Homework for Head of Data Science Candidates

"The one who knows all the answers has not been asked all the questions."

Confucius

Introduction

Create a notebook or source code that details how you developed a predictive model using Python for the data in the attached zipped *dataset.parquet* file. It should describe all phases of your modelling process that you went through, from exploratory data analysis, through feature engineering, to model selection, training, and validation.

You can use any Python data science, machine learning, or deep learning library.

Your eventual trained model will be evaluated on unseen data not found in *dataset.parquet*. While a high-performing model is always welcome, it is just as important to delineate your thought process and experiment design with concise comments, markdowns, and apt visualizations, and showcase your best clean coding practices.

In other words, your submission will not only be evaluated based on how well-performing your model is, but how you think and go about solving a prediction problem, and how neatly you can present them.

Details about dataset.parquet

- Its index is a CE(S)T-localized DatetimeIndex in hourly resolution starting from 2023-01-16 01:00 to 2023-09-25 00:00. It represents the target time but is not necessarily contiguous.
- It has 6048 rows.
- It has 34 columns:
    - 30 columns named *x01* … *x30* are features known at prediction time.
    - 1 column named *y* is the target variable not known at prediction time, corresponding to the dataset's index, a.k.a. the target time.

    - 1 column named *x_y_lagged* is also a feature and represents a lagged value of the target variable y known at prediction time.
    - 1 column named *z* is neither a feature nor a target variable, but a helper column to calculate our evaluation metric, as explained later. It is not known at prediction time and therefore must not be used during modelling, only when evaluating the model.
    - 1 column *x_z_lagged* is also a feature and represents a lagged value of helper column *z* known at prediction time.
- To reiterate, the dataset has three types of columns:
    - 32 features known at prediction time, named *x01* … *x30*, *x_y_lagged*, and *x_z_lagged*.
    - 1 target variable not known at prediction time named *y*.
    - 1 additional helper column named *z*, which is not a feature nor a target variable, and is not known at prediction time and should no be used during modelling, only when evaluating the model.
- The dataset contains a multi-dimensional timeseries, but it is not a conventional timeseries. Target variables have to be predicted in batches but their realized value is known after their target times. This happens according to the below schedule:
    - On day *D-1* at 11:30 CE(S)T, prediction is done at once for the below 24 hours:
        - Day *D* 01:00 … day *D* 23:00
        - Day *D+1* 00:00
    - Then on the same day *D-1* at 13:00 CE(S)T, the 24 realized *z* helper column values for day *D* 01:00 … day *D+1* 00:00 are published at once. This means that realized *z* values are not known as the target time (hours) pass one after another, but they are known the day before (for target time day *D+2* 00:00, two days before).
    - However, target variable *y* values are known one by one some time after the target time (hour) is passed.
    - This batch dynamic can be seen when looking at the values of *x_y_lagged* as compared to *y*, and the values of *x_z_lagged* as compared to *z*.
    - This schedule needs to be taken into consideration to correctly use lagged values of other features, as illustrated by the following:
        - On day *D* at prediction time, all 32 feature values corresponding to the 24 to-be-predicted hours day *D* 01:00 … day *D+1* 00:00 are available and known at once.
        - This means that for a given feature *x* corresponding to target time day *D* hour *n*, not only “past” lagged values from before day *D* hour *n-1* could be used, but also “future” lagged values from day *D* hour *n+1* to day *D+1* 00:00, but not further, because feature *x*’s value corresponding to day *D+1* 01:00 and beyond is not known at prediction time on day *D-1*.

        - Of course, given the batch nature of the underlying timeseries, these “future” values are not actually future values but are known at prediction time.
        - Just to reiterate, the above is true for *x_y_lagged* and *x_z_lagged* as they are features, but is not the case for *y* and *z*.

Evaluation metric

Models will not be evaluated using conventional accuracy metrics, but by the following custom metric named *pnl_score,* calculated as follows:

- Let’s call *yhat* the prediction of *y*.
- If at target time *t* ($yhat \geq z$ and $y \geq z$) or ($yhat < z$ and $y < z$), i.e. *yhat* is on the same side of *z* as *y*, then we define the following four numbers as:
    - $pnl_t = |y_t - z_t|$
    - $pnl\_max_t = |y_t - z_t|$
    - $hit_t = 1$
    - $hit\_max_t = 1$
- If at target time *t* ($yhat \geq z$ and $y < z$) or ($yhat < z$ and $y \geq z$), i.e. *yhat* is on the opposite side of *z* as *y*, then the definition of the above numbers are:
    - $pnl_t = -|y_t - z_t|$
    - $pnl\_max_t = |y_t - z_t|$
    - $hit_t = 0$
    - $hit\_max_t = 1$
- Over *n* samples (rows) of prediction, we define the below metrics:

$$pnl\_pct = \frac{\sum_{t=1}^{n} pnl_t}{\sum_{t=1}^{n} pnl\_max_t} * 100$$

$$win\_pct = \frac{\sum_{t=1}^{n} hit_t}{\sum_{t=1}^{n} hit\_max_t} * 100$$

$$pnl\_score \ = pnl\_pct \ * win\_pct \ / \ 100$$

While a good model maximizes *pnl_score*, it does not mean that your machine learning algorithm(s) of choice should use the above as an objective or loss function, but that your modelling process’ ultimate goal should be to maximize *pnl_score*.

Because of the binarized nature of the evaluation metric *pnl_score*, you can treat the problem not as a regression, but as a classification. However, your eventual predictions should still be real numbers and not classes.

Accepted submission file formats

- Python file with .py extension
- Jupyter (Lab) Notebook with .ipynb extension

Special requirement

- Please indicate the specific libraries and their versions alongside with the Python version used in an appropriate format so that your experiment be reproducible.