Organizing repositories

Alberto Tonda, Ph.D., Senior permanent researcher (DR)

UMR 518 Mathématiques et Informatique Appliquées - PS, INRAE, U. Paris-Saclay
UAR 3611 Institut des Systèmes Complexes Paris Ile-de-France, CNRS

alberto.tonda@inrae.fr

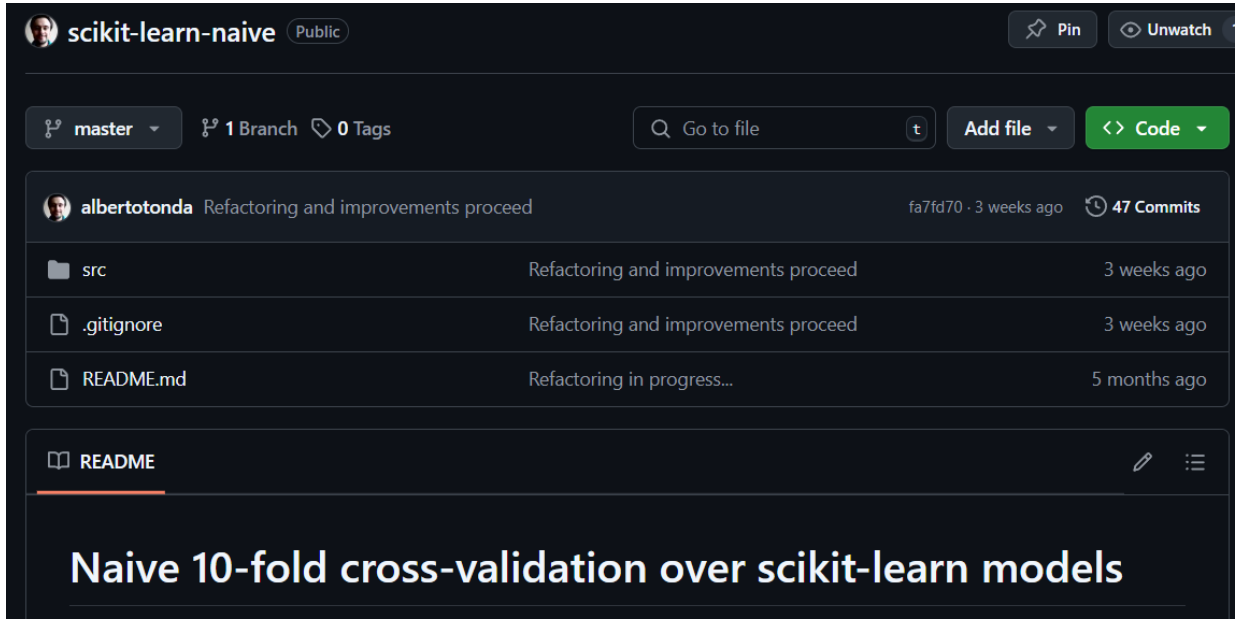
Another example of title for a slide

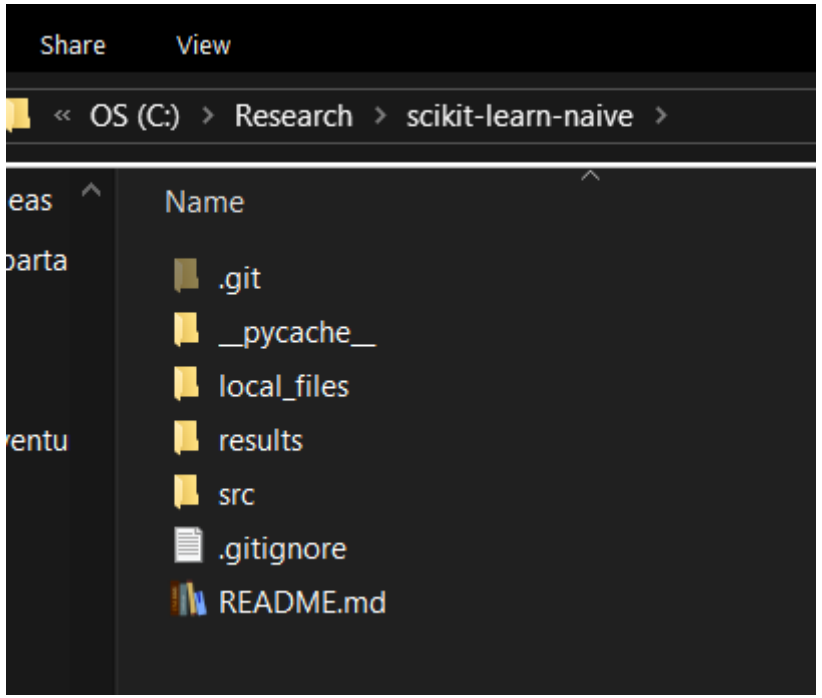
- This is some text
  - And some smaller text

Caveat

- This is mostly a mix of recommendations + experience

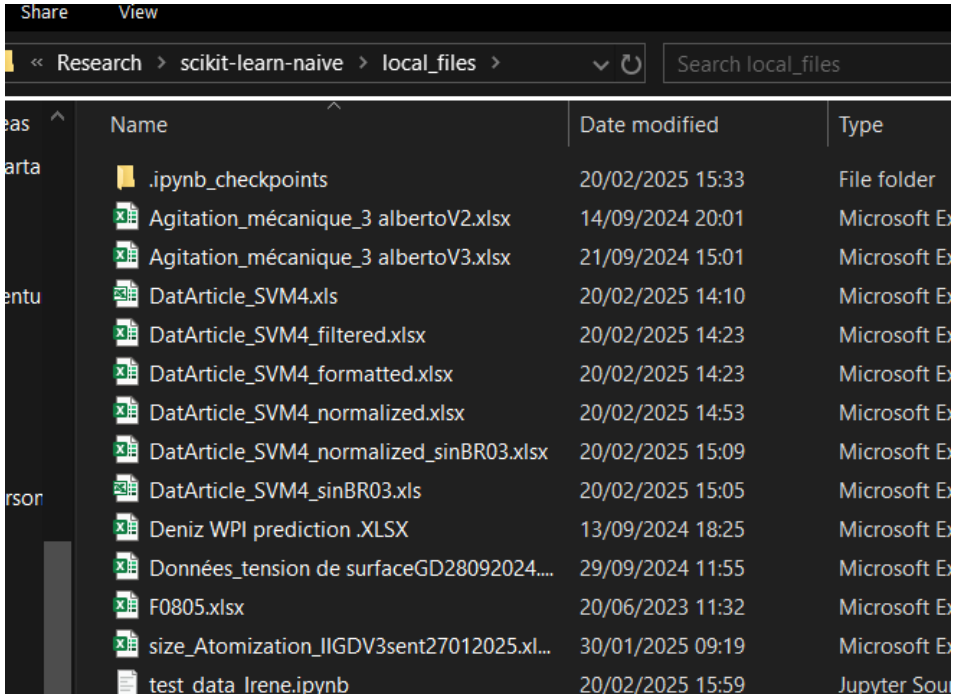
Exploit functionalities of git





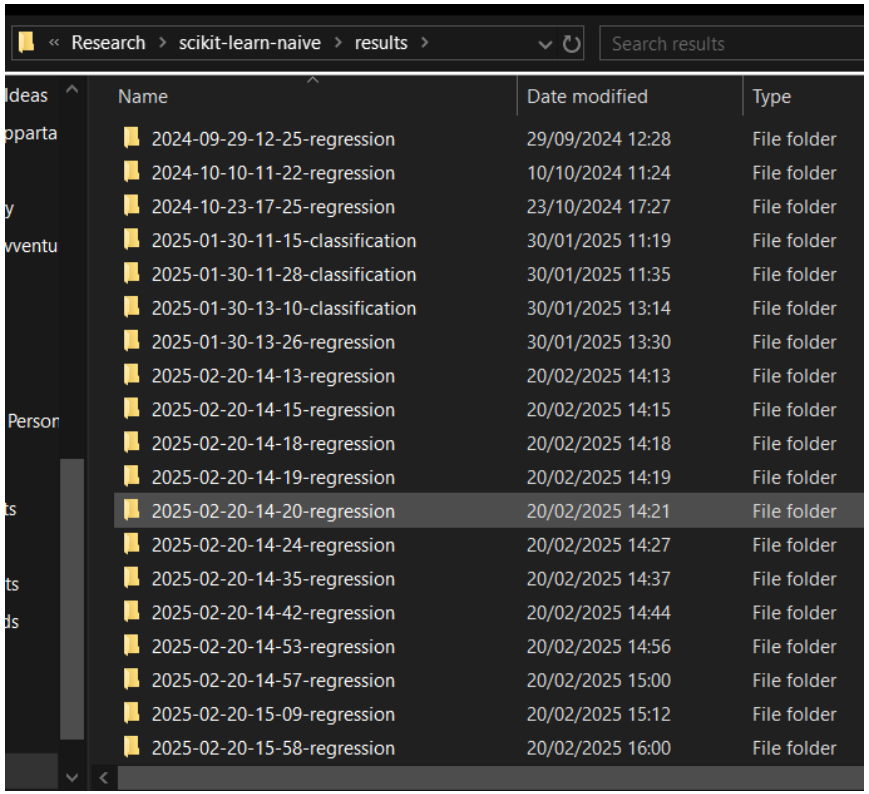
Exploit functionalities of git

- You can ask git to ignore entire folders!
  - Good for files you don’t want to put under version control
  - E.g. sensitive data from your collaborators



Exploit functionalities of git

- But also useful to not clutter folder with results!
  - Put all results of your scripts (text files, CSV, etc.) in another folder



Create meaningful subfolders for results

- Each time you run your script, create a subfolder
  - How to name subfolder? **datetime**
  - If you go Year-Month-Day-Hour-Minute-Second, your system will sort alphabetically in a descending order by recency!

```
# create uniquely named folder
folder_name = datetime.datetime.now().strftime("%Y-%m-%d-%H-%M") + "-classification"
folder_name = os.path.join(result_folder_name, folder_name)
if not os.path.exists(folder_name) :
    os.makedirs(folder_name)
```

How to save results?

- CSV tables are pretty convenient
- The best workflow I found so far is
  - Create **dictionary** of **lists**
  - Keys are the column names, lists are elements in each row
  - Use **pandas** to convert dictionary to a pandas.DataFrame
  - Save dictionary to CSV using built-in **to_csv()** method

```
# now, here we can write a final report; it's probably a
# probably the best way to go is to first create a dictio
df_dict = dict()
df_dict["classifier"] = []
df_dict["preprocessing"] = []
for metric_name in metrics :
    for t in ["train", "test"] :
        df_dict[metric_name + " " + t + " (mean)"] = []
        df_dict[metric_name + " " + t + " (std)"] = []
df_dict["AUC (mean)"] = []
df_dict["AUC (std)"] = []
```

```
# now that the dictionary is ready, convert it to a DataFrame
df = pd.DataFrame.from_dict(df_dict)

# since we are using a lot of different metrics, we have to pick one that will be used
df.sort_values(reference_metric + " test (mean)", ascending=False, inplace=True)

final_report_file_name = os.path.join(folder_name, final_report_file_name)
logger.info("Final results will be written to file \"" + final_report_file_name + "\"..
df.to_csv(final_report_file_name, index=False)
```

Don’t hard-code values in your code!

- Hard-coding means writing values by hand in the code
  - Try to limit this as much as possible
  - Way better to read values from outside (e.g. CSV, text or JSON file)
  - Why? Don’t modify code every time you change something
  - JSON is more convenient to parse if data don’t fit a table

Logging

- Sometimes the output of your program is LONG
  - Going through the console is too difficult
  - Save output to console AND text file at the same time
  - Some messages just in the text file
- Logging! Using the **logging** module
- Creates a **logger** object (needs to be passed to functions)

Scripts or notebooks?

- Always scripts, except for images or data summaries
  - For example, notebooks for plotting images for papers
  - And also generating tables as text files (Latex)
  - If you create images during a run, save them as part of script

- Examples from **symbolic-regression-ode-systems**

alberto.tonda@inrae.fr