Assignment 2. ChatGPT report

Errasti Domínguez, Nuria and Levenfeld Sabau, Gabriela

January 2024

Summary of how we have used ChatGPT, some important prompts, some places where ChatGPT went wrong and how we solved it.

1. **Coding easy with ChatGPT**.
   Whenever we had to import a module we didn't know the library it belonged to we drew upon ChatGPT to help us. For example, we asked the AI where from we had to import the functions *f_regression* and *mutual_info_regression* and the answer was *sklearn.feature_selection* which was correct.

2. **ChatGPT as a refactor helper**.
   When coding the assignment, our initial focus was on making the code work. Once we ensured it worked properly, we turned to ChatGPT to assist us in the refactoring process. The AI helped us by deleting unnecessary code that was no longer in use and creating new functions to clarify the script.
   We realised that by offering some "hints" about the desired goals of these new functions, we enhanced the quality of ChatGPT's responses. By doing so, it was clear that we were able to save time. Thus, for example, we did not need to spend time determining the variables to pass to these functions. Some examples are the generation of functions such as *plot_predictons*, *eval_metrics_model* or *split_data* function.

3. **ChatGPT as a debbuger assistant**.
   Similarly to the previous situation, the main advantage lies in the significant amount of time saved thanks to the AI. It helped us with understanding, detecting as well as fixing the errors that the python console provided us. With ChatGPT there was no need to spend time on internet searches to decipher error messages. Furthermore, most times the mistakes were related to typos in variables' names or misplaced commas, it optimized the debugging process.

4. **Transforming ideas into reality with ChatGPT**.
   Often, when we had a clear vision of our coding goal but struggled with the initial step, ChatGPT proved to be a good tool at providing a good starting point. It not only generated coded sketches but also included the nec-

essary libraries. However, we need to emphasise the importance of formulating well the prompts. This will mark the difference for obtaining a high quality output. In order to do so, we realised it is highly recommended to be specific, by providing details about the desired functionality. It is also a good practice to include a brief summary about the general task (`"I am coding a regressor KNN using sklearn library in Python. This project is about predicting wind energy production..."`), sharing relevant variables and ensuring clarity about what ChatGPT must cover on the answer.

5. **ChatGPT as a coding mentor**.
   Since we are not coding experts, we often had doubts about the different types of variables existing in Python. Our common questions were about understanding the distinctions between different ways of initializing variables and setting them in our code. ChatGPT, helped us as a reference, making clear explanations.
   For instance, we asked for the difference between initializing the search space variables using the command `list(range(4, 40, 1))` and the command `list(np.random.randint(2, 16, 2))`. Another example was on the difference between doing `y_train = train_train['energy'].values` and `y_train = train_train.energy`.

6. **ChatGPT gets into an infinity loop**.
   One of the worst challenging issue we encountered while using ChatGPT was getting into an infinitive loop, where the AI repeted the same information endlessly, making it impossible to reach a solution. In such a situation, the only way to get out of it was by starting from scratch a new chat.
   For example, we asked for help selecting the relevant parameters of the KNN and Tree models and to propose an appropriate range for those parameters in the random-search. ChatGPT responded by describing the parameters of the model and emphasizing that the range depends on the specific problem. Then, we could not get into a clear conclusion and we had to rely on our own resources. This lead us into exploring the models library and looking for references over the internet in order to formulate the desired ranges.

7. **Fixing ChatGPT**.
   On Question 7, which consists on feature selection for KNN, we had some trouble at getting the **selected_feature_names**. We identified this mistake when printing "Selected features and scores" and it did not include the highest score. However, ChatGPT kept saying that our code was correct (we were starting to enter an infinity loop). So after a long discussion, we had to specify to review a designated line of code and had to be really clear about what was going on.

[ME]
selected_feature_names =
X_train.columns[reg_fs_grid.best_estimator_.named_steps['select'].get_support()]
Can you check this line of the code, because i am not sure it is working properly

[ChatGPT]
The provided line of code appears correct...

[ME]
The problem is that when I print all the feature scores I observe that there are higher values than the ones that have been selected during the process. So I think my code is not completely well defined...
For example, this is a selected value: 0.0441070770219083, but there are other features where the value is higher and does not appear on the selected ones (0.16457657).
...
Attached also to this prompt was the value of feature_scores parameter, the loop for displaying "Selected features and scores" as well as some more code on this question.

[ChatGPT]
I see the issue now. The problem seems to be related to how you are retrieving the feature scores after fitting the...

Finally, the issue was solved.