Exercise #04

IT University of Copenhagen (ITU)
Data Mining KSD (DAMIN)
(Autumn 2025)

16 September 2025

Exercise 04.01. *Batsh*t Data* (45-90 minutes) – For this exercise, we will work on incomplete, unstandardized, poorly encoded data. In other less polite words, s**ty data. The problem with them is that if loaded while trying to extract useful information, it will fail.

- One tries to plot them; one integer is filled with a string or is empty.
- Calculate things; only errors are calculated.
- Try splitting the data; you will certainly miss something.

There are many ways data is not standardized. Let us discuss some of the possibilities.

- **Instructions:**
  - If you **haven't read** the material on this week's plan, read Section 5.1 of Textbook #02 right now. After that, let us discuss and work on the following. Talk a little bit with your 隣 (tonari), and go through the following points while taking notes.
    - What are the options to deal with missing data? (`NaN`)
    - What are the options for non-standardized data labeling?
    - What are the options for data of different kinds? (think `str` and `int`)
    - Why is it not okay to just "complete" the data table you may have with the average value if there are too many missing data? (Think about statistics!)
  - Do you like penguins? Those weird birds found in the Antarctic continent and a few islands? We hope you do. Look for the following files in the course's GitHub Repository: `penguins.csv`. You should be able to load it to a Python shell.

Listing 1: Load the penguins.csv file into a Pandas dataframe.

```python
# Importing the libraries.
from typing import List

import math

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import random as rnd
import seaborn as sns  # Seaborn make some plots way easier!

# Theme for seaborn.
sns.set_theme()

# Check the directories in you code! Copy&Paste is not enough!
data = pd.read_csv("../data/penguins.csv")  # Load penguins.csv files
```

- **Data Exploration:**
  - The command `data.species` lists the column "species" and can be used, among other uses, to locate things. Such as in `data[data.species="Gentoo"]`. Use it to look around species and other attributes.
  - Many other commands allow exploring the data. But nothing is like a plot. Look at Visualizing Distributions of Data at the Seaborn tutorial. It uses the original data set.
  - Use the Seaborn plotting capabilities to plot the data. Use the `sns.displot` command to plot the data. It will show a lot of information. Use the `hue` parameter to show the species.

```python
sns.displot(penguins , x="flipper_length_mm", hue="species")
```

  - Remember to investigate other attributes other them `"flipper_length_mm"`. Use what is in there to have an idea about what you have. It will be useful later.
- **The one where we actually do things.** Here's a secret for you: We intentionally messed with the data in the file. We hold an "original copy" of the dataset. (How can a copy be original?) Load this copy, which will be corrected! Load the new data set and compare it with the original one:

```python
data2 = pd.read_csv("../data/penguins2.csv")  # The messed csv file.
```

1. Now, it's time to make the data more standard. First, we messed with the data, creating redundant species that differ only by a dot or capitalization, changing some data types to string, and setting some to `NaN`. How one goes about that?
   - To correct the redundant species! One can use Pandas data frame commands such as: `data.species[data.species == 'Adelie']` list all Adelies. One can also change the value of a given entry as a variable or be really "distinctive and correct" and use `data. at` to do it.
   - Correct string values should be in the columns where there is a float. Use the `np.float64` command. (tip: What happens if you convert a float to a float?)
   - To solve the `NaN` values, one can either somehow replace them with guessed numbers or remove all lines with some `NaN` value. Look here for inspiration on how to do it. Be aware that too many of them can be pretty problematic.

2. Correct the redundant species. List all species, identify the redundant ones, and use your acquired knowledge of `for`, `else` and functions to do it.
3. Work and compare the `NaN` in the two ways possible.
   (a) Data Amputation of lines with `NaN`.
   (b) Data Estimation using:
      - Average.
      - Proximal values (look at the slides!).
      - (*Extra!*) Create a routine that calculates the average only in each species. Use it to find possible estimates for each missing value.
      - Compare each of them.
   (c) Using your new acquired plotting with Seaborn, and plot the corrected data. Compare it with the original data.
   (d) Make a routine to guess the species for the data with unknown species using the data for each species (hard bonus exercise for the show offs that finish early).
      - Extract the data with missing 'species' data.
      - Complete the missing data that using averages of the physical characteristics of each species.
4. Can you use the plotting capabilities of Seaborn to compare the corrected and the original data? How would go about?