UNIVERSITY OF EDINBURGH
Business School

2022-23

CMSE11528 - Financial Machine Learning II (Practical)

Limit Order Book – Mid-Price Prediction using Neural Networks

B214479

1. Problem Statement and Introduction:

The current report aims to investigate the predictive performance of LSTM and CNN models for Mid-Price forecasting. Predicting Mid-Prices from a set of input features and some engineered features is a challenging task, given the complexity of structuring, engineering, and training basic and derived features to achieve a meaningful objective. The research will leverage LSTM and CNN models to develop an accurate and efficient Mid-Price prediction system.

2. Data Pre-Processing:

Data Used : *Seminar 1 material – S092215-v50-AMZN_OCT2_states.mat*

The input data is a .mat file which are binary files used by MATLAB. To read the file, a specialized python package 'hdf5storage'is installed. All columns are specifically named and read into a dataframe since the raw data doesn't by default has column names. The dataset has Epoch UNIX time by default – It is changed to a proper time format before we proceed to any key step in the process. The 'Time' column values are converted to '%Y-%m-%d %H:%M:%S.%f' format for better readability and to aid operations. In the next process, the dataframe is filtered to have datapoints between 0930 Hours and 1600 Hours. This ensures that we have data during the Market-hours. One of the important steps also includes setting the Time as Index, which is one of the key steps in handling time series data. Data is also resampled to take last value of every '20 millisecond' interval – This is being done to accommodate the whole data into the model, reduce complexity, RAM usage, and save time.

Next step involves a robust function to clean data which is crucial. The NaN values can seriously affect the prediction accuracy and increase loss functions. In this function, if a column has more than 40% missing data (Seminar 1/Lab 1 discussion) that column is dropped since it carries no weight to the input features. If the missing data is less than 40%, then the first five NaN values will be filled using ffill and bfill. After that, moving average of 5 rows is calculated for the missing NaN values. However, these are simple and basic approaches to handle missing data. It has been implemented here only because there is no/negligible missing data in raw dataset. There are sophisticated approaches to handle missing or corrupt data – Rolling Statistics, Interpolation, Imputation (Suhardi, Langi and Gondokaryon, 2016).

3. Feature Engineering:

In this section, we will discuss about the features derived from the current dataset and plausible explanation. As described by (Ntakaris, Kanniainen, *et al.*, 2019), There are several market and technical indicators that can be extracted from the available set. But to reduce time and complexity, only 3 derived features have been used here – Order Imbalance, Mid Price Difference and Simple Moving Average with a window of 5. The author tried to implement more features but the autoencoder took significant amount of time to extract features and therefore, they were reduced to 3 derived features. These are also some of the important Time-Insensitive and quantitative measures (Ntakaris, Kanniainen, *et al.*, 2019) that can be instrumental in predicting Mid-Price (Ntakaris, Mirone, *et al.*, 2019).

Order Imbalance: Calculating the order imbalance of the first bid-ask volume can help the LSTM model understand the supply-demand dynamics of the market.

Mid-Price Difference and Simple Moving Average: Calculating the mid-price difference and simple moving average can help the LSTM model understand the trend of the market. If the mid-price is increasing over time, it may suggest that the market is bullish, while if the mid-price is decreasing, it may suggest that the market is bearish.

Total Average Bid - Ask Price and Bid – Ask Volume: Calculating the total average bid and ask price and volume can help the LSTM model understand the liquidity of the market. Higher liquidity can lead to tighter bid-ask spreads, which can affect the mid-price.

Bollinger Bands: Calculating the Bollinger Bands can help the LSTM model understand the volatility of the market. If the mid-price is trading near the upper band, it may suggest that the market is overbought and due for a correction, while if the mid-price is trading near the lower band, it may suggest that the market is oversold and due for a rebound.

4. Feature Extraction:

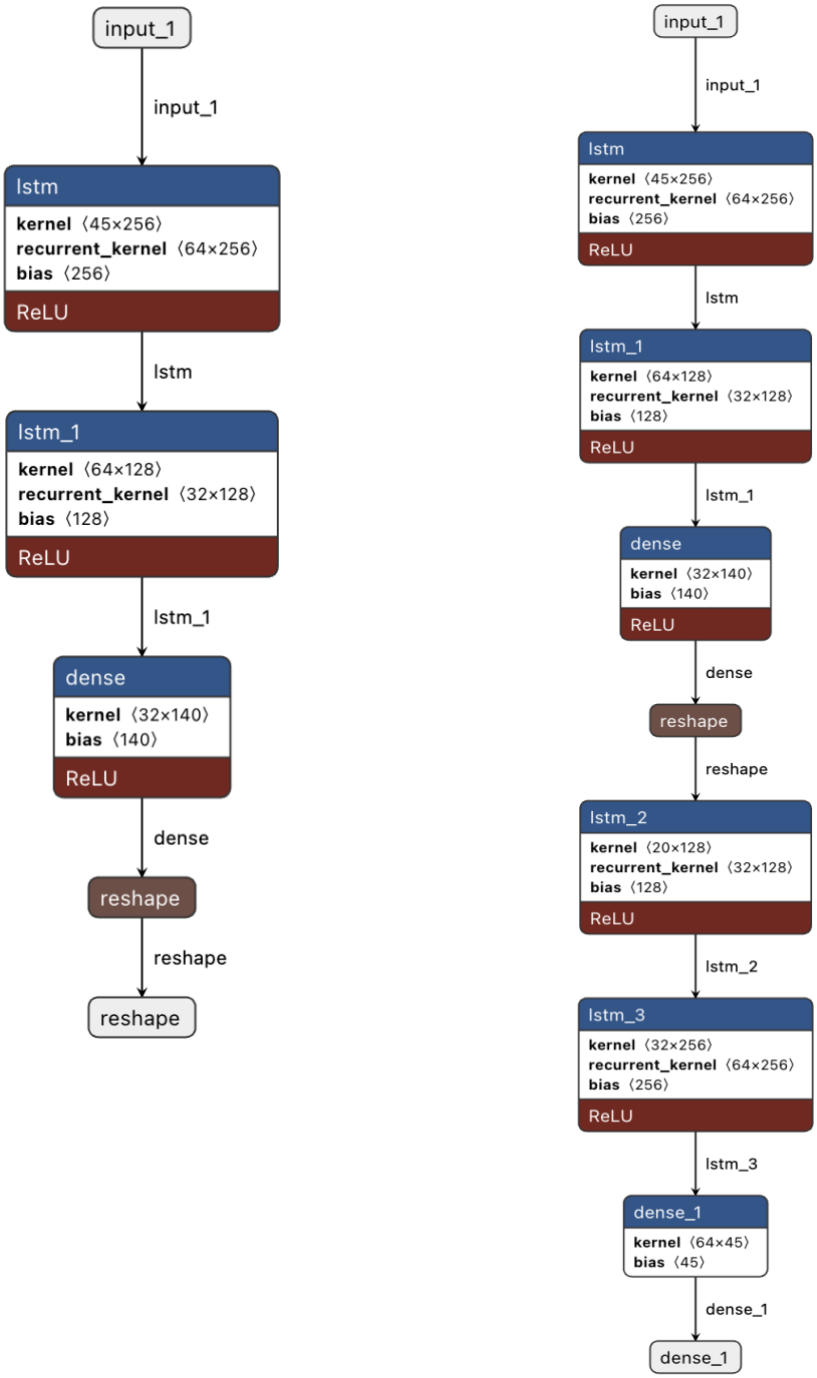


Fig 1: LSTM Encoder (left) and LSTM Auto-Encoder Architecture (Right)

The above charts (generated using ***Netron***) define implemented autoencoder architecture using LSTM layers to extract features from the input time series data. The autoencoder is used to compress the input data into a lower dimensional representation using an encoder, and then

decode it back to the original dimensions using a decoder. The compressed representation can be seen as a summary of the input time series data, capturing the most important information.

The encoder part of the autoencoder consists of two LSTM layers with decreasing number of neurons, followed by a fully connected dense layer with the number of neurons equal to the product of compressed dimension and look_back. The compressed representation is then reshaped to have the same shape as the input.

The autoencoder is compiled with the mean squared error loss function and the Adam optimizer. The new autoencoder architecture is well-suited for time series data because it can capture temporal dependencies and extract features from the input data that are most relevant for predicting the target variable. The LSTM layers are particularly useful for processing sequential data as they can remember information from previous time steps.

5. Model Architecture:

The below architecture diagrams outline the structure and design of the implemented models of CNN and LSTM. Data is trained on Both models simultaneously to observe which one yields the best results. As discussed in the Feature-Extraction section, the compressed representation from the encoder is used as the training/testing after concatenating the same with - input data to the CNN and LSTM models. Combined, the number of features is over 45. To implement the online manner, the initial dataset into divided into 2 halves. The first segment is directly fed into the models, trained and predictions are made using that. On the other hand, the segment 2 is divided into multiple chunks to illustrate new data arrival and predictions. The function create_data_sequence_IO() is an important component in the model where the X -y pairs are generated in real time and “look_back” number of steps are utilized to predict “horizon” number of steps. The function is suitable for other look_back – horizon combinations as well.

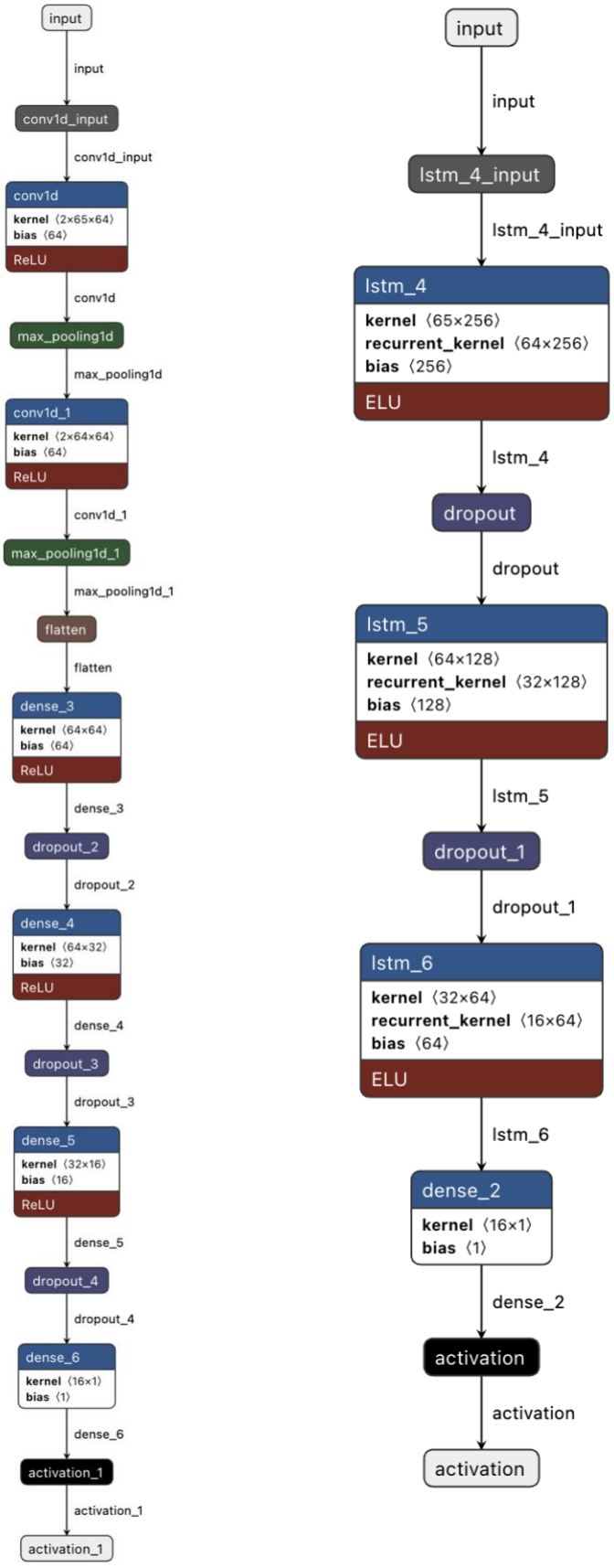


Fig 2: CNN (left) and LSTM (right) Models

Model Selection:

Before implementing the CNN/LSTM models, other potential models were also considered for implementation. When it comes to time series particularly stock data, the chosen model should be able to capture linear, non-linear, temporal and seasonal patterns from the data. With that in mind, we may not be able to utilize Linear models such as ARIMA / AR / ARMA (Selvin *et al.*, 2017). The reason is they do not capture the dynamics in the stock data. On the other hand, deep neural architectures can effectively overcome this issue. Even with Recurrent Neural Network, we have Vanishing gradient issue and also the use of tanh/relu activation functions for long sequences has been proved ineffective. But for time series data, ReLu function can be very helpful. LSTM overcomes these issues and trains using back-propagation. Although, CNN has been implemented here – LSTM is more interpretable than CNN since it keeps track of the internal state of the network over time, which can be used to interpret the predictions. In contrast, CNN operates in a feedforward manner, making it harder to interpret its results. LSTM is also better at modelling sequential data compared to other models.

6. Normalization:

This method is often preferred over other normalization methods for several reasons. Z-score normalization does not change the shape of the distribution of the original data. In contrast, other normalization methods such as min-max scaling can compress or expand the range of the data, which can alter the distribution. Z-score normalization is robust to outliers because it is based on the mean and standard deviation, which are less affected by outliers than other summary statistics like the median and interquartile range. Z-score normalization is particularly useful here, when comparing data from different distributions or when combining data from different sources. We combine compressed representations from encoder and the original input sequence. Here, it is enforced by normalize_ftn() while denormalization is done using denormalize_ftn()

7. Activation and Optimizer:

ELU (Exponential Linear Units) activation function is used here in LSTM model because it addresses the vanishing gradient problem which can occur in long sequence models. The activation function allows for negative values which helps to push the mean activation closer to zero, avoiding saturation of the activation function and improving the learning of the model. On the other hand, RELU (Rectified Linear Units) activation function is used in autoencoder model since it has been found to be more

efficient in training deep neural networks. The activation function allows the model to learn non-linear relationships between the input and output data and has been found to produce sparse representations, which can be useful for feature extraction. Linear activation function is often used in output layers everywhere because it produces unbounded values which is useful when predicting continuous variables. It does not distort the predicted values and maintains the linearity between the input and output variables. RELU (Rectified Linear Units) activation function is often used in CNN (Convolutional Neural Network) models because it has been found to be more computationally efficient.

Adam Optimizer: Adam optimizer is able to handle sparse gradients effectively and is robust to noisy or uninformative gradient updates. It is also able to adapt its learning rate on a per-parameter basis, allowing it to converge faster and with better accuracy than other optimization algorithms. Here, the models are designed to accept other optimizers as well.

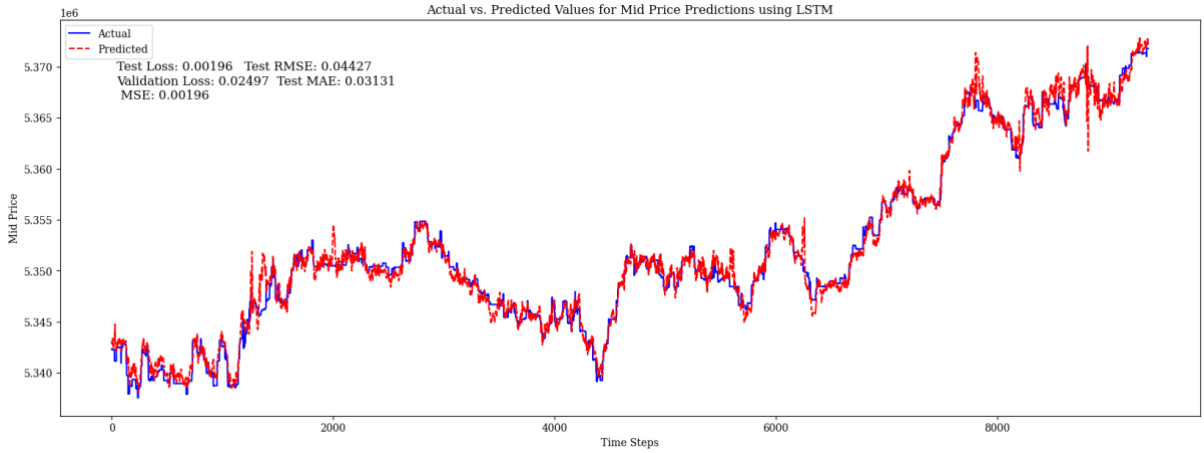
Regularization: Early-stopping, learning rate and drop rate parameters ensure that the model is not overfitted to a great extent.

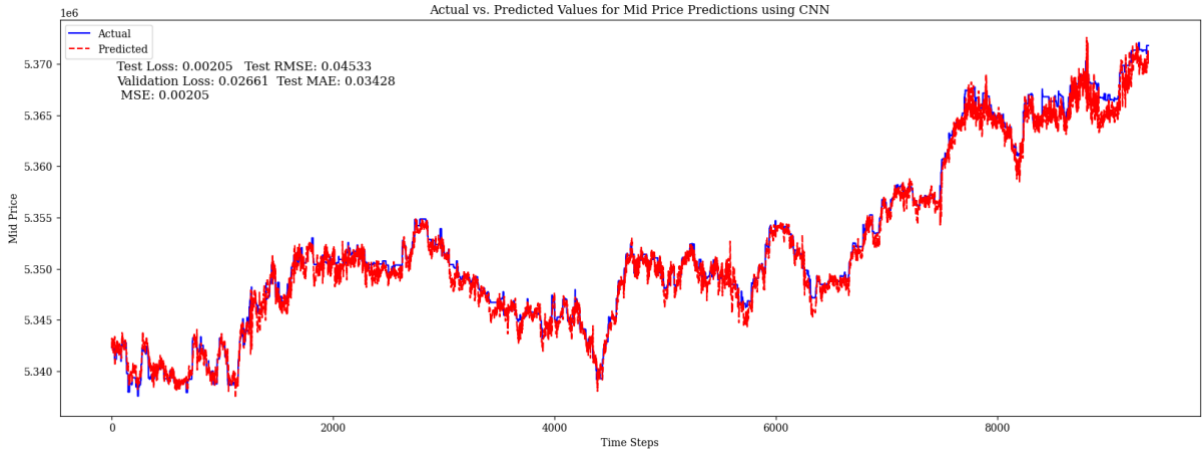
8. Metrics:

Mean squared error (MSE): This loss function measures the average squared difference between the reconstructed data and the original data. It is commonly used when modelling continuous data. MSE and RMSE are the key metrics chosen for all models including autoencoder. MSE (mean squared error) and RMSE (root mean squared error) are popular loss functions for deep learning models such as LSTM, CNN, and autoencoders. They are useful for measuring the difference between predicted and actual values and can be used for both regression and classification tasks. Their advantages include being easy to interpret and compute and penalizing larger errors more than smaller ones.

9. Results: (From Previous Run)

	MSE	RMSE	Test MAE	Val_loss
LSTM Model	0.00196	0.04427	0.03131	0.02497
CNN Model	0.00205	0.04533	0.03428	0.02661
LSTM – Chunk 1	0.02166	0.14718	0.12917	0.00048
CNN Model – Chunk 1	0.04476	21157	0.20275	0.00132





10. Conclusion:

In the context of Mid-Price prediction, the obtained results suggest that both LSTM and CNN architectures can be effective in predicting financial time series. However, the LSTM models show a slightly better performance in terms of prediction accuracy and convergence, potentially due to their ability to capture long-term dependencies and patterns in the data. The CNN models, on the other hand, have shown promising results in capturing the local temporal features in the data.

11. Reference:

Ntakaris, A., Mirone, G., *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.

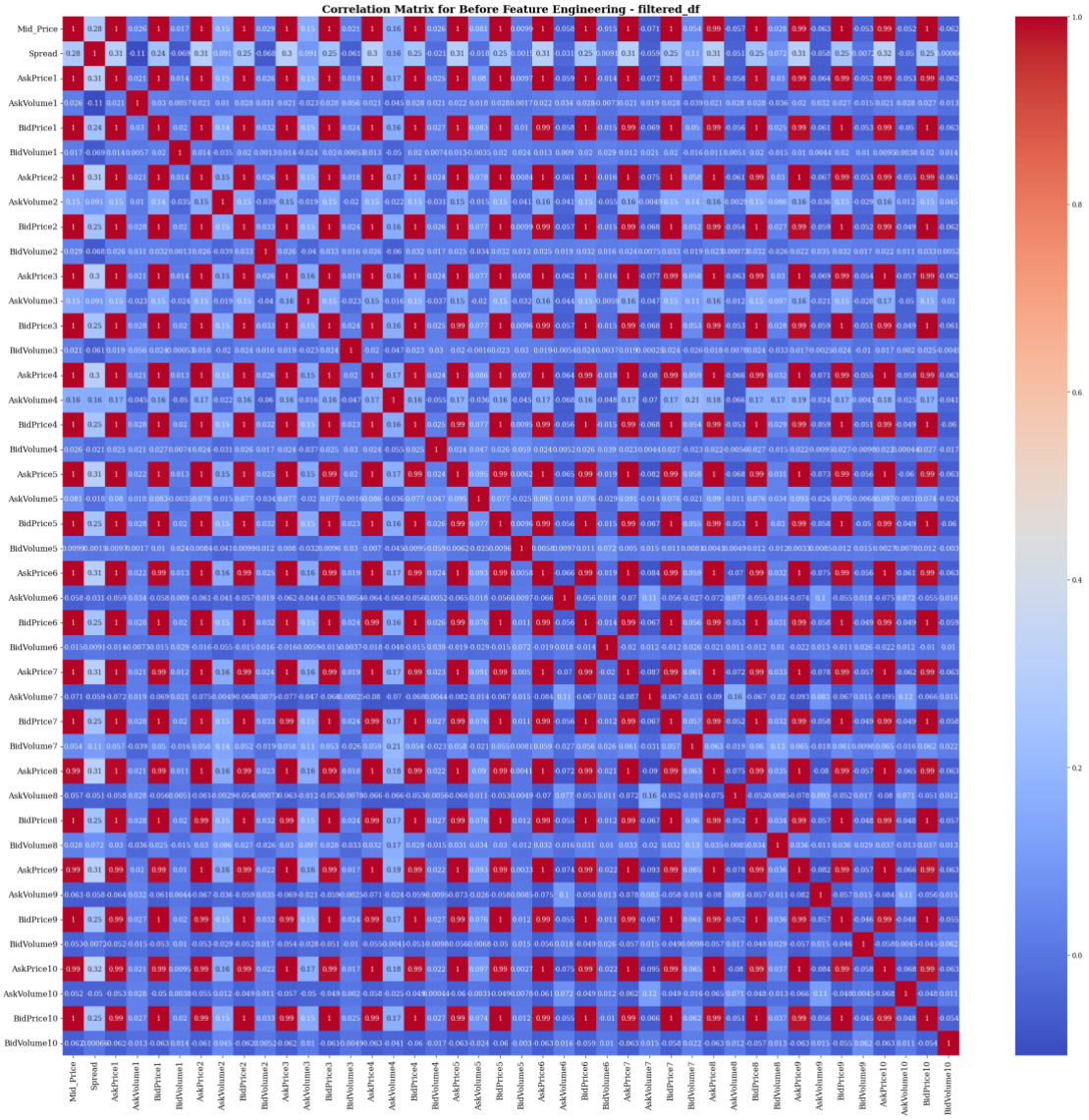
Ntakaris, A., Kanniainen, J., *et al.* (2019) 'Mid-price Prediction Based on Machine Learning Methods with Technical and Quantitative Indicators'. Available at: https://doi.org/10.48550/ARXIV.1907.09452.

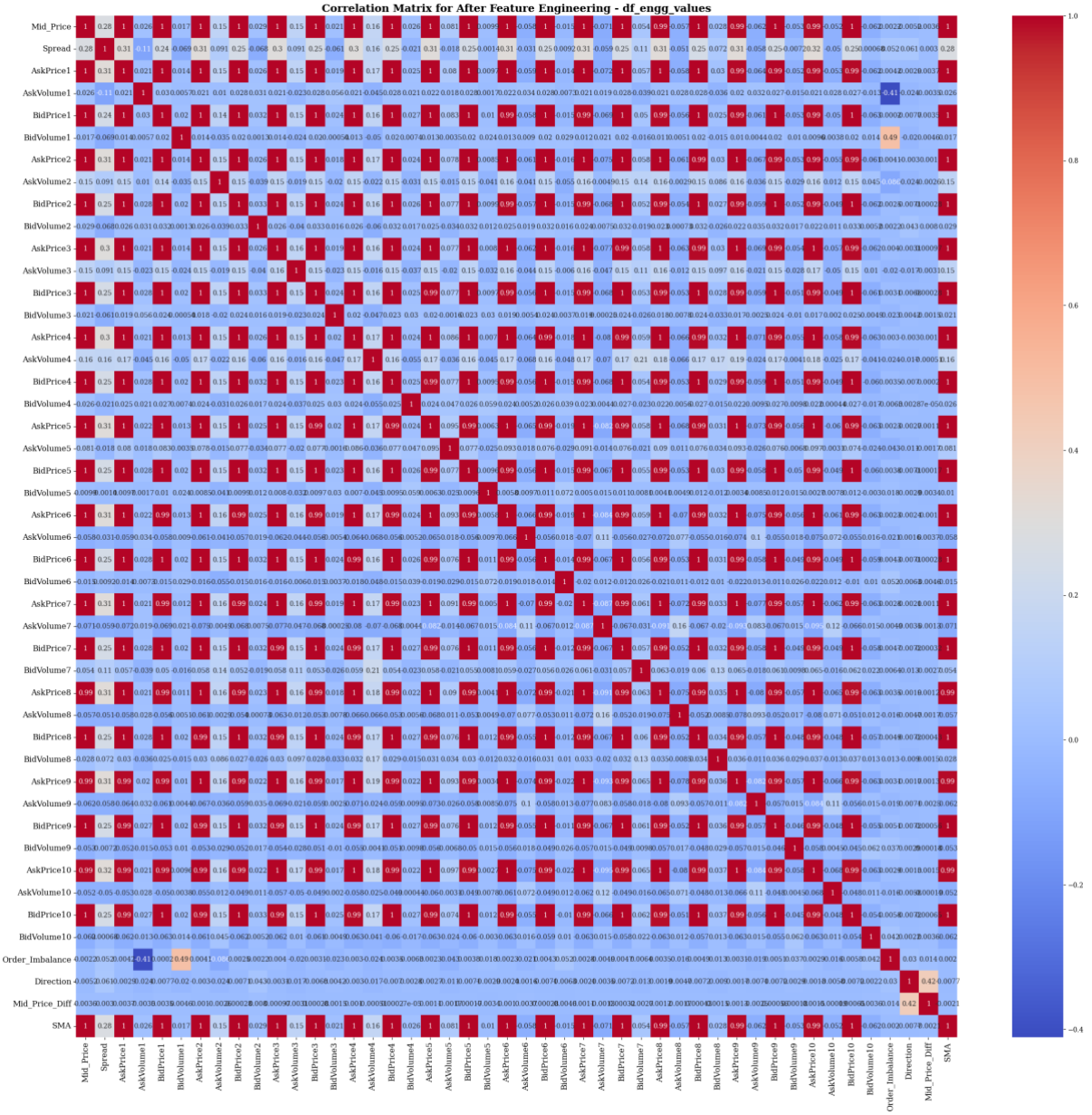
Selvin, S. *et al.* (2017) 'Stock price prediction using LSTM, RNN and CNN-sliding window model', in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi: IEEE, pp. 1643–1647. Available at: https://doi.org/10.1109/ICACCI.2017.8126078.

Suhardi, Langi, A.Z.R. and Gondokaryon, Y.S. (eds) (2016) '2016 International Conference on Information Technology Systems and Innovation (ICITSI): proceedings : October 24-27, 2016, Bandung - Bali, Indonesia'.

Appendix:

1. Feature Engineering Impact on correlation




Correlation Matrix for After Feature Engineering - df_engg_values
Mid_Price
Spread
AskPrice1
AskVolume1
BidPrice1
BidVolume1
AskPrice2
AskVolume2
BidPrice2
BidVolume2
AskPrice3
AskVolume3
BidPrice3
BidVolume3
AskPrice4
AskVolume4
BidPrice4
BidVolume4
AskPrice5
AskVolume5
BidPrice5
BidVolume5
AskPrice6
AskVolume6
BidPrice6
BidVolume6
AskPrice7
AskVolume7
BidPrice7
BidVolume7
AskPrice8
AskVolume8
BidPrice8
BidVolume8
AskPrice9
AskVolume9
BidPrice9
BidVolume9
AskPrice10
AskVolume10
BidPrice10
BidVolume10
Order_Imbalance
Direction
Mid_Price_Diff
SMA
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4