MLCB 2024

19/3/2024 (100 points)

Assignments No 1

Assignment goal: The first assignment for this semester is split into two parts, one is related to setting up a development environment and following proper software design principles, using python files to build up your codebase and notebooks for the presentation or operation you will perform, once such as plotting and data exploration, cleaning. The second part aims to familiarize yourselves with some basic data cleaning, as data in the real world is rarely in the proper form for ML tasks. Additionally, you will learn to implement one of the most popular techniques for model hyperparameter tuning, namely cross-validation.

Task 1. Set up your development environment:

1. Select and download your integrated development environment (IDE), one of the best free options is Visual Studio Code.
2. Establishing a Linux environment is essential for this assignment. Linux users can skip this step. However, Windows users should install Linux alongside Windows without the need for a dual-boot by utilizing the Windows Subsystem for Linux (WSL). You can also connect Visual Studio Code with your WSL environment [tutorial].
3. Install Python to your Linux WSL enviroment via the terminal with the Anaconda package manager [tutorial].
4. Create a GitHub account and install git, a powerful version control software, via the Linux terminal [tutorial].

Task 2. Download and clean the dataset:

You will be working with time-series data of the confirmed COVID-19 cases for each day from 22-01-2020 to 31-3-2022 for forecasting purposes using regularized polynomial regression. Use "git clone https://github.com/MLCB2024Class/Assignment_1 " with your terminal to download the project directory to your computer from github, this is the directory you will submit when you finish the project.

1. First create the sub-directories to structure your project. These should be "src" for the source code, "notebooks" for the jupyter notebook files and "models" to store your trained model instances. The *Johns Hopkins Coronavirus Resource Center* dataset should exist in the ./data directory. Create a notebook "data_cleaning.ipynb".

2. In the ./notebooks directory, load the dataset and keep the COVID-19 cases for the following countries: 'Korea, South', 'Japan', 'Italy', 'Spain', 'Germany', 'Greece' and all dates. Process the dataset and once it is in a format you deem final for forecasting create two datasets, one with the first 700 days (data-points) and one with the first 800 days of the pandemic and export them as .csv files named "final_data_700_days.csv" and final_data_800_days.csv" respecrtively in the ./data directory.
3. Plot the complete, 700 points and 800 points dataset for all countries in three separate figures.
4. Explain the reasoning behind your cleaning/pre-processing operations. Are all the columns necessary? What should the index of the final dataset be since we are dealing with a time series forecasting task and why? Propose a component

Task 3. Time Series forecasting using polynomial regression:

In the ./src directory, create a "functions.py" file, there you will write your codebase to Identify the optimal model instance from a given simple parametric family of regularized polynomials of degree up to M=9 (as in Bishop's book example) that best characterizes the growth curve of COVID-19 cases, using polynomial features with ridge regression (l2 regression from class). You should implement:

- Cross-validation to tune the hyperparameters of your model. It should be able to take as input (arguments) the data used for training and output an optimal trained model instance with tuned hyperparameters based on the minimization of the Root Mean Squared Error (RMSE) across the validation splits.
- Code to establish a baseline model for each country to set a minimum performance benchmark and reference point for more complex models.

The output model instances are to be stored for each country in the ./models directory to be loaded for inference on unseen data.

1. After setting up your codebase, create a notebook named "model_analysis.ipynb". Import your codebase from "functions.py" and load the dataset "final_data_700_days.csv". To train candidate models for each country, use 80% of the data points for training and the remaining 20% for testing. Generate a figure with subplots for all countries, distinguishing between train and test sets with different colors to verify the split.
2. Generate a baseline model for each country and populate ./models .Plot true and predicted data for both training and test sets on a single figure for each country, with distinct colors for each dataset. Titles should include the country, selected polynomial degree, and alpha values. Store the RMSE on the train, test set for each country for future comparisons.

3. For each country, perform hyperparameter tuning with cross-validation to populate the ./models directory with the optimal model instance. Create plots and store the RMSE values as in 2.
4. Create a single barplot to compare the RMSE of the baseline and optimal model instances both on the training and test sets. Discuss your findings.
5. Load the dataset "final_data_800_days.csv" and repeat steps 1 through 4. Then, compare the resulting RMSE bar plots with those obtained using the time series with 700 data points. Comment on and provide an explanation for the observed results.
6. Establish a directory named "final_models" to house only the 6 best model instances, one for each country, either trained with the 700 or 800 points dataset located in the "./models" folder. Your work will be evaluated based on your optimal model instances per country.

In any case, to guarantee reproducibility of your results you should use a seed=42.

For extra Bonus points:

The following bonus parts are independent of each other. You can do as many as you want, and **they are all optional**. How bonus points contribute towards your final class grade will be explained in class. If you choose to complete bonus tasks do so in a notebook named “Bonus.ipynb” in the ./notebooks directory.

1. [For 40 points]. Add an alternative option for hyperparameter tuning in your “functions.py” codebase via an input parameter using the Optuna package with trial pruning and Bayesian optimization. Compare and discuss the performance of the resulting models against the optimal models from task 5.
2. [80 points] Well-known signal processing methods exist for short-time prediction applicable to time series data. Review the related literature. Then select and implement one of them. Compare and discuss the performance of this method against your best results from task 5.
3. [80 points] If you are not satisfied with the performance of your optimal models in an 800-point dataset, investigate alternative machine learning methods and implement one of them following the same steps. Compare and discuss the performance of this machine learning method against your best results from task 5.

Reporting: Your code should be in Python (scikit-learn preferred), well-structured, and well-documented. Y*ou should not copy* code from other solutions found on the internet. Every student should work alone on this assignment. Your Jupyter notebooks should be structured with headers describing the task or bonus task of each section and **contain ample markdown** where discussion is needed. Special emphasis should be put on the visualization of the results, which should be of good quality and informative, with correct axis limits and labeling. Please include everything in the assignment directory in a single compressed (zip) file with filename format <YourLastName_FirstName_IDnumber>.zip. Ensure that the models stored in the "./final_models'' directory can directly predict unseen data. This includes appropriately handling the input required for the model. If direct prediction is not possible, develop a Python script named "evaluation.py" in the "./src" folder to address this issue.

An important note on LLM usage for coding: Learning programming/Data Science isn't about mastering a single thing. It's mostly about gathering bits and pieces of knowledge and fitting them together like a puzzle. Once you have built "muscle memory" from practicing with assignments like this one, you can easily use your knowledge to tackle similar problems in the future. However, if you rely on LLMs without 100% understanding of the code they provide, you enter an infinite cycle where you will keep returning to it for the same tasks over and over again, and each time it is varied, often error-prone solutions will only lead to further confusion. It is much better to use them as a teaching assistant if needed, although the sci-kit learn package provides great documentation with in-depth explanations for all of the tools it provides, and it is a great idea to familiarize yourself with its use as early as possible. After all, assignment solutions that utilize copy/pasted LLM code are most often distinguishable and we have the experience to spot them!

Hints:

- It is a good practice to follow the "Single responsibility principle" (SRP) when creating the functions for your codebase, which states that each function should have a single specific purpose.
- Repeated code in your codebase signals the need for modularization through functions.
- Since time series data are by nature sequential you should avoid shuffling your training, validation as well as test datasets. They should follow temporal ordering.
- Take a look at sci-kit learn Pipelines.

Deadline: You should upload your deliverables (.zip file) to e-class by **Monday, April 8, 2024**.