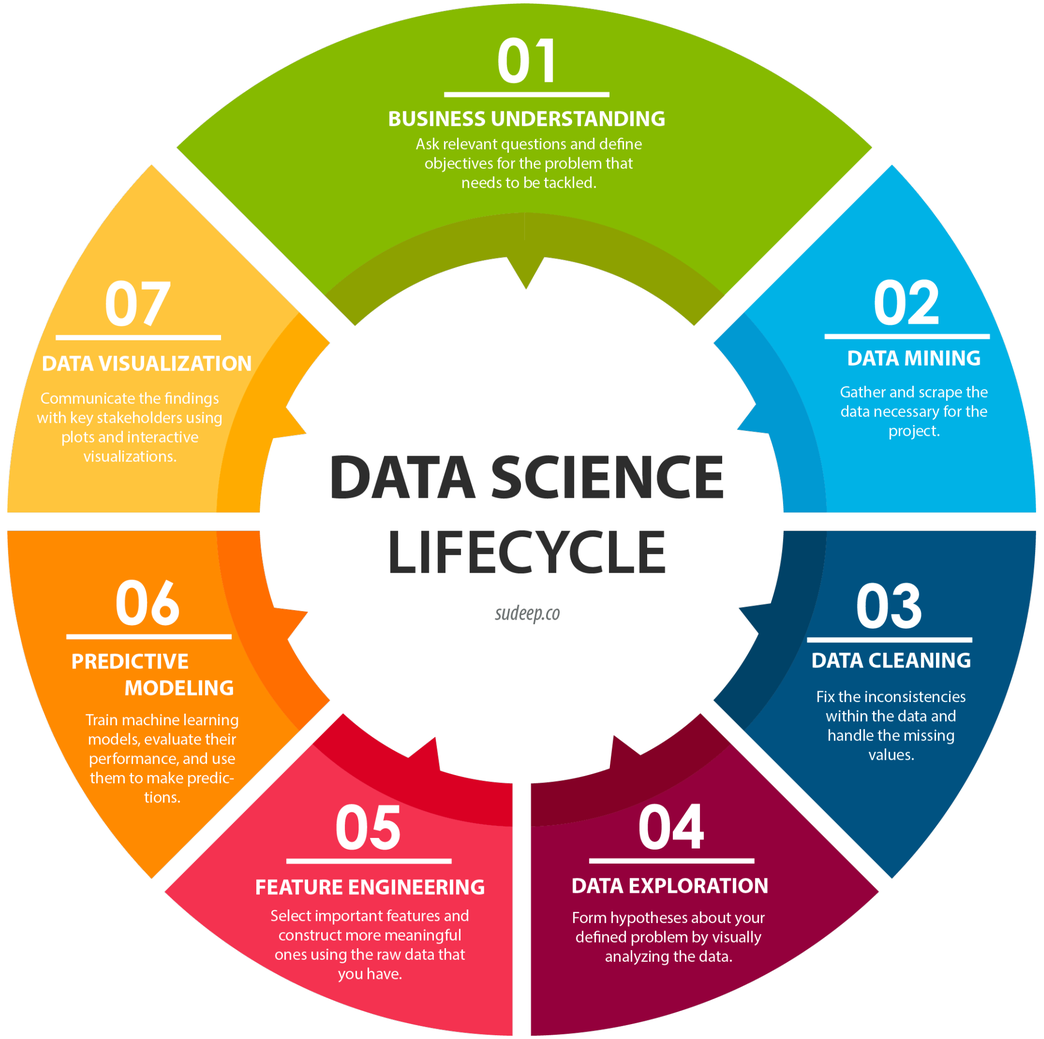
SQUARE METERS	PRICE	SQUARE METERS	PRICE
60	280.000 €	150	450.000 €
80	300.000 €	110	430.000 €
90	290.000 €	120	420.000 €
Mean	**77**	**290.000 €**	**127**	**433.334 €**

Exercise

- Download the same dataset again (updated version) and 1970 is not present there anymore and the year is 1990. Do that and run the previous work again.
- Filter out the row that contains a mistaken grade for the math score

Validate your data

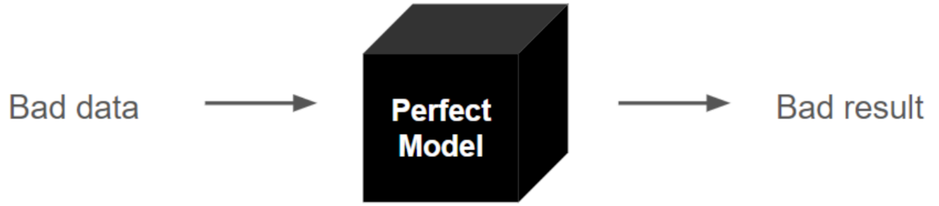
- Do you have enough data for your needs?
- Is it uniformly formatted in a design or language that your analysis tools can work with?
- This is an iterative process you may need to go back and fix errors that you find later on. Experience plays a role here.



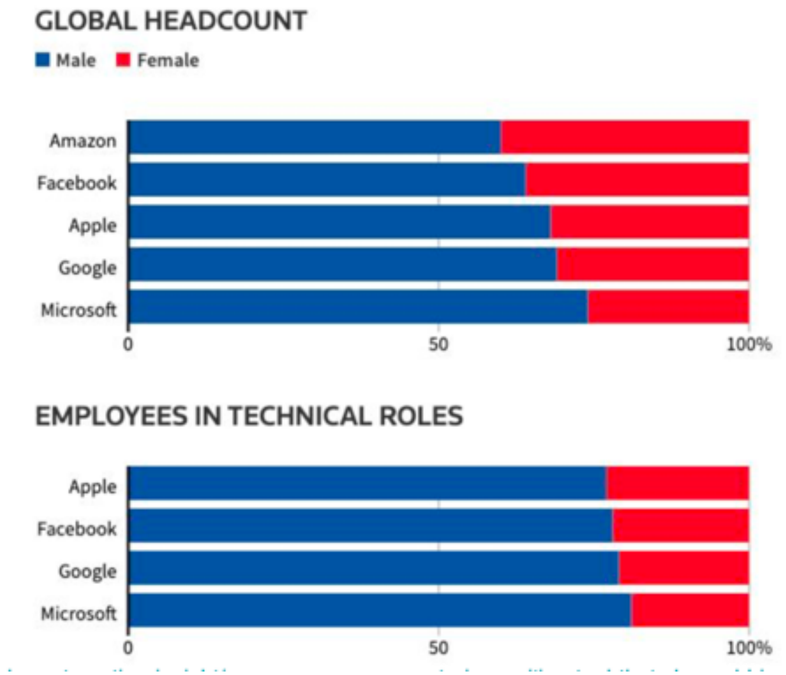
esade

Topic 5: Data Preparation - Data Exploration

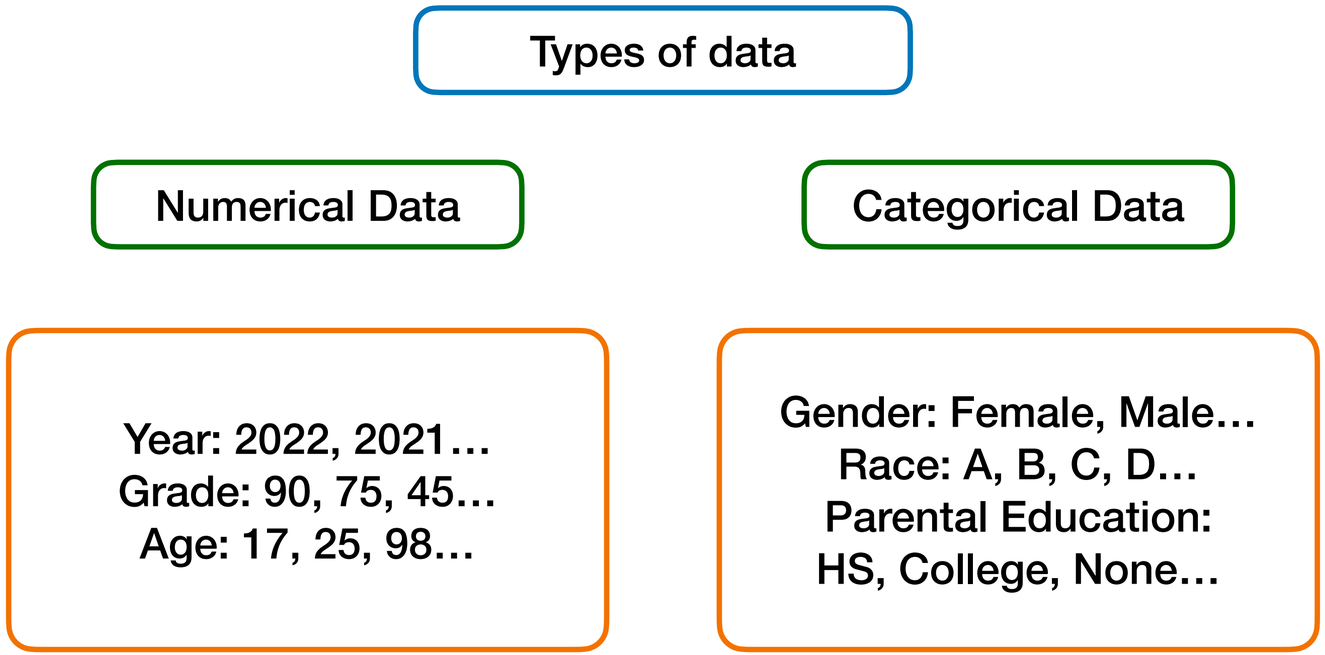
Data Exploration - Exploratory data analysis (EDA)



- A process where users look at and understand their data with statistical and visualization methods.
- Identify patterns and problems in the dataset
- Decide which model or algorithm to use in subsequent steps.



Preliminary data processing



Preliminary data processing

- Changes to the types of data of a column, i.e.:
    - Numerical to a string: int —> str
    - String to a date time: str —> datetime

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 962 entries, 0 to 1009
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       962 non-null    object
 1   race/ethnicity               962 non-null    object
 2   parental level of education  962 non-null    object
 3   lunch                        962 non-null    object
 4   test preparation course      962 non-null    object
 5   math score                   962 non-null    int64
 6   reading score                962 non-null    int64
 7   writing score                962 non-null    int64
 8   id_student                   962 non-null    int64
 9   Year                         962 non-null    int64
 10  Age                          962 non-null    float64
dtypes: float64(1), int64(5), object(5)
memory usage: 90.2+ KB
```

Exercise

- Change the data type of the id_student column from int to str

EDA steps

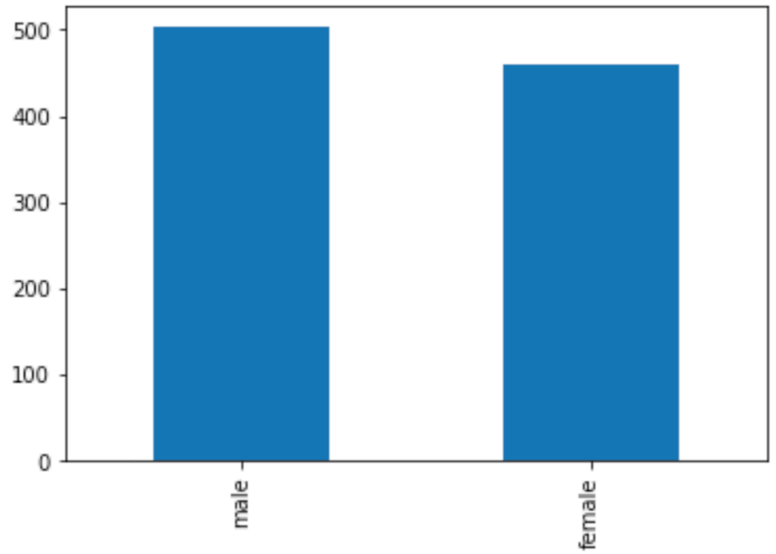
- Categorical EDA
- Numerical EDA
    - Univariate analysis
    - Bivariate analysis (Categorical + Numerical)
    - Multivariate analysis

Categorical EDA

- Initial step: Explore group counts of categorical columns

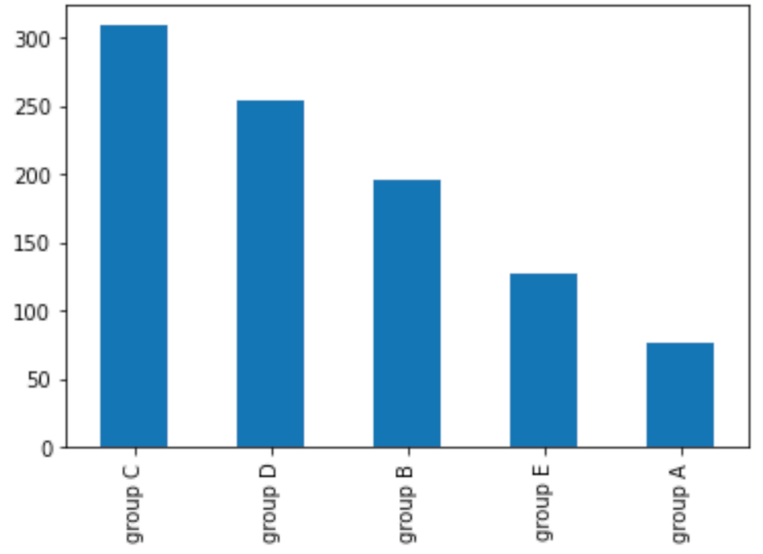
```
df['gender'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```




```
df['race/ethnicity'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```



Numerical EDA

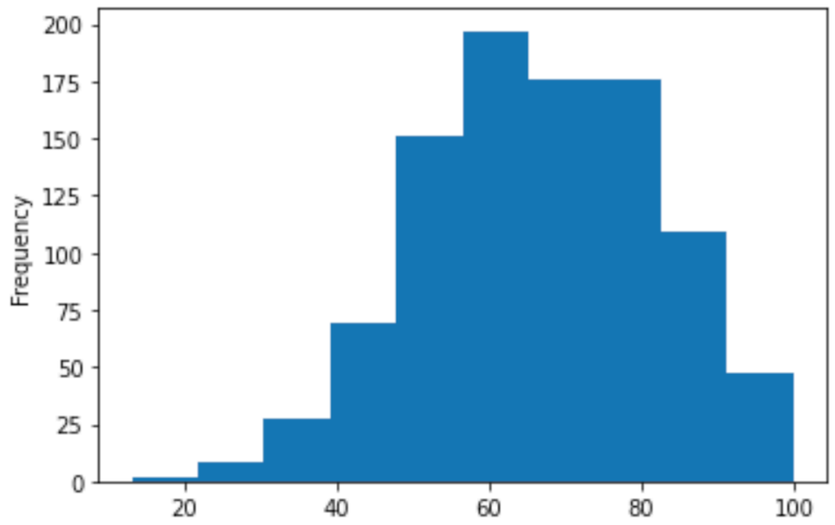
- How is the numerical data distributed?
- Can we find anything interesting?

```
df.describe()
```

	math score	reading score
count	962.000000	962.000000
mean	66.426195	68.943867
std	15.396706	14.768592
min	13.000000	27.000000
25%	55.250000	59.000000
50%	67.000000	70.000000
75%	77.000000	79.000000
max	100.000000	100.000000

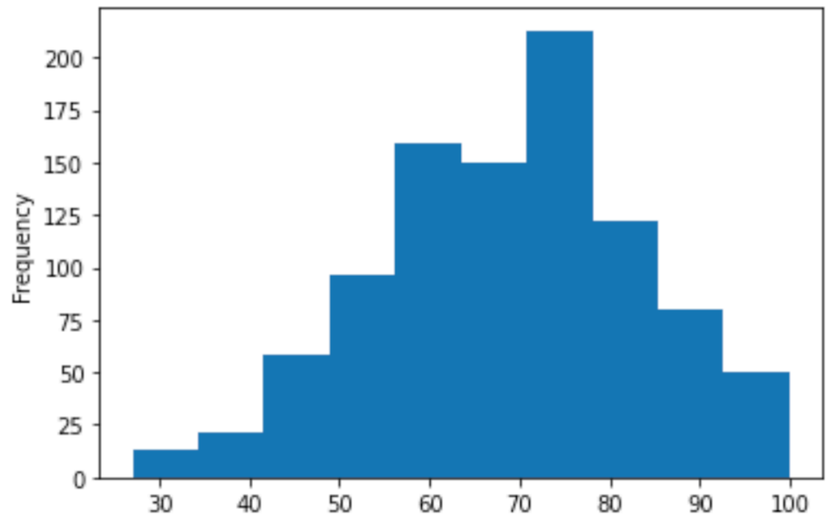
```
df['math score'].plot(kind='hist')
```

```
<AxesSubplot:ylabel='Frequency'>
```




```
df['reading score'].plot(kind='hist')
```

```
<AxesSubplot:ylabel='Frequency'>
```



Exercise

- Get a bar plot that shows the number of students that completed the test preparation course and those who did not do it.
- Plot the histogram of the writing scores. Do you see anything interesting?

esade

Documentation

Datacamp Courses

Python Fundamentals

Are you ready to gain the foundational skills you need to become a Python programmer? In this track, you'll learn the Python basics you need to start on your programming journey, including how to clean real-world data ready for analysis, use data visualization libraries, and even how to write your own Python functions.

Your instructor Hugo will introduce you to how companies worldwide use Python to gain a competitive edge. Through hands-on coding exercises you'll then learn how to store, manipulate, and explore data using NumPy. Then it's time to level-up as you learn how to visualize your data using Matplotlib, manipulate DataFrames and dictionaries using pandas, and write your own functions and list comprehension. Start this track to add these essential Python skills to your data science toolbox.

Switch Track

Python 15 hours 4 Courses 1 Skill Assessment

Data Scientist with Python

Gain the career-building Python skills you need to succeed as a data scientist. No prior coding experience required.

In this track, you'll learn how this versatile language allows you to import, clean, manipulate, and visualize data—all integral skills for any aspiring data professional or researcher. Through interactive exercises, you'll get hands-on with some of the most popular Python libraries, including pandas, NumPy, Matplotlib, and many more. You'll then work with real-world datasets to learn the statistical and machine learning techniques you need to train decision trees and use natural language processing (NLP). Start this track, grow your Python skills, and begin your journey to becoming a confident data scientist.

Resume Track

Python 88 hours 23 Courses 6 Projects 3 Skill Assessments

Free Books to learn Python & data analytics