Jane Street: Real-Time Market Data Forecasting

[*]Machine Learning HW2: Kaggle Competition

Liang-Jen Huang
Institute Of Data Science, NCKU
December 23, 2024

***Abstract*—This competition involves building machine learning models to predict the behavior of responders in financial markets using provided historical market data. These behaviors may include trading decisions, order executions, and other related activities. Accurate predictions can significantly enhance the effectiveness of trading strategies. The competition details are available at: https://www.kaggle.com/competitions/jane-street-real-time-market-data-forecasting**

I. Introduction

Financial markets are highly dynamic and influenced by a variety of factors, such as market volatility, economic indicators, and global news events. Predicting the behavior of market participants is critical for designing effective and profitable trading strategies.

The competition provides a dataset that captures the historical state of the financial market at multiple time points, along with trading activities. The objective is to analyze this data, extract meaningful features, and develop predictive models to forecast future market responder behaviors. The primary target variable is *responder6*, which represents specific market behavior.

The dataset comprises several components to facilitate model development. The training data includes historical market data and responder behaviors. Lagged data offers responder values from the previous day, enabling the incorporation of temporal dependencies. A test set mimics the structure of data served during deployment, providing a framework for real-time forecasting. Metadata for the market features and responders is available in features.csv and responders.csv. To ensure data confidentiality, the financial products, market features, and responders have been anonymized.

This competition represents a practical application of machine learning in a complex and fast-paced financial market environment. It provides an opportunity to address real-world challenges in market forecasting and to develop robust solutions under realistic conditions.

II. Method

In this analysis, various data processing techniques and modeling strategies were progressively employed to address the prediction problem effectively. The steps undertaken are detailed as follows:

A. Handling Missing Values

To handle missing values, two methods were tested. The first approach involved filling the missing values with the median of the corresponding column, which is simple and computationally efficient. The second approach used the K-Nearest Neighbors imputation method, which fills missing values based on the relationships between neighboring data points. However, the experimental results indicated little difference in performance between the two methods, while KNN imputation was significantly more time-consuming. Thus, for efficiency and accuracy, the median imputation method was chosen for the remainder of the analysis.

B. Incorporating financial instrument into the Analysis

Initially, only the market features were used for modeling. However, considering that financial products might significantly influence the response behavior, symbol_id was incorporated into the analysis. To achieve this, symbol_id was transformed into multiple dummy variables using One-Hot Encoding. This allowed the model to better capture the potential impact of financial products on the target responses.

C. Adding Lagged Features

Lagged data was introduced to capture the temporal characteristics of the market. These lagged features were designed based on the previous day's date, time, and corresponding financial product data. This additional context provides valuable information for prediction. However, since lagged features require matching the same product at the same time from the previous day, not all records had corresponding data, leading to a reduction in the number of samples in the training dataset.

D. Feature Normalization

To ensure stability during model training, two normalization methods were tested: Min-Max Normalization and Standard Normalization. Both methods aim to scale features to a consistent range, thereby reducing the impact of varying feature scales on the model. The experimental results showed negligible differences between the two methods, so the choice of normalization method was tailored to specific scenarios as needed.

E. Splitting the Data into Training, Validation, and Testing Sets

The dataset was split based on temporal order, with the last 20% reserved as the test set. The remaining data was used for training and validation. This time-based split ensures that the evaluation simulates real-world prediction scenarios while avoiding data leakage from future data into the training process.

F. Feature Selection Using LightGBM

Feature importance scores from the LightGBM model were used to perform feature selection. This method evaluates the contribution of each feature to the model's predictions, allowing the selection of the most impactful features. By focusing on the most relevant features, the efficiency and accuracy of the model were improved.

G. Modeling

Based on previous experience with time series models, three models were employed: Lasso, LightGBM, and XGBoost. Cross-validation was used to tune the hyperparameters of each model:

For Lasso, the alpha parameter was adjusted to control the regularization strength, balancing model complexity and robustness. For LightGBM, Grid Search was used to optimize hyperparameters such as learning rate and number of leaves. For XGBoost, Grid Search was also employed to fine-tune learning rate and max depth to control the learning rate and tree complexity, respectively. Through these steps, the models were comprehensively evaluated, and the best-performing model was selected based on its accuracy and stability for forecasting in financial markets.

III. Experiments

Due to memory limitations, I sampled 400,000 records from each of the ten data partitions. After adding Lag features, the final dataset retained 382,094 records for training. Without Lag features, 100,000 records were sampled from each partition for training. The details of the experimental versions are as follows:

- **ver.1**: Used only market feature columns and filled missing values with the median.
- **ver.2**: Included financial product information by encoding it into dummy variables with One-Hot Encoding and filled missing values with the median.
- **ver.3**: Added Lag features and filled missing values with the median.
- **ver.4**: Added Lag features, filled missing values with the median, and applied LightGBM's Feature Importance to pre-select important features for modeling.

However, my submissions for ver3 and ver4, which included lagged data, encountered errors when uploading to the Kaggle competition. As of writing this report, the issue remains unresolved. As a result, the experiment results presented here are based on datasets I split locally. My best performance on Kaggle so far was achieved with a version that did not include lag but utilized LightGBM for feature selection with the XGBoost model, achieving an R^2 score of 0.0047.

In this experiment, XGBoost performed the best across all scenarios, demonstrating its strong nonlinear modeling capabilities and adaptability to sparse features. However, adding Lag features and performing feature selection (Ver.3 and Ver.4) did not yield the expected improvements and even resulted in some decline in performance. This may be due to the process of handling Lag features, where data was filtered to ensure proper alignment of time and date, which reduced the training dataset size. The insufficient data volume further limited the model's performance, particularly for models like LightGBM that heavily rely on large datasets.

TABLE I
Ver.1

	R2	RMSE	MAPE
LASSO	0.0027	0.8489	0.1402
LGBM	0.0061	0.8588	0.2782
XGB	0.0064	0.8412	0.3209

TABLE II
Ver.2 (+financial product)

	R2	RMSE	MAPE
LASSO	0.0029	0.8416	0.1361
LGBM	0.0065	0.8401	0.2613
XGB	0.0067	0.8401	0.2839

TABLE III
Ver.3 (+lag)

	R2	RMSE	MAPE
LASSO	0.0027	0.8412	0.0512
LGBM	0.0038	0.8509	0.0219
XGB	0.0044	0.8619	0.0518

TABLE IV
Ver.4 (+lag, select features)

	R2	RMSE	MAPE
LASSO	0.0027	0.8473	0.0480
LGBM	0.0040	0.8467	-0.021
XGB	0.0045	0.8465	0.0443

IV. Conclusions

This experiment demonstrated that XGBoost performed the best across all scenarios, showing strong adaptability to sparse data and limited dataset sizes. However, the results after adding Lag features and performing feature selection did not meet expectations, likely due to the reduced data volume impacting the model's learning capability. Future work should focus on implementing more thorough preprocessing steps, such as optimizing the Lag features by ensuring consistent Lag values for the same day or developing more effective methods to impute missing Lag values. This will help improve the robustness and accuracy of the predictive model.

V. Discussion & Future Work

- I suspect that the submission issue related to the lag-added data arises from the interaction between the hidden test data and the lag combination. Although the code executes smoothly in the Kaggle online Notebook, successfully generating the expected output and log files, the submission fails during the competition upload without providing any specific error messages. Despite dedicating significant time to troubleshooting this problem, I have not yet identified an effective solution. The relevant submission file can be accessed at the following link:https://www.kaggle.com/code/edogawaliang/version3

- I had considered grouping different financial products, so I attempted to perform PCA on all features, reducing them to one dimension, and Figure 1 showing the changes of financial products over time. However, these charts did not reveal any distinct time trends among the financial products, indicating that there were no significant clustering patterns. This may suggest that the features of the financial products in the current dataset do not exhibit strong time-related characteristics, or that the clustering structure is too subtle to be directly observed through this approach.
- Figure 2 shows that the target variable (Responder 6) exhibits low volatility, making it easier for the model to learn and fit. However, this may lead the model to produce predictions close to the mean, limiting its generalization ability in more complex scenarios. To address this issue, the target variable can be transformed, such as using a difference transformation to represent the target as relative changes. This allows the model to focus on learning the change patterns rather than the absolute values. After the model completes learning, the predicted differences can be accumulated to restore the original scale, ensuring the results align with practical requirements.

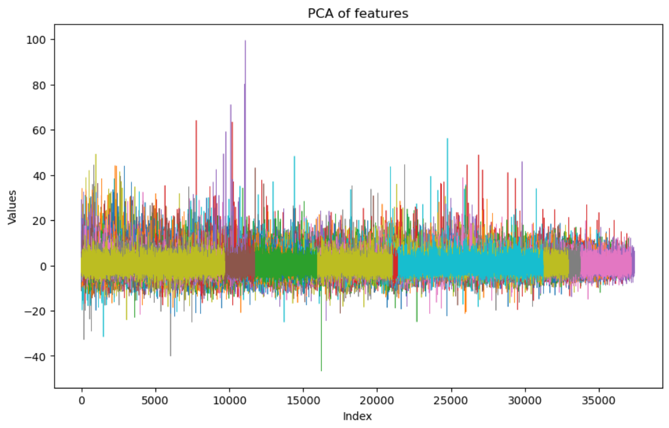

Fig. 1. Feature Variation of Different Financial Products Over Time

S

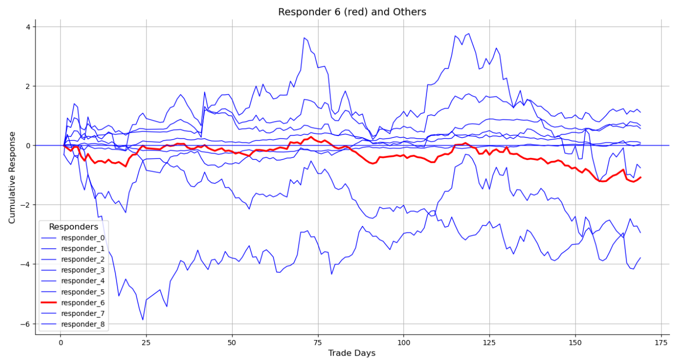

Fig. 2. Responder 6 Volatility Over Time