Data Engineering – Project 2: Working with data

March 16, 2023

1 Introduction

Welcome to the second project. This set of exercises is focused on working with real-life data.

As before, they will require that you load a dataset and process it as required, saving the indicated file. This time, we will focus on loading datasets and some basic data cleaning operations like type casting, value filtering and mapping, as well as using categories and encodings.

2 Instructions

As before, please write one Python program which performs all the specified tasks in a sequence, producing the desired results.

Please remember about the good practices of analytic script preparation:

1. Creating your software as a Jupyter notebook is probably a good way to start.
2. When your program is ready, check if it runs correctly as a whole (instead of using individual cells in a REPL environment) – you can do this by selecting *Kernel > Restart Kernel and Run All Cells…* from the top menu, or by simply clicking the "fast forward" button in the notebook toolbar.
3. Finally, you can download your notebook as a `.py` file by selecting *Save and Export Notebook As… > Executable Script* (or using JupyText). Remember to launch a terminal (or open a terminal tab in JupyterLab), activate the environment, and run the program in pure Python.

Your soluton should be committed to the repository as `project02/project02.py`.

3 Exercises

3.1 Exercise 1: Load the file

2 points

Create a DataFrame from a file called `proj2_data.csv`, knowing that:

- it contains more than one column,
- the separator is either `|`, `;` or `,`,
- at least one column contains pure floating-point numbers,
- the decimal part is separated using `.` or `,`,
- the decimal separator does not collide with the column separator,
- thousands are not grouped,
- all columns containing pure floating-point numbers have the same format,

- the file also contains text columns.

The DataFrame obtained by loading the file will be referred to as our *initial DataFrame*.

Save the DataFrame, as imported, to `proj2_ex01.pkl`.

3.2 Exercise 2: Value scale

3 points

The file `proj2_scale.txt` contains strings, one in each line, forming a scale. Subsequent values are (implicitly) associated with natural numbers $1, 2, 3, \ldots, n$. For instance, given a file with values:

```
very bad
bad
average
good
very good
```

the value `very bad` should be associated with 1, and the value `very good` – with 5.

Create a copy of the initial DataFrame. Locate columns in which the values are a subset of the values loaded from the text file. In these columns, replace the values with their numeric counterparts.

Save the resulting DataFrame to `proj2_ex02.pkl`.

3.3 Exercise 3: Categories

3 points

Create another copy of the initial DataFrame. Change the type of columns identified in Exercise 2 to `categorical`. Set the categories for these columns to reflect the entire list loaded from the text file, even if not all values are present in the source data.

Save the resulting DataFrame to `proj2_ex03.pkl`.

3.4 Exercise 4: Number extraction

4 points

Scan the strings in all non-numeric columns of your DataFrame for numbers, following the guidelines below:

- include both integers (e.g. `123`) and floating-point numbers (e.g. `4567.89`),
- include negative numbers (e.g. `-3.14`),
- the decimal can be separated by a dot (`.`) or a comma (`,`),
- thousands are not separated,
- if there is more than one number, pick the first one.

Create a DataFrame with just these columns where any numbers were extracted and save it to `proj2_ex04.pkl`.

3.5 Exercise 5: One-hot encoding

3 points

In the initial DataFrame, find columns which:

- contain text data,
- contain no more than 10 unique values,
- only have values consisting of small letters, i.e. the `[a-z]` range,
- have values that do not appear in the text file loaded in Exercise 2.

For these columns, perform *one-hot encoding*, obtaining a separate DataFrame with encoded values for each original column. The column names should match the values within the column, without any prefixes or suffixes.

For example, for a column that contains 3 distinct values, `red`, `green`, and `blue`, column names in the resulting DataFrame should be exactly that – `red`, `green`, and `blue`.

Save the DataFrame created for each column to files named `proj2_ex05_X.pkl`, where `X` is a subsequent natural number, e.g. `1`, `2`, `3`, etc. (e.g. `proj2_ex05_1.pkl`, `proj2_ex05_2.pkl`).