Machine Learning 2
Lab Practice: Stream Processing
Course 2023-2024

For this laboratory exercise, you will put together all you know about using the stream learning with River. The first task to tackle would be to choose a dataset suitable to be solved by applying a stream learning approach. Therefore, keep in mind that a temporal dimension to the problem could be helpful.

Being this exercise the evaluable part of stream learning for the practice, this exercise has a weight of 4 points in the final mark of the course (laboratory part). At this point, it could be worth mentioning that you may use a dataset provided by River for this little project, however, the evaluation of your work may be impacted according to the complexity of the process and task to be tackled.

This assignment is going to be developed in groups of 3 or 4 students. Each of the groups would have to present their results on February 22nd during the practical hours (10-12 minutes maximum for each presentation). Additionally, each of the groups would also have to submit a notebook by the task enable on the Campus Virtual of each university, which contains at least the following points:

1. Select a stream database. A dataset that is most useful should have a temporal component.

   File-based dataset: The datasets could be from a static file. You would then use the river library to create a generator and iterate over records. Some publicly available datasets with a time-dependent component that can be used with the River library in Python. Nonetheless, you are free to select another dataset.

   - NYC Taxi trip data (https://www.kaggle.com/c/nyc-taxi-trip-duration)
   - Stock prices (https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs)
   - Energy consumption data (https://www.kaggle.com/robikscube/hourly-energy-consumption)
   - Movie Lens Recommendations (https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset)
   - Weather data (https://www.kaggle.com/selfishgene/historical-hourly-weather-data)

   API Endpoint: It doesn't have to be a static file but could also be an endpoint. Here are some public API endpoints that can be used for stream processing with the River library in Python:
   - Twitter API (https://developer.twitter.com/en/docs/tweets/filter-realtime/api-reference/post-statuses-filter) - for processing real-time tweets.

- GitHub API (https://docs.github.com/en/rest/guides/getting-started-with-the-rest-api) - for processing events such as new commits, pull requests, and issue updates.
- Stock market API (https://iexcloud.io/docs/api/) - for processing real-time stock market data.
- News API (https://newsapi.org/docs) - for processing real-time news articles from various sources.
- Weather API (https://openweathermap.org/api) - for processing real-time weather data.

2. Objectives: Describe the problem you are trying to solve. What are you trying to predict? Is the data imbalanced? Could your dataset be subject to concept drift? If so, why?

3. Data type conversion: Develop a base model must be studied and define the data type conversion (casting) that is required to be able to use River. Does your data require multiclass classification? Be aware of how to encode the output in that case.

4. Concept drifts. Use at least two of the concept drift detectors to check if there are any on your problem, in other case, could be an indication that your problem is not suitable for stream learning.

5. Batch learning with base model: For this, load the dataset (or data collected from the API), split the dataset into a training set and a test set, train with ML method, and then evaluate. This should provide a baseline for understanding default ML parameters. Put special attention in checking how the technique deals with the concept drifts.

6. Stream learning. Define and implement a “River” stream pipeline. Next, define a metric from the river API that could be used to evaluate the performance of your ML model (for example, those described in Unit 1 and Unit 2 of the tutorials).

7. Model selection/comparison. Use at least three complementary ML models within Stream with their appropriate pipelines. How should your models/code be modified to deal with the fact that the default values may not be suitable. For one of these, investigate the use of a Hoeffding Tree model (as described in Unit 2 and Unit 3).

8. Include some plot to exemplify the differences between the approaches and/or models.

9. Explain the results and give some conclusions.

Notes:

You are free to create the pipeline you consider appropriate to deal with minimum proposed requirements. However, your efforts will be considered for the evaluation. For instance, you can apply data engineering to create new features or remove some of the current ones. There are also different preprocessing operations that you can apply like normalization. The

pipeline can also include anomaly detection systems and drift detectors. There are many options in River that you can test. This practice will be developed in groups, so you can try different alternatives.