Symbolic Regression GECCO Competition - 2023 - Track 2 - Interpretability

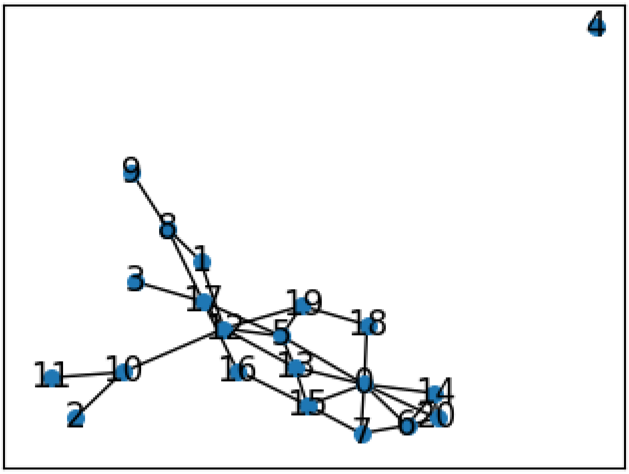
Team: C-Bio-UFPR
Participant 1: Adriel Macena Falcão Martins
Email: am.adriel.martins@gmail.com
Participant 2: Prof. Aurora Trinidad Ramirez Pozo
Email: aurora@inf.ufpr.br

Disclaimer: I'd first like to say that we've worked only for a month on this project. This is such an interesting project and we had many ideas to implement, but time was a constraint. We hope we can still contribute to this competition somehow.

Pipeline

First, we notice that our bike data could be seen as a spatial-temporal problem. We wanted to use spatial properties and also time properties to solve it. So we framed the problem of SR as a graph, where the nodes would be the bike lanes and the edges would only exist for bike lanes that were connected in reality.

So for that we turned each bike lane into a number. Also, as the information about bike lane connectivity is not easy to get, we simply did an experiment with ChatGPT where we asked him what the connectivity was. ChatGPT could be hallucinating, we are not sure. But still we thought it was a valid experiment, for the whole methodology will not be affected once the user of the techniques has more valid edges for the graphs.



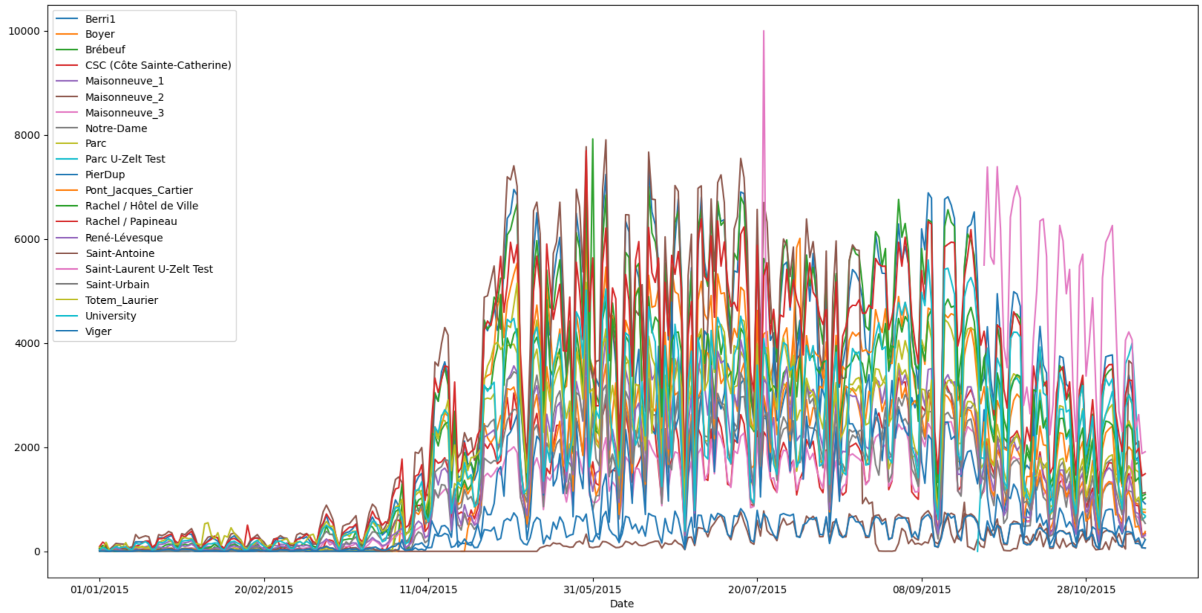
So what we actually want is to build multiple graphs, one for each date. Also, we want our nodes to carry two vectors: x and y. Where x means the K-lagged observations. y means the current (for that date) observation of that bike lane. In our case, we choose to set K=7, that means that each bike lane will carry the information of one week prior, besides its current value for that date. Because of the lack of time, we had to cut the 7 first observations for each bike lane. So at most, a bike lane will have 312 graphs, one for each date. One way to remedy this is for those 7 particular initial observations, use reverse lagged observations, going from "future to past", rather than "past to future".

After having our multiple graphs, we need to create our final dataframe. What we want is to use connected nodes plus the node itself to predict the value of y for any particular node. Or, to put it in another way, you want x from your neighbours and your own x, to predict your own y. So we constructed that in our code, a dataframe that reflects that.

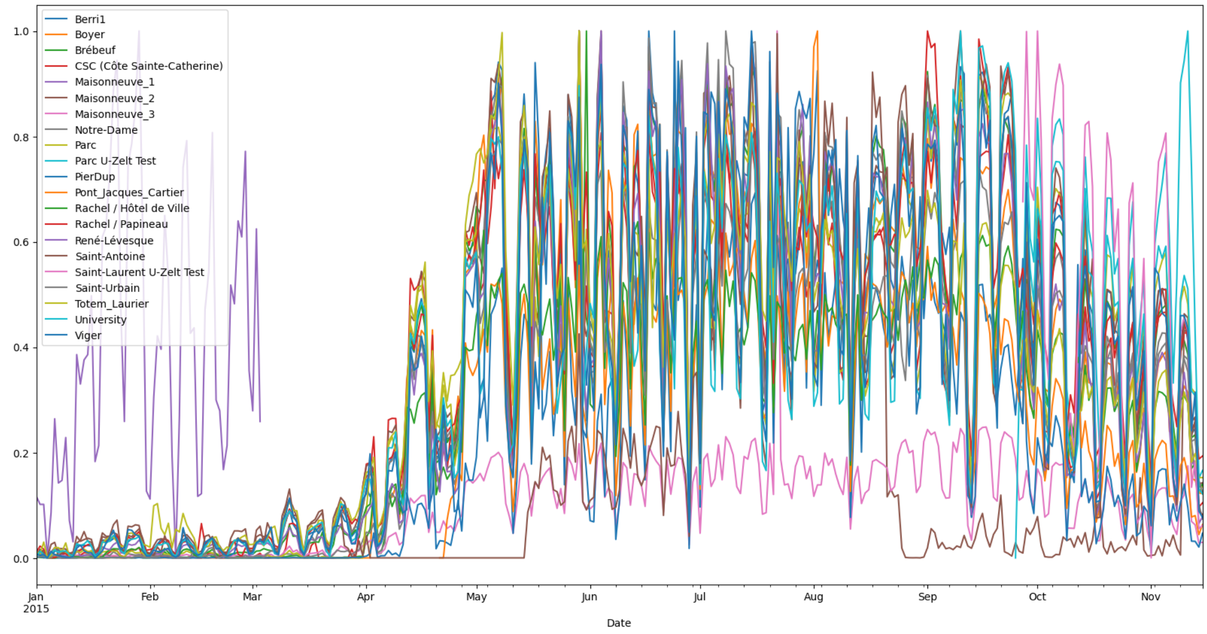
Finally, we create 21 models, one for each node, in which we relate spatial-temporal features via lagged observations and their neighbours lagged observations.

One thing that is vital is that we noticed that the data was not really stationary across time. This would pose difficulties for our SR algorithm. So we scaled the data using the MinMax algorithm for each bike lane separately. The good thing about this scaler is that it is a simple linear scaling that you can easily go back and forth.

So we went from this plot, which represents the count of people per bike lane per date:



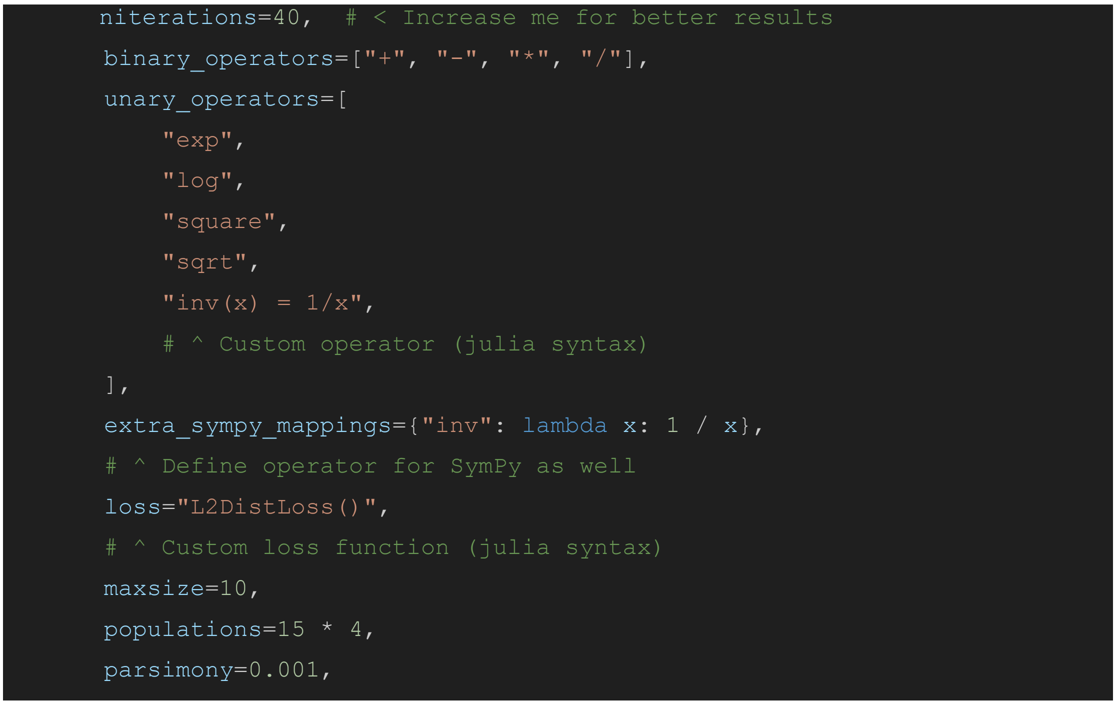
To this:



Algorithm

We used the classic tree based genetic programming using PySR library. One particular thing that we used was that as a bike lane could have many neighbours the data could then become a truly high-dimensional data so we choose to implement a simple random forest classifier to always select at most the top 10 features via this random forest classifier. We also used an L2 loss.

To cite some parameters of our tree-based GP:



```
niterations=40,  # < Increase me for better results
binary_operators=["+", "-", "*", "/"],
unary_operators=[
    "exp",
    "log",
    "square",
    "sqrt",
    "inv(x) = 1/x",
    # ^ Custom operator (julia syntax)
],
extra_sympy_mappings={"inv": lambda x: 1 / x},
# ^ Define operator for SymPy as well
loss="L2DistLoss()",
# ^ Custom loss function (julia syntax)
maxsize=10,
populations=15 * 4,
parsimony=0.001,
```

- Maxsize: Max complexity of an equation.
- Parsimony: Multiplicative factor for how much to punish complexity.
- The rest of the parameters were taken to be the default ones and can be checked here: https://astroautomata.com/PySR/api/

Analysis of the real world data set

There was not enough time to complete that in only one month.