Predict Daily Dollar Volume of S&P500 Stocks

Daiying Yin

January 12, 2025

1 Introduction

In this report, we provide a basic attempt to predict the daily dollar volume of the stocks within the S&P500 index.

2 Data Download and Preparation

Using the package *yfinance*, we download stock data of the stocks from 2006-01-01 to 2019-12-31. For simplicity, we only keep the stocks with enough data, i.e., those that have at least 1500 data points (only a few stocks are exceptions). The datasets contain features: Open, Close, High, Low, Volume, Adj Close.

We combine the data frames of each stock by creating a feature called "Ticker", which represent the name of the stock in the final data frame. We convert the Date into datetime object and set it as the index.

For simplicity, we split the train and test set by the splitting time 2016-01-01. Of course, in the future, we can do better by updating the train and test set in a yearly-rolling fashion.

We also create a feature called "Sector" by going to the wikipedia page and find for each "Ticker" which sector it belongs to.

3 Data Preprocessing

We visualize the features in the dataset against Date grouping by "Ticker", and find four important observations to take care of.

1. The features contain many outliers. Therefore, for each Ticker and each feature, we compute the moving average and then the residuals with respect to the moving average. We then calculate the quantiles Q1 (25% quantile) and Q3 (75% quantile) and the IQR=Q3-Q1. For residuals that are below Q1-IQR or above Q3+IQR, we cap the residual at Q1-IQR or Q3+IQR, similar to the idea of John Tukey method. We would like to stress that, because the data is in time series, we can only cap the residuals but not the original features. (Otherwise, imaging the stock is monotonically increasing, then capping the stock price does not make sense.)

2. The features are skew-distributed, and the variance varies a lot with time. This is reflected in the histogram of the features grouped by Ticker. The daily dollar volume, which can be approximated by the product of Close and Volume, also turns out to be highly skewed. In this case, suppose we want to build a linear model, then we hope the residuals to be roughly normally distributed, therefore we would like to take the log-transformation of the features, as well as the target. This also helps these time series to appear stationary.

3. The stock prices exhibit different scales. If we do not scale them accordingly during the data preprocessing, then it will be troublesome to define a metric later for evaluating the prediction performance (because in that case, we have to compute a "weighted" metric). We choose to standardize the features within each stock.

4. The features are quite noisy, especially the Volume. Therefore, we decided to keep the original features and compute a smoothed version of the features later during feature engineering.

To summarize, we first deal with the outliers only in the training set (not the test set because we cannot see the future, and we don't do rolling window for simplicity). We then compute the target variable "Dollar_Volume" as the product of "Close" and "Volume" after dealing with outliers. We take the log-transformation of all the features and the target. Some visualizations are available in the Jupyter Notebook provided.

We perform additive seasonal_decomposition on the transformed data. This decomposition extracts the trend, seasonal and residuals of the daily dollar volume time series. From the QQ plots, it can be seen that the residuals are roughly normally distributed now.

4 Selection of Evaluation Metrics

1. Mean Absolute Error:

MAE is easy to understand and interpret. It treats all errors equally, and hence is more robust to outliers. Given the noise and variance of the daily dollar volume, we believe MAE is the most important metric to look at. Since we have scaled the Dollar_Volume within each Ticker, it is safe to stack the error of different stocks all together and compute the MAE (otherwise we have to worry about the weights).

2. Root Mean Squared Error:

RMSE is more sensitive to outliers compared to MAE. We use RMSE indeed of MSE because it is in the same scale as the original data.

3. Mean Absolute Percentage Error (MAPE):

Although this metric is scale-independent, we realized it is *not* very useful here, because we found that, there are certain dates and certain stocks for which the volume is zero, then the percentage change will be huge.

4. R-square and Adjusted R-square:

Adjusted R-square accounts for the fact that adding more variables to the model always increases R-square, regardless of whether the new variables are truly relevant.

5. Trimmed MAE:

As shown in Figure 1, there are unavoidable outliers in the testing data, leading to huge residuals. To address this, we trim the upper and lower 0.1% of the residuals, before computing the MAE.

6. Trimmed Residual Variance:

We compute the variance of the residual after trimming the extreme values. Since the QQ plot of the residual in the end is approximately normal, we believe the residual variance provides insight of the quality of the prediction.

7. Residuals Analysis:

Residual analysis helps in detecting model mis-specification and patterns that the model hasn't captured. We use the QQ plot of the trimmed residual to investigate the behavior of the residuals.

8. Directional Accuracy:

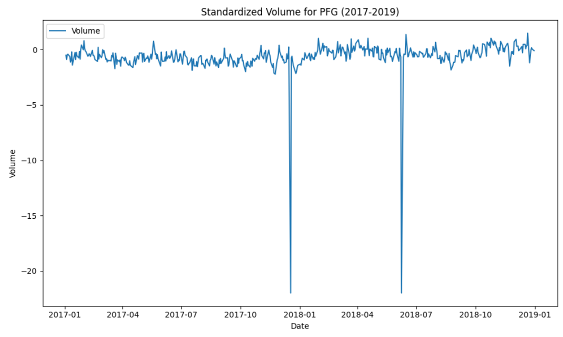
This metric computes the proportion of predictions that correctly forecasts whether the Dollar_Volume the next day is increasing or decreasing. Due to the time constraints, we only implement this metric for the baseline model.

5 Model Implementation and Optimization

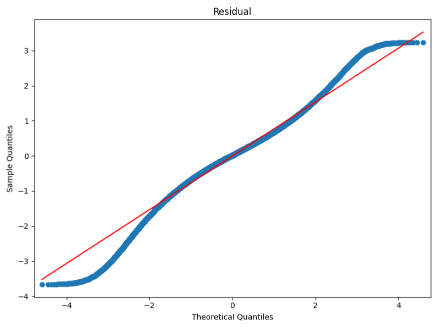
We decided to merge the model implementation and research paper optimization into the same section, because there are many overlaps between the two, for example, feature engineering, input sequence generation, and hyperparameter tunning, etc. The part that used ideas/model architectures from the research paper [3] will be marked in RED.

Baseline

The baseline is given by predicting the next-day Dollar_Volume using the observation today. We draw the QQ plot of the residual, there seem to be several large outliers. The explanation for this is that, we only deal with the outlier in the training set, but during the test period, there are indeed some dramatic changes in Volume, and we cannot change the outlier behaviour in the test set. This can be seen in the following example, where we visualize the Volume of the stock "PFG" during 2017-2019. For this reason, from now on, we only visualize the QQ plot of the trimmed residuals.



(a) PFG Volume (2017-2019)



(b) Baseline Residual Trimmed

Figure 1: Baseline Residual and PFG Volume (2017-2019)

Feature Engineering

Before fitting any model, we have to perform feature engineering. The "raw" features are Open, Close, High, Low, Volume, Adj Close. The added feature consists of both self-proposing features and the ones used in the research paper [3]. In the following, we shall refer to the added features "augmented features".

1. We onehot encode the "Sector" to make it numerical.

2. We extract some datetime features including dayofmonth, dayofweek, month, year, quarter due to the seasonality exhibited in the time series data.

3. We include the Dollar_Volume as one of the feature to predict the next day.

4. We include the cumulative sum, simple moving average, exponential moving average of the features “Volume”, “Close” and “Dollar_Volume” because as pointed out before, the raw features are very noisy and hence needs smoothing.

5. We compute the technical indicators including Momentum, CCI, RSI, RSI-based oversold and overbought, ATR, CMF, KDJ, WR and MACDH, where some of them are proposed in [3].

6. We compute the aggregated features of “Volume”, “Close” and “Dollar_Volume” via grouping by “Sector” and compute the mean and standard deviation.

Create Sequence

Here we propose three different method of constructing input into models

1. *create_sequence*: Instead of using just the features available at the current time, this function aggregate the lagged past features of several timestamps in order to predict the next Dollar_Volume. For example, it uses features at time $t-3, t-2, t-1$ to predict value at time t.

2. *create_sequence_shift*: This function prepares features at time $t-3, t-2, t-1$ to predict simultaneously the Dollar_Volume at $t-2, t-1, t$. We believe this will enhance the robustness of the training. This sliding window approach is proposed in [3].

3. *create_sequence_shift_all*: Considering that the target Dollar_Volume is directly related to Close and Volume This function prepares features at time $t-3, t-2, t-1$ to predict simultaneously the (Close, Volume, Dollar_Volume) at $t-2, t-1, t$.

Sequence_length is a parameter that indicates how many past days are used for prediction. In our experiment, we mostly set this parameter as 3, as in [3].

Lasso Regression

We first run a lasso regression with the raw features (without feature engineering), and with sequence_length=1.

We then run two lasso regressions with the augmented features, and with sequence_length equals 1 and 3 respectively. Lasso is very good at *feature selection*. Although our features are created in a way that some of the features could have a high correlation, we expect a good regularization of the Lasso regression will help to select features.

We also run a linear regression with a PCA applied to reduce dimensionality and to remove redundant variance explanation. This is inspired by [2].

Model optimization is via grid-search cross-validation to find the best regularization alpha.

XGBoost Regressor

XGBoost is an optimization method based on gradient-boosting trees, which allows to train a tree model with gradient method, aimed at correcting the residuals computed from the previous iteration. The family of boosting method focuses on reducing the bias, but potentially increases the variance.

We run an XGB regressor with the augmented features, and with sequence_length=3.

Model optimization is also via grid-search cross-validation to tune the hyperparameters that prevent overfitting in tree models, including max_depth, subsample ratio, max_features, L_1 regularization, min sample to split.

Feed-Forward Neural Network

We first implement the same Feed-Forward Network architecture as in the paper [3], i.e. single layer with 48 nodes and Dropout = 0.6, for the input data generated by i.e. *create_sequence*. We tune the hyperparameters, and set the number of nodes as 200 for input data generated by *create_sequence_shift*. Similarly, we set the number of nodes as 300 for input data generated by *create_sequence_shift_all*. This makes sense because the dimensions of the output are increased from 1 to 3 and then to 3×3. All the experiments are with the augmented features and sequence_length=3. We perform hyperparameter tunning for the batch size, learning rate, early_stop_rounds, etc.

LSTM model

We implement the same LSTM architecture as in the paper, i.e. two LSTM blocks each with 32 nodes and Dropout = 0.6, combined with the two different input-generating schemes, i.e. *create_sequence*, *create_sequence_shift*, all with sequence length=3. We perform hyperparameter tunning for the number of layer, nodes, batch size, learning rate, early_stop_rounds, etc.

Ensemble Stacking

In the paper [3], it is proposed to apply a linear regression without intercept and with all coefficients constrained to be nonnegative, in order to stack ensemble the output of several individual models, including a Random Forest, a Feed-Forward Network, as well as two LSTMs. In our case, the best-performing models are Lasso (MAE 0.5038), and the single-layer Feed-Forward Network with 3 different types of data generation schemes (MAE 0.5058, 0.5056, 0.5050 respectively). The XGB model gives a slightly larger MAE of 0.5085. Therefore, we propose two schemes of ensemble stacking. We also found that including an intercept for the linear regression works better in our case.

Scheme 1: Ensemble Lasso, XGBoost, FFN, FFN_shift and FFN_shift_all.

Scheme 2: Ensemble Lasso, FFN, FFN_shift and FFN_shift_all.

Another Idea of Formulating the Target

As mentioned in the previous section, we choose to predict the target variable, being the standardized dollar volume (standardized within each stock so that we don't need to care about the scale of different stocks). Indeed, another way to remove the effect of scale, is to predict the percentage return after the log-transformation, as what is done in the paper [3]. An advantage of this approach is that, the log-return of the daily dollar volume $\log\left(\frac{S_t \cdot V_t}{S_{t-1} \cdot V_{t-1}}\right) = \log\left(\frac{S_t}{S_{t-1}}\right) + \log\left(\frac{V_t}{V_{t-1}}\right)$. In this manner, the targeting data would have a smaller variance, and the problem of predicting the daily volume can be decomposed into two problems, where for each problem it will be easier to find a benchmark in the literature. (Perhaps one way to interpret it is to) Due to the time constraint, we leave this as a future work.

What we took from the paper [3]

The log-transformation to correct skewness and stabilize variance, The scaling which makes different stocks comparable, The directional accuracy as a metric, Selection of technical indicators as features, RandomForest model (not performing well), Feed Forward Network architecture, LSTM architecture, The idea of creating shift when generating the input sequence, Ensemble Stacking method.

Reason for choosing this paper [3]

The paper was chosen for many reasons.

First, this paper looks interesting. It proposes to use technical indicators, financial reports, and online text data to predict stock returns, although for the project we in the end only implemented the technical indicators due to the time constriant. Similar to our project requirement, the paper also implemented linear models, tree-based models as well as AI models. By looking at the numerical results in this paper, we could have an idea what the relative performance of the models could be. Moreover, the paper comes with code on Github, although we write our own code, it is good to check theirs first and to double confirm the logic and some of the details that need attention.

Secondly, we encounter a very similar phenomenon that appears in this paper, that is, the numerical performance does not necessarily improve much when more complicated models are fitted. For example, see Table 9 of the paper [3], the MAE from random forests to simple FFN to LSTM are almost the same, and sometimes could get worse. In our experiments, it turns out that the FNN model is slightly better than Lasso regression, which performs better than XGBoost, whereas the LSTM model suffers from overfitting. *Given this similar situation, we believe that we should not think about adding fancy tricks or more complexity to the model, but should think about how to improve the performance with what we have.* Hence, we are curious to see whether the Ensemble Stacking method proposed in [3] indeed improves the robustness of the performance. As we know, ensemble method is simple but powerful and especially widely applied on the industrial level because of its robustness and variance reduction.

As a final remark, we point out that the research paper [3] is under the setting of predicting the stock price return. Predicting the return makes it easier to compare the value among different stocks. On the other hand, we start the project with another way to deal with the different scales, by standardizing the data within each stock group. Since both the paper [3] and our approach transform the data independent of the stock, we believe the methodology proposed in [3] is also applicable in our setting, perhaps just with different model assumptions.

6 Results Discussion

Metric Analysis

We summarize the experimental results in the table below. (Note that l=3 refers to sequence_length=3, i.e. using 3-day lagged features.)

Comparing row 2 and row 3, we see that feature engineering significantly improve the result in the Lasso model. Therefore, for the rest of the models, we stick to use the augmented features.

Comparing row 3 and row 4, it seems that it is much better to let Lasso perform the feature selection for us, than using PCA to reduce the dimension and then keep all the features given by PCA.

Comparing row 3 and row 5, we see that lagged features slightly improve the result in the Lasso model.

Row	Model	**MAE**	RMSE	Adj R-Square	Trim Resid Variance	Trim MAE
1	Baseline	**0.5854**	0.8264	0.5724	0.5983	0.5771
2	Lasso Raw Feature (l=1)	**0.5560**	0.7726	0.6265	0.5250	0.5484
3	Lasso (l=1)	**0.5055**	0.7160	0.6792	0.4512	0.4980
4	Linear (l=1) with PCA	**0.5787**	0.7873	0.6121	0.5461	0.5711
5	Lasso (l=3)	**0.5038**	0.7122	0.6823	0.4490	0.4963
6	XGBoost (l=3)	**0.5085**	0.7176	0.6775	0.4502	0.5017
7	FFN (l=3)	**0.5058**	0.7140	0.6807	0.4502	0.4985
8	FFN shift (l=3)	**0.5056**	0.7135	0.6812	0.4463	0.4983
9	FFN shift all (l=3)	**0.5050**	0.7102	0.6841	0.4492	0.4977
10	LSTM (l=3)	**0.5472**	0.7552	0.6429	0.5147	0.5399
11	LSTM shift (l=3)	**0.5366**	0.7552	0.6428	0.5027	0.5292
12	Ensemble scheme 1	**0.5059**	0.7135	0.6812	0.4476	0.4990
13	Ensemble scheme 2	**0.5019**	0.7082	0.6860	0.4439	0.4954

Table 1: Performance Metrics for Different Models

Comparing row 5 and row 6, we see that, after feature engineering, Tree-based models do not necessarily outperform linear models. We believe that *feature engineering* already compensates for some of the nonlinearity that the linear model lacks of. On the other hand, XGBoost, which is a boosting tree model, could suffer from overfitting, even though we have set early_stop_rounds to regularize.

Comparing row 5 and rows 7,8,9, we see that the FFN models can achieve performance comparable to the Lasso model, for example, row 9 has beat row 5 in terms of adjusted R-square. Among the rows 7,8,9, we see that the trick "shift" and "shift_all" indeed improves the performance.

Looking at the rows 10 and 11, it is clear that LSTM models suffer from overfitting, at least with the current effect of hyperparameter tunning. We think that with more detailed tunning in the future, the LSTM models perhaps could improve.

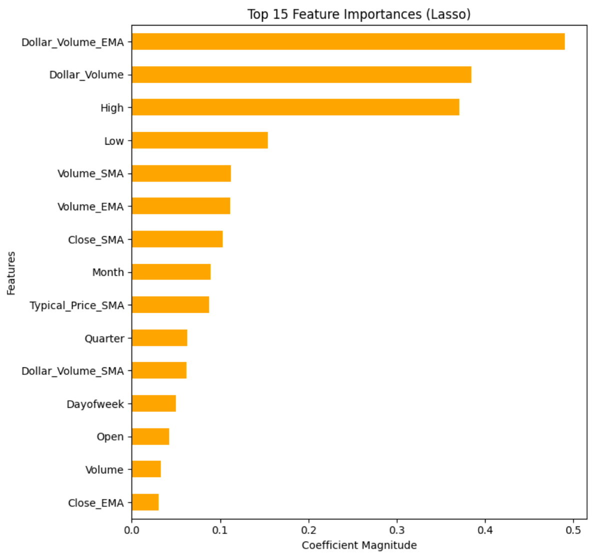
Finally, since the performance of the individual XGBoost model does not match that of Lasso and FFN, the MAE of ensemble scheme 1 does not improve much, but its residual variance decreases significantly. In addition, the ensemble scheme 2 outperforms every single other model in all metrics significantly, demonstrate the effectiveness of ensemble stacking.

Feature Importance

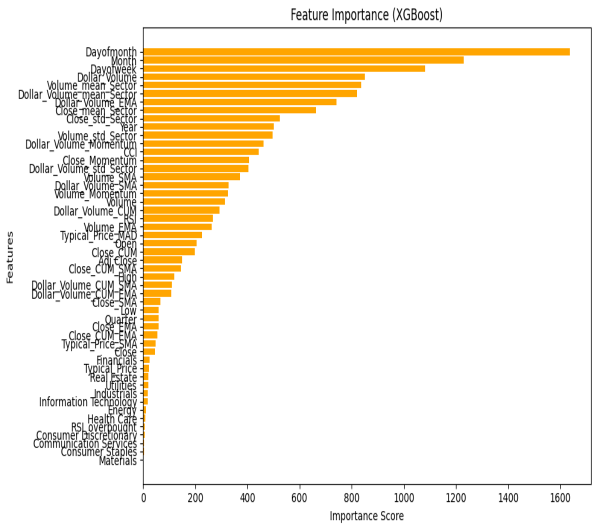
We visualize the feature importance of Lasso and XGBoost because for neural networks it is not feasible to do so. Feature importance of Lasso is characterized by the coefficient magnitude, whereas that of the XGBoost model is characterized by the variance explanation. We observe that the previous value, simple moving average, exponential moving average as well as the momentum of Dollar_Value, Close and Volume constitute the most important part of the features, which makes sense because they are the quantities that directly affect dollar volume.

In addition, the datetime feature such as month, dayofmonth and dayofweek also plays an important role. This is also expected because as visualized in the Jupyter Notebook, the dollar volume exhibits clear seasonality on various levels of time span.

Moreover, the XGBoost model seems to capture better the sector-aggregated features, e.g. "Dollar_Volume_mean_Sector", "Volume_mean_Sector", "Close_std_Sector", which computes the mean and standard deviation of stocks in the same Sector, trying to capture the sector behaviour.



(a) Lasso Feature Importance



(b) XGBoost Feature Importance

Figure 2: Comparison of Feature Importance: Lasso vs. XGBoost

Residual Analysis

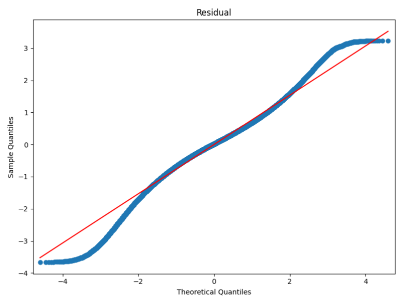
In addition, we visualize below the trimmed residuals of some of the models, and compare it with the baseline. We see that the baseline residual has both a light left tail and a light right tail. (Indeed, by trimming we are already creating a slightly lighter tail on both sides. But let's just assume that the 0.1% trimmed residuals are outliers). Compared with the baseline, we observe that most of the models are trying to correct the left tail, and at the same time make the right tail even lighter. A lighter right tail indicates that, after training, there are fewer times when the groundtruth dollar volume is significantly higher than the predicted dollar volume. At the same time, a heavier left tail indicates that there are more times when the groundtruth dollar volume is significantly lower than the predicted dollar volume. More precisely, the result indicates that, compared to the baseline, the trained models tend to predict higher values, that is, *the models believe that it is generally more likely that the dollar volume tomorrow is going to be higher than that of today.* This makes sense because the stock market is doing great from 2006 to 2016 except for a short time in the 2008 crisis. In addition, take the Ensemble 2 model for example, *the behavior of the residual tells us that, the model predicts small values normally, but predicts aggressively large values.* On the other hand, since we have taken log-transformation and standardization in the beginning, the groundtruth dollar volume is more or less normal as visualized in the code. Therefore, a future work is to deal with the tailed-behaviour of residuals, for example using ARIMA to check if a pattern exists.

7 Potential Improvements

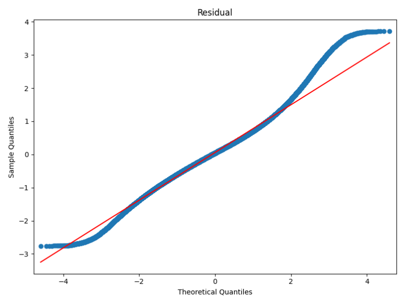
1. For the data preprocessing, we exclude several stocks because they have insufficient data points from Yahoo. Also, there are still certain periods during which some stock prices are missing (although very few). This leads to several discontinuities of dollar volumes (creating a "jump"), which could negatively affect the training. The paper [3] provides a more detailed treatment by combining different data sources for the missing values. This has to be one of the future improvements.

2. As mentioned above, in view of $\log\left(\frac{S_t \cdot V_t}{S_{t-1} \cdot V_{t-1}}\right) = \log\left(\frac{S_t}{S_{t-1}}\right) + \log\left(\frac{V_t}{V_{t-1}}\right)$, we could have tried to predict separately the percentage return and the percentage volume change, which might lead to better numerical stability. Also there will be more benchmarks to compare with.

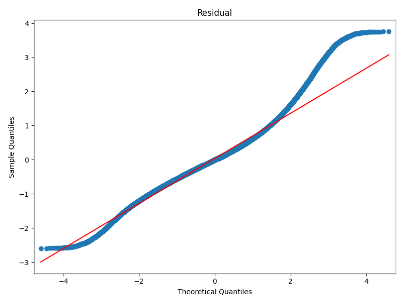
3. One easy improvement is to perform the training and testing in a rolling fashion. For example,



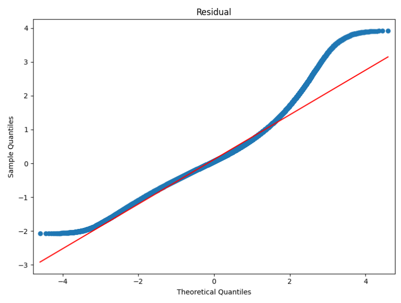
(a) Baseline



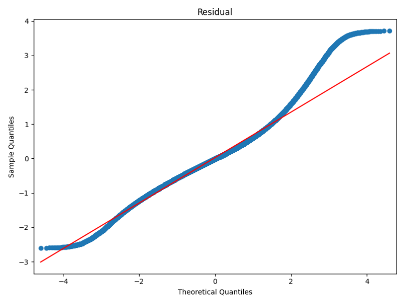
(b) Lasso raw features



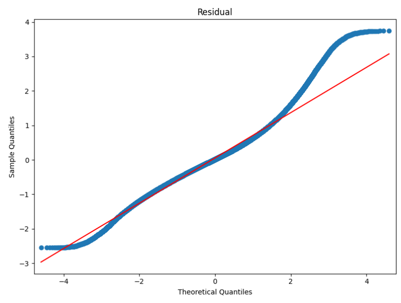
(c) Lasso l=3



(d) XGBoost



(e) FFN shift all



(f) Ensemble 2

Figure 3: QQ-plot of Trimmed Residuals

when predicting the year 2017, we could add the year 2016 as training data. In this way, we can investigate the performance for each year.

4. It is clear that LSTM model suffers from overfitting issues, although it works in [3]. Resolving this issue can be left as future work.

5. As mentioned in the result analysis, it remains to further investigate the distributional behavior of the residuals. Perhaps use time series models, e.g., ARIMA, to investigate if certain pattern exists.

6. Due to the time constraints, we didn't perform feature selection except for the PCA in Lasso. For future work, we can use the pairwise correlation matrix to filter out highly correlated features, and select the features that are mostly correlated with the target. Speaking of correlation, we can try to input the stock correlation structure as features.

7. There could be more "personalized" ways to encode the information of "stock ticker" into the model. Our current approach involves one-hot encoding of the "Sector" of the stock as a feature, which we believe is more robust. In [3], they propose to fine-tune the LSTM model using the stock's data before prediction, however, we believe this could lead to an even more severe overfitting. One solution to this is to add data augmentation following the approach in our own research paper [1].

References

[1] Ariel Neufeld, Julian Sester, and Daiying Yin. Detecting data-driven robust statistical arbitrage strategies with deep neural networks. 2024.

[2] Muhammad Waqar, Hassan Dawood, Ping Guo, Muhammad Bilal Shahnawaz, and Mustansar Ali Ghazanfar. Prediction of stock market by principal component analysis. In *2017 13th International Conference on Computational Intelligence and Security (CIS)*, pages 599–602, 2017.

[3] Shanli Zhong and David B. Hitchcock. S&p 500 stock price prediction using technical, fundamental and text data. *ArXiv*, abs/2108.10826, 2021.