Optiver – Trading at the Close

Zilu Ma, Shaozong Wang

Introduction

In this project, we aim to take the Kaggle competition hosted by Optiver on predicting US stock closing movements. We will have auction data delivered through API offered by Optiver.

We are given time series data for 20 stocks within some time frame, each with 13 features including ask/bid prices, ask/bid sizes, wap, etc, where wap is the weighted sum of ask and bid prices according to their sizes. The goal is to use these features to predict 60 second future move in the wap of the stock, less the 60 second future move of the synthetic index. Concretely, the target is

$$\frac{\text{StockWAP}_{t+60}}{\text{StockWAP}_t} - \frac{\text{IndexWAP}_{t+60}}{\text{IndexWAP}_t}.$$

Submissions are evaluated on the Mean Absolute Error (MAE) between the predicted move and the observed target. Concretely, $\text{MAE}(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$, where $y, \hat{y} \in \mathbb{R}^n$ are observed and predicted values for n samples, respectively.

Approach

We first generate more features with financial significance, which plays a major role according to our experiments. For example, we consider imbalance and difference between various prices and statistics of prices and sizes, etc. Since we deal with time series data, we also consider discrete derivatives, rolling averages, etc. Eventually we generated 100-200 features for various experiments.

Since the number of features is relatively small, we mainly use gradient boosting tree model, which is one of the favorite models for this type of competitions. Now there are efficient and robust packages such as XGBoost, CatBoost, LightGBM, etc.

Results

Among numerous experiments, we select a few representative results to present. Due to the limits of submissions and time constraints, we mainly test the models on validation sets and submit a few.

The performance does improve after introducing more features such as discrete derivatives of prices and sizes. The models are quite insensitive to the change of learning rates and other hyperparamters in the tree models. XGBoost models tend to be trained faster but with higher loss, and CatBoost models have less loss and they seem to be more flexible dealing with noisy data. We eventually tested on ensembles of the two models with larger weights on CatBoost.

Models	#features	learning rate	validation loss	submission loss
CatBoost	106	0.005	5.89	5.99
CatBoost	106	0.05	5.86	6.21
CatBoost	128	0.005	5.89	5.58
CatBoost	138	0.005	5.89	5.60
XGBoost	128	0.05	5.986	5.60
XGBoost	138	0.05	5.986	5.60
Ensemble(0.7×Cat+0.3×XGB)	138	0.005, 0.05	5.90	**5.53**