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 `cars_synthtic.csv` found on the class website.
   (b) Using pandas, load the file and check its content using for instance `.head(2)`
2. Column Types
   (a) Using `df.info()`, check the type that pandas assigned automatically to each column.
   (b) One column has not been converted to the expected numerical type. Try to force conversion using `pd.to_numeric`. An error should occur. This is because a row is problematic. 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)
3. Data Quality
   (a) Compute the classic descriptors of the `length` column using pandas' `describe` function. Check the mean, std, percentiles, and extreme values...
   (b) You should observe suspicious values, too low and too high. Keeping false values in the dataset would bias the results. We will replace them later with `np.nan`, but we need to explore the data to know which values to remove.
4. Missing values
   (a) Check the number of missing values in the `color` column. This value was already present when you did the `df.info`, but you can also use `df[col].isna().sum()` to compute it for one column.
   (b) For columns with few missing values, remove the corresponding rows. You can use the `dropna()` function. It has a `subset` parameter to take only some columns into account. For columns with many missing values, keep them for now.
5. Data Exploration
   (a) Plot the distribution of the `length` variable using a histogram. You can directly use pandas plotting tools (`df[col].plot.hist()`). Vary the number of bins using the `bin` parameter and observe the changes. Use a `kde` plot instead of a `hist`.
   (b) You should now have enough information to consider what are *aberrant* values for length. replaces those values with `np.nan`. You can use for instance `np.where`.

(c) Interactive plots are often convenient to explore data. Pandas allows replacing the plotting backend with an interactive library such as plotly. Install `plotly` library if needed, set it as the pandas plotting backend using `pd.options.plotting.backend = "plotly"`, and then write the same line as before for plotting the histogram. Observe that it is now an interactive plot

(d) To really understand your data, you will however often have to spend time designing your own plots. In this example, use plotly's `px.scatter` function to design a plot in which: x is the `year`, y is the `price`, the symbol shape depends on the `type`, the symbol color corresponds to car's `color` and the symbol size corresponds to the car's `weight`. Try to check if you see some patterns in it. For instance, does it seem that the color or the type has an influence on the price?

6. Distributions

   (a) Plot the cleaned distribution of the `length`. Does it look like a normal distribution? What about the distribution of length for the SUV only? Standard cars only?

   (b) To know if a variable follows or not a given distribution, the best is to use a *statistical test*. 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`).

   (c) Evaluate if the variable `length` follows a normal distribution, for instance considering a p-value of 0.05, or 0.01 ?

7. Dispersion, Correlation

   (a) For the following questions, we will focus on the numerical variables only (length,weight,width,price,year). 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. It is also interesting to compare what happens with and without removing the incoherent values of length: it affects a lot the results!

   (b) Compute the variance and standard deviation (you can use, e.g., `std,var` functions from pandas) for the variables.

   (c) Compute the covariance matrix, e.g., with `cov` function from pandas. Check the relation with the variance. Remember how to interpret those values: you can say something about the sign, but the magnitude alone is not directly interpretable.

   (d) Compute the correlation coefficient between those variables, for instance using the `corr` function from pandas. By default, it uses the Pearson correlation coefficient. Interpret those coefficients. Using the class, check how it is computed from the covariance matrix.

   (e) Remember that the assumption made when computing Pearson correlation is that the relation between the two variables is linear. Use `pd.plotting.scatter_matrix` to have a look at the relation between all the variables. Find relations that are not linear.

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

2 Advanced

8. (a) On the class page, you can find a dataset corresponding to real data about used cars, for one brand. Donwload it (you can also find the reference to the original dataset, containing other brands, if you prefer).

   (b) Apply a similar analysis on this real data.