Plotting in Python (Bonus)

Solve the following exercises and upload your solutions to Moodle until the specified due date. Make sure to use the *exact filenames* that are specified for each individual exercise. Unless explicitly stated otherwise, you can assume correct user input and correct arguments.

Exercise 1 – Submission: `ex1.py` 15 Points

Write a function `plot_lifts(data: dict, save_path: str = None)` that plots powerlifting weights of the three lift categories squat, bench press and deadlift. `data` contains these weight lifts in the following format: The key (string) is the name of the category, and the value is a list of integers representing the lifted weights. The function must create the following plot using `matplotlib`:

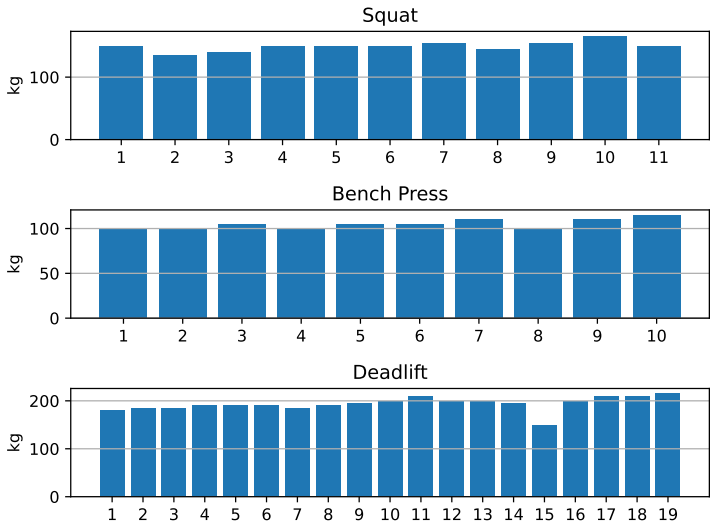
- Three bar plots in three separate rows showing the individual lifted weights per category.
- The x-ticks must range from 1 to the number of lifted weights with a step size of 1.
- The plots must show a grid on the y-axis.
- The y-axis label must be set to `"kg"`.
- The title of each row must be set to the corresponding category name.

If `save_path` is not `None`, the plot must be saved to the specified path.

Example program execution (given some example data in `ex1_data.csv`):

```
lifts = dict()
with open("ex1_data.csv") as f:
    for line in f.readlines():
        lift_name, weights = line.split(",", maxsplit=1)
        lifts[lift_name] = [int(w) for w in weights.split(",")]
plot_lifts(lifts)
```

Example output (might differ due to matplotlib versions and settings):



Exercise 2 – Submission: `ex2.py` 15 Points

Write a function `plot_classes(data: dict, save_path: str = None)` that plots 2D data points which are part of classes. `data` contains these data points in the following format: The key (string) is the name of the class, and the value is a NumPy array of shape `(n, 2)`, where `n` is the number of samples of this class. The function must create the following plot using `matplotlib`:

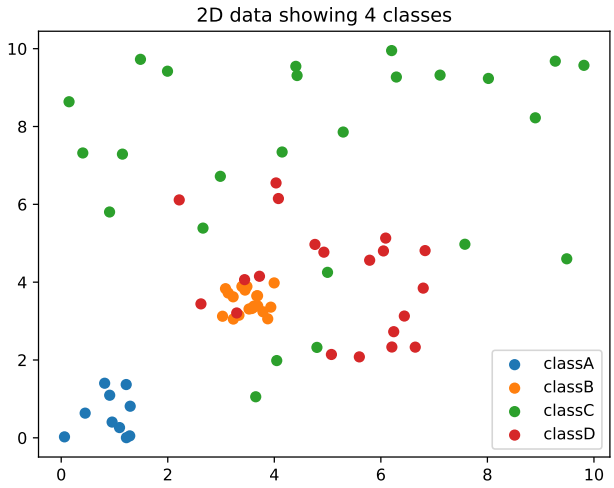
- A scatter plot showing all samples, colored according to their classes.
- The plot must contain a legend that lists all classes.
- The axis title must be set to `"2D data showing c classes"`, where `c` is the number of classes.

If `save_path` is not `None`, the plot must be saved to the specified path.

Example program execution (using the `create_data` function from Exercise 3 of Assignment 10):

```
plot_classes(create_data([
    {"id": "classA", "n": (10, 2), "a": 0, "b": 1.5},
    {"id": "classB", "n": (20, 2), "a": 3, "b": 4},
    {"id": "classC", "n": (25, 2), "a": 0, "b": 10},
    {"id": "classD", "n": (20, 2), "a": 2, "b": 7},
], 0))
```

Example output (might differ due to matplotlib versions and settings):



Exercise 3 – Submission: `ex3.py` 20 Points

Write a function `plot_eval_metrics(data: dict, save_path: str = None)` that plots various evaluation metrics. `data` contains these evaluation metrics in the following format: The key (string) is the name of the evaluation metric, and the value is yet another dictionary. This inner dictionary has the following two entries: `"values"` refers to a list of n NumPy arrays containing float numbers in the range $[0, 1]$, and `"labels"` refers to a list of n strings that indicate the names/IDs of these arrays (i.e., arrays and IDs match, there is exactly one array per ID). The function must create the following plot using `matplotlib`:

- For each evaluation metric in separate columns, a box plot showing data of the n arrays in distinct colors according to some fixed colormap of your choice.
- The box plots must be drawn vertically.
- The median line (enabled by default) must be set to the color black.
- The x-ticks of each evaluation metric box plot must be set to the n IDs.
- The y-axis is shared across all plots, and the y-ticks must range from 0.0 to 1.0 (inclusive) with a step size of 0.1. Additionally, make sure that the plots include some space below and above 0.0 and 1.0, respectively, i.e., the borders/limits of the plot should not be 0.0 and 1.0.
- The title of each column/box plot must be set to the corresponding evaluation metric name.
- The box plots must show a grid on the y-axis with gray dashed lines.

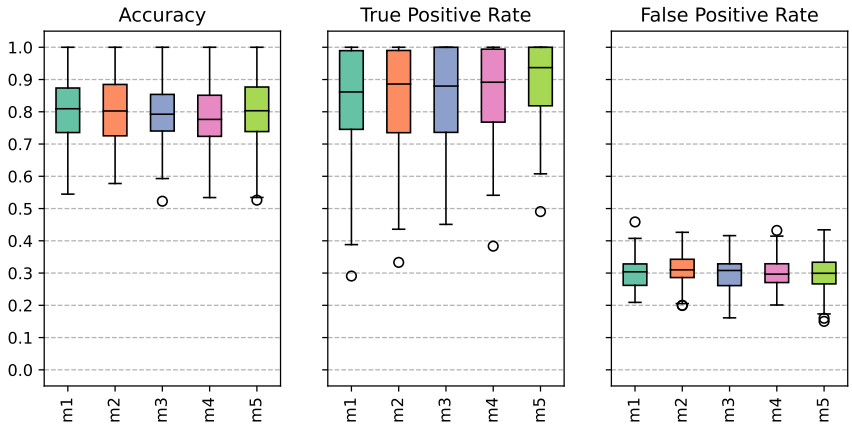
If `save_path` is not `None`, the plot must be saved to the specified path.

Example program execution (given some example data in `ex3_data.pkl`):

```
import dill as pkl

with open("ex3_data.pkl", "rb") as f:
    ex3_data = pkl.load(f)
plot_eval_metrics(ex3_data)
```

Example output (might differ due to matplotlib versions and settings):



Hints:

- When you want to manually use a colormap (i.e, not via an existing function argument), you can write `cmap = plt.get_cmap(name)` and then `cmap(i)`, where `i` can either be an integer indicating the `i`-th color of this map (useful for qualitative colormaps) or a float in the range $[0, 1]$ indicating the $100 \cdot$ `i`-th percent along this colormap line (useful for sequential, diverging or cyclic colormaps). The example above uses the qualitative colormap *Set2*.
- To fill box plots with color, this example might be helpful. To change the color of the median line, pass some dictionary to the `medianprops` parameter. Take a look at the possible keyword arguments that you can specify in this dictionary.