Prediction of traded volume

Martin Selecký

August 19, 2020

1 Introduction

In this report we describe three methods for forecasting next day traded volume of S&P 500 index. The modeling methods we used represent three different approaches to time series modeling – autoregressive models, random forests, and recurrent neural networks. We describe their advantages, disadvantages, and compare their results with each other's as well as with a reference model $\hat{E}[v_{d+1}] = v_d$.

2 Exploratory Data Analysis

The financial data used for modeling and evaluation were downloaded from Yahoo Finance, in particular these are S&P 500 index data for time period 1.1.2000-31.12.2018. The volume and close price data are shown in Figure 1 and Figure 2 respectively, and we can observe both trend and seasonality in these data, which is more visible in the seasonal-decompose graph in Figure 3.

From scatter plot showing correlations of individual values in Figure 4 it is visible that individual price data are highly correlated with each other unlike with the traded volume.

The partial autocorrelation graph in Figure 5 shows how previous lags influence the actual traded volume. There is a strong influence of previous 5 days as well as a bit weaker influence of lags 9, 10, 18, and 19 on the current traded volume.

Also, the box plot in Figure 6 shows distribution of the volume data as well as the outlayers in terms of larger traded volumes in some limited number of days.

Historical events Apart from the basic financial data, we have tried to enhance the data with dates and importances of historical financial events. There is an API[1] where these events can be obtained, however it is not free. Thus, we have downloaded these events from publicly available economic calendar[2], however there, at least for first couple years, there is many events missing which may have negative impact on the models predictive capability.

[1] https://tradingeconomics.com/analytics/api.aspx
[2] https://www.fxstreet.com/economic-calendar

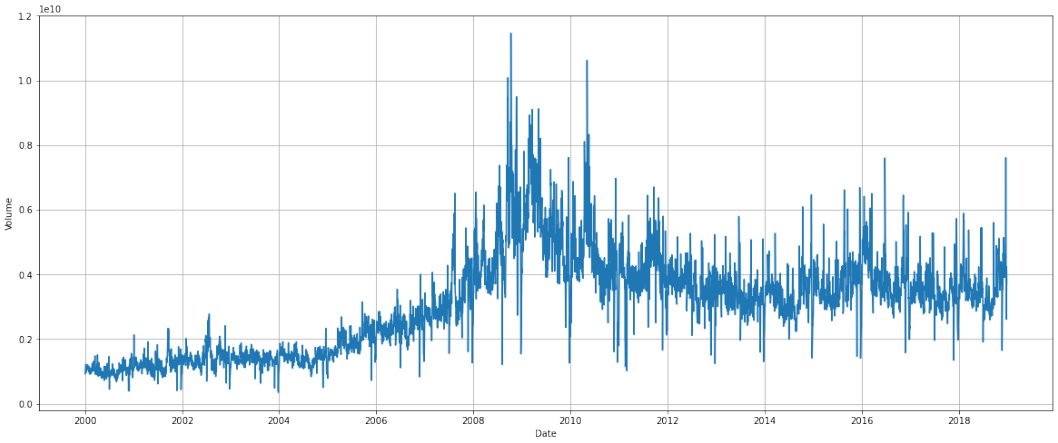


Figure 1: Traded volume in the observed time interval

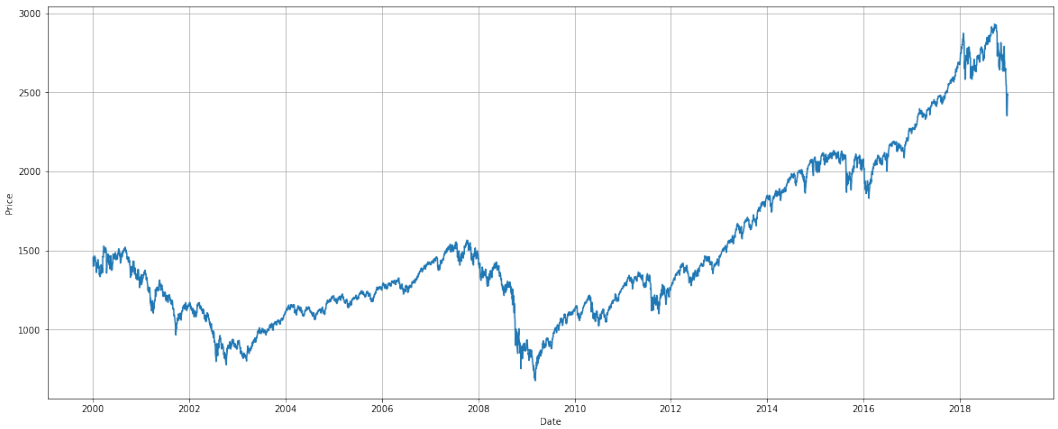


Figure 2: Historical close prices in the observed time interval

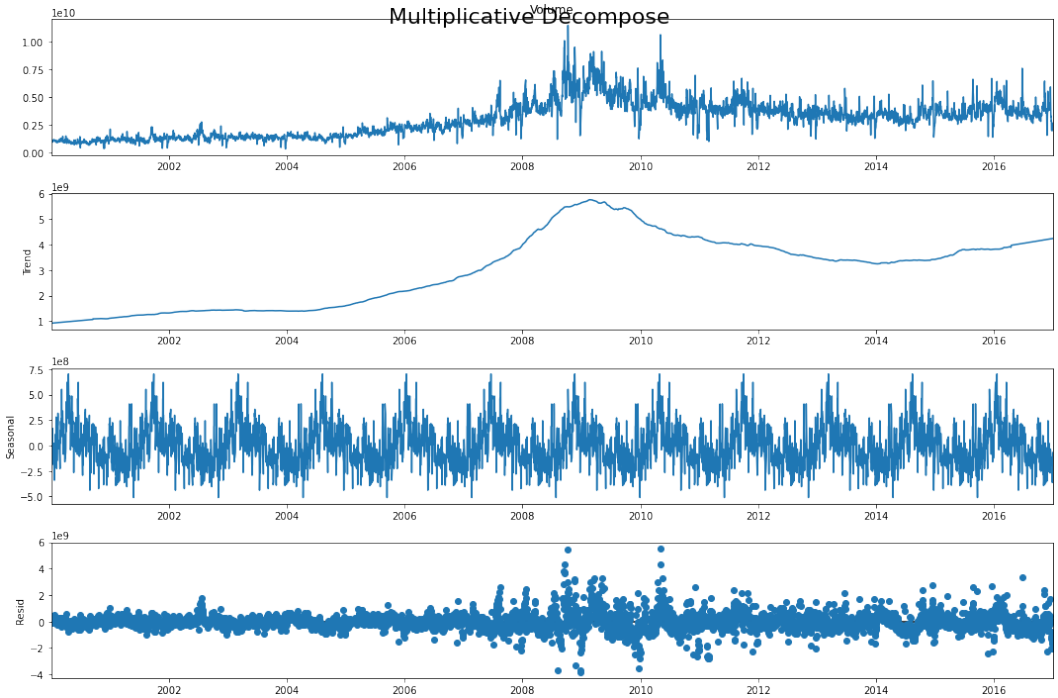


Figure 3: Seasonal decomposition of the volume data with period of 360 days

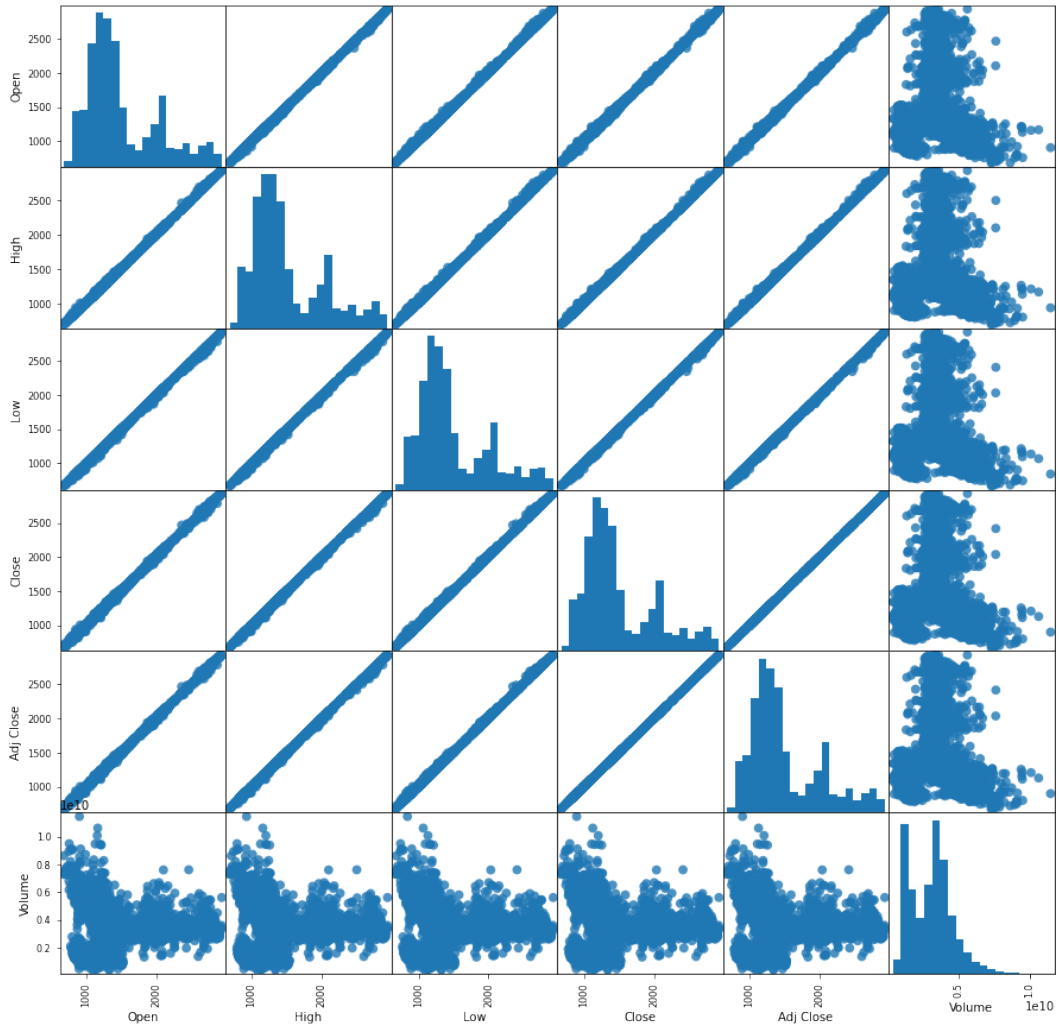


Figure 4: Dependence of individual data values

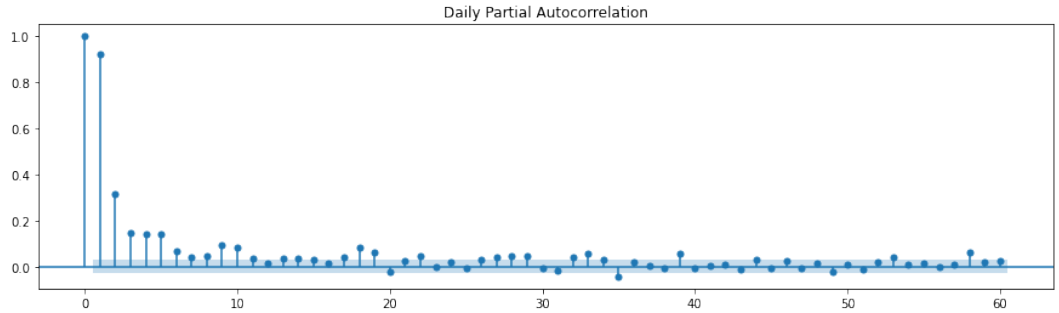


Figure 5: Partial autocorrelation function of daily volume data

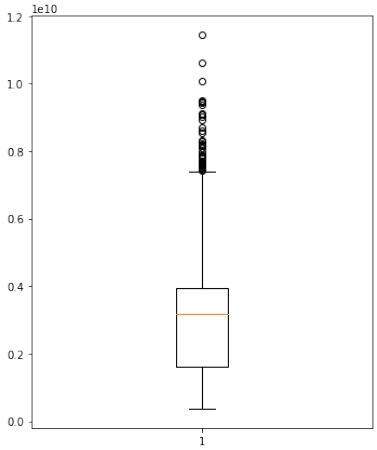


Figure 6: Box plot of volume data showing the distribution of values and outlayers

3 Volume Data Modeling

In this section we describe theoretical properties of the three approaches for time series forecasting – autoregressive models, random forests, and recurrent neural networks.

3.1 SARIMAX model

Autoregressive models are the most frequently used methods for time series analysis and forecast. They are are essentially linear regression models that utilize the lag(s) of the series itself as predictors. There are several modifications using various moving averages, or even exogenous data. The advantage of these models is that they could be applied on relatively short time series and are well explainable, since the models can be described by relatively simple equations.

On the other hand, if the time series data are difficult to be modeled by linear models, autoregressive models may have problem with good fit.

In particular, we decided to use SARIMAX model (Seasonal Autoregressive Integrated Moving Average with eXogenous data) since the volume data have both trend and seasonality and this model allowed for using also other features, such as daily close price, to be used in the model. The formal definition of SARIMAX model is stated in the following equation

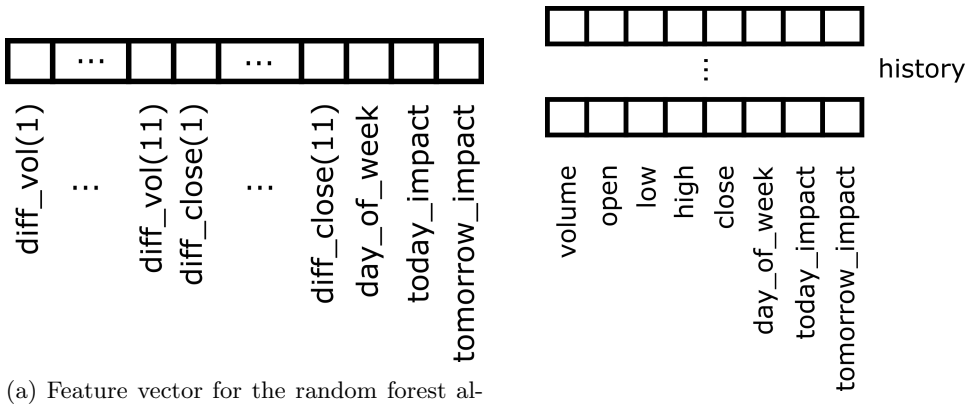
$$\Theta(L)^p\theta(L^s)^P\Delta^d\Delta_s^d y_t = \Phi(L)^q\phi(L^s)^Q\Delta^d\Delta_s^d\epsilon_t + \sum_{i=1}^{n}\beta_i x_t^i \tag{1}$$

which basically combines polynomials of autoregressive model and moving averages over the error terms with regular and seasonal lags, as well as the weighted exogenous variables in time steps 1 to n.

3.2 Random Forest

Random forest is an ensamble learning technique based on the bagging algorithm where multiple decision trees are combined in a way to reduce over-fitting of single decision trees, to reduce their prediction variance and therefore improve their accuracy. Random forests are stable, robust to the outliers and can handle non-linear parameters efficiently. The explainability of of these algorithms is good as well. Impact of individual inputs can be estimated using e.g. the permutation feature importance algorithm introduced by Breiman [2].

On the other hand, in general, tree-based algorithms don't fit very well increasing or decreasing trends which are usually encountered when dealing with time-series analysis, they do not extrapolate. However they can be and have been used for time series prediction e.g. by Dudek [1]. To be used on time series, data need to be first detrended and the lag values then used as features.



(a) Feature vector for the random forest algorithm

(b) Feature tensor for LSTM network

Figure 7: Formats of the input data for random forest and LSTM network

3.3 Recurrent Neural Network

Recently, recurrent neural networks, in particular variations of LSTM networks showed efficient in predicting time series data [3] since they can model complex nonlinear dependencies in the data, and when combined with attention mechanism, they can also provide explainability of their predictions.

For this task, we used network with LSTM layer of variable size, taking variable history length feeding outputs into a fully-connected layer of variable size and finally feeding values to a fully-connected layer with one output neuron. We used Adam optimizer with MSE loss function and trained it for variable number of epochs. These variable hyperparameters are described in section 4.2 which deals with hyperparameter optimization.

4 Model training

This section describes different data preprocessing steps needed for individual models, as well as applied techniques for model quality improvement.

4.1 Data preparation

For model training we used the financial data enhanced with information about important financial reports as described in section 2. These data were divided into training and testing sets according to the assignment and then preprocessed individually for each of the used algorithms.

SARIMAX model We used SARIMAX implementation from Python module *statsmodels* that accepts Pandas *DataFrames* as an input. We first differentiated the volume data to detrend it (even though SARIMX model should be able to deal with the trends) and used differentiated historical close prices as exogeneous variables of the model. We did not take all the history since the model fitting take too long and also since too old data may not be that relevant for the future prediction.

Random Forest As stated in section 3.2, random forests cannot work with trends and does not take time dependance of the samples into account. Thus, the data were first differentiated for detrending and then only specific history lags were used as a features of each training sample. Each sample contained differentiated volumes and close prices from lags 1 - 11, day of the week of the day of the target and importance of financial events of current day and yesterday. See Figure 7a for the idea.

LSTM network Recurrent neural networks should be able to cope with trends, however we differentiated the volume data as well to simplify the data normalization and denormalization. As the features, we used all the data at disposal together with their history (history of OLHC prices and volume, day of week, financial events), this is visualized in Figure 7b.

Further more, the features were re-scaled with a scaler fitted on the training data, to have comparable values as well as smaller overall values for the neural network, which helps it train faster.

4.2 Hyperparameter tuning

As mentioned above, all the used methods have hyperparameters of their models to be tuned. Hyperparameter tuning is a time demanding process a often require training and evaluating a model for all possible combinations of these hyperparameters. For that we need to split the data again, so that the effect of hyperparameters tuning is not evaluated on the same data. Generally a cross-validation is used, however with time series data this is tricky, since we need to take caution, not to shuffle the data blocks and not to train the model on data that may be later on used for evaluation. *sci-kit learn* package provides a TimeSeriesSplit cross-validation method, that is suitable for this type of data, see the vizualization of the generated splits in Figure 8.

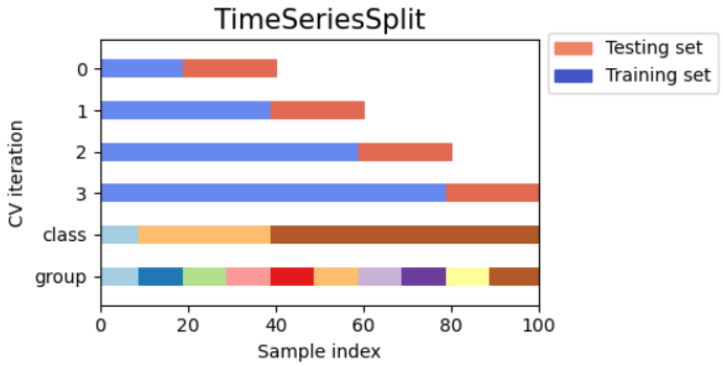


Figure 8: Visualization of TimeSeriesSplit cross-validation

The hyperparameters and their examined values are depicted in Table 1. To examine all the possible hyperparameter combinations, we have used Grid Search algorithm [4]. However, the combination of TimeSeriesSplit cross-validation scheme with the Grid Search algorithm was not suitable for our implementation where we predicted just a single value for a set of historical samples. Thus, we modified the implementation of the Grid Search algorithm such that it evaluates all the samples in validation set one by one.

Table 1: Model hyperparameters to be tuned and inspected combinations. The selected best hyperparameter values are marked red.

Model	Parameters				
SARIMAX	**model order**	**trend**	**seasonal period**		
	[4, 5, 10]	[None, const.]	[5, 10, 20]		
Random Forest	**num trees**	**max depth**	**min sample split**		
	[100, 500, 1000, 2000]	[10, 50, 100]	[2, 3, 5]		
LSTM NN	**LSTM units**	**Dense units**	**History len**	**epochs**	**leaky alpha**
	[10, 20, 40]	[50, 100, 500]	[20, 100, 200]	[20, 30]	[0.1, 0.3]

4.3 Retraining model in evaluation phase

Another improvement that was tried to improve the models predictions was to periodically re-train the models on the test data, i.e. add already evaluated data to the training set and once in a while retrain the models on the up to now collected samples. The idea was that if there are changes in the behavior of the time series in the test data that were not observed in the training set, this re-training would help in discovering them and improving the predictions.

Unfortunately, this re-training mechanism had no significant effect on the quality of the predictions. Moreover, this re-training increases the overall time of model evaluation. It is possible that there is some bug in the implementation but for time reasons, we did dive deeper into this problematics.

5 Evaluation

This section shows the results of the models evaluation and discusses properties of individual models.

5.1 Prediction quality

Table 2 summarizes the evaluation of the used algorithms, it shows their forecast quality in terms of R^2 score, MAE relative to the mean of the testing data, and maximal error relative to the mean of the testing data. Apart from the model quality statistics, we also noted times needed for the model training and evaluation on the testing data. For comparison, we show also the results of the reference model $\hat{E}[v_{d+1}] = v_d$.

Table 2: Evaluation of used algorithms

Model	R^2	MAE	Maximal Error	time [s]
Reference model	0.1234	0.1070	1.4221	0
SARIMAX	0.2813	0.0971	0.8813	2200
Random Forest	0.3942	0.0897	0.7064	230
LSTM NN	0.2883	0.0951	0.8702	790

Table 2 shows the performance of the models using best found set of hyperparameters (shown in Table 1). The best results are achieved by the random forest algorithm, which also needed reasonable amount of time for the training and evaluation.

The training and evaluation times for SARIMAX were largely dependent on the hyperparameters used and ranged from minutes to hours. This made the hyperparameter tuning difficult. However, this problem could be addressed by some sort of early-stopping mechanism for time-consuming or unpromising hyperparameter combinations.

Results of the LSTM network are comparable to those of SARIMAX model, however given more data and maybe more training epochs, the results could get better.

Figure 9 shows the actual predictions of the individual algorithms. The reference model returns basically one day shifted volumes, so even though visually the predictions seem precise, they are not that good. From the rest of the graphs, we can see how good individual algorithms are in predicting extreme values – the most extreme cases are problematic for all of them since such outliers may be caused by some unpredictable events. LSTM network has also problems with some larger volume changes, this may be caused by not being trained for enough epochs, or by the volume scaler being fitted on data that had much lower values.

5.2 Explainability

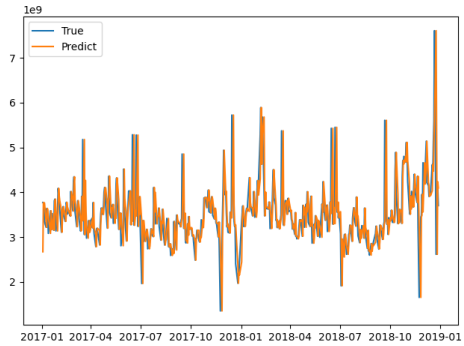
Explainability of the models is another important property of the algorithms, since it helps in debugging problems with the model and overall analysis. SARIMAX model from its nature provides the best explainability of its predictions, we can see the coefficients of all the variables used in the model, see Figure 10.

Random forest also provide good explainability of its predictions. In particular, thanks to the permutation feature importance algorithm, it provides importance measures of its features – see Figure 11.

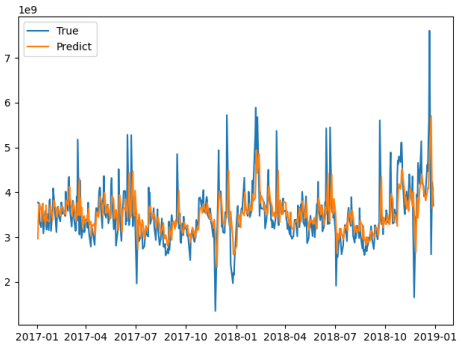
Thus the worst explainability is provided by the LSTM model, which is expected since this is one of the biggest drawbacks of the recent implementations of the neural networks. However, recently there are approaches using attention mechanisms, which can provide at least analysis and visualizations of the intermediate outputs of the neural network layers.

6 Conclusion

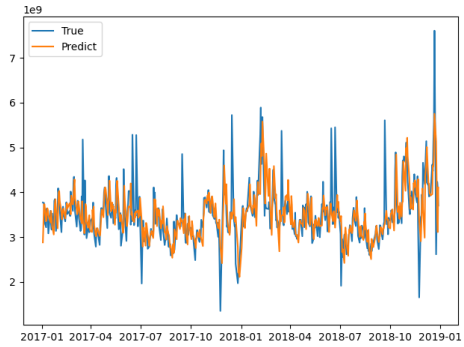
We have implemented and compared three significantly different methods for prediction of the traded volume of S&P 500 index. As the features we used both historical data and occurrence of significant financial events. Model hyperparameters were fine-tuned using modified Grid Search hyperparameter optimization with TimeSeriesSplit cross-validation. We have experimented with re-training of the models during evaluation.



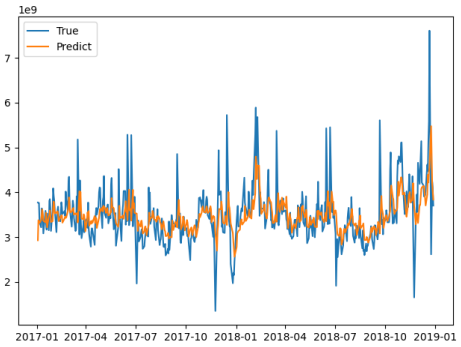
(a) Results of reference model



(b) Results of SARIMAX model



(c) Results of random forest model



(d) Results of LSTM model

Figure 9: Restults of prediction of individual algorithms

	coef	std err	z	P>|z|	[0.025	0.975]
Close	1.061e+06	3.04e-07	3.49e+12	0.000	1.06e+06	1.06e+06
ar.L1	-0.4402	0.302	-1.459	0.145	-1.031	0.151
ar.L2	-0.3946	0.158	-2.502	0.012	-0.704	-0.086
ar.L3	-0.1550	0.224	-0.691	0.490	-0.595	0.285
ar.L4	0.3714	0.115	3.220	0.001	0.145	0.597
ma.L1	-0.0781	0.304	-0.257	0.797	-0.673	0.517
ma.L2	0.0499	0.295	0.169	0.866	-0.528	0.627
ma.L3	-0.2092	0.260	-0.805	0.421	-0.719	0.300
ma.L4	-0.6555	0.242	-2.706	0.007	-1.130	-0.181
ar.S.L5	0.0574	0.759	0.076	0.940	-1.429	1.544
ar.S.L10	-0.1357	0.777	-0.175	0.861	-1.659	1.387
ar.S.L15	-0.3223	0.565	-0.571	0.568	-1.429	0.784
ar.S.L20	0.0965	0.385	0.251	0.802	-0.658	0.851
ma.S.L5	0.0907	0.758	0.120	0.905	-1.395	1.576
ma.S.L10	0.1382	0.791	0.175	0.861	-1.412	1.688
ma.S.L15	0.2971	0.622	0.477	0.633	-0.923	1.517
ma.S.L20	-0.0069	0.425	-0.016	0.987	-0.841	0.827
sigma2	3.097e+17	3.52e-17	8.8e+33	0.000	3.1e+17	3.1e+17

Figure 10: Explainability of SARIMAX model

References

[1] Dudek, G., 2015. Short-term load forecasting using random forests. In Intelligent Systems' 2014 (pp. 821-828). Springer, Cham.

[2] Breiman, L., 2001. Random forests. Machine learning, 45(1), pp.5-32

[3] Kim, S. and Kang, M., 2019. Financial series prediction using Attention LSTM. arXiv preprint arXiv:1902.10877.

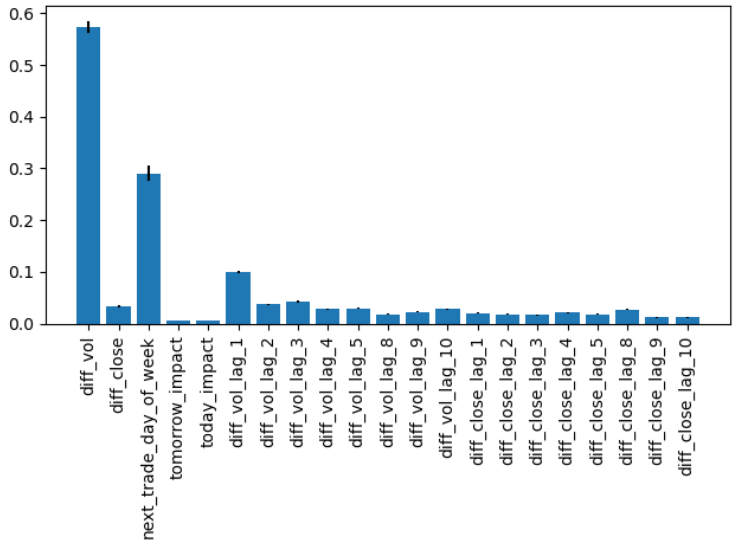


Figure 11: Explainability of random forest model - the black lines corresponds to the standard deviations of the feature importance estimation

[4] Lerman, P.M., 1980. Fitting segmented regression models by grid search. Journal of the Royal Statistical Society: Series C (Applied Statistics), 29(1), pp.77-84.