# LightGBM Model

LightGBM (Light Gradient Boosting Machine) is a distributed high-performance framework that uses decision trees for ranking, classification, and regression tasks (Saha, 2023). In contrast to the level-wise(horizontal) growth in XGBoost, LightGBM carries out leaf-wise (vertical) growth that results in more loss reduction, higher accuracy while being faster (Saha, 2023). However, this approach may lead to the overfitting of the training data (Saha, 2023).

For the experiment and the improvement attempts, the training data set will be used. The data set will be split for cross-validation with k = 10. The experiments with LightGBM are organized as below:

* Exploring preprocessing techniques:
  + An initial run was done with all the features available in the training data set, using Scikit-learn’s TimeSeriesSplit for Cross-Validation, without scaling the data using the StandardScaler(), and using all default hyperparameter values of LGBMRegressor.
  + Test the effect of applying StandardScaler() with the rest of processing and setting equals to the initial run.
  + Test the effect of using normal k-fold cross-validation (non-time series data) with the rest of processing and setting equals to the initial run.
* Exploring Feature Engineering
  + The first test will check if removing seemingly unrelated features will affect or improve the training performance.
  + The second test will check if extra engineered features help.
    - First extra sets– imbalance calculator
    - Second extra sets – incorporate the movement of the reference price, this include the first derivative, which is the price of current minus the price of previous 10 seconds, and the second derivative, this is the change of the first derivative in the last 10 seconds.
* Exploring hyperparameter tunings of the LightGBM model
  + The hyperparameters considered include n\_estimators, learning\_rate, num\_leavers, max\_depth, min\_data\_in\_leaf, max\_bin, min\_gain\_to\_split

# Exploring preprocessing techniques

## Initial Trial

Figure 1 shows the training and validation scores for the initial run using all features available in the training data set and Scikit-learn’s TimeSeriesSplit of 10 for cross validation. The main observations are as follow:

Observation 1: The average validation scores are about 5% higher than the average training scores, this indicates potential overfitting of the model to the training data.

Observation 2: The validation score seems to be improving when more data is used for training. This indicates that the series might be stationary without a trend or seasonal component. Note when using the Scikit-learn’s TimeSeriesSplit function, the testing data of the first fold is added into the training data of the second fold (see figure 2 for illustration), thus the training data set will continue expands.

Since the number of days in the training data set is 480, one concern is the original split (TimeSeriesSplit of 10 will split the training data set into 11 parts) may contain fractional data of the day which affect the performance of the model. A quick test was performed using TimeSeriesSplit setting of 9 (which split the data into 10 parts) with result shown in figure 3. As the resulting models didn’t perform significantly better than the previous models, it is concluded that the number of splits has no significant affect to the performance.

A screenshot of a computer screen

Description automatically generated

Figure : Training scores and Validation scores for the initial run

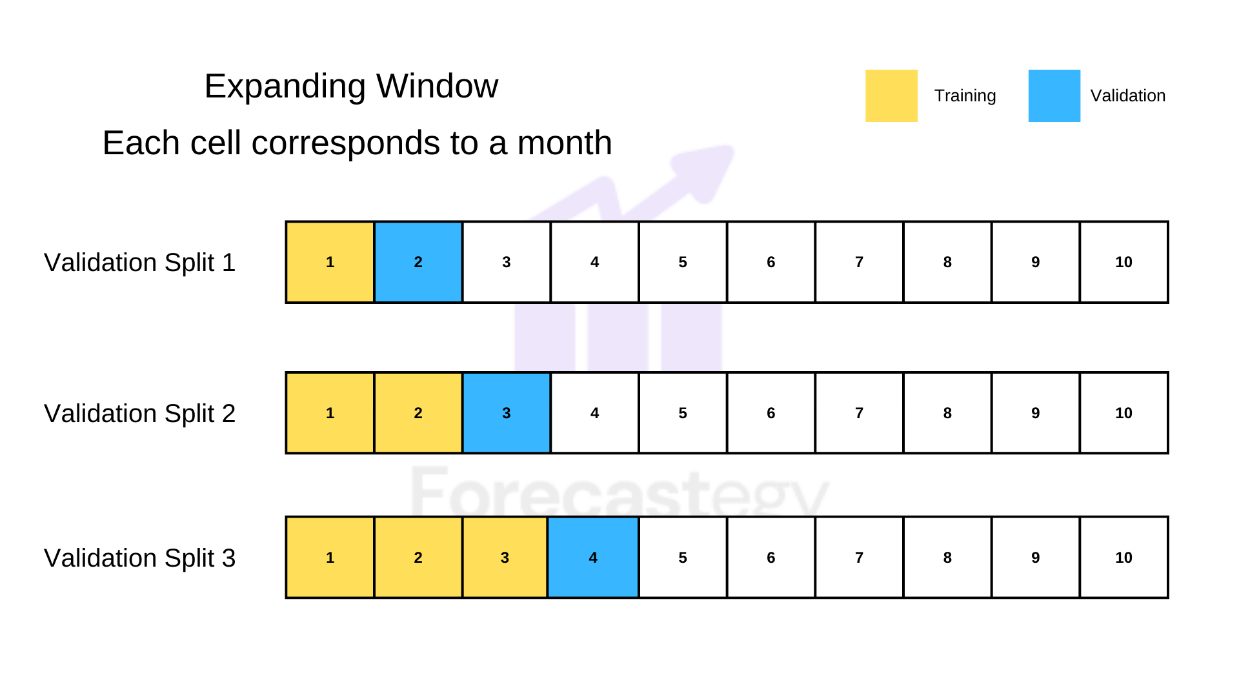


Figure : Expanding Window Time Series Split Validation (Source: (Filho, 2023))

A screenshot of a computer screen

Description automatically generated

Figure : Training scores and Validation scores for the initial run, timeseries split of 9

## Initial Trial with Standard Scaler Applied

This test applied StandardScaler() to the following numerical categories: imbalance\_size, reference\_price, matched\_size, far\_price, near\_price, bid\_price, bid\_size, ask\_price and ask\_size. The results are shown in figure 4. It doesn’t seem to have significant improvement compared to training data without StandardScaler() applied.

A screen shot of a computer

Description automatically generated

Figure : Training scores and Validation scores for the initial run with standard scaler applied.

### Initial Trial with k-fold cross validation

This test uses the k-fold instead of TimeSeriesSplit for cross validation. The results are shown in figure 5. On average the model performs better compared to the model trained using the TimeSeriesSplit. Besides, there are two observations:

Observation 1: The is one part of the data (the first part) where the model predicts well.

Observation 2: The model performs better in the validation set when it is underfit the training data. This happens when the validation score is lower than the training score achieved.

A screenshot of a computer screen

Description automatically generated

Figure : Training scores and Validation scores for the initial run with k-fold cross validation

## Summary for Preprocessing Techniques

Table 1 shows the summary of the comparison of the performance of the models using different preprocessing techniques. Based on the results, it can conclude that,

* The training data exhibits stationary property with the variations independent of time, hence processing the data in time series is not necessary.
* The best performing model is likely to be the model that underfits the training data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Title | Training Score | | Validation Score | |
| Average | Best Validate | Average | Best |
| Initial Run | 6.14128 | 6.34218 | 6.47656 | 5.88523 |
| With TimeSeriesSplit of 9 | 6.15360 | 6.34312 | 6.47738 | 5.90742 |
| With StandardScaler | 6.14125 | 6.34143 | 6.48478 | 5.88302 |
| With kfold | 6.29844 | 6.42056 | 6.32005 | 5.23449 |

Table : Summary comparison of performance of the models for different preprocessing techniques

# Reference

1. Saha, S. (2023, August 8). *XGBoost vs lightgbm: How are they different*. neptune.ai. <https://neptune.ai/blog/xgboost-vs-lightgbm>
2. Filho, M. (2023, July 12). How to do time series cross-validation in Python. Forecastegy (Alt + H). https://forecastegy.com/posts/time-series-cross-validation-python/