esade

Intermediate Python & a Glimpse into AI applications

Class 3

Pepe Bonet Giner

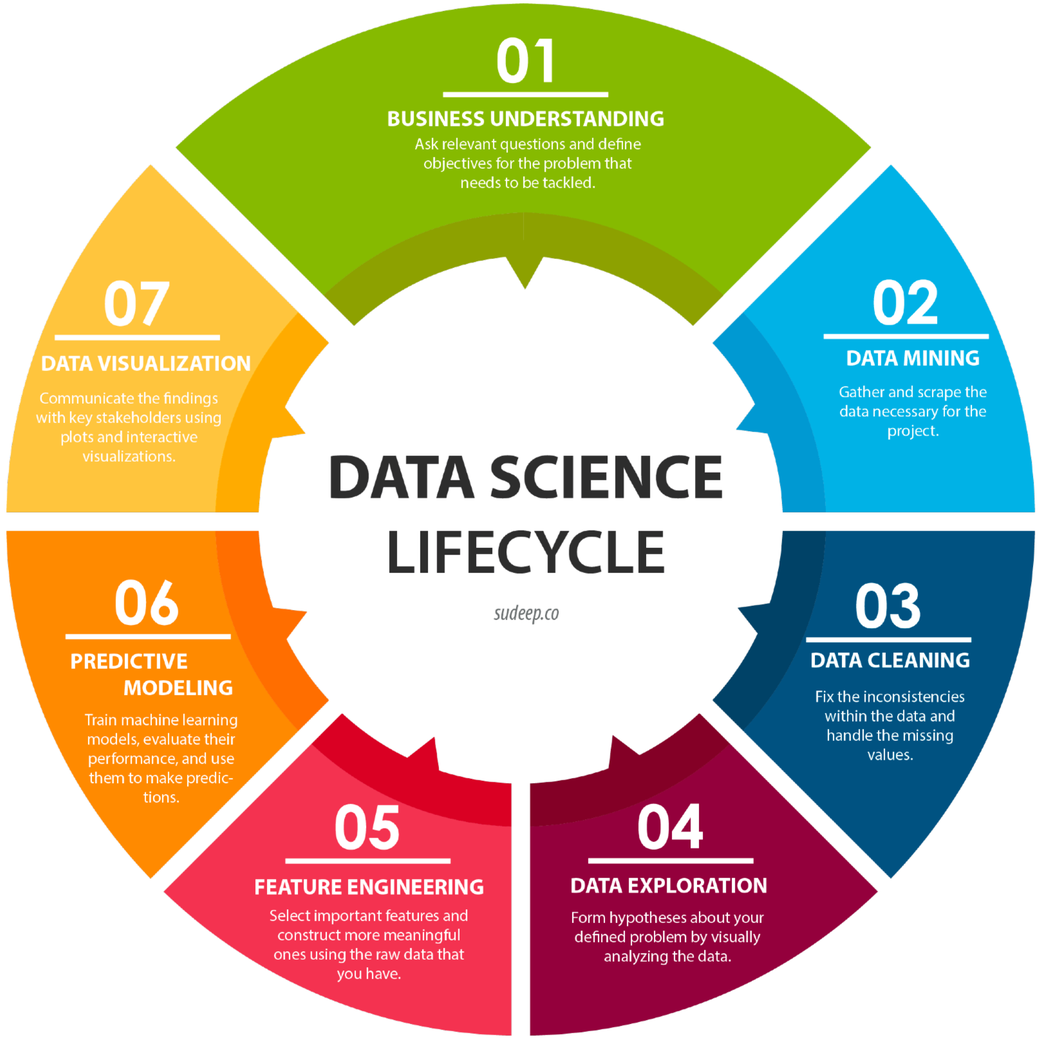
13th January 2023

Index Class 3

esade

Topic 1: Recap

Recap Class 2: The Data Science Lifecycle



Recap Class 2: 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

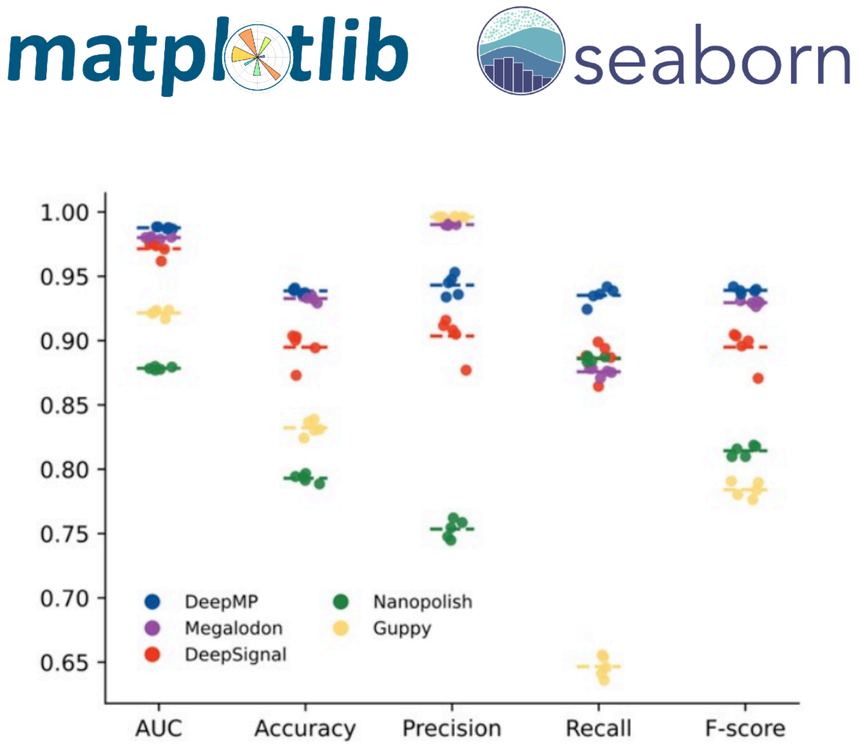
Data Exploration: EDA steps

- A process where users look at and understand their data with statistical and visualization methods
- Preliminary Data Processing
- Categorical EDA
- Numerical EDA

esade

Topic 2: Matplotlib & Seaborn

Matplotlib & Seaborn



- Both are Python libraries to produce figures
- To explore the data —> visualizations of your results are needed
- Seaborn is built on top of matplotlib. It provides an easy-to-use interface for drawing attractive and informative graphics (Also from pandas dataframes)

Matplotlib building blocks

Load Libraries

```
import matplotlib.pyplot as plt
import seaborn
```

Fully customizable

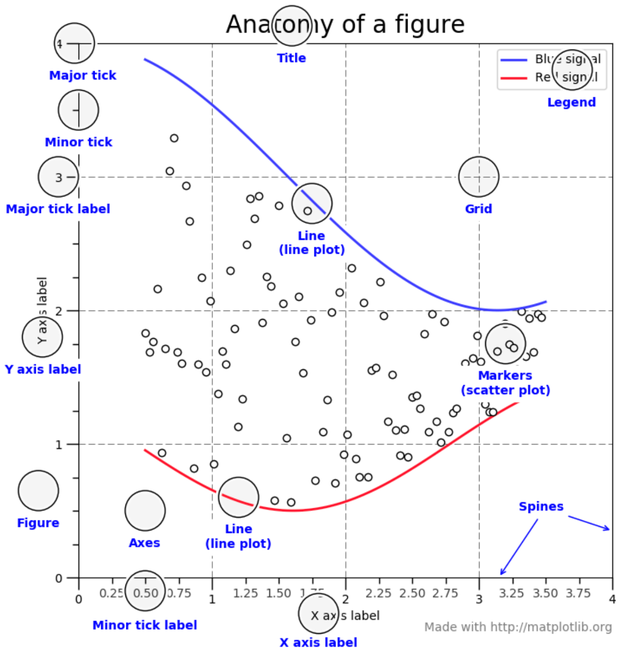
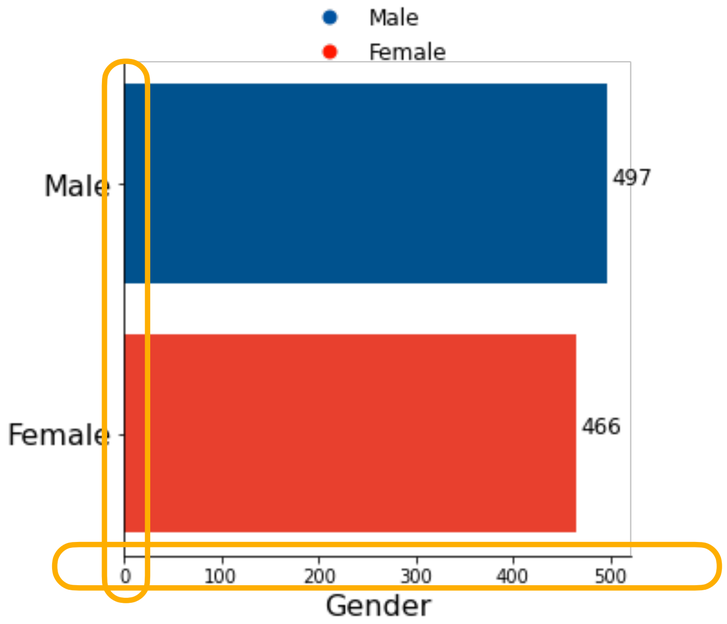


Figure structure

```
#Start Figure
fig, ax = plt.subplots(figsize=(5, 5))

#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
            palette=['#08519c', '#f03b20'])

#Change Axes
ax.set_xlabel("Gender", fontsize=12)
ax.set_ylabel("", fontsize=12)
ax.set_yticklabels(['Male', 'Female'])
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Add Numbers to plot
for index, row in to_plot.iterrows():
    ax.text(row.gender + 25, index, row.gender,
             color='black', ha="center", fontsize=12)

## Add legend
custom_lines = []
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:
    custom_lines.append(
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',
        mew=0, color=el[1], label=el[0])[0]
    )
ax.legend(
    bbox_to_anchor=(0., 1.05, 1., .102),
    handles=custom_lines, loc='upper center',
    facecolor='white', ncol=1, fontsize=10, frameon=False
    )
#Save or show
plt.show()
```

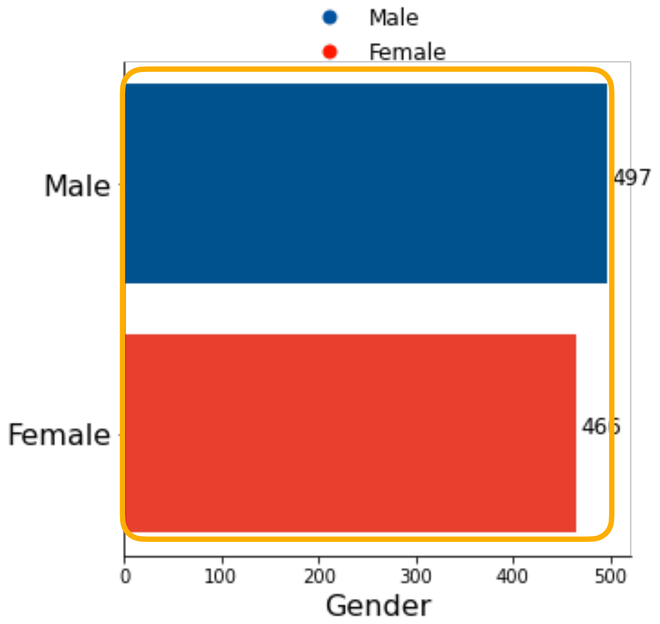
Code for figure

```
#Start Figure
fig, ax = plt.subplots(figsize=(5, 5))

#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
            palette=['#08519c', '#f03b20'])

#Change Axes
ax.set_xlabel("Gender", fontsize=16)
ax.set_ylabel("", fontsize=16)
ax.set_yticklabels(['Male', 'Female'], fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Add Numbers to plot
for index, row in to_plot.iterrows():
    ax.text(row.gender + 25, index, row.gender,
             color='black', ha="center", fontsize=12)

## Add legend
custom_lines = []
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:
    custom_lines.append(
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',
        mew=0, color=el[1], label=el[0])[0]
    )
ax.legend(
    bbox_to_anchor=(0., 1.05, 1., .102),
    handles=custom_lines, loc='upper center',
    facecolor='white', ncol=1, fontsize=12, frameon=False
    )
#Save or show
plt.show()
```



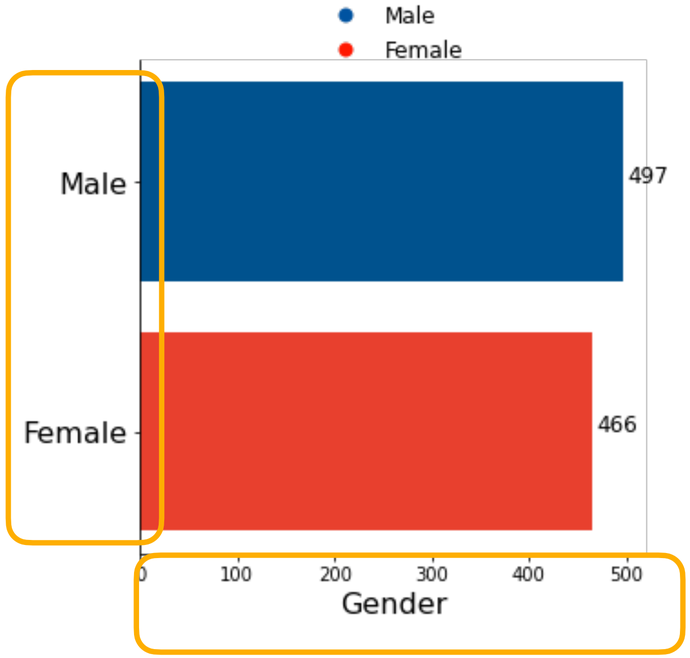
Code for figure

```
#Start Figure
fig, ax = plt.subplots(figsize=(5, 5))

#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
            palette=['#08519c', '#f03b20'])

#Change Axes
ax.set_xlabel("Gender", fontsize=16)
ax.set_ylabel("", fontsize=16)
ax.set_yticklabels(['Male', 'Female'], fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Add Numbers to plot
for index, row in to_plot.iterrows():
    ax.text(row.gender + 25, index, row.gender,
             color='black', ha="center", fontsize=12)

## Add legend
custom_lines = []
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:
    custom_lines.append(
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',
        mew=0, color=el[1], label=el[0])[0]
    )
ax.legend(
    bbox_to_anchor=(0., 1.05, 1., .102),
    handles=custom_lines, loc='upper center',
    facecolor='white', ncol=1, fontsize=12, frameon=False
    )
#Save or show
plt.show()
```



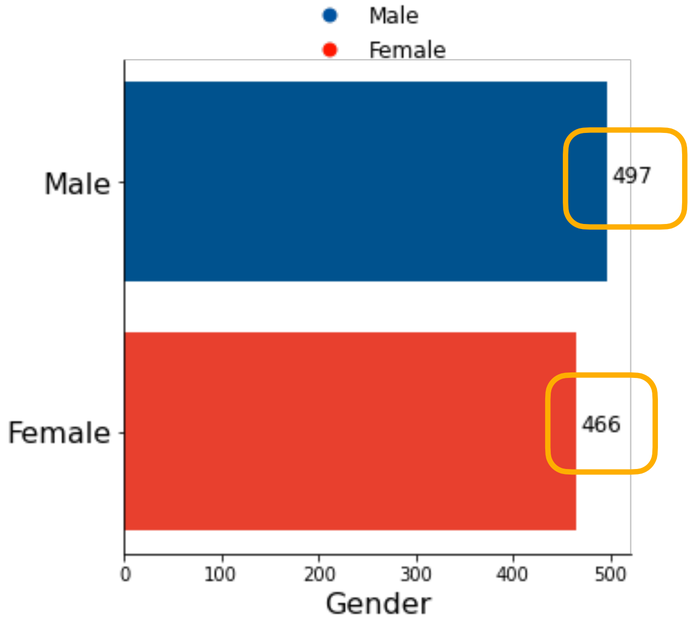
Code for figure

```
#Start Figure
fig, ax = plt.subplots(figsize=(5, 5))

#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
            palette=['#08519c', '#f03b20'])

#Change Axes
ax.set_xlabel("Gender", fontsize=16)
ax.set_ylabel("", fontsize=16)
ax.set_yticklabels(['Male', 'Female'], fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Add Numbers to plot
for index, row in to_plot.iterrows():
    ax.text(row.gender + 25, index, row.gender,
             color='black', ha="center", fontsize=12)

## Add legend
custom_lines = []
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:
    custom_lines.append(
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',
        mew=0, color=el[1], label=el[0])[0]
    )
ax.legend(
    bbox_to_anchor=(0., 1.05, 1., .102),
    handles=custom_lines, loc='upper center',
    facecolor='white', ncol=1, fontsize=12, frameon=False
    )
#Save or show
plt.show()
```



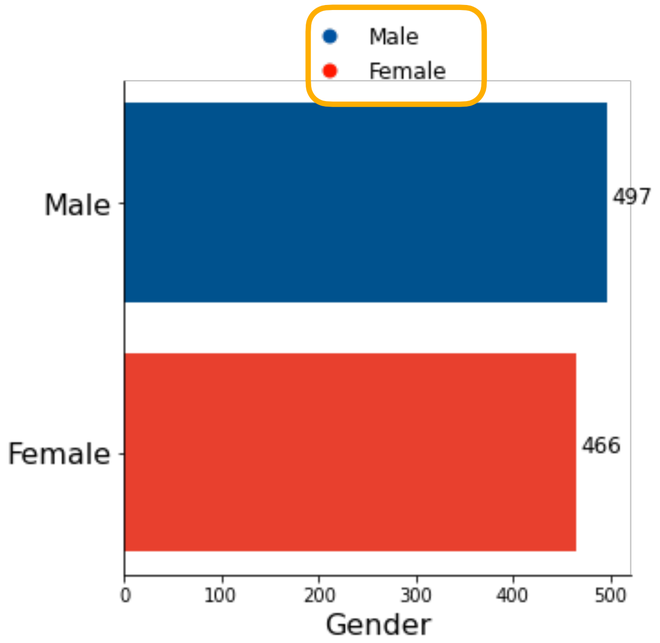
Code for figure

```python
#Start Figure
fig, ax = plt.subplots(figsize=(5, 5))

#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
            palette=['#08519c', '#f03b20'])

#Change Axes
ax.set_xlabel("Gender", fontsize=16)
ax.set_ylabel("", fontsize=16)
ax.set_yticklabels(['Male', 'Female'], fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Add Numbers to plot
for index, row in to_plot.iterrows():
    ax.text(row.gender + 25, index, row.gender,
             color='black', ha="center", fontsize=12)

## Add legend
custom_lines = []
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:
    custom_lines.append(
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',
        mew=0, color=el[1], label=el[0])[0]
    )
ax.legend(
    bbox_to_anchor=(0., 1.05, 1., .102),
    handles=custom_lines, loc='upper center',
    facecolor='white', ncol=1, fontsize=12, frameon=False
    )
#Save or show
plt.show()
```



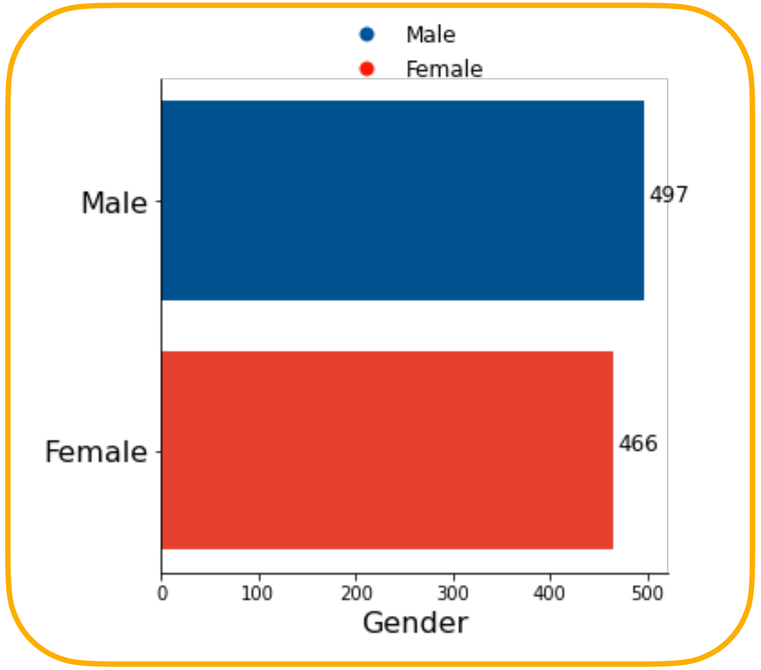
Code for figure

```python
#Start Figure
fig, ax = plt.subplots(figsize=(5, 5))

#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
            palette=['#08519c', '#f03b20'])

#Change Axes
ax.set_xlabel("Gender", fontsize=16)
ax.set_ylabel("", fontsize=16)
ax.set_yticklabels(['Male', 'Female'], fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Add Numbers to plot
for index, row in to_plot.iterrows():
    ax.text(row.gender + 25, index, row.gender,
             color='black', ha="center", fontsize=12)

## Add legend
custom_lines = []
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:
    custom_lines.append(
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',
        mew=0, color=el[1], label=el[0])[0]
    )
ax.legend(
    bbox_to_anchor=(0., 1.05, 1., .102),
    handles=custom_lines, loc='upper center',
    facecolor='white', ncol=1, fontsize=12, frameon=False
    )
#Save or show
plt.show()
```



Code for figure

```python
#Start Figure
fig, ax = plt.subplots(figsize=(5, 5))

#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
            palette=['#08519c', '#f03b20'])

#Change Axes
ax.set_xlabel("Gender", fontsize=16)
ax.set_ylabel("", fontsize=16)
ax.set_yticklabels(['Male', 'Female'], fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

# Add Numbers to plot
for index, row in to_plot.iterrows():
    ax.text(row.gender + 25, index, row.gender,
             color='black', ha="center", fontsize=12)

## Add legend
custom_lines = []
for el in [('Male', '#08519c'), ('Female', '#f03b20')]:
    custom_lines.append(
        plt.plot([],[], marker="o", ms=8, ls="", mec='black',
        mew=0, color=el[1], label=el[0])[0]
    )
ax.legend(
    bbox_to_anchor=(0., 1.05, 1., .102),
    handles=custom_lines, loc='upper center',
    facecolor='white', ncol=1, fontsize=12, frameon=False
    )
#Save or show
plt.show()
```

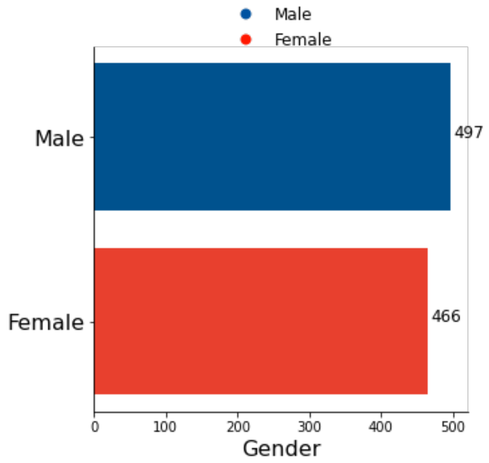


Barplot

- Graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent

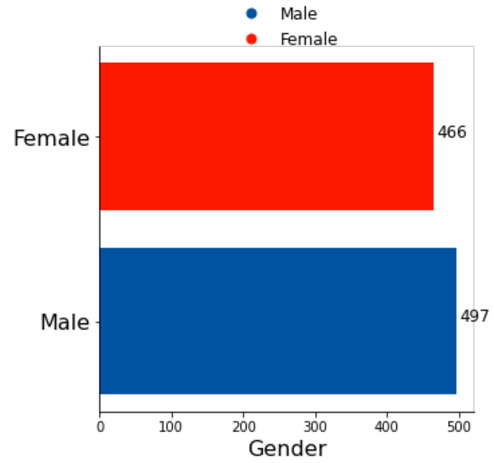
seaborn

```
#Body of the figure to build and the data to use
sns.barplot(x=to_plot['gender'], y=to_plot['index'],
            palette=['#08519c', '#f03b20'])
```



matplotlib

```
#Body of the figure to build and the data to use
plt.barh(width=to_plot['gender'], y=to_plot['index'],
         color=['#08519c', '#f03b20'])
```

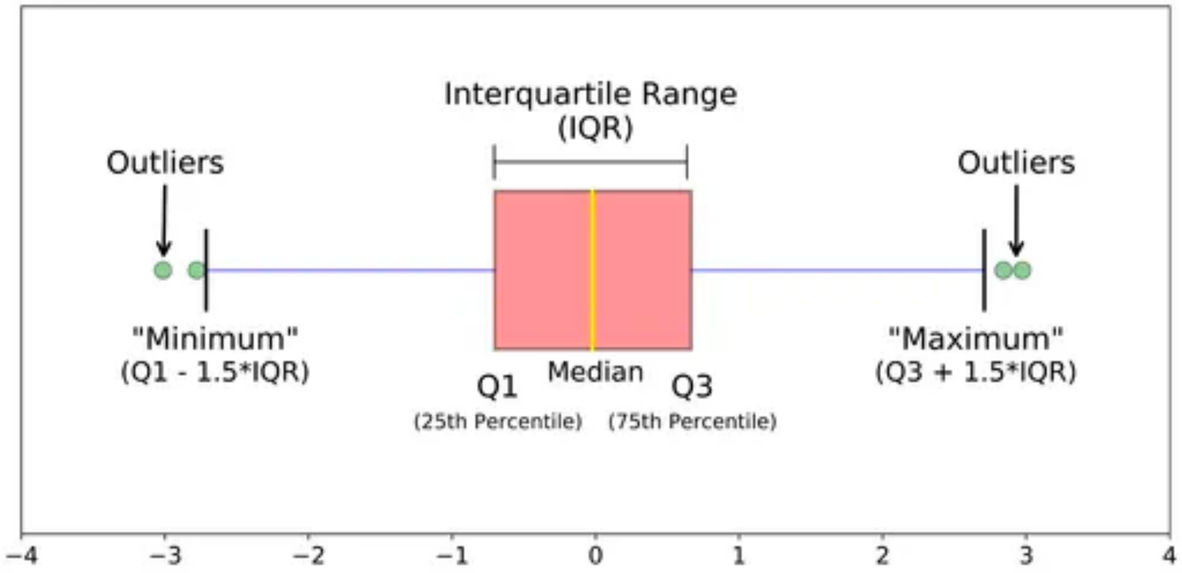


Exercise

- Write the barpot Python code for a categorical column that is not gender

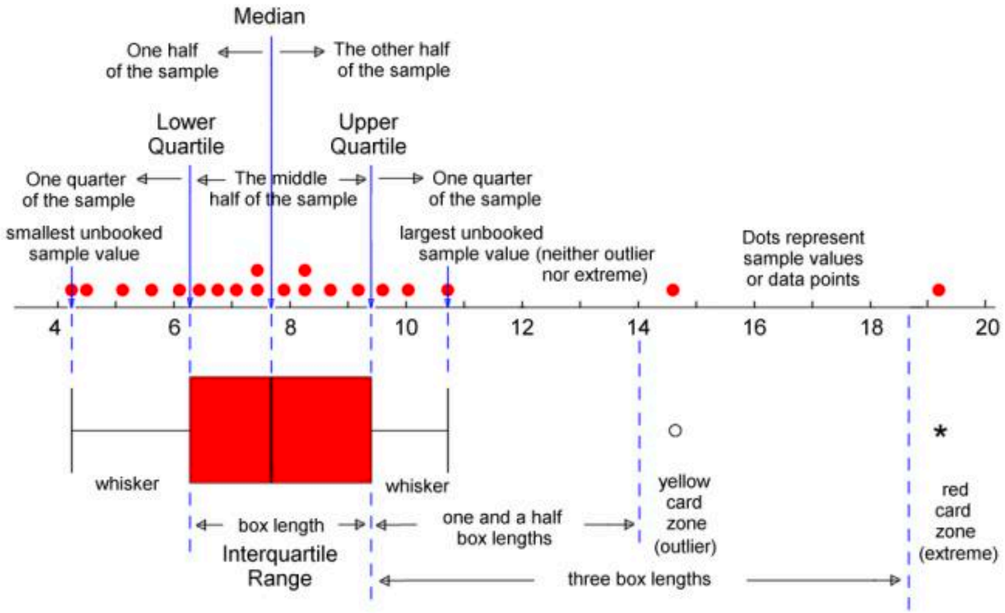
Boxplot

- 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”)
- Graphically demonstrates the locality, spread, and skewness groups of numerical data through their quartiles



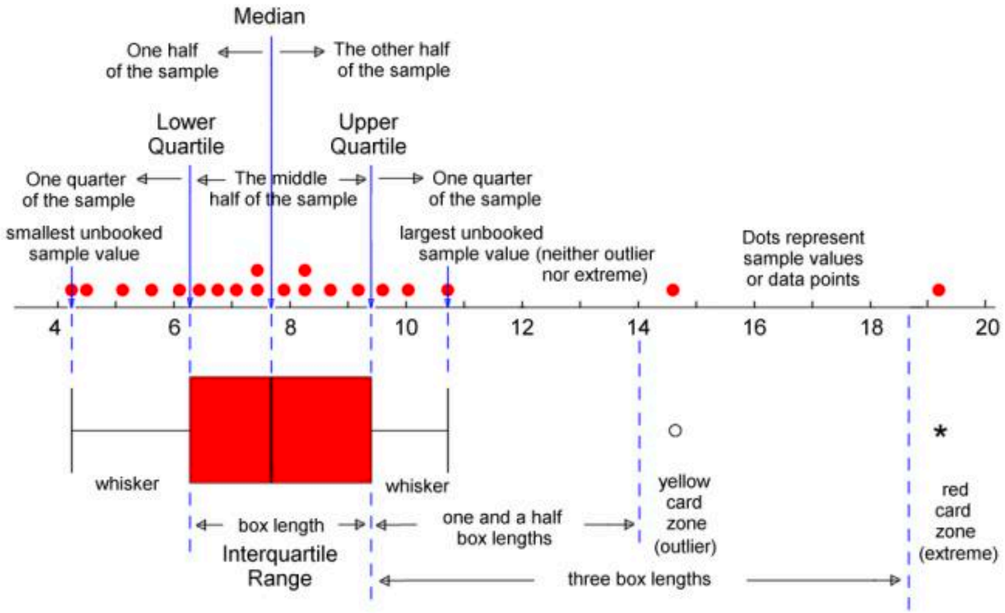
Boxplot Calculation

Interpretation



Boxplot Calculation

Interpretation

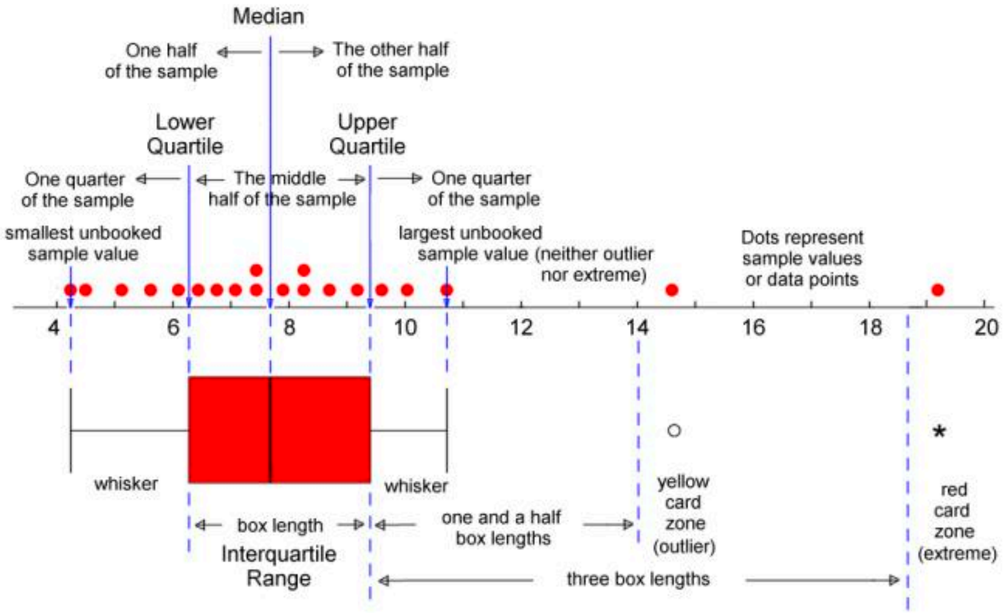


Data

	math score
count	963.000000
mean	66.234683
std	15.250540
min	13.000000
25%	56.000000
50%	66.000000
75%	77.000000
max	100.000000

Boxplot Calculation

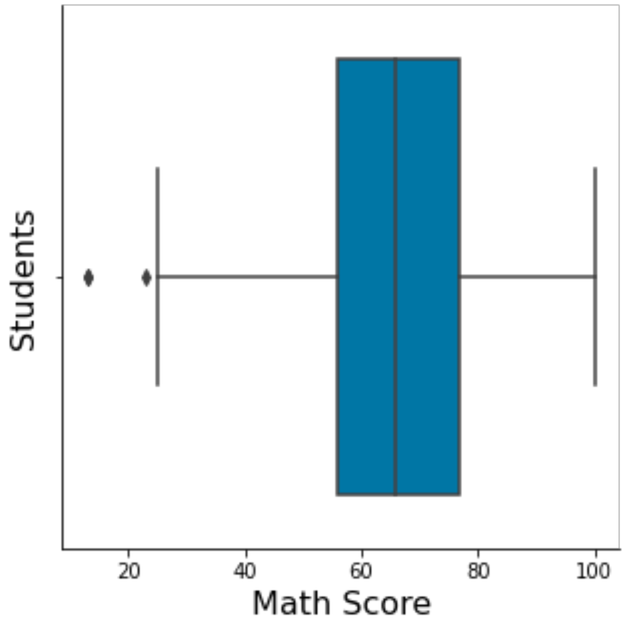
Interpretation



Data

	math score
count	963.000000
mean	66.234683
std	15.250540
min	13.000000
25%	56.000000
50%	66.000000
75%	77.000000
max	100.000000

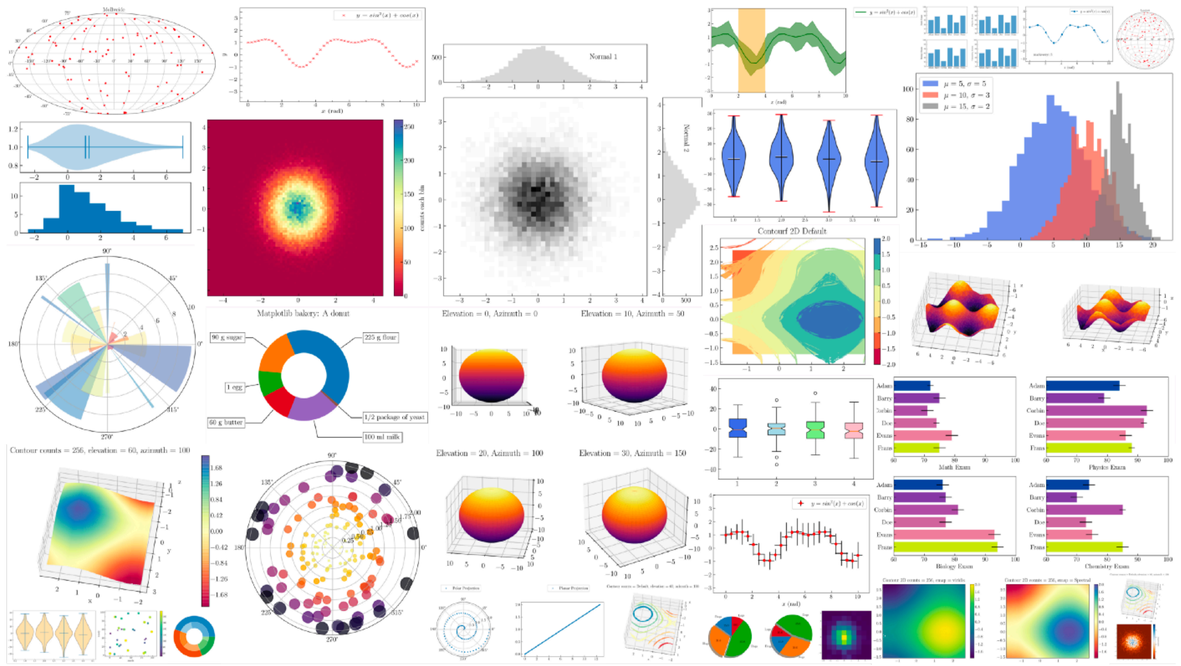
Figure Result



Exercise

- Write the boxplot Python code for the writing score column

There are infinite plot types that you can look into and use



https://matplotlib.org/stable/gallery/index.html

- Important to know how to use some of them quickly to explore your data:
    - Barplots and Boxplots mainly. Also, violin plots, scatter plots, etc…
- Plotting figures is an art and is crucial for communicating your results
- It is a time-consuming process. Go simple and improve when you want to show something

esade

Topic 3: Data Exploration & Visualization

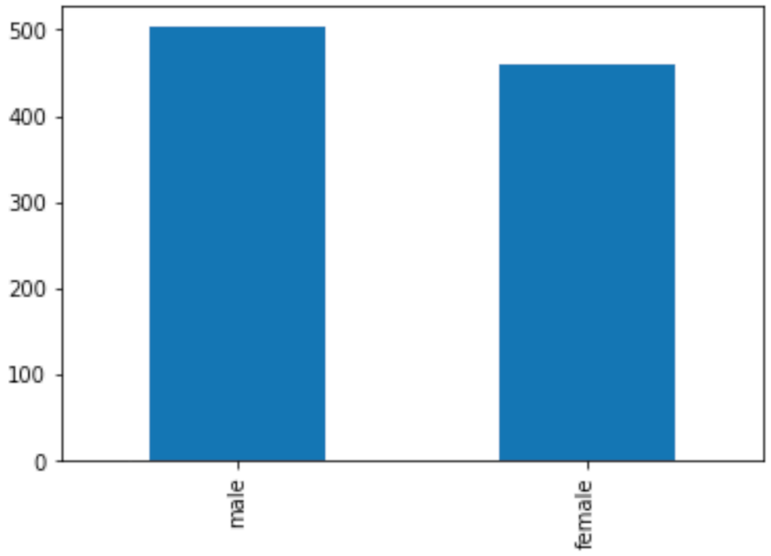
EDA steps

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

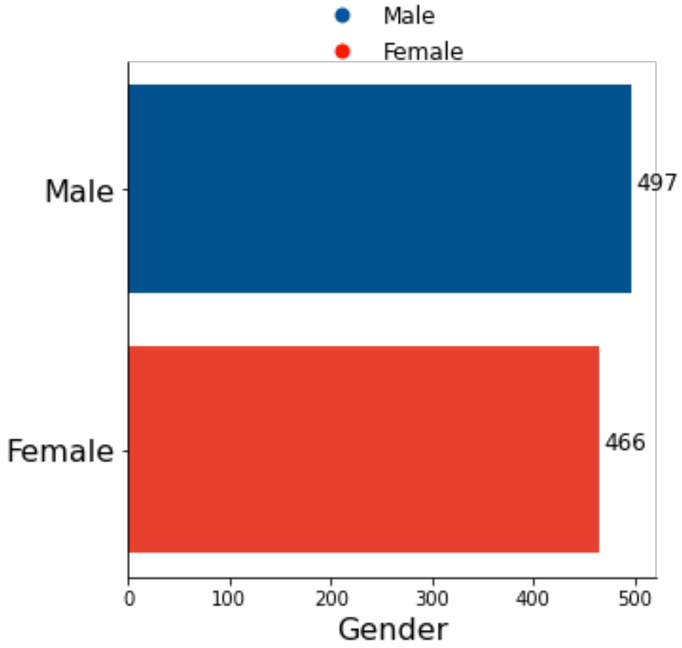
Categorical EDA

- Improve our plots: give them color, change axis names, etc…

Ugly but little code



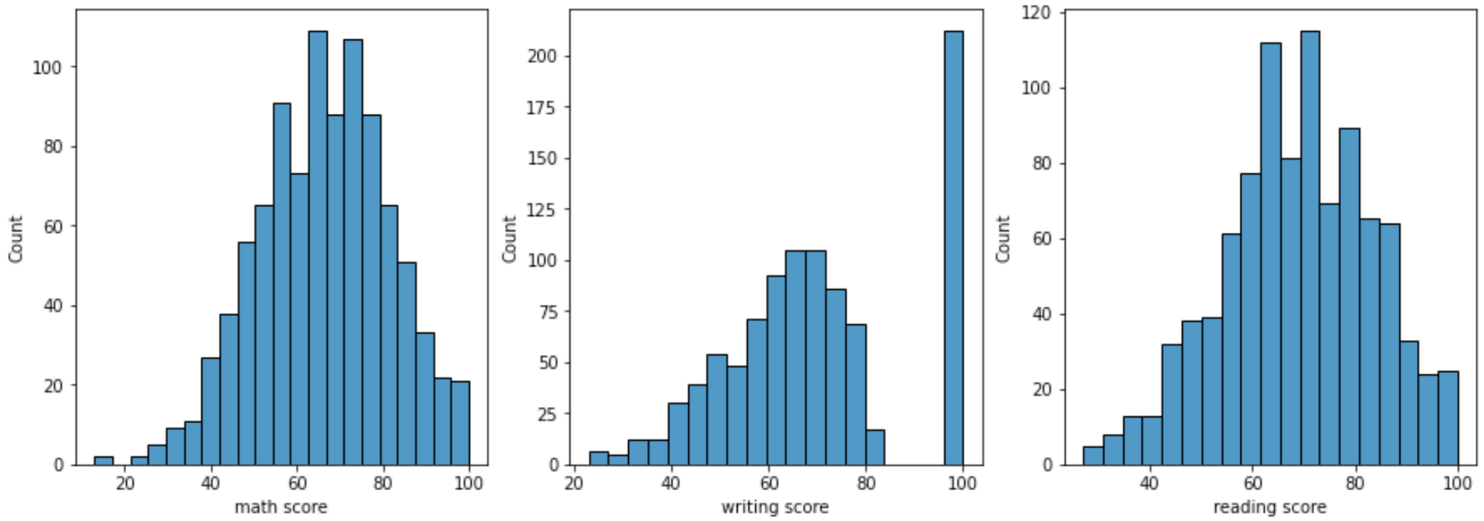
More informative but time-consuming



Numerical EDA - Univariate Analysis

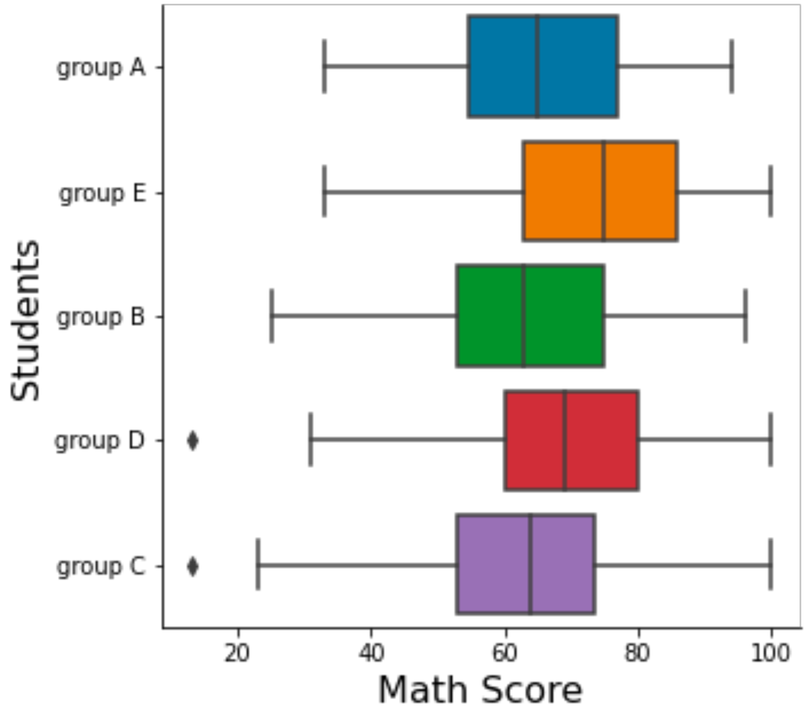
- Study of single variables of the data by analysing the data distributions —> Histograms & Boxplots
- Is my data normally distributed? Can I identify unseen outliers?

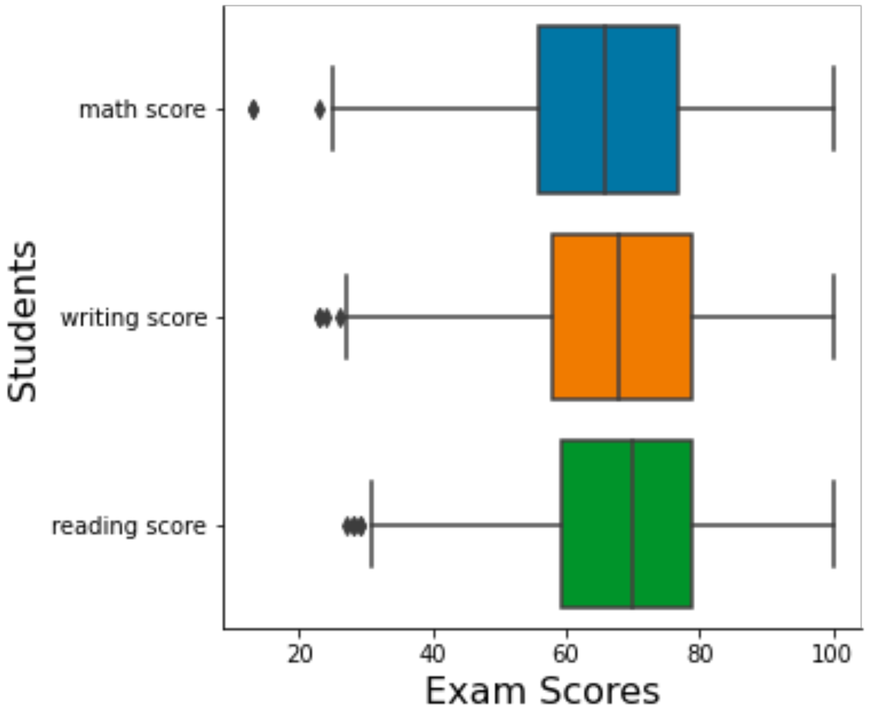
```
#Start Figure
fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(15, 5))

#Body of the figure to build and the data to use
sns.histplot(x=df['math score'], ax=ax1)
sns.histplot(x=df['writing score'], ax=ax2)
sns.histplot(x=df['reading score'], ax=ax3)

#Save or show
plt.show()
```



Numerical EDA - Bivariate Analysis (Categorical + Numerical)

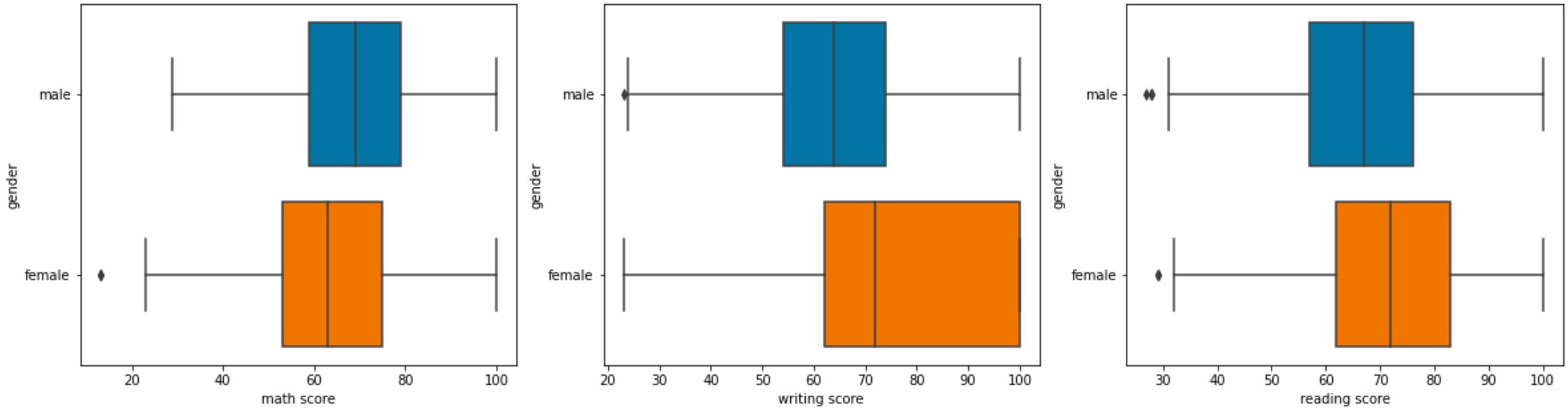
- Combination of variables through boxplots





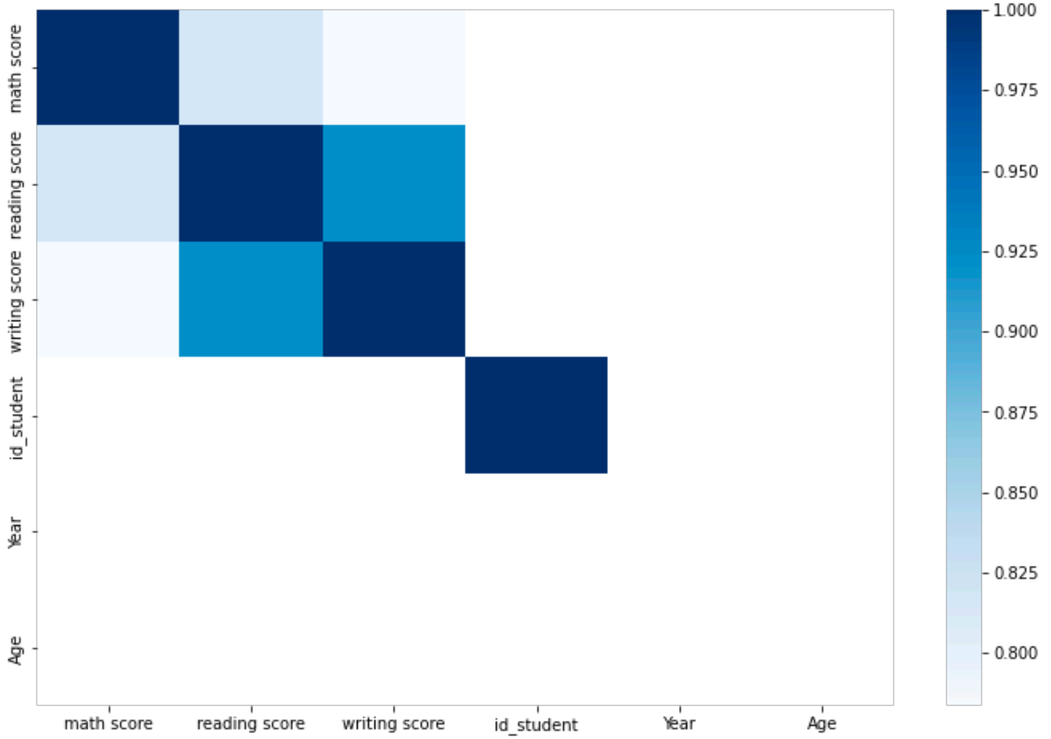
Exercise

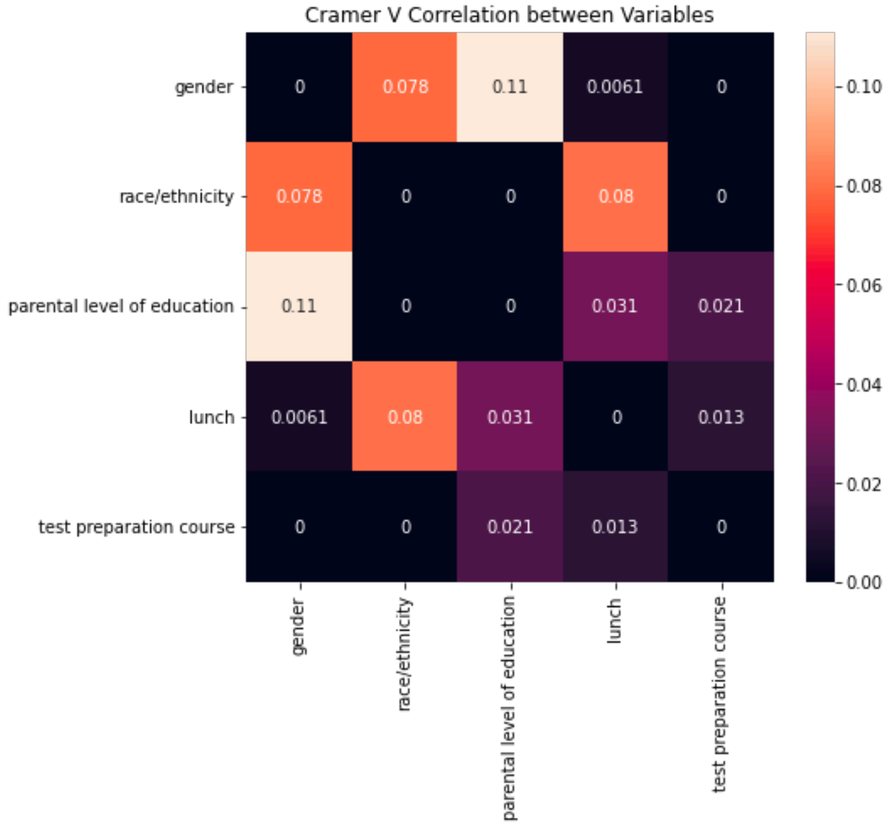
- Can you get the plot below (They can be separated as well)?



Numerical EDA - Multivariate Analysis

- Correlation plots to see how all numerical or categorical variables are related
- This helps identify possible correlations in the data that made affect downstream models. We will come back to this

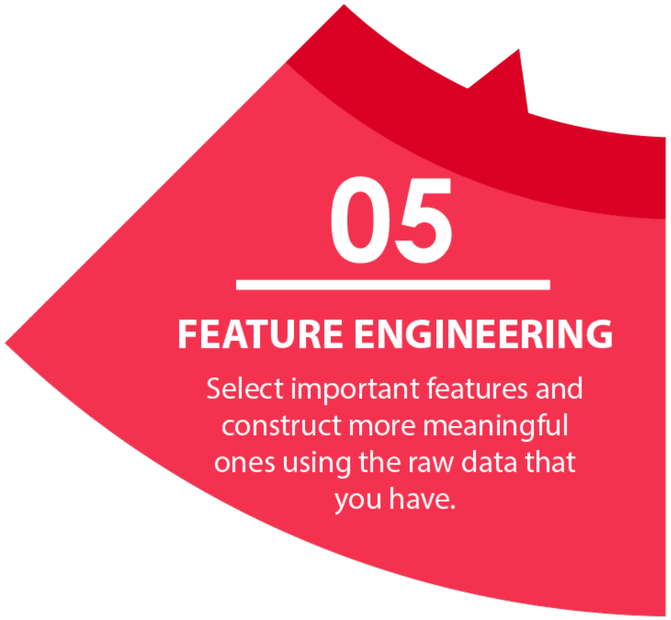




esade

Topic 4: Feature Engineering

Feature engineering



- Is the process of using domain knowledge to extract features (characteristics, properties, attributes) from raw data
- Examples of current projects:
  - Chatbot: #TimesAsking4help, #Words
  - Universities: #PassedSubjects, GradeMean

Feature engineering

Transform the parental level of education

```
df['parental level of education'].value_counts()
```

```
some college          219
associate's degree    196
high school           194
some high school      180
bachelor's degree     107
master's degree        67
Name: parental level of education, dtype: int64
```

```
new_col = []
for el in df['parental level of education'].tolist():
    if el in ["master's degree", "bachelor's degree", "some college"]:
        new_col.append('went to college')
    else:
        new_col.append('no college')
df['Education'] = new_col
```

Feature engineering

Transform the parental level of education

```
df['parental level of education'].value_counts()
```

```
some college          219
associate's degree    196
high school           194
some high school      180
bachelor's degree     107
master's degree        67
Name: parental level of education, dtype: int64
```

```
new_col = []
for el in df['parental level of education'].tolist():
    if el in ["master's degree", "bachelor's degree", "some college"]:
        new_col.append('went to college')
    else:
        new_col.append('no college')
df['Education'] = new_col
```

BMI Calculation

```
df['height'] = 1.70
df['weight'] = 70
```

```
df['BMI'] =  round(df['weight'] / (df['height'] * df['height']), 2)
```

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

Exercise

- Add a new feature (column) to the dataset named *scores mean* that represents the mean of all three grades (math, writing and reading)

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