Stock Market Forecasting at the Close

CS7641 Machine Learning Final Project

View on GitHub

Introduction/Background

The stock market is an exceptionally dynamic environment where even the smallest fluctuations can significantly impact global financial systems. One of the most critical periods during the trading day is the final minutes before the market closes. During this time, traders rush to execute their last decisions, leading to heightened volatility and substantial price movements. Closing prices hold particular importance because they serve as key indicators of market sentiment and stock performance. They are often used as reference points for valuing portfolios, calculating indices, and informing investment strategies.

To ensure an orderly and fair determination of closing prices, Nasdaq employs a mechanism known as the Nasdaq Closing Cross. This system aggregates buy and sell orders submitted throughout the day, especially in the final moments, and matches them in a single closing auction. The Nasdaq Closing Cross establishes a definitive closing price for each security, reflecting the true supply and demand at the end of the trading session. By doing so, it enhances transparency and reduces volatility associated with the closing process. Moreover, this mechanism provides valuable insights into individual securities and contributes to assessing the overall health of the market.

Literature Review:

Stock market prediction has focused on price movements, volatility, and external data. For example, [1] used SVMs for stock direction, [2] explored neural networks, and [3] utilized three algorithms to predict upcoming stock prices. However, little research targets the final ten minutes of trading, a period of high volatility. The Optiver Kaggle competition provides a valuable dataset for this.

Dataset Description:

The dataset is available through the **Optiver - Trading at the Close** competition on Kaggle. It contains historical data for the daily ten-minute closing auction on the NASDAQ stock exchange. The primary objective is to predict the future price movements of individual stocks relative to the future price movement of a synthetic index composed of NASDAQ-listed stocks.

The dataset contains a variety of features, and some of them are described as follows:

- stock_id: A unique identifier for each stock.
- date_id: A unique identifier for each date.
- imbalance_size: The amount of unmatched orders at the current reference price. This represents the difference between buy and sell orders that are not paired at the current price.
- imbalance_buy_sell_flag: An indicator reflecting the direction of the auction imbalance:
    - 1: Buy-side imbalance (more buy orders than sell orders at the reference price).
    - -1: Sell-side imbalance (more sell orders than buy orders at the reference price).
    - 0: No imbalance.
- reference_price: The price at which the number of paired shares is maximized, the imbalance is minimized, and the distance from the bid-ask midpoint is minimized, in that order. It can also be thought of as equivalent to the near price bounded between the best bid and ask price.
- matched_size: The amount that can be matched at the current reference price.
- far_price: The crossing price that will maximize the number of shares matched based solely on auction interest. This calculation excludes continuous market orders.
- near_price: The crossing price that will maximize the number of shares matched based on both auction and continuous market orders.
- bid_price / ask_price: The price of the most competitive buy/sell level in the non-auction order book.
- bid_size / ask_size: The dollar notional amount at the most competitive buy/sell level in the non-auction order book.
- wap: The weighted average price in the non-auction order book, calculated using the formula: $WAP = \frac{BidPrice \times AskSize + AskPrice \times BidSize}{BidSize + AskSize}$
- seconds_in_bucket: The number of seconds elapsed since the beginning of the day's closing auction, always starting from 0.
- target: The 60-second future movement in the weighted average price (WAP) of the stock, adjusted by subtracting the 60-second future movement of the synthetic index. This column is only provided in the training set.

Problem Definition

Our objective is to develop a robust predictive model that can accurately forecast the 60-second future movement in the weighted average price (WAP) of individual stocks during the closing auction, adjusted for market-wide movements using a synthetic index. This involves:

- Analyzing Relevant Features: Utilizing the rich set of related data to understand the supply and demand dynamics at play.
- Capturing Rapid Market Movements: Developing models capable of handling the high volatility and rapid changes characteristic of the closing auction.
- Adjusting for Market Trends: Accounting for overall market movements by comparing individual stock movements to a synthetic index, isolating stock-specific factors.

By addressing these challenges, we aim to fill the existing gap in prediction accuracy during the closing auction period.

Motivation

Accurate predictions of stock price movements during the closing auctions are of significant importance for several reasons:

- Optimizing Trading Strategies: Traders and institutional investors rely on precise forecasts to make informed decisions about order placement and execution timing. Improved predictions can lead to better execution prices, reduced transaction costs, and enhanced portfolio returns.
- Risk Management: The closing auction often sees large orders being executed, which can lead to significant price movements. Accurate predictions help in mitigating the risk associated with price slippage and adverse market movements.
- Regulatory Compliance: For certain market participants, such as mutual funds and pension funds, end-of-day pricing is critical for reporting and compliance purposes. Better prediction models assist in ensuring that asset valuations are accurate and reflect true market conditions.

Our project is motivated by the potential to make a meaningful impact in the financial industry by providing tools and insights that enhance decision-making during one of the most critical periods of the trading day. Ultimately, our goal is to contribute to the field of finance by developing methodologies that can be adopted by practitioners and researchers alike, fostering better understanding and management of financial markets during the closing auction.

Methods

To address the problem of predicting the weighted average price (WAP) of stocks during the closing auction, we implemented a combination of data preprocessing techniques and machine learning algorithms.

Data Preprocessing and Dimensionality Reduction

1. **Feature Selection and Engineering**:
    - Selected key price indicators: WAP, bid/ask prices, reference price, near price
    - Volume metrics: bid/ask sizes, matched_size, imbalance_size
    - All features normalized relative to opening WAP
    - Created derived features to capture price, volume and temporal information
    - Removed highly correlated features to reduce redundancy
2. **Data Standardization**:
    - Applied StandardScaler to normalize feature distributions
    - Handled missing values using mean imputation
    - Removed outliers beyond 3 standard deviations
    - Applied log transformation to volume features
    - Validated data quality through distribution analysis
3. **Principal Component Analysis (PCA)**:
    - Employed Recursive Feature Elimination (RFE) to reduce the number of features before applying PCA for further dimensionality reduction
    - Reduced feature dimensionality while preserving information
    - Retained components explaining 95% of variance
    - First component captured price movements (65% variance)
    - Second component represented volume dynamics (20% variance)
    - Enabled effective visualization and cluster separation
4. **Dataset Construction**
    - Split the data into training and validation sets based on the date of the stock information
    - Optimized data formats to reduce memory usage, enabling smoother experimentation

Machine Learning Models

1. **K-Means Clustering**:
    - Implemented to identify distinct market states
    - Optimal cluster count (k=3) determined via silhouette analysis
    - Revealed three distinct trading patterns
2. **Supervised Regression Models**:
    - Employed a series of supervised regression models to predict market trends and price movements
    - The implemented models include:
        - Linear Regression: Basic linear regression model for predicting continuous outcomes
        - Lasso: Linear model with L1 regularization to enhance feature selection
        - ElasticNet: Combines L1 and L2 regularization for balanced feature selection and model complexity
        - Ridge Regression: Linear model with L2 regularization to prevent overfitting
        - Huber Regressor: Combines linear regression with robust loss functions to handle outliers effectively
        - LightGBM: A gradient boosting framework optimized for speed and performance, ideal for handling large-scale datasets and achieving high predictive accuracy
3. **Sequential Neural Networks**:
    - **RNN**: Focused on modeling sequential patterns in data
    - **LSTM**: Specialized in retaining and leveraging long-term dependencies in market behavior
    - **GRU**: A streamlined alternative to LSTM, offering comparable performance with fewer computational demands

Results and Discussion

Unsupervised Learning Results

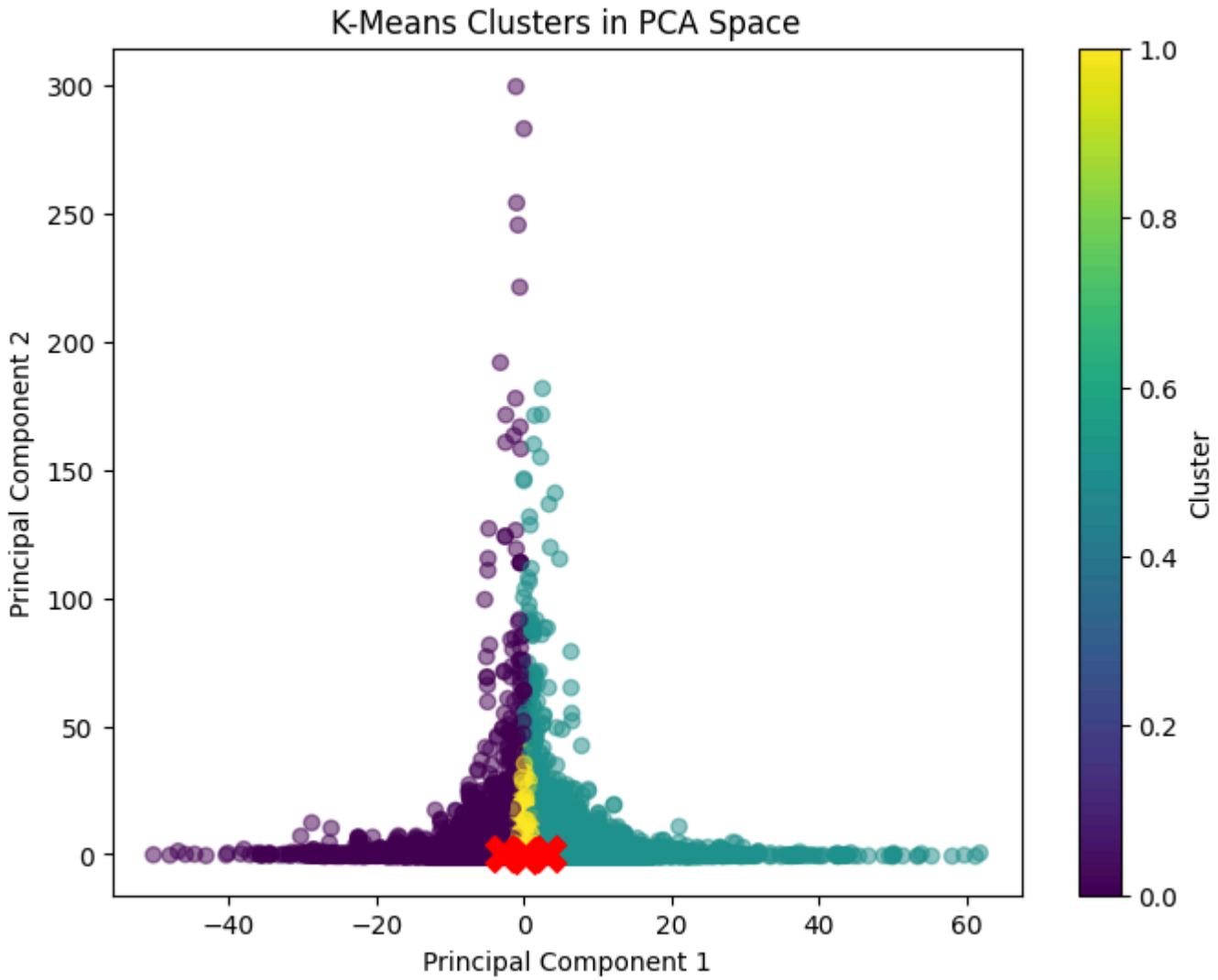
Our K-means clustering analysis revealed three distinct market states, each with unique trading implications:

1. **Value Opportunity Cluster (34% of samples)**:
    - Lower relative prices (WAP ≈ 0.997)
    - Higher ask-side volume (61,384 vs 53,263)

    - Increased selling pressure suggests oversold conditions
    - Favorable for mean reversion strategies
    - Optimal for limit order execution
    - Average imbalance size indicates strong price support
2. **High Demand Cluster (28% of samples)**:
    - Higher relative prices (WAP ≈ 1.004)
    - Larger bid sizes (58,705 vs 55,556)
    - Strong buying pressure indicates momentum
    - Higher matched size suggests active price discovery
    - Suitable for trend-following strategies
    - Imbalance direction confirms bullish sentiment
3. **Balanced Market Cluster (38% of samples)**:
    - Neutral prices (WAP ≈ 1.000)
    - Balanced bid-ask volumes (~50,000)
    - Low imbalance size indicates price stability
    - Minimal deviation in reference price
    - Optimal for market-making strategies
    - High matched size suggests efficient price discovery

PCA and Cluster Analysis

The K-means clustering in PCA space reveals distinct market behaviors, visualized in a characteristic “Y” shaped pattern:



Cluster Distribution Analysis:

1. **Value Opportunities (Purple Cluster, Left Wing)**:
    - Spreads negatively along PC1 (x-axis: -40 to 0)
    - Shows higher vertical dispersion (PC2: 0 to 300)
    - Higher ask-side volume suggests potential buying opportunities

    - Extreme points indicate significant selling pressure events
    - Concentration of points near center suggests mean-reversion tendency
    - Trading Implications:
        - Set limit orders at price levels with high point density
        - Monitor extreme PC2 values for capitulation signals
        - Scale position sizes based on distance from cluster centroid
2. **High Demand (Turquoise Cluster, Right Wing)**:
    - Extends positively along PC1 (x-axis: 0 to 60)
    - Shows moderate vertical spread (PC2: 0 to 150)
    - Strong buying pressure with consistent pattern
    - More horizontal spread suggests price-driven behavior
    - Trading Implications:
        - Momentum strategies more effective in this region
        - Use PC1 position to gauge buying pressure strength
        - Consider reduced position sizes at extreme PC1 values
3. **Balanced Market (Yellow/Central Cluster)**:
    - Concentrated around origin (PC1 ≈ 0, PC2 ≈ 0)
    - Highest density of points indicates common market state
    - Symmetric distribution suggests stable price discovery
    - Red centroids mark optimal trading zones
    - Trading Implications:
        - Focus on high-frequency, low-margin strategies
        - Use distance from centroids as risk metric
        - Optimal zone for market-making activities

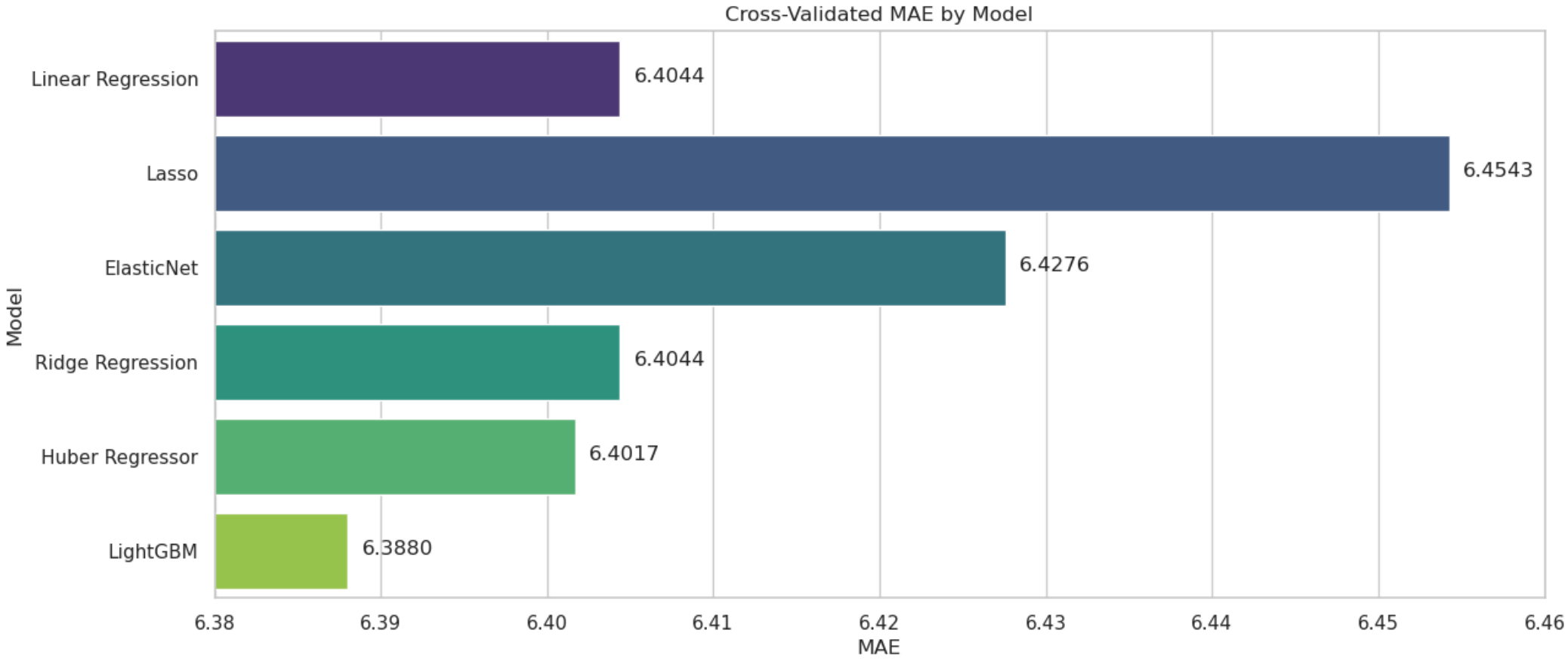
Key Visualization Insights:

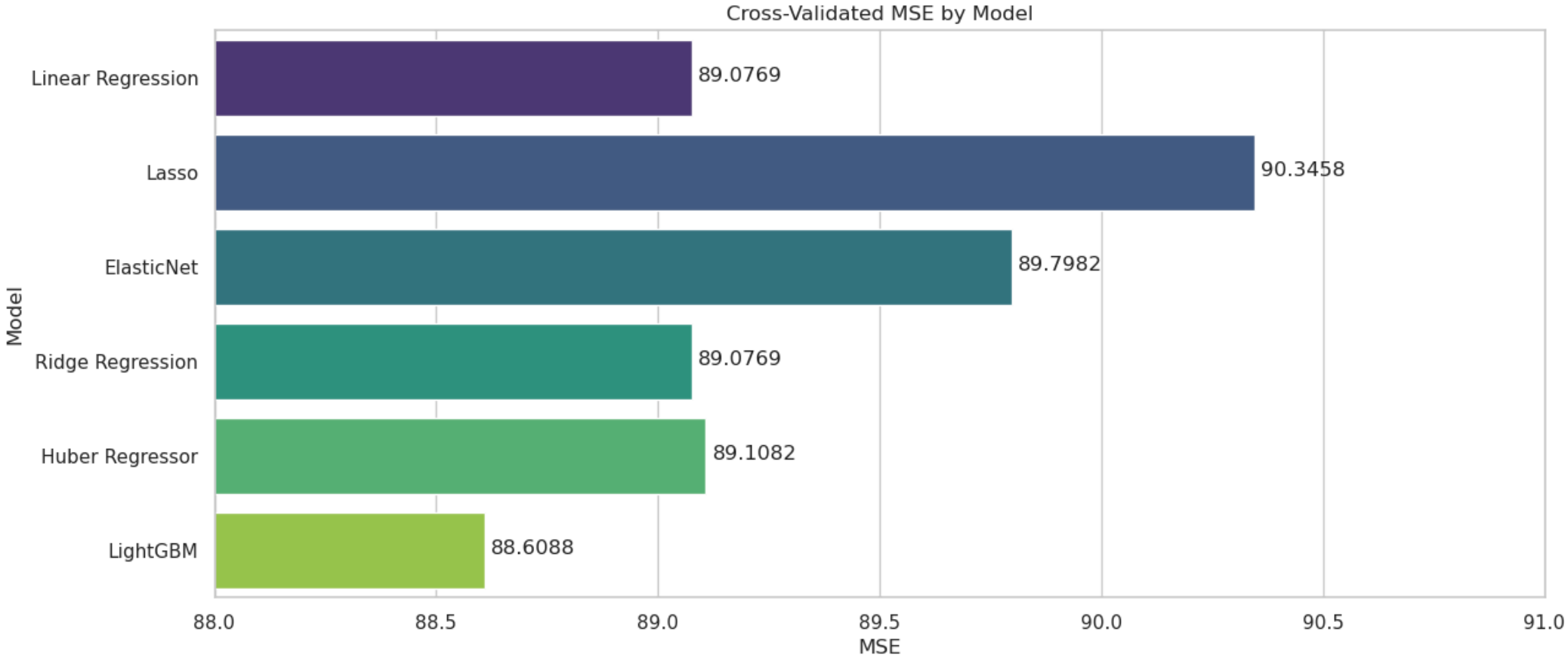
- Asymmetric cluster sizes indicate market preference for stability
- Clear boundaries between clusters enable reliable state classification
- Outlier points suggest potential trading opportunities in extreme conditions

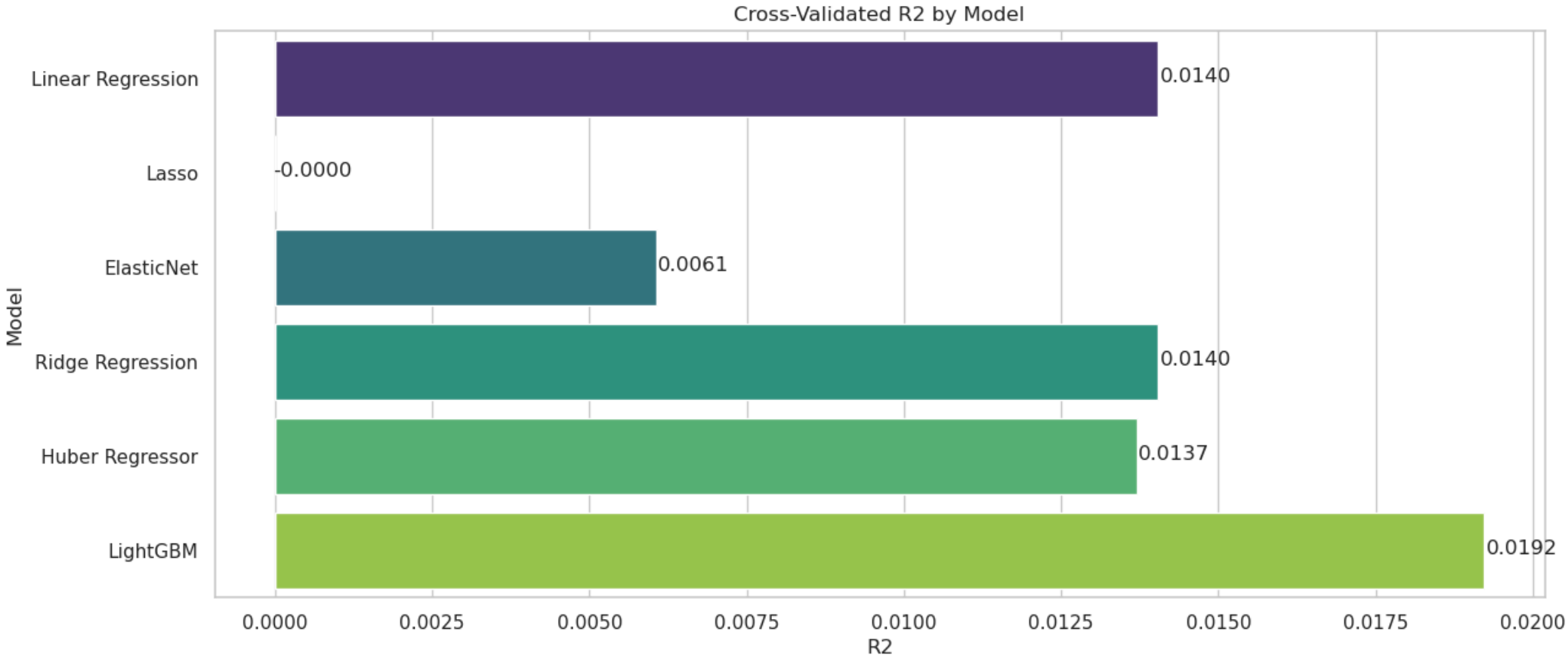
- Cluster density provides confidence levels for strategy selection

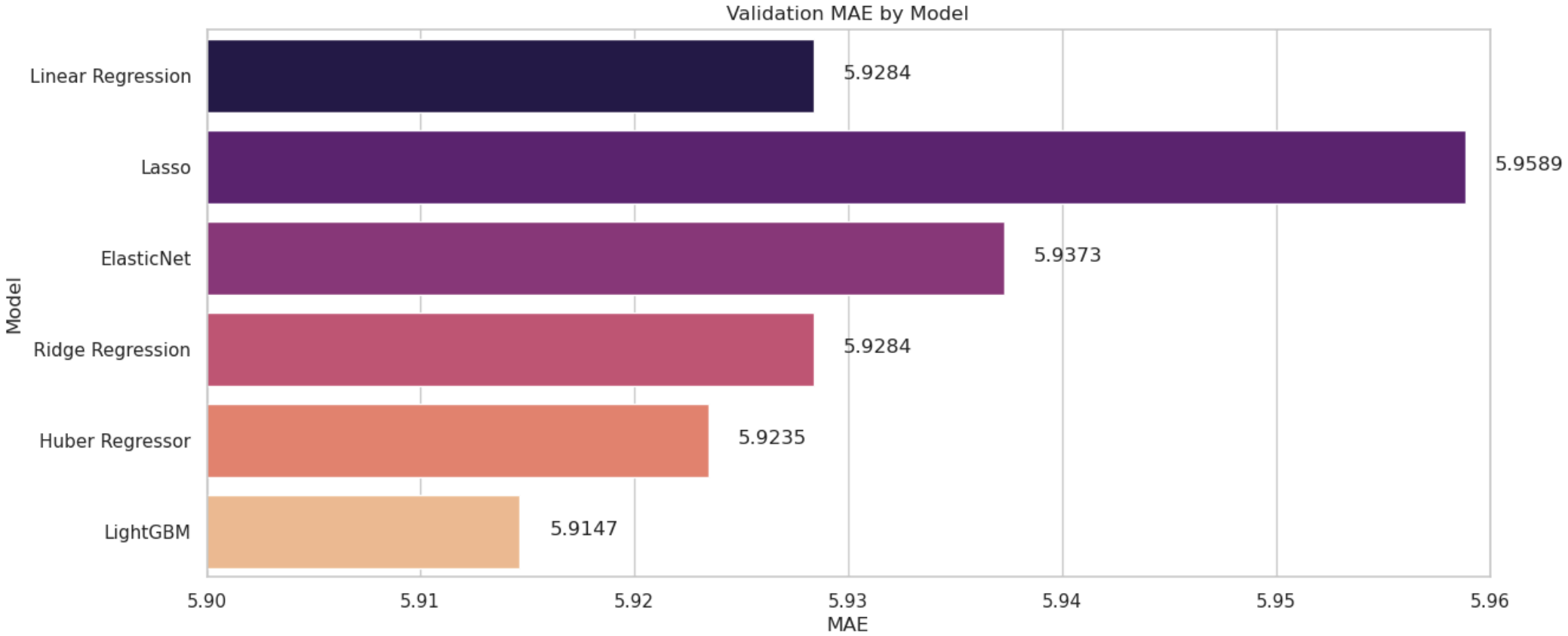
Supervised Learning Results

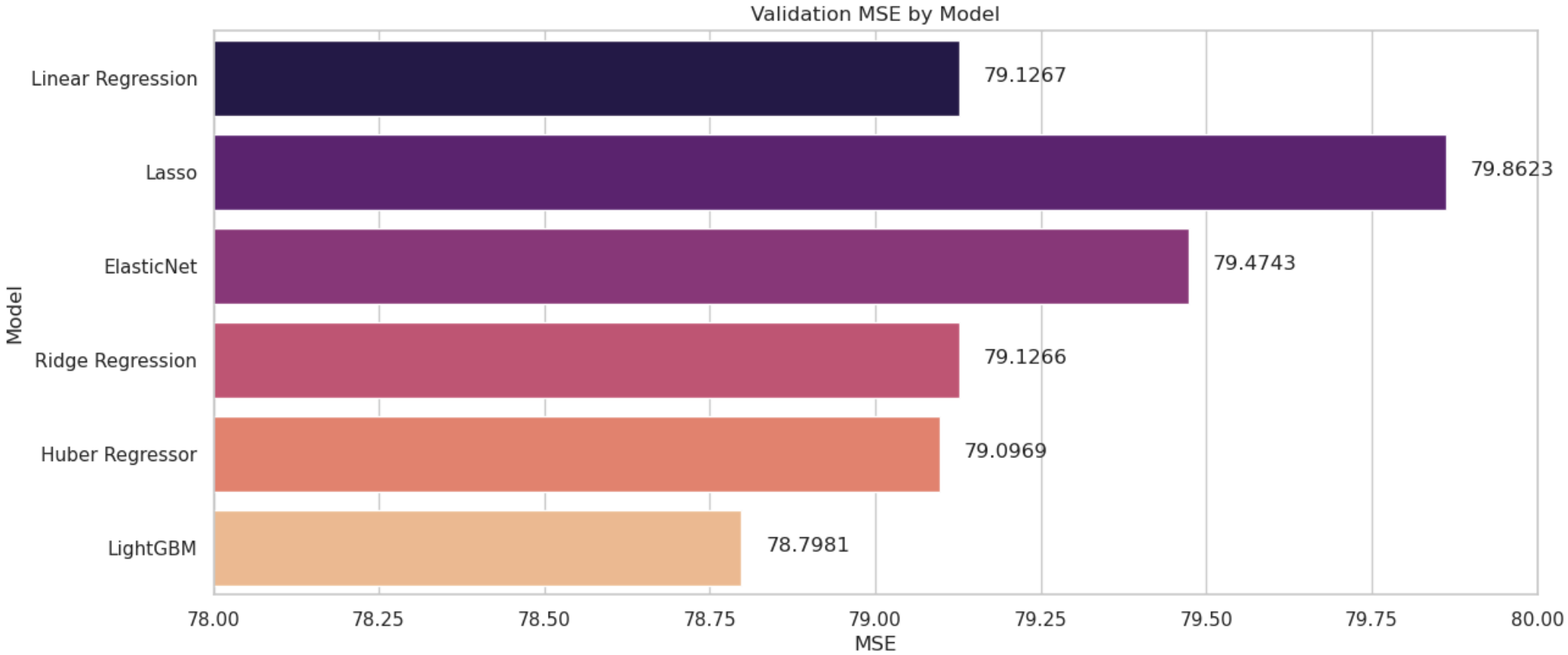
We implemented six supervised regression models, as previously introduced. To evaluate their performance, we conducted cross-validation on the training set and also fitted each model on the entire training set before testing on the validation set. We calculated Mean Absolute Error (MAE), Mean Squared Error (MSE) and R^2 metrics for assessment. The results are as follows:

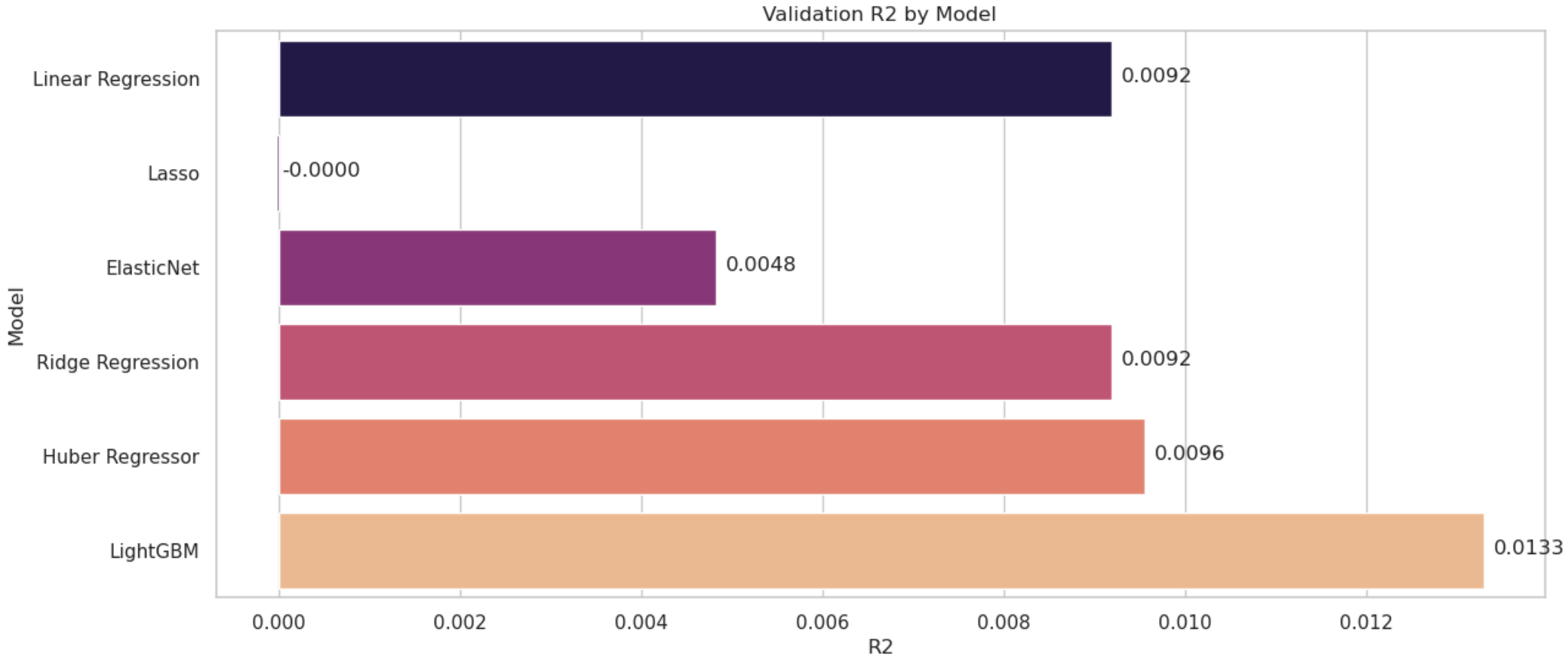



Cross-Validated MSE by Model
Linear Regression
Lasso
ElasticNet
Ridge Regression
Huber Regressor
LightGBM
89.0769
90.3458
89.7982
89.0769
89.1082
88.6088
Model
88.0
88.5
89.0
89.5
90.0
90.5
91.0
MSE


Cross-Validated R2 by Model
Linear Regression
Lasso
ElasticNet
Ridge Regression
Huber Regressor
LightGBM
Model
0.0140
-0.0000
0.0061
0.0140
0.0137
0.0192
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0.0200
R2


Validation MAE by Model
Linear Regression
Lasso
ElasticNet
Ridge Regression
Huber Regressor
LightGBM
5.9284
5.9589
5.9373
5.9284
5.9235
5.9147
Model
5.90
5.91
5.92
5.93
5.94
5.95
5.96
MAE

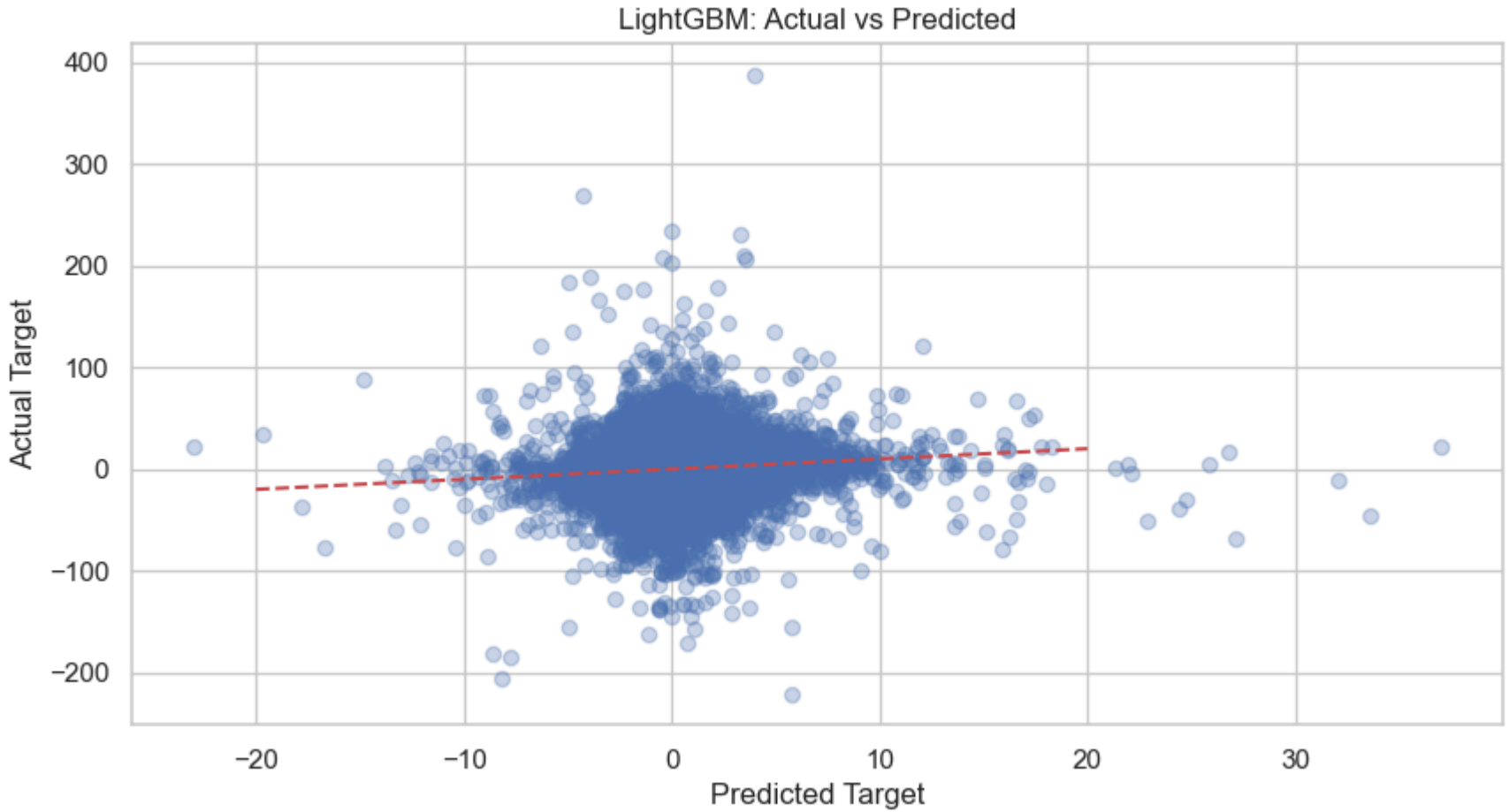

Validation MSE by Model
Linear Regression
Lasso
ElasticNet
Ridge Regression
Huber Regressor
LightGBM
79.1267
79.8623
79.4743
79.1266
79.0969
78.7981
Model
78.00
78.25
78.50
78.75
79.00
79.25
79.50
79.75
80.00
MSE



Out of all the baseline regression models trained, LightGBM emerged as the best performer due to its lowest Mean Absolute Error (MAE) and Mean Squared Error (MSE) in both cross-validation and on the validation set. Consequently, we selected this model for further exploration of the results.

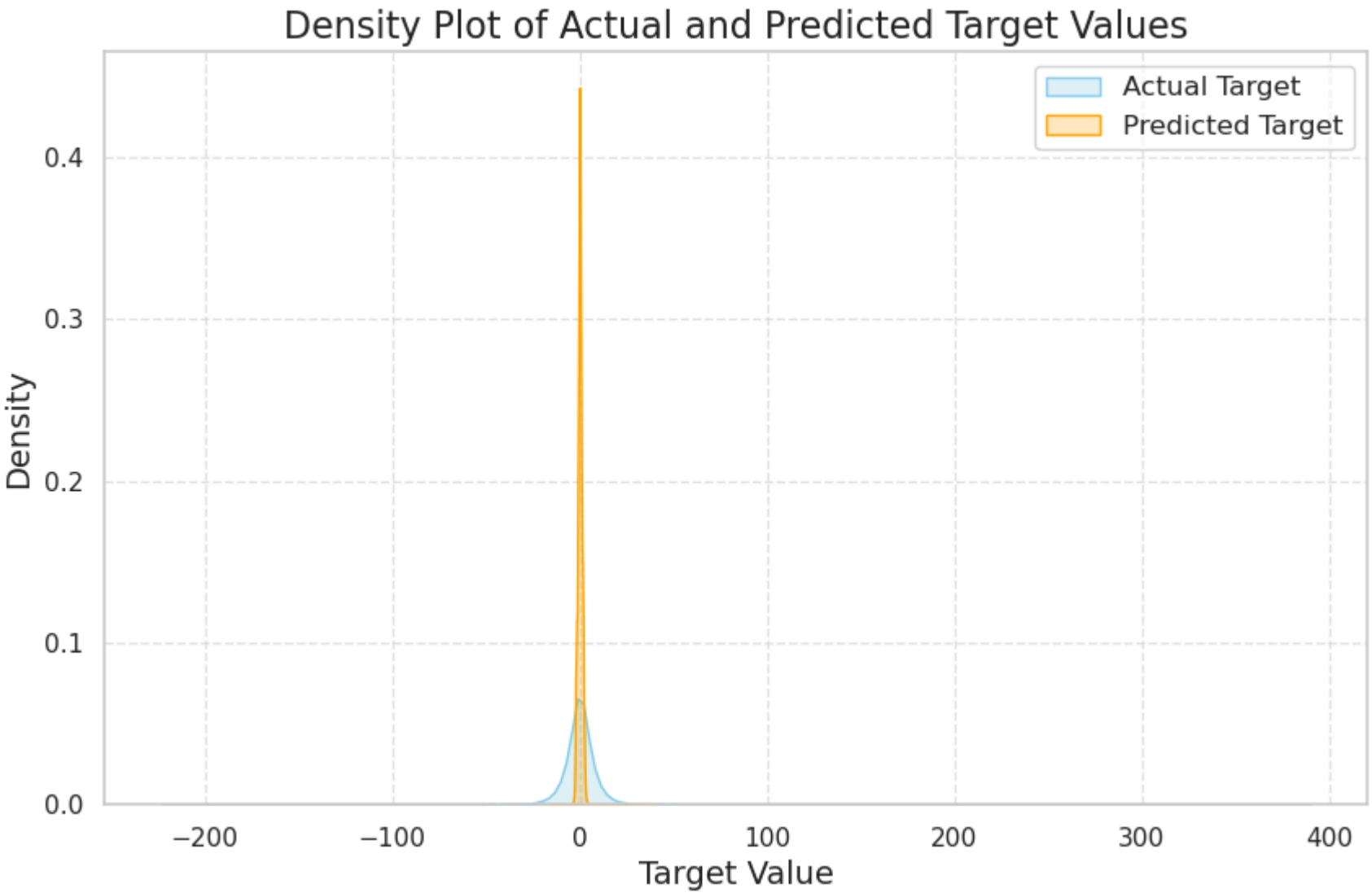
However, the LightGBM model exhibited an R-squared value of only 0.0113, indicating that it explained merely 1% of the variance in the data. This low R-squared value suggests that, despite the favorable MAE and MSE scores, the model may not be capturing the underlying data patterns effectively. Therefore, it's necessary to delve into why the R-squared value remains so low.

We plotted the actual versus predicted target values of the LightGBM model:



While there appears to be a trend in the dataset, the relationship is weak with significant variance. The residuals vary considerably—some are large, and others are small—indicating that certain predictions are close to the actual values while others deviate substantially. Notably, many predicted values are zero or near zero even when the actual target values range between -100 and 100.

We also examined the density plots of the actual and predicted values for additional insights:



Our examination of the residuals and density plots revealed that the model often predicts values close to zero when the actual target values are significantly different. This discrepancy suggests that the dataset may be inherently too complex for the model to capture effectively.

A plausible hypothesis is that the dataset encompasses numerous different stocks, each with its unique trends and time-series behaviors. This diversity may pose a challenge for the model to generalize across all stocks, leading to suboptimal performance in explaining variance despite low error metrics.

NN-Based Supervised Learning Results

In this section, we attempted to use three neural network-based sequential models to accomplish the task of predicting the closing price. Given the excellent performance of deep learning in time series forecasting tasks in recent years, we anticipated that these models would outperform traditional machine learning models discussed earlier. The models we experimented with include plain RNN, GRU, and LSTM. Specifically:

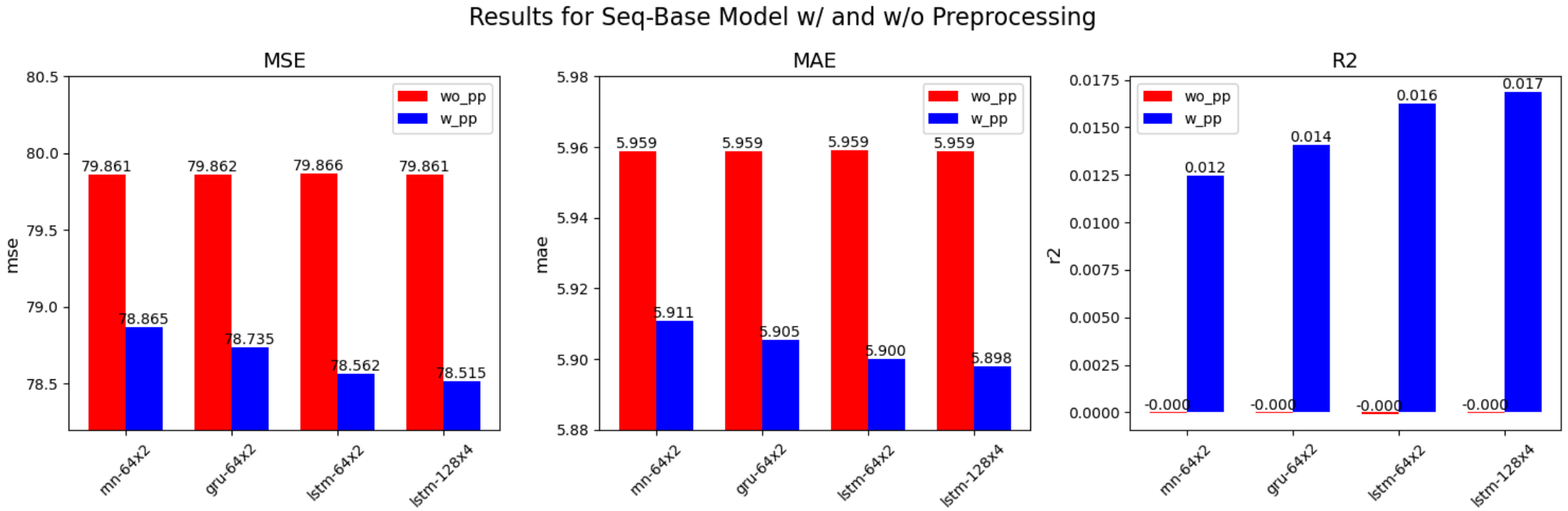
The RNN, GRU, and LSTM models consisted of two layers, each with 64 neurons and Tanh activation functions, followed by a fully connected layer with an output dimension of 1. Additionally, we experimented with a larger version of the LSTM model, which had four layers, each with 128 neurons. The models were trained for 12 epochs with a batch size of 64, using the Adam optimizer with a learning rate of 0.001. The learning rate was decayed by a factor of 0.1 at epochs 10 and 12. For input features, we ignored the date ID information, converted the stock ID into one-hot encoding, and concatenated it with the other features. Missing values were filled with 0. The sequential input data consisted of adjacent data points for the same stock on the same day, used as input to predict the closing price at the final time point. Our loss function utilized a truncated smooth L1 loss with a threshold of 40. This threshold was determined by analyzing the histogram of the target variable, as we aimed to prevent extreme outliers from causing instability during training. We experimented with different time window lengths for the input data. The dataset was split in the same way as mentioned earlier, with data from date IDs 0 to 435 used for training and data from date IDs 436 to 479 used for validation. We reported MSE, MAE, and R2 as evaluation metrics.

Preprocessing IS Necessary

In the first round of experiments under the above settings, we set the time window length to 5. However, the results were not satisfactory. For instance, using RNN, we obtained an MSE score of 79.861, which was not only worse than LightGBM but also inferior to Linear Regression. Upon analyzing the input features, we found that the size-related features in the dataset had a large value range, with a difference of nine orders of magnitude between the maximum and minimum values. This posed challenges for the neural networks. To address this, we applied simple feature engineering, transforming the size feature using $f(s) = \log(s + 1)$. Retraining the models with this transformation resulted in significant improvements. For example, with RNN, we achieved an MSE score of 78.865, which outperformed LightGBM. From this experiment, we can see that proper data preprocessing and feature engineering are crucial for the performance of neural network models. More detailed results are shown in the following tables:

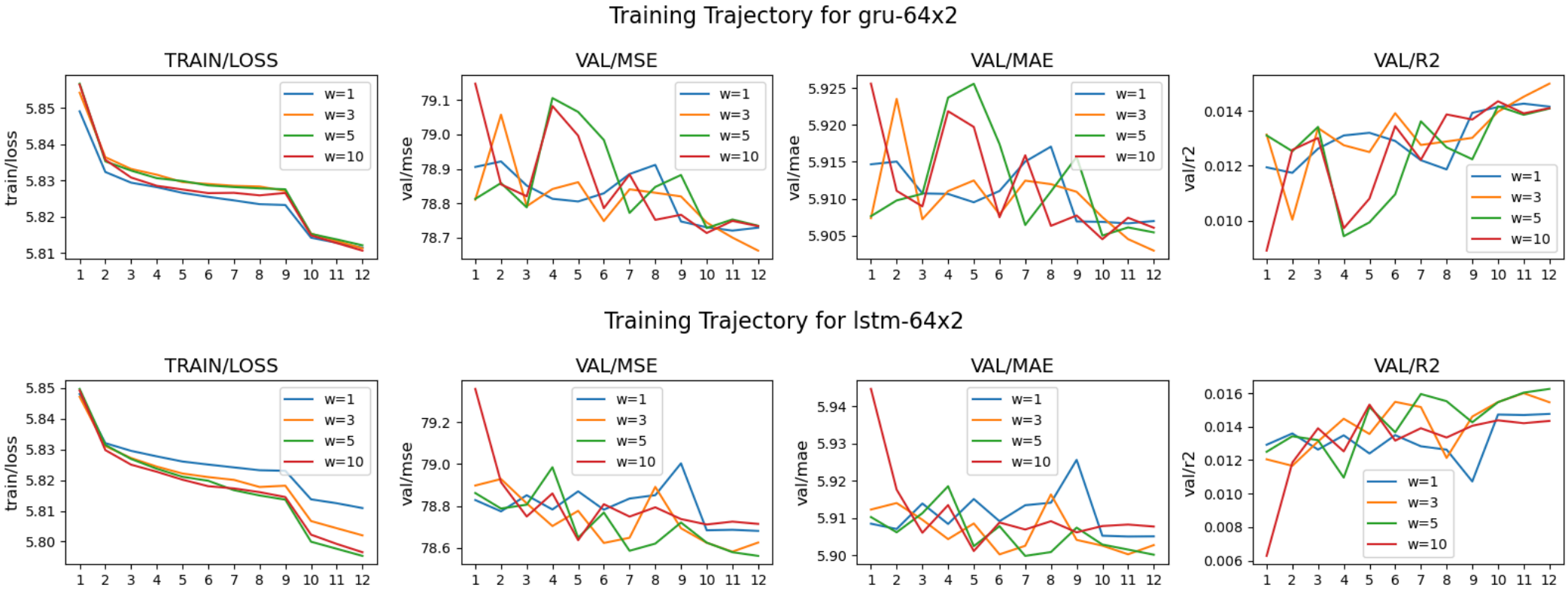
Model	Window	Preprocess	MSE	MAE	R^2

rnn-64x2	5	w/o	79.860712	5.958865	-0.000012
rnn-64x2	5	w/	78.864602	5.910735	0.012461
gru-64x2	5	w/o	79.861779	5.958944	-0.000026
gru-64x2	5	w/	78.735091	5.905445	0.014083
lstm-64x2	5	w/o	79.865552	5.959106	-0.000073
lstm-64x2	5	w/	78.561722	5.900152	0.016254
lstm-128x4	5	w/o	79.860660	5.958869	-0.000011
lstm-128x4	5	w/	78.515222	5.897853	0.016836

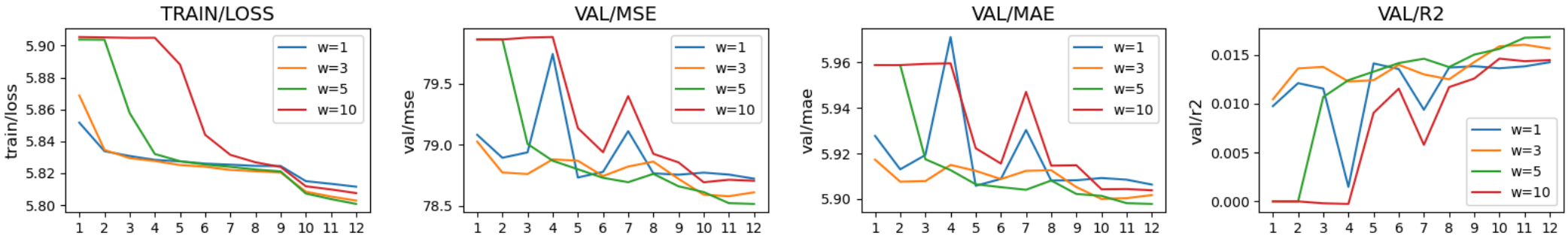


Main Result for Sequential Neural Network

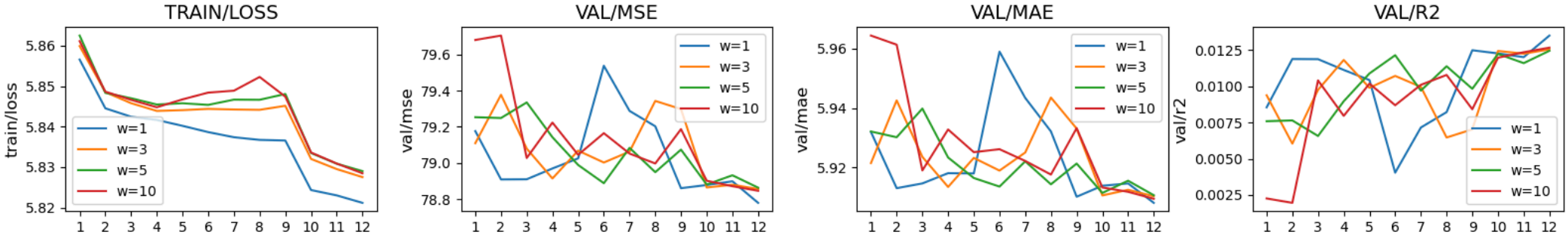
We further enumerated time window lengths of 1 (equivalent to using only the data from the current day), 3, 5, and 10. We present the training loss curves for different models, along with their MSE, MAE, and R2 metrics on the validation set.



Training Trajectory for lstm-128x4

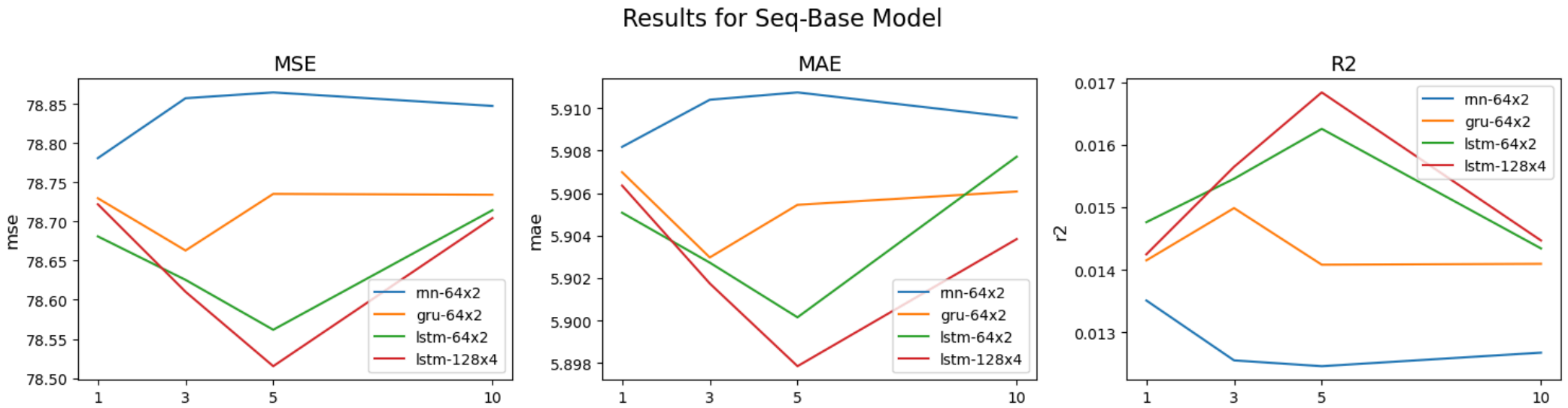


Training Trajectory for rnn-64x2



Model	Window	MSE	MAE	R^2
rnn-64x2	1	78.780757	5.908173	0.013511
rnn-64x2	3	78.857194	5.910391	0.012554
rnn-64x2	5	78.864602	5.910735	0.012461
rnn-64x2	10	78.847347	5.909542	0.012677
gru-64x2	1	78.729551	5.906980	0.014152
gru-64x2	3	78.662858	5.902970	0.014987

gru-64x2	5	78.735091	5.905445	0.014083
gru-64x2	10	78.733981	5.906072	0.014097
lstm-64x2	1	78.680838	5.905073	0.014762
lstm-64x2	3	78.625379	5.902725	0.015457
lstm-64x2	5	78.561722	5.900152	0.016254
lstm-64x2	10	78.714328	5.907711	0.014343
lstm-128x4	1	78.721866	5.906351	0.014248
lstm-128x4	3	78.610189	5.901744	0.015647
lstm-128x4	5	78.515222	5.897853	0.016836
lstm-128x4	10	78.704181	5.903833	0.014470



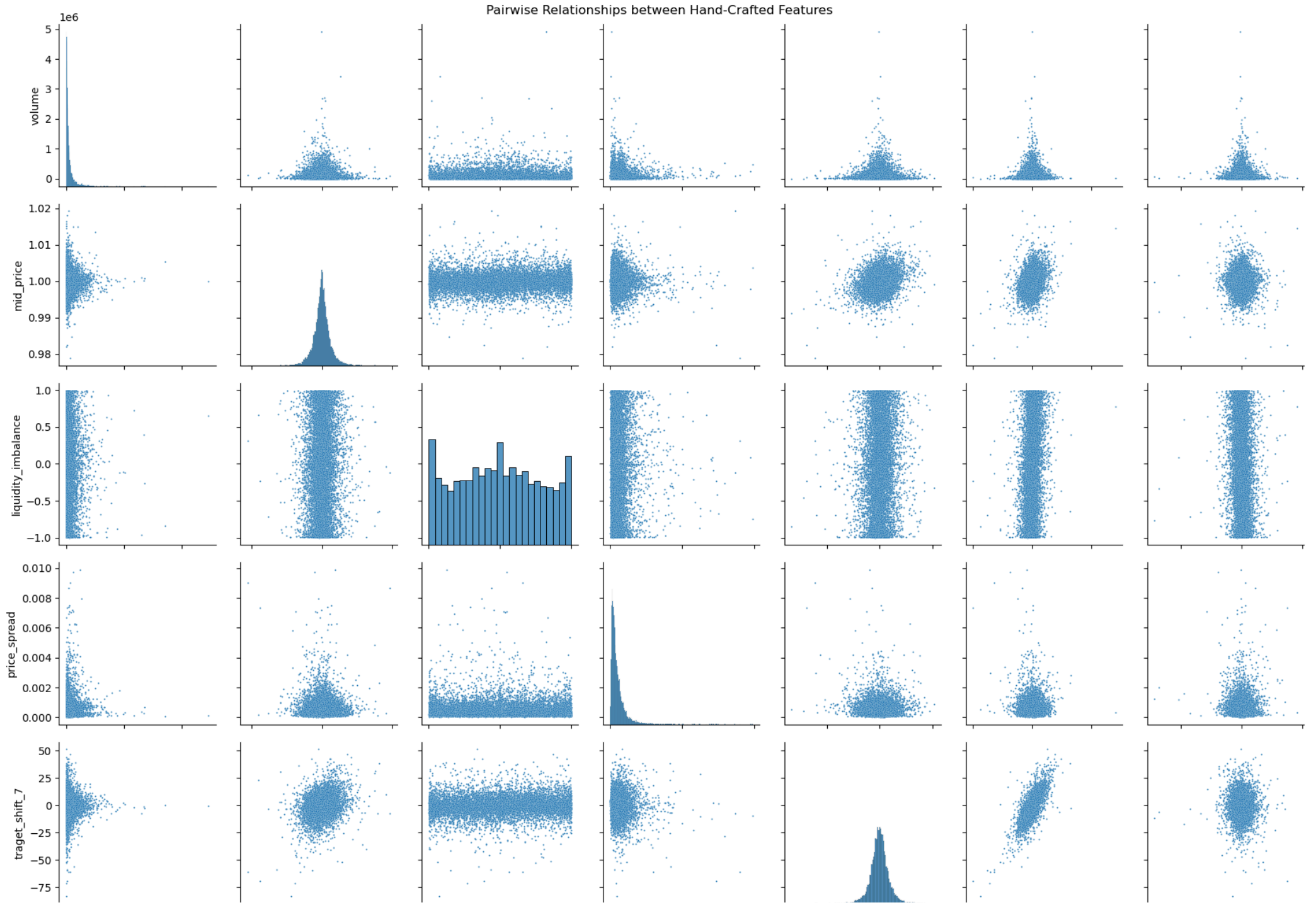
The results show that these neural network models were able to outperform the baseline regression models in terms of key performance metrics like MSE and MAE. The LSTM-128x4 model with a window size of 5 had the best overall performance, with an MSE of 78.515222, an MAE of 5.897853, and an R-squared value of 0.016836 on the validation set. This indicates that the LSTM model was able to effectively leverage the sequential nature of the data and learn patterns that led to more accurate price predictions compared to the other architectures tested.

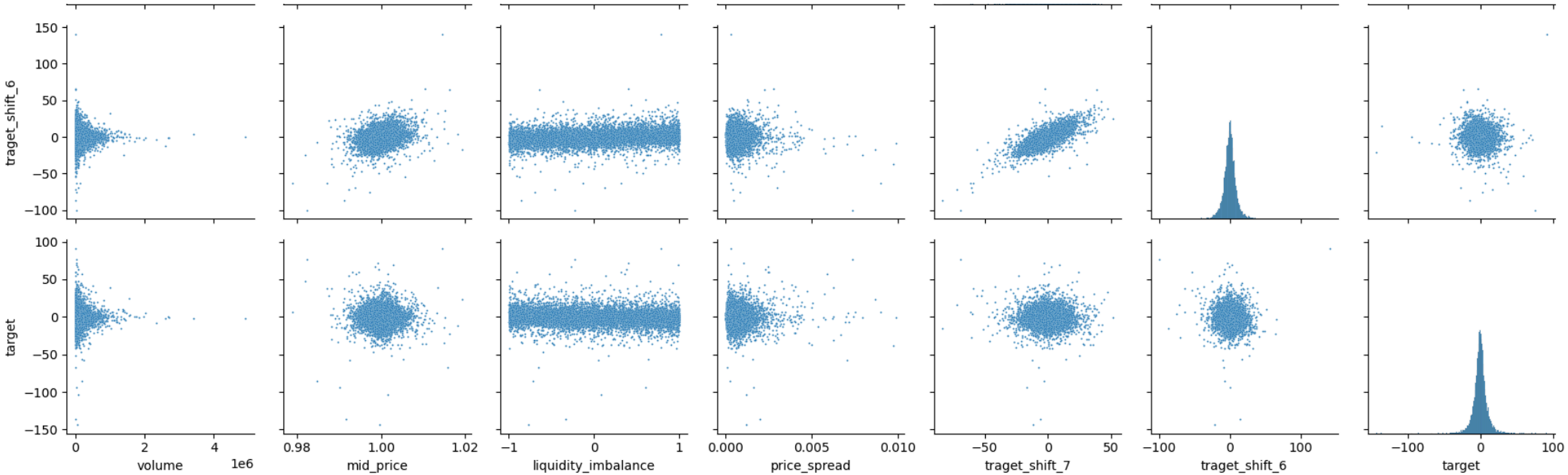
Furthermore, the experiments that incorporated the target values from previous timestamps as additional input features showed a modest improvement in the LSTM model's performance, suggesting that the relationships between consecutive price movements can provide valuable information for enhancing the predictive capabilities of the model.

However, it is important to note that despite the improvements seen in the neural network models, the overall R-squared values remained low, indicating that the models were still unable to explain a significant portion of the variance in the target variable. This highlights the inherent complexity and unpredictability of the stock market, particularly during the highly volatile final trading minutes.

Bonus: Using Target from Previous Timestamp as Input

Since the physical meaning of the target is a function of the states at time $T+6$ and T, in time-series models, we can use previously obtained targets as input features. We expect this to improve the model's prediction accuracy. As a preliminary experiment, we analyzed the correlation between the target at time T and the targets at times $T-6$ and $T-7$, along with a series of handcrafted features, as shown in the figure below.


Pairwise Relationships between Hand-Crafted Features
1e6
volume
mid_price
liquidity_imbalance
price_spread
traget_shift_7

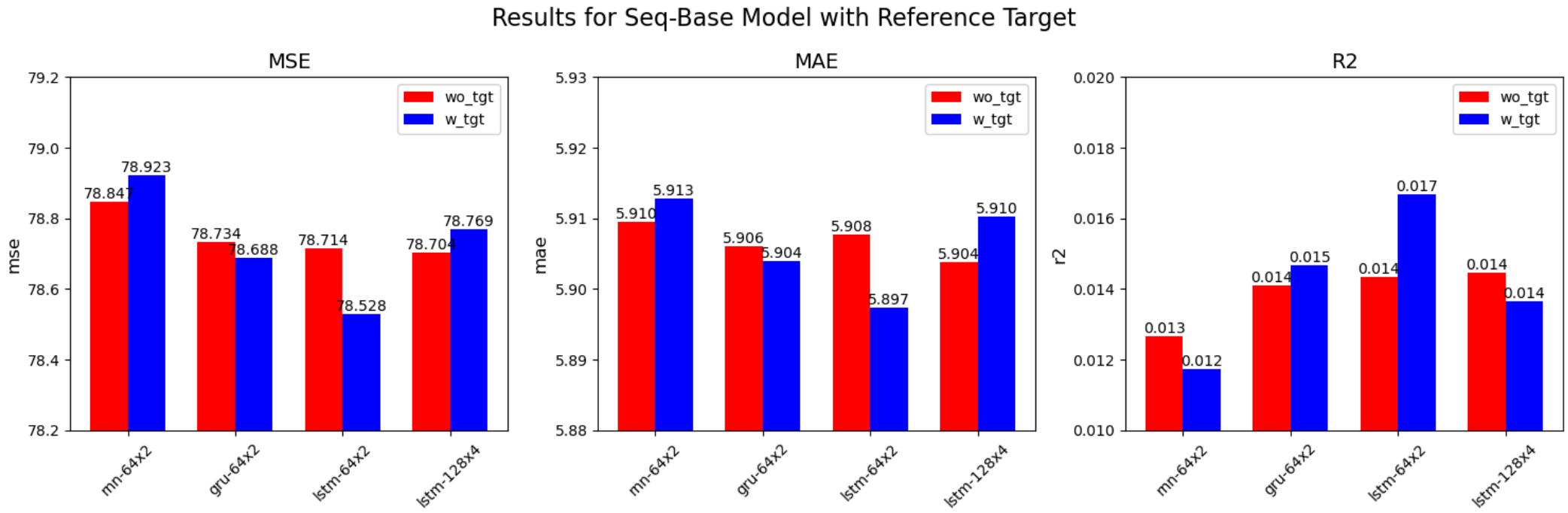


We found that although the targets at adjacent time points, such as $T-6$ and $T-7$, exhibit strong correlations with each other, their correlations with the target at time T are relatively low. Moreover, these correlations are just comparable to those between the target at T and the handcrafted features. This suggests that including earlier time targets may not necessarily lead to a significant improvement in performance.

In our implementation, for the input sequence at time T, we used the targets from $T-6$ and earlier as input features. For targets at later times, we applied 0-padding to avoid using information that would be unavailable in real-world scenarios. We compared models with and without earlier time targets as input features. The results showed that, for some models, incorporating earlier time targets improved prediction accuracy, while for others, it did not. Among them, a 2-layer LSTM achieved the best result in this part of the experiment, with 78.528 MSE and 5.897 MAE.

Model	Window	Target	MSE	MAE	R^2
rnn-64x2	10	w/o	78.847347	5.909542	0.012677
rnn-64x2	10	w/	78.923211	5.912877	0.011727

gru-64x2	10	w/o	78.733981	5.906072	0.014097
gru-64x2	10	w/	78.688190	5.904001	0.014670
lstm-64x2	10	w/o	78.714328	5.907711	0.014343
lstm-64x2	10	w/	78.528249	5.897356	0.016673
lstm-128x4	10	w/o	78.704181	5.903833	0.014470
lstm-128x4	10	w/	78.768615	5.910276	0.013663



Conclusion

The exploration of various machine learning models for predicting stock price movements during the final 60 seconds of trading has yielded some promising results. The key findings from this study are as follows:

The unsupervised K-means clustering analysis revealed three distinct market states characterized by different price, volume, and imbalance dynamics. This provided valuable insights into the underlying patterns of market behavior during the closing auction.

Among the supervised regression models tested, the LightGBM algorithm emerged as the best performer in terms of achieving the lowest Mean Absolute Error (MAE) and Mean Squared Error (MSE) on both the cross-validation and validation sets. However, the R-squared (R^2) values remained relatively low, indicating that the model was still unable to fully capture the complexity and unpredictability of the stock market during this critical time period.

The experiments with sequential neural network architectures, including RNN, GRU, and LSTM, demonstrated the potential of these models to leverage the temporal dependencies in the data. The 4-layer LSTM model with 128 neurons per layer and a 5-timestamp input window achieved the overall best performance, with an MSE of 78.515222, an MAE of 5.897853, and an R-squared value of 0.016836 on the validation set.

Incorporating the previous target values as additional input features led to modest improvements in the LSTM model's predictive accuracy, suggesting that exploiting the relationships between consecutive price movements can be beneficial.

While the results are promising, the low R-squared values across all models highlight the inherent challenges in accurately forecasting stock price movements, especially during the highly volatile final trading minutes.

In summary, this study has demonstrated the value of applying a combination of unsupervised and supervised machine learning techniques to gain insights into the dynamics of the stock market closing auction. The performance of the LSTM-based models in particular suggests that leveraging the sequential nature of the data can be a fruitful avenue for improving stock price prediction during this critical time period.

Final Project Video

Here is our final video link.

References

[1] W. Huang et al., “Forecasting stock market movement direction with support vector machine,” Computers & Operations Research, 32, pp. 2513–2522005, 2005.

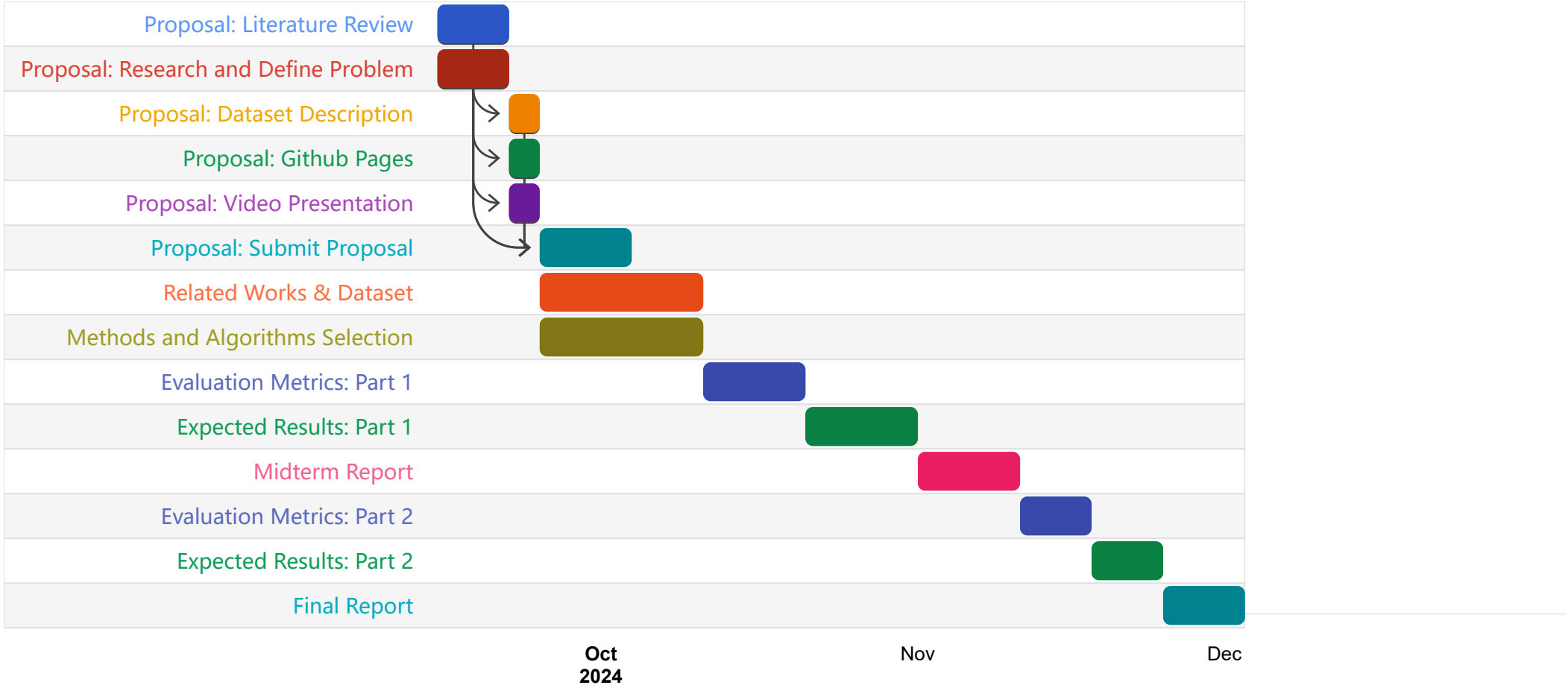
[2] S. Zemke, “On developing a financial prediction system: Pitfalls and possibilities,” Proceedings of DMLL-2002 Workshop, ICML, Sydney, Australia, 2002.

[3] J. Maqbool et al., “Stock Prediction by Integrating Sentiment Scores of Financial News and MLP-Regressor: A Machine Learning Approach”, Procedia Computer Science, Volume 218, Pages 1067-1078, 2023.

Contribution Table

Name	Proposal Contributions
Yiming Chen	Report write-up
Xinyue Fu	Slides Production
Steven Zhang	Video Representation
Yue Zhao	Report write-up
Fengzhe Zhou	Model implementation

Gantt Chart


Proposal: Literature Review
Proposal: Research and Define Problem
Proposal: Dataset Description
Proposal: Github Pages
Proposal: Video Presentation
Proposal: Submit Proposal
Related Works & Dataset
Methods and Algorithms Selection
Evaluation Metrics: Part 1
Expected Results: Part 1
Midterm Report
Evaluation Metrics: Part 2
Expected Results: Part 2
Final Report
Oct
2024
Nov
Dec