Enhancing Time Series

Methods and How To

FEV.2022

About Me

Leandro Sartini

Age of 26, currently working for mercado livre

as a Data Scientist, focused on MarketPlace

Education:
Computer Engineering, University SENAC, Brazil

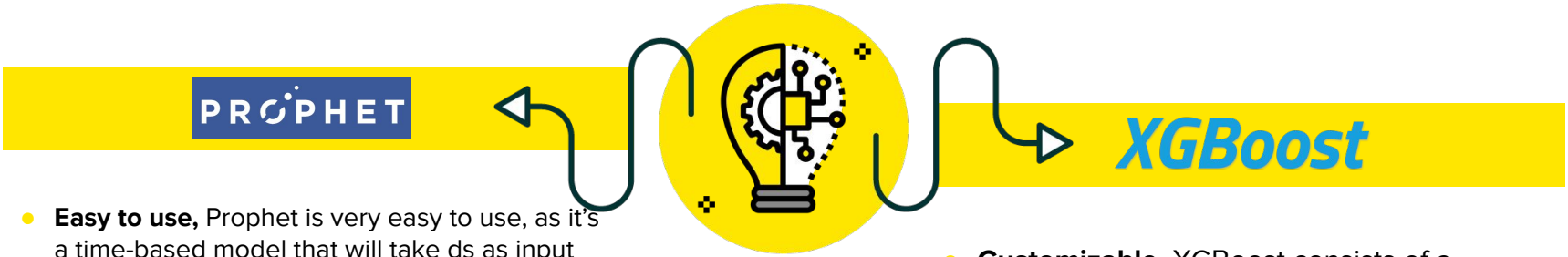
Hobbies:
Gamer;
Gym Rat;
Snowboarder.

LinkedIn Profile

Agenda

- Models used | XGBoost & Prophet
- XGBoost as a Time Series? How
- Validation Methods | OOT & OOS
- What is Train, Test and Val?
- Conclusion and Links

Prophet & XGBoost - Pros and Cons



- **Easy to use,** Prophet is very easy to use, as it's a time-based model that will take ds as input variable and y as output variable.
- **Nice built-in features,** You can create plots to determine whether your signal is a noise or an actual signal, you can predict the future in a plot with one line of command.
- **Not a category friend,** you will have to create many models for each category, and that goes for evaluation too.
- **Imagination helps but not so much,** since it's a time based model, it accepts "regressors" as additional variables and while they do make a change, sometimes your imagination with variables won't work so well.

- **Customizable,** XGBoost consists of a gradient-boosted decision tree that has loads of customizable parameters.
- **Creativity builds the model,** It can accept all your created variables, and you can test them how you want.
- **Category Friend,** you can build one model for all your troubles, leaving it easier to when automating the model.
- **Harder to use,** while it's easy to use, to make it a good time based model, will request a lot of effort.

XGBoost as Time Based Model

To create our time based XGBoost we will use data that already exists in our past, a sliding window of that data has to be created, and our model will be stuck with that prediction range.

You have to define how long you need to predict, it could be 24h, 1 week, 1 month, etc.

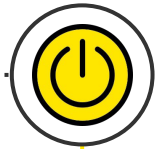
Define your prediction range

Create sliding windows with your time variables in different columns.

Create sliding windows

With XGBoost you can use custom variables other than the data.

Aggregate custom variables



Make use of the evaluation methods for users and time.

Evaluate

Automate your single model and predict with that

Automate & Predict

We have to be **extremely careful** on this part, specially because we are using our **data from the past to make predictions in the future** AKA our **sliding window**, some examples to avoid mistakes:

- You have to use your **day, month, year variables into separate columns**, sometimes year it's not very useful since the tree never saw that.
- Since our problem is with a timeseries the **data that the model is reading is extremely important**, then **aligning your day of week with day of week** can be very helpful, for example using **-7 days** will always align **last Tuesday with actual Tuesday**, or **-28 days** with the **same Tuesday from last month.**

Evaluating & Predicting

Building

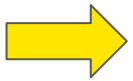
XGBoost as Time Based Model

```
In [28]: df_xgb['day'] = df_xgb['ds'].dt.day
         df_xgb['month'] = df_xgb['ds'].dt.month
         df_xgb['year'] = df_xgb['ds'].dt.year
```

```
In [29]: df_xgb.sample(2)
```

Out[29]:

	ds	postcode	y	propertyType	bedrooms	property_type	day	month	year
1757	2009-11-04	2615	472500	house	4	1	4	11	2009
8379	2014-02-11	2611	461000	house	3	1	11	2	2014



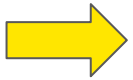
Creating Time Variables

```
In [30]: df_xgb.sort_values(['propertyType','postcode','ds'],ignore_index = True,inplace = True)
```

```
In [31]: df_xgb.sample(2)
```

Out[31]:

	ds	postcode	y	propertyType	bedrooms	property_type	day	month	year
4051	2019-07-20	2605	708000	house	3	1	20	7	2019
21950	2017-05-08	2913	835000	house	4	1	8	5	2017



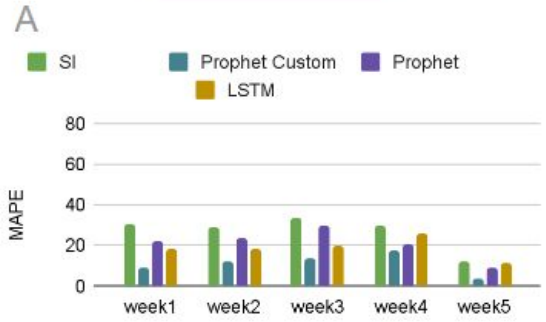
Sliding variables 28 days

This is tricky, we have to group it by our other "category" variables and then we have to shift without the dat grouped df

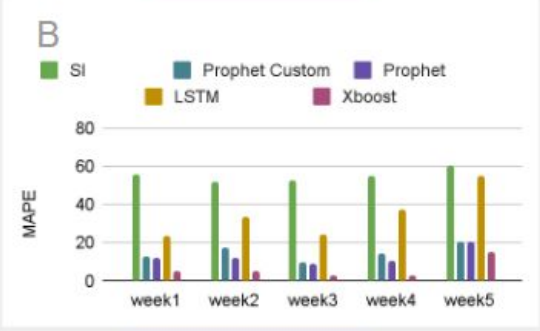
```
In [32]: df_xgb['bedrooms_var'] = df_xgb.groupby(['propertyType','postcode'])['bedrooms'].shift(28)
```

Results in real world scenario

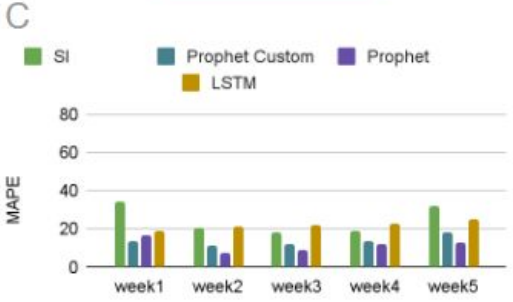
CATEGORY A



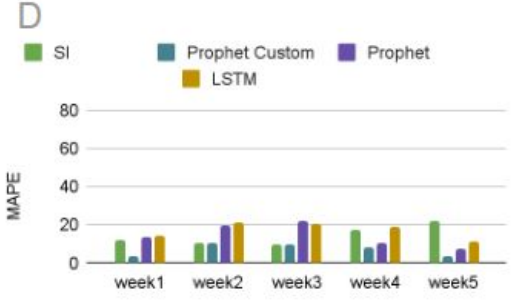
CATEGORY B



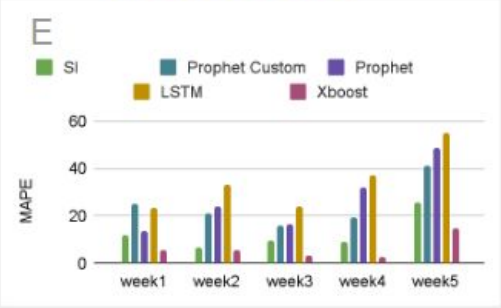
CATEGORY C



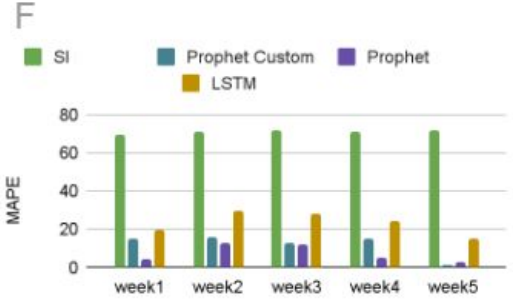
CATEGORY D



CATEGORY E



CATEGORY F



Each category is unique and we can see that for category B and E, which were the most volatile categories, we have a very nice response of XGBoost against all the other categories.

What is Train, Test and Val?

Coloque o subtítulo, quando for necessário

01

Train

Train also know as fit is where your model will learn from your data

02

Test

Test and some regions can call validation, is when you are testing your trained model hyperparameters

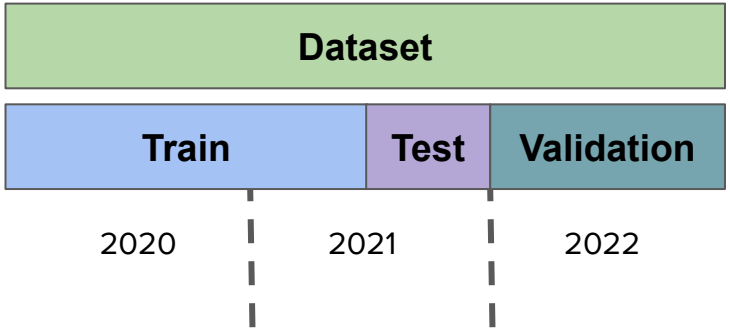
03

Validation

Validation that can be called test, is the final step where you are testing your scenario, it could be time, sample, etc.

Out of Time Validation

1. You have to separate the last time period of your data for out of time validation.
2. It will help you know how your model will work in the future while automated.
3. It helps you prevent overfitting and put your mind at ease with an automated model.



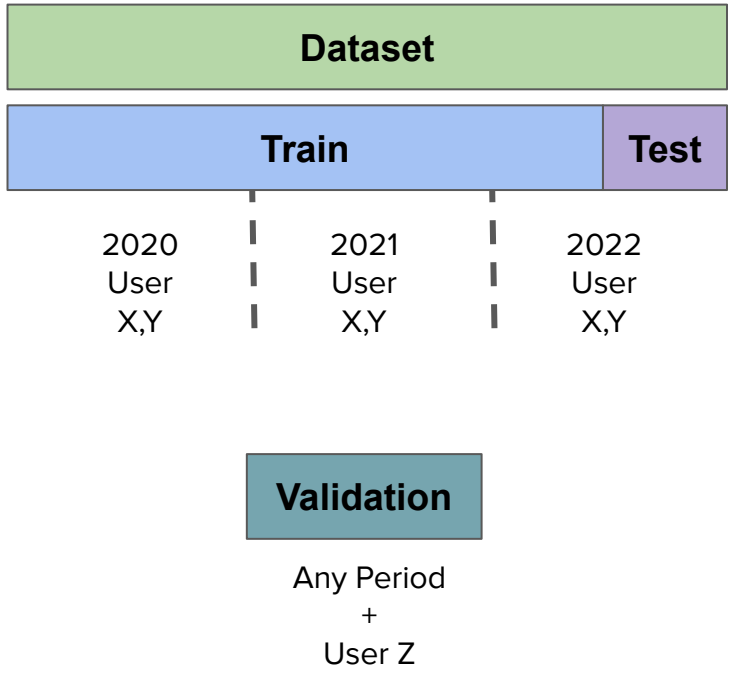
Out of time validation

```
In [33]: train_df = df_xgb.loc[df_xgb['ds'] < '2019-05-01'].reset_index(drop = True).copy()

In [34]: train_df = train_df.loc[train_df['bedrooms_var'].notnull()].reset_index(drop = True).copy()

In [35]: test_df = df_xgb.loc[df_xgb['ds'].between('2019-05-01','2019-05-31')].reset_index(drop = True).copy()

In [36]: val_df = df_xgb.loc[df_xgb['ds'] >= '2019-05-31'].reset_index(drop = True).copy()
```

Out of time Validation focusing on knowing if our time series model will deprecate too much in time

Out of Sample Validation

1. You have to separate the last time period of your data for out of time validation.
2. It will help you know how your model will work in the future while automated.
3. It helps you prevent overfitting



- Use Case GitHub with a test on House Property Sales | Kaggle -> GitHub Link
- How to Use XGBoost for Time Series Forecasting
- XGBoost docs
- Prophet Quick Start