Introduction to Data Science

FCUP

Practical Class 2
Data Pre-processing
Exercises

1 Preprocess the dataset Diamonds2

Use the file Diamonds2.csv in the moodle

This dataset is a subset of the dataset Diamonds retrieved from Kaggle:

https://www.kaggle.com/datasets/shivam2503/diamonds

Check there to find more info.

2. Execute the following steps loading the required libraries. Use the Cheat Sheets

Libraries: *pandas, numpy, os, seaborn, sklearn.preprocessing, matplotlib.pyplot*

1. Change to the directory with the dataset and load the dataset **diamonds2.csv**

2. Change the name of the very first column to *idx* and set this variable as the index of the dataset

- Tip: first get the columns names into a list, change the first element of the list to "idx" and set the index with set_index()

3. Check how many (and which) features have non-null values

- Tip: use the .info() function or sum all the .isnull() values per column

4. Create a subset of your original dataset with only the attributes: carat, cut, depth, price

- Tip: Create a list with the selected features. e.g., df[['width','length','species']] --> Select multiple columns with specific names.

5. Use the function pairplot() in seaborn to plot all features in a pairwise manner.

- Tip:
    - Use the hue parameter to set the colors according to the value of the attribute “cut”
    - Set the parameter vars as the list of variables that you want to plot except the cut feature, i.e., ["carat", "depth", "price"]

6. Check if there are negative values.

7. Set the variables that are negative to null values

- Tip:
    - Null values: use float("NaN")
    - Select attributes with negative values using loc: e.g. df.loc[df['a'] < 0, ['a','c']]
    - Select rows meeting logical condition, and only the specific columns

8. Set the null values to the mean of the features excluding the null values

- Tip:
    a. test all entries in the feature that are not null, use pandas isna() function to test e.g. test = pd.isna(df[“a”]) == False
    b. select the cases where the above test holds using .loc[], e.g. df.loc[test, "a"]
    c. apply the mean function to the resulting column
    d. set this value where the test above is True, i.e., pd.isna(df[“a”]) == True

9. Set the null values in “cut” to the most frequent value

- Tip:
    a. use the function value_counts() to find the most frequent value
    b. set the attribute cut as the most frequent value for rows with null values, i.e., the rows that match the condition pd.isna(df[“cut”]) == True

10. Redo the figure in step 5 and save to file fig.pdf

- Tip:
    - use the savefig function from matplotlib.pyplot

11. Check if the feature clarity needs processing

- Tip

- convert all the values to upper case. Use the function upper() applied to strings, e.g. x.upper()
- create a list of all the values in clarity converted to upper case
- use list comprehension e.g. [x.upper() for x in dmd.clarity]
- set the column clarity with this new value
- recheck that all values are in upper case (use the value_counts() applied to the column)

12. Encode the variable cut as a numeric value; create a new feature called cut_num as the numeric encoding of cut

- Tip:
  - use the function LabelEncoder from sklearn.preprocessing
  - check the sklearn cheat sheet for an example

13. encode the variable cut_num as a one hot encoding value; set a new feature called cut_ohe

- Tip:
  - use the function OneHotEncoder from sklearn.preprocessing
  - first use the fit function on the OneHotEncoder object
  - use reshape(-1, 1) in the selected column, e.g. df.cut_num.reshape(-1, 1)
  - apply the transform function on the OneHotEncoder object
  - use the toarray() to obtain the converted matrix

14. Calculate a new feature called price_zscore as the standard Z-score of the feature price

15. Calculate a new feature called price_min_max as the min max scaling of the feature price

- check its range visualizing as a histogram