UNIVERSITY OF EDINBURGH
Business School

2023-24

CMSE115282023
Financial Machine Learning II (Practical)

Regression Modeling for Mid-Price Forecasting

B246085

Word count: 2,013

Introduction

This report focuses on developing Machine Learning models to predict stock mid-price movements using a sample dataset. By analyzing the ask and bid prices and volumes, this aims to forecast mid-price in the next time event. The approach uses a vector-based representation of stock, derived from multi-dimensional time series data at a specific time moment t, to anticipate mid-price direction at a future time t+1.

Below is an overview of the Machine Learning pipeline, as employed in previous research (Dash and Dash, 2016; Passalis *et al.*, 2017; Tran Thanh *et al.*, 2017):

(a) Data Pre-processing: Involves filtering, normalizing, and preparing the data for analysis.

(b) Feature Selection: Identifies and selects key features for the analysis.

(c) Feature Engineering:

(d) Data Splitting: Splits the dataset into training, validation, and test subsets.

(e) Model Selection and Training: Describes the chosen models for regression tasks, detailing the training process.

(f) Model Evaluation: Outlines the metrics used to evaluate model performance for an in-depth discussion of the results.

This structured approach facilitates a thorough investigation into forecasting stock mid-price movements, combining theoretical concepts with practical applications.

1. Dataset Description

The dataset comprises high-frequency trading limit order book data for Amazon, containing a total of 562,650 raw entries recorded between 5:00 and 21:00. Each entry encompasses 46 columns, detailing various attributes related to the trading environment. The dataset contains bid and ask prices and volumes, at levels ranging from the best (level 1) to the tenth level. Each level provides information on the depth of liquidity in the market at different price levels, with level 1 indicating the most competitive prices and level 10 representing prices further away from the current market price.

Key columns include "Time" indicating the timestamp of each observation and was converted from unix timestamp to a readable format for analysis; and "Mid-Price," denoting the average value of the bid and ask prices at level 1 as a reliable proxy for trade price.

(a) For insights into the distribution of Mid-Price values, refer to Appendix [1].
(b) To explore time series visualizations of Mid-Price and Spread, see Appendix [2] and [3], respectively.

2. Data filtering

The following filtering steps were performed in the data preparation phase:

(a) Variables with 30% or more missing values were deemed unreliable and thus removed from the dataset.
(b) Entries with missing values (NaN, 0, inf) were deleted, as these do not represent valid market transactions.
(c) The dataset was narrowed down to include only transactions that occurred during the official New York Stock Exchange trading hours, from 9:30 AM to 4:00 PM, to align with standard market activity periods.

3. Feature Selection

I conducted feature selection through two methods: variance and correlation analysis of the columns. Variance-based selection involved identifying and retaining columns with a variance above a certain threshold, filtering out columns with low variance. However, in the case of a LOB, all columns, including those presenting price and volume, shows high variance.

Likewise, correlation analysis aimed to detect columns with high correlation to each other, helping to identify redundant or highly correlated features that may be removed to improve model performance. However, in the case of the Limit Order Book (LOB), each level of Ask and Bid showed strong correlation, which is the inherent structure of the LOB. Therefore, based on these 2 analysis, I decided to retain all variables for regression analysis.

However, future research could explore advanced ensemble feature selection techniques to further enhance the feature selection process and potentially improve model performance. (Saeys, Abeel and Van de Peer, 2008)

4. Feature engineering

According to (Ntakaris *et al.*, 2019), handcrafted features overperformed the fully automated feature extraction process. Also, it states that the task of selecting suitable features is customized according to data, stock and model availability. Based on (Ntakaris *et al.*, 2019), I selected and added 3 handcrafted features for training the models:

(a) Lag Ask and Bid Price: create a lag of 1 event, which means for each current event being analyzed, the model looks back 1 event prior (the 'lagged event') and uses information from that lagged event to make the prediction.

(b) Rolling mean: For each row in the dataframe, it calculates the average of the current and the preceding 3 price values. The window of 4 moves down one row at a time. As in volatile markets like HFT, prices can fluctuate rapidly, this feature helps to smooth out these fluctuations to highlight underlying trends.

(c) Log return: commonly used in market microstructure because of their its properties and it can simplify the calculation of compound returns over time. The log return for a single period is calculated using the natural logarithm, based on the closing price of an asset at two different times. It is calculated as:

$$r = \ln\left(\frac{P_t}{P_{t-1}}\right)$$

Where:
r is the log return
P_t is the mid-price at time t
P_{t-1} is the mid-price at previous timestamp

5. Normalization

This procedure is performed based on MinMax normalization for the feature matrix and the forecasting variable before running the regressors. This approach is supported by a prior study (Lee and Wang, 2018); and subsequnt regression run, both with and without normalization, have demonstrated that normalization enhances performance and reduces model training time.

As can be seen in appendix [1], [3], mid price or other features vary widely in scale, which is a common issues with financial data. Also, since there are added features with different scales, normalization ensures all features contribute equally to model training by preventing bias. Lastly, the activation function for CNNs used in this research is Relu, which is known to become insensitive to small changes as the input value is too large. Normalization help resolve this issue. By scaling data, it is also observed to help CNNs learn more effectively, potentially reduce overfitting based on the results compare before and after normalization. The formula is as below:

$$X_{normalize} = \frac{X_i - X_{min}}{X_{max} - X_{min}}$$

Where:
$X_{normalize}$: normalized value.

X_i : original value.
$X_{min}\ and\ X_{max}$: minimum and maximum values of the features x, respectively.

6. Overview of Machine Learning Models

Machine learning algorithms are carefully chosen based on their specific strengths and suitability for the dataset's characteristics and complexities, as evidenced by prior research. The choice of algorithm depends on the nature of the data, especially when dealing with time series data.

(a) CNN – Forecasting: It comprises two convolutional layers, max-pooling, and dense layers with ReLU activation.

(b) Convolutional Neural Network (CNN) – Autoencoder: It prepares data with a look-back window of 10 time steps. The architecture includes convolutional layers with upsampling for reconstruction. The model is trained and compiled with MSE loss and Adam optimizer, then features are extracted using the encoder portion.

(c) Radial Basis Function Neural Network (RBFNN): The pipeline includes KMeans clustering, RBFSampler for feature mapping, and Ridge regression for prediction.

(d) Multi-Layer Perceptrons (MLPs): Three MLP architectures are explored: shallow, deep, and very deep.

(e) Long Short-Term Memory (LSTM) : The LSTM architecture includes a single LSTM layer with 256 units followed by a dense layer for prediction. It excels in capturing long-term dependencies and temporal patterns inherent in sequential data (Deshmukh and Sonkar, 2020).

(f) Random Forest Regressor: an ensemble learning method that constructs multiple decision trees during training and outputs the mean prediction of the individual trees.

(g) Gradient Boosting Regressor: an ensemble learning technique that builds a series of weak learners (typically decision trees) sequentially, with each new tree fitting to the errors made by the previous ones.

(h) Bayesian Regression, specifically Bayesian Ridge regression, incorporates Bayesian principles to estimate regression parameters. Unlike neural networks, it doesn't require feature scaling due to its regularization technique, which automatically adjusts coefficients. Additionally, it assumes a probabilistic model where features are assumed to be normally distributed, eliminating the need for normalization as it accounts for feature distribution during training.

(i) Autoregressive Integrated Moving Average (ARIMA): ARIMA is a time series forecasting method that models the relationship between a variable and its historical values, as well as the errors from past predictions. In this implementation, an ARIMA model is fitted to the target variable (Forecasting_Variable) without feature engineering. I used this as it is a classical time series model for comparison (Pole, West and Harrison, 1994; Han *et al.*, 2021).

For all Regressors, TimeSeriesSplit is used for cross-validation, and data is normalized using MinMaxScaler.

The Rectified Linear Unit (ReLU) activation function has been used for Convolutional Neural Networks (CNNs) and MLP due to its simplicity and efficacy (McKerahan *et al.*, 2023).

The primary motivation behind using ReLU stems from the need to introduce non-linearity into the network without significantly increasing the computational burden. Since sigmoid and tanh functions suffer from the vanishing gradient problem, where gradients become very small, effectively halting the network's training, ReLU was used as a solution to allowing model to learn faster and perform better. For positive inputs, the gradient is always 1, so gradients do not vanish during backpropagation as they can with sigmoid or tanh functions.

The ReLU function is mathematically defined as:

$$f(x) = \max(0, x)$$

This means that for any positive input, the output is equal to the input, and for any negative input, the output is zero.

However, the ReLU activation function can cause neurons to "die" and stop learning if they receive negative inputs, complicates optimization due to non-zero-centered outputs, and may lead to learning issues from its unbounded positive output.

7. Evaluation metrics

In the context of regression tasks aimed at predicting future mid-price values within time series data, two prevalent evaluation metrics are the Root Mean Squared Error (RMSE) and Mean Percentage Error (MPE). These metrics serve as quantitative measures to assess the performance of regression models. The expressions for RMSE and MAPE are as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \hat{y}_i)^2}$$

$$MPE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right) \times 100$$

Where n is the number of observations, $y_i$represents the actual values, and $\hat{y}_i$ represents the predicted values. These evaluation metrics provide a quantitative assessment of the predictive performance of regression models, with RMSE indicating the average magnitude of prediction errors in the same units as the target variable, and MPE representing the average percentage deviation between predicted and actual values.

For CNN Autoencoder, calculating RMSE and MPE is challenging due to their focus on learning compressed representations of input data through non-linear transformations. The latent space representation learned by autoencoders does not correspond to the original data and dimensionality reduction makes traditional regression metrics less applicable. Alternative metrics like MSE or qualitative assessments through visualization are preferred. For a comparison with other regressors' performance, RMSE has been calculated based on MSE. (Appendix 4).

The motivation of choosing RMSE is that it minimizes the average magnitude of errors, given the rapid split-second price changes in HFT (McKerahan *et al.*, 2023). It squares the errors before averaging, thus penalizing larger errors. Its values are in the same units as the predicted outcome (mid-price movements), making it easier for market makers and traders to interpret the model's accuracy. RMSE is also efficient in training of complex models like neural networks often used in HFT LOB predictions.

The downside of RMSE is its emphasis on squaring errors can disproportionately impact model training by over-penalizing outliers and large errors, potentially misaligning with strategies that favor consistent, smaller gains in HFT LOB contexts.

8. Results and Discussion

Table 1 and 2 presents the performance metrics of regression models. Each model's Root Mean Squared Error (RMSE) and Mean Percentage Error (MPE) are demonstrated across five folds of cross-validation, along with the average performance across all folds. Among these, CNN Autoencoder demonstrates the highest RMSE across all folds, indicating larger prediction

errors compared to other models. Conversely, Bayesian Regression achieves the lowest RMSE, suggesting higher predictive accuracy.

The MPE values indicate the average percentage deviation of predicted values from actual values, with negative values denoting underpredictions and positive values denoting overpredictions. Overall, Bayesian Regression exhibits the most favorable performance in terms of both RMSE and MPE.

It's also worth noting that Very Deep MLP Model also demonstrated good performance and is a potential alternative for the task.

Table 1: Regression models' performance

	Model	RMSE	MPE (%)
Fold 1	CNN Forecasting	28,100.88	-0.522
	CNN Autoencoder	769,187.39	NaN
	RBFNN	20,679.23	0.004
	Shallow MLP	5,055.82	0.052
	Deep MLP	12,322.61	0.217
	Very Deep MLP	11,943.14	0.172
	LSTM	16,749.59	0.310
Fold 2	CNN Forecasting	8,518.21	-0.143
	CNN Autoencoder	506,402.47	NaN
	RBFNN	6,902.89	0.001
	Shallow MLP	1,683.53	0.004
	Deep MLP	1,243.72	0.000
	Very Deep MLP	778.28	-0.004
	LSTM	38,325.64	0.680
Fold 3	CNN Forecasting	2,667.18	0.040
	CNN Autoencoder	479,098.48	NaN
	RBFNN	4,214.74	0.001
	Shallow MLP	1,269.76	0.006
	Deep MLP	1,028.36	0.001
	Very Deep MLP	567.87	0.003
	LSTM	16,851.98	0.309
Fold 4	CNN Forecasting	846.17	0.006
	CNN Autoencoder	379,790.21	NaN
	RBFNN	13,814.02	0.002
	Shallow MLP	903.99	0.002
	Deep MLP	1,050.84	0.006
	Very Deep MLP	503.04	0.002
	LSTM	5,087.12	-0.055
Fold 5	CNN Forecasting	4,544.77	-0.080
	CNN Autoencoder	347,467.35	NaN
	RBFNN	2,999.89	0.000
	Shallow MLP	1,634.58	0.004
	Deep MLP	1,670.33	-0.006
	Very Deep MLP	758.45	0.004
	LSTM	4,870.53	-0.009
Average	CNN Forecasting	8,935.44	-0.140
	CNN Autoencoder	496,389.18	NaN
	RBFNN	9,722.15	0.002
	Shallow MLP	2,109.54	0.013
	Deep MLP	3,463.17	0.044
	Very Deep MLP	2,910.16	0.035
	LSTM	16,376.97	0.247

Table 2: Regression models' performance

	Model	RMSE	MPE (%)
Fold 1	Random Forest Regressor	13,086.52	-0.19272215
	Gradient Boosting	13,376.61	-0.19890741
	Bayesian Regression	0.00	1.62E-10
	ARIMA	15,988.37	NaN
Fold 2	Random Forest Regressor	245.90	0.00037396
	Gradient Boosting	332.27	-3.36E-05
	Bayesian Regression	0.01	-3.77E-10
	ARIMA	24,724.97	NaN
Fold 3	Random Forest Regressor	129.88	-0.00017848
	Gradient Boosting	294.43	-0.00097813
	Bayesian Regression	0.06	-8.43E-10
	ARIMA	7,679.59	NaN
Fold 4	Random Forest Regressor	323.52	-9.30E-04
	Gradient Boosting	398.32	-1.46E-03
	Bayesian Regression	0.19	-2.56E-09
	ARIMA	9,414.48	NaN
Fold 5	Random Forest Regressor	326.99	0.00016276
	Gradient Boosting	483.94	-0.00163115
	Bayesian Regression	0.44	5.20E-07
	ARIMA	30,977.25	NaN
Average	Random Forest Regressor	2,822.56	-0.0386587
	Gradient Boosting	2,977.11	-0.04060207
	Bayesian Regression	0.14	1.0322E-07
	ARIMA	17,756.93	NaN

References

Dash, R. and Dash, P.K. (2016) 'A hybrid stock trading framework integrating technical analysis with machine learning techniques', *The Journal of Finance and Data Science*, 2(1), pp. 42–57. Available at: https://doi.org/10.1016/J.JFDS.2016.03.002.
Deshmukh, S. and Sonkar, S.K. (2020) 'Mid-Price Stock Prediction with Deep Learning', *International Research Journal of Engineering and Technology* [Preprint]. Available at: www.irjet.net.
Han, Z. *et al.* (2021) 'A Review of Deep Learning Models for Time Series Prediction', *IEEE Sensors Journal*. Institute of Electrical and Electronics Engineers Inc., pp. 7833–7848. Available at: https://doi.org/10.1109/JSEN.2019.2923982.
Lee, K.H. and Wang, H. (2018) *BIG DATA ANALYTICAL ALGORITHMS AND SYSTEMS FOR HIGH FREQUENCY TRADING AND CONTRACT NETWORK ANALYSIS*.
McKerahan, T. *et al.* (2023) *Harnessing the Power of Artificial Intelligence in Stock Market Trading*. Available at: https://www.ijresm.com.
Ntakaris, A. *et al.* (2019) 'Feature Engineering for Mid-Price Prediction With Deep Learning', *IEEE Access*, 7, pp. 82390–82412. Available at: https://doi.org/10.1109/ACCESS.2019.2924353.
Passalis, N. *et al.* (2017) 'Time-series classification using neural Bag-of-Features', in *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE, pp. 301–305. Available at: https://doi.org/10.23919/EUSIPCO.2017.8081217.
Pole, A., West, M. and Harrison, J. (1994) *Applied Bayesian Forecasting and Time Series Analysis*. 1st edn. New York.
Saeys, Y., Abeel, T. and Van de Peer, Y. (2008) 'Robust Feature Selection Using Ensemble Feature Selection Techniques', in W. Daelemans, B. Goethals, and K. Morik (eds) *Machine Learning and Knowledge Discovery in Databases*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 313–325.
Tran Thanh, D. *et al.* (2017) 'Tensor Representation in High-Frequency Financial Data for Price Change Prediction'.

Appendix

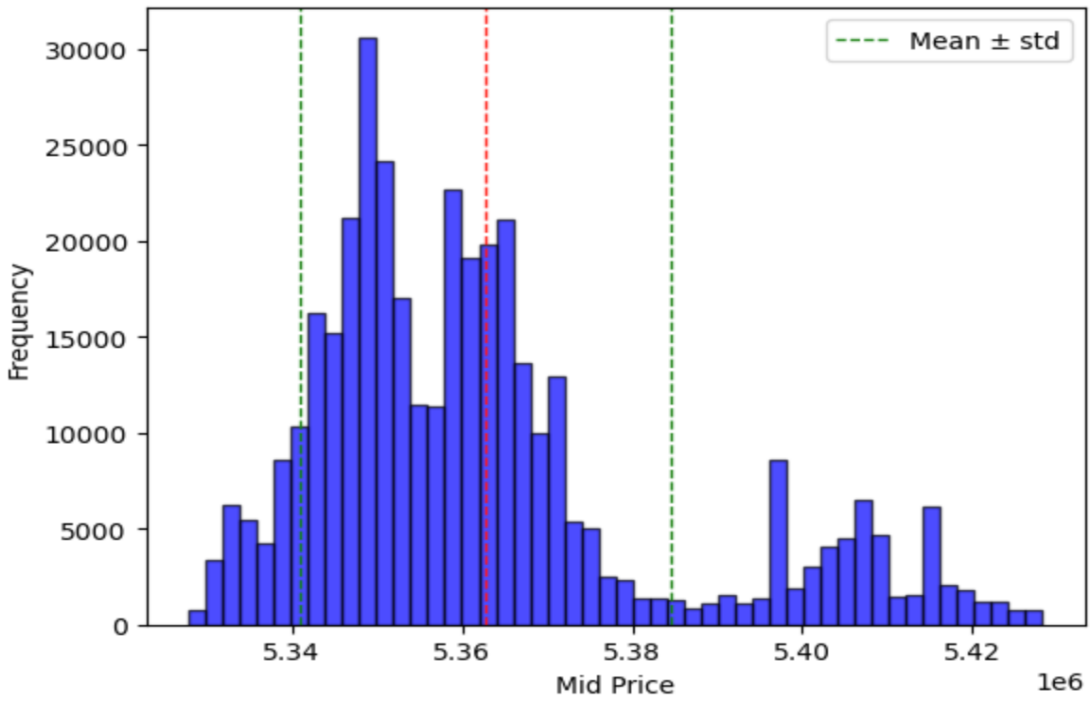

Figure 1. Histogram of Mid Price Focused around Mean and STD

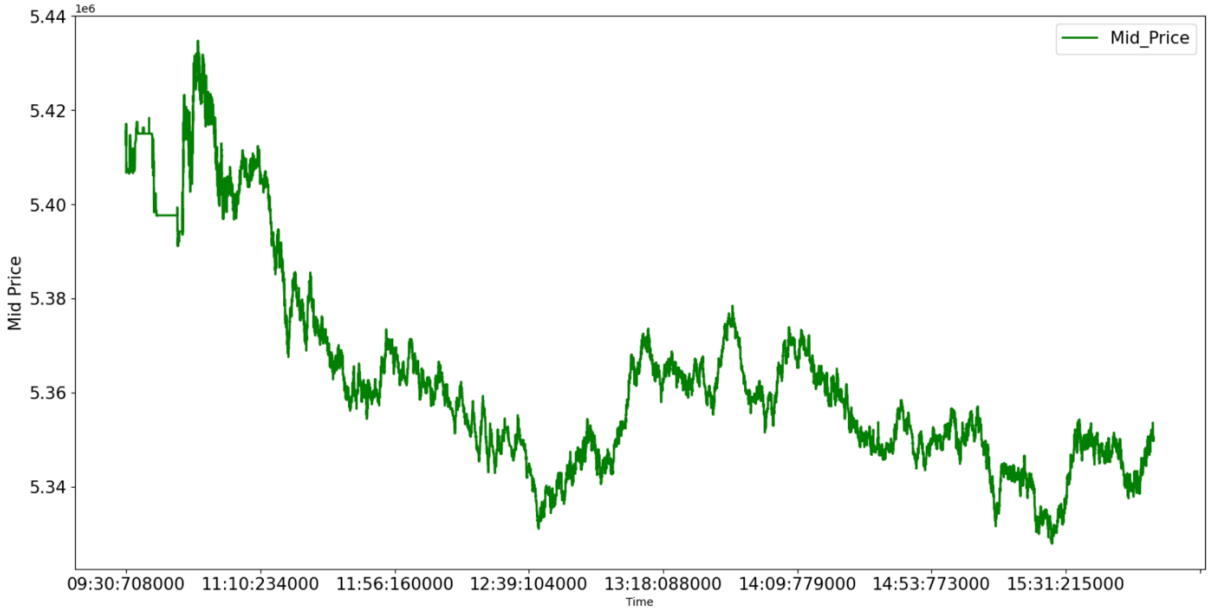

Figure 2. Time Series Trends of Mid Price

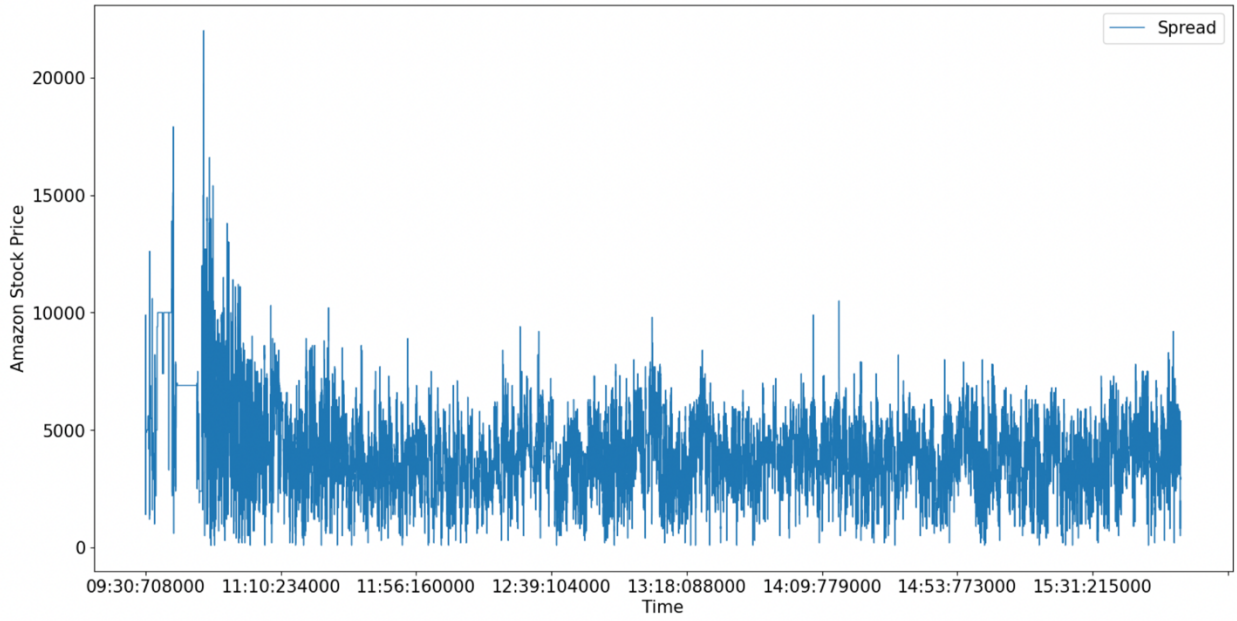

Figure 3. Time Series Visualization of Spread

CNN Autoencoder	MSE	RMSE
Fold 1	5.916492E+11	769,187.39
Fold 2	2.564435E+11	506,402.47
Fold 3	2.295354E+11	479,098.48
Fold 4	1.442406E+11	379,790.21
Fold 5	1.207336E+11	347,467.35

Figure 4. RMSE calculation for CNN Autoencoder

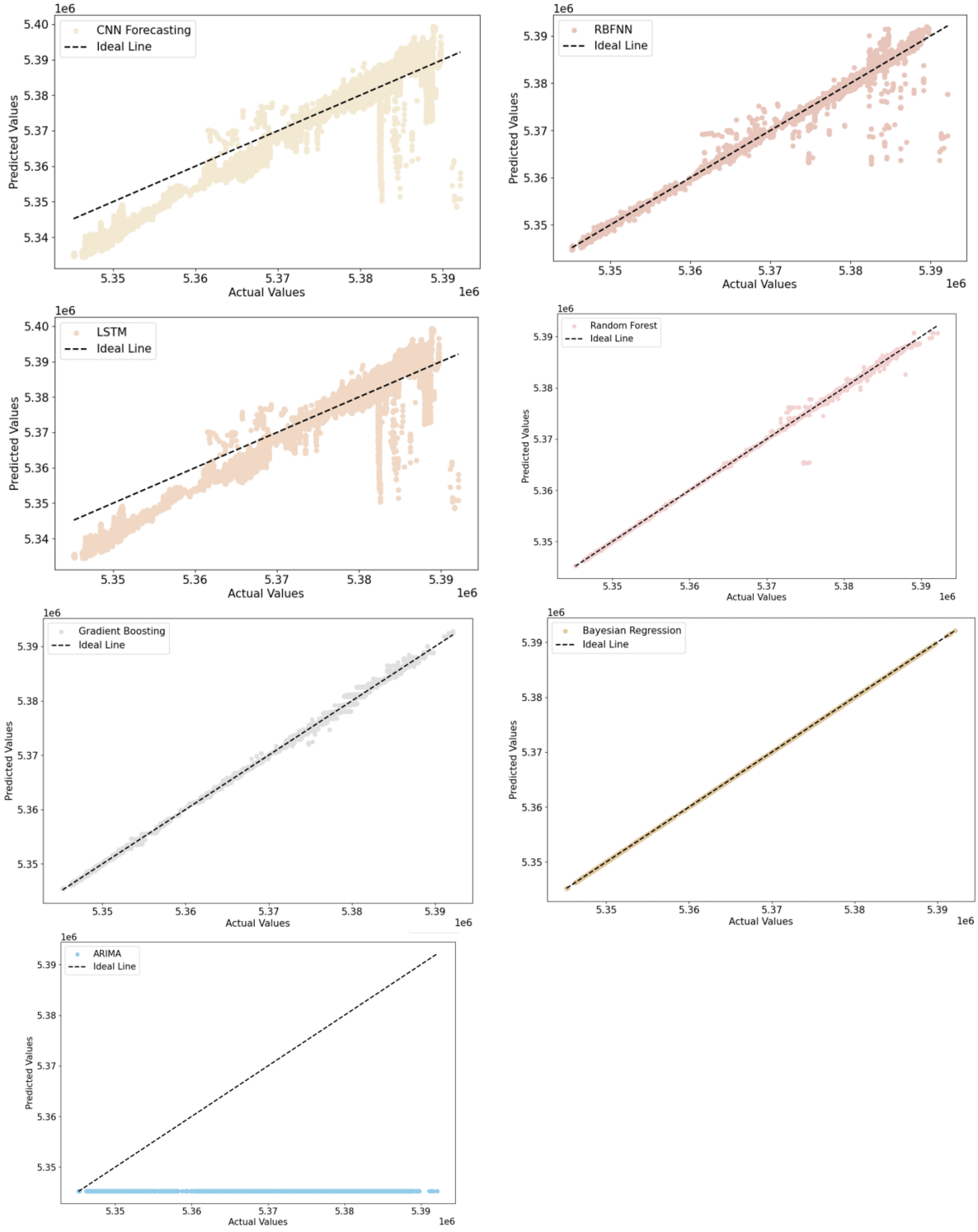

Figure 5. Actual vs Predicted Values for Multiple Regressors

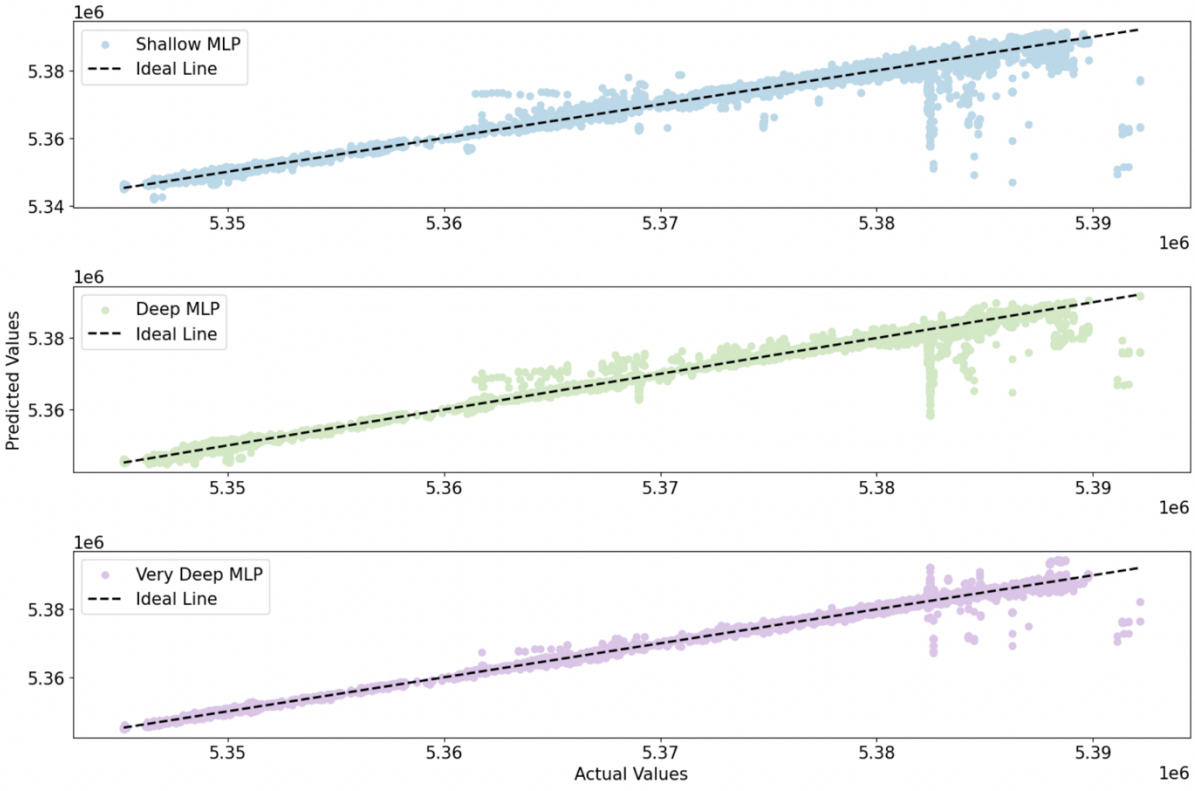

Figure 6. Predicted vs Actual Values for 3 level MLP