Data Exploration

The objective of those exercises is to familiarize yourself with the manipulation of a complex dataset, having multiple types of features. We will use python. I recommend to work with notebooks. You can work either by installing python on your computer, or using google colab. If you are not familiar with pandas library, here is a short introduction: `https://colab.research.google.com/github/Yquetzal/Teaching_notebooks/blob/main/Pandas_hands_on.ipynb`

1 Fundamentals

1. Loading the data
   (a) Download the dataset found on the class website.
   (b) Using pandas, load the `movies_metadata.csv` file and check its content
   (c) Discuss in small groups about the nature of each feature.
2. Cleaning the data
   (a) Using `df.info()`, check the type that pandas assigned automatically to each column.
   (b) It appears clearly that some columns have not been converted to the expected numerical type. Try to force conversion using `pd.to_numeric`. An error should occur. This is because the data is unclean (welcome to the real world :)). You can use the option `errors="coerce"` to ignore those errors (nb.: you'll certainly introduce new errors doing so, but let's start with a *quick and dirty* approach)
   (c) Compute the classic descriptors of the `budget` column using pandas' `describe` function. What do you observe about the percentiles ? Keeping false values will bias future analysis, replace them with `np.nan` (e.g., using `replace` function.)
   (d) Using a plotting library (easiest: seaborn, interactive: plotly...), plot the distribution of budget. What do you think of this distribution ? Try using logarithmic sized bins (e.g., using `log_scale=True` in seaborn, or defining your own bins with `np.logspace`.). Find the plot that in your opinion better explain the data and keep it for discussion.
   (e) Do the same with other numerical values. Which one are, visually, following a bell curved, and which one aren't ?
3. Dispersion, Correlation
   (a) For the following questions, we will focus on the revenue, runtime, vote_average and vote_count variables. It might be easier to create a new dataframe with only those variables. You can use `df[['col1', 'col2']]`. Keep only lines in which all values are not NaN.
   (b) Compute the variance, the standard deviation and the mean average deviation (you can use, e.g., `mad()` function from pandas) for the variables. Note the differences between mad and std.
   (c) Compute the covariance matrix, e.g., with `cov` function from pandas. Check the relation with the variance. Can you say something about the other values in this matrix?
   (d) Compute the correlation coefficient between those variables, for instance using the `corr` function from pandas. By default, it uses the Pearson correlation coefficient. Check how it is computed from the covariance matrix. Interpret those coefficients.

(e) Remember that the assumption made when computing Pearson correlation is that the relation between the two variables is linear. Use `sns.pairplot` to have a look at the relation between those variables.

(f) Check the documentation of the `df.corr` function to check how to compute the Spearman correlation. Compare the results.

(g) What do you think of those correlations ?

2 Advanced

4. Statistical tests

   (a) To know if a variable follows or not a given distribution, the best is to use a *statistical test*. The `runtime` is a reasonable candidate to follow a normal distribution: it looks somewhat bell-shaped, and it makes sense intuitively that there is a *typical* movie duration. The Shapiro-Wilk test is a classic method to check normality for a variable. Check the Wikipedia page to see how to interpret it, then see how to run it in python (`scipy.stats.shapiro`).

   (b) Evaluate if the variable follows a normal distribution. The more data there is, the hardest to follow exactly a theoretical distribution, so try with subsets of different sizes

   (c) Compare visually the distribution with a proper normal distribution with the same mean and std (you can use `np.random.normal`). Observe the various ways in which the variable differ from a theoretical normal law.

5. Normalization

   (a) Using sklearn preprocessing tools, apply Rescaling(Normalization), (`MinMaxScaler`), and Standardization (`StandardScaler`). You can use `fit_predict`, and transform the result into a dataframe using `pd.DataFrame(res,columns=previous.columns)`, with res the result of the transformation and previous the original dataframe.

   (b) Compute the covariance matrix and the correlation matrix for both of them. Plot the correlation between the variables. What do you observe compared with before the rescaling?

   (c) Check the formula of the standardization, and think about what it does when the variable distribution is very different from a normal distribution, for instance a power law?

3 Going Further

6. Design an experiment to show how non-linear relations can lead to unexpected results with Pearson Correlation Coefficient, while being correctly captured with Spearman's.

7. Design another experiment in which you set a clear correlation between the two variables, but in which both tests find no strong correlation.

8. Have a look at a real, rich dataset, for instance that one: `https://www.kaggle.com/datasets/benoit72/uk-accidents-10-years-history-with-many-variables`, characterize the variables, how you would encode them, the fraction of missing values, etc.