Forecasting Outstanding Shares by Traditional and ML Methods:

Implementing Competing Supervised Learning Models to Forecast the Logarithmic Fraction of All Outstanding Shares, a Project Overview

Tanner Woods

21 March 2023

ECON 425T

Table of Contents

Initial Analysis of Dataset

Statement of Purpose

Forecast daily for dates 1980-1-1 to 1986-12-31 using training set of dates 1962-12-03 to 1979-12-31. This project will implement the following methods:

1. Baseline (strawman).
2. Autoregressive model.
3. Autoregressive model with multilayer perceptron tuning.
4. Random forest algorithm model.
5. Boosting algorithm (Scikit-learn) model.
6. Boosting algorithm (XGBoost) model.
7. Long short-term memory model with blocking and baseline time series splitting on Bayesian searches.

Base Variable Definitions

Dow Jones return (*DJ_return*, r_t): This is the difference between the log of the Dow Jones Industrial Index on consecutive trading days.

Log trading volume (*log_volume*, v_t): This is the fraction of all outstanding shares that are traded on that day, relative to a 100-day moving average of past turnover, on the log scale.

Log volatility (*log_volatility*, z_t): This is based on the absolute values of daily price movements.

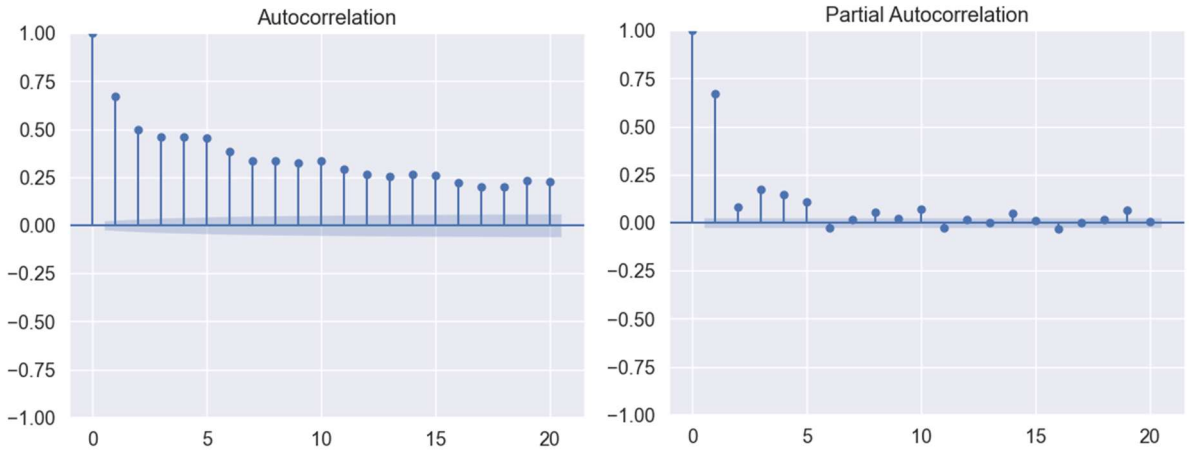
Day of the week (*day_of_week*): This is the day the observation was made on.

Variables for lagged time-periods of up to five are from the first three baseline variables above and are included in all but baseline model. Both the LSTM and baseline model omit the *Day of week* categorical dummies.

Autocorrelations (ACF) and Partial Autocorrelations (PACF)

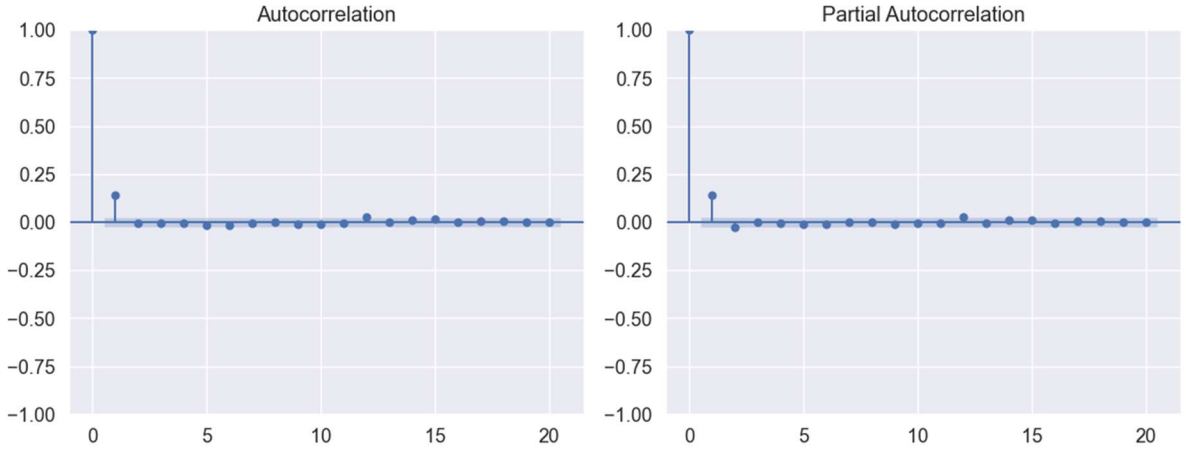
ACF and PACF of Log Trading Volume

Figure 1. ACF and PACF of Log Trading Volume



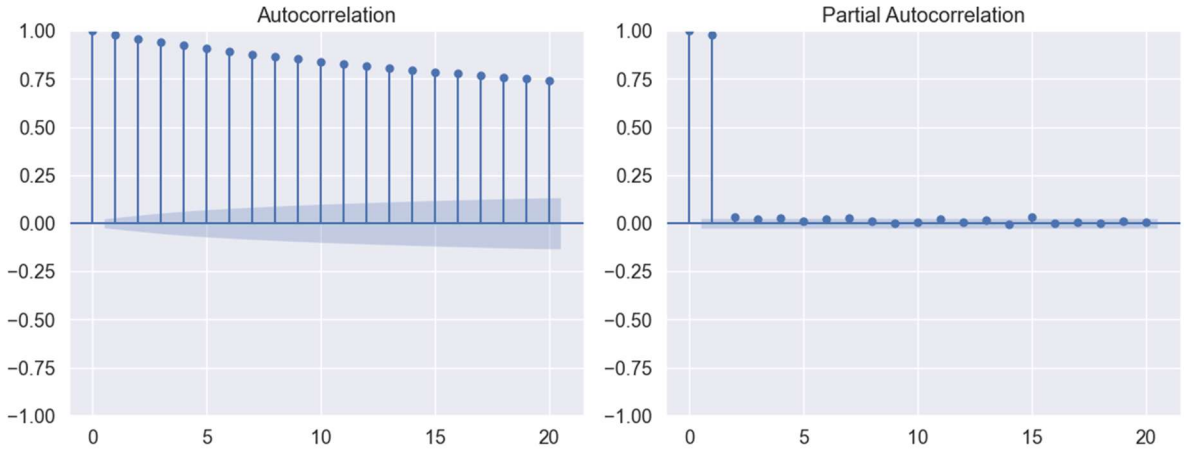
ACF and PACF of Log Dow Jones Returns

Figure 2. ACF and PACF of Log Dow Jones Returns



ACF and PACF of Log Volatility

Figure 3. ACF and PACF of Log Volatility



Splitting Time Series Data for Cross-Validation

After an 80/20 split in the dataset for training and testing, we are faced with an interesting dilemma: how does one properly manipulate time series data for cross-validation? The initial impulse may be to treat the data in the same manner as non-time series data. However, this results in a majority of validation actions adding nothing to — if not degrading — the performance of our model. This method is visualized in **Figure 4** below:

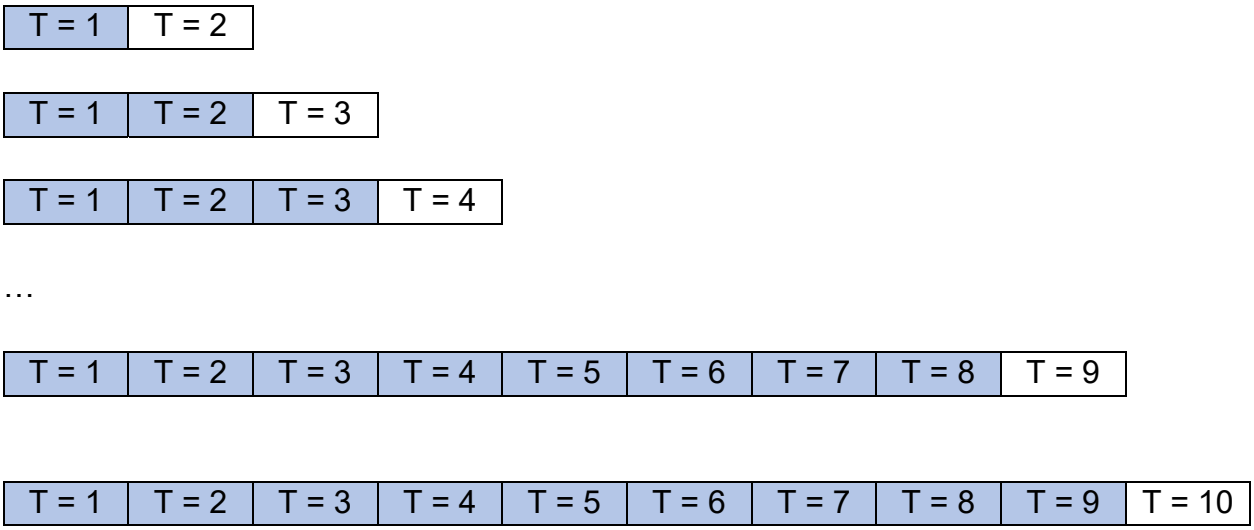
Figure 4. *Improper CV Splitting of Time Series Data*

T = 1	T = 2	T = 3	T = 4	T = 5	T = 6	T = 7	T = 8	T = 9	T = 10

Training-validation comparisons: $\begin{cases} [T=1]\ vs.\ [T=2] \\ \dots \\ [T=1]\ vs.\ [T=10] \\ \dots \\ [T=10]\ vs.\ [T=9] \end{cases}$

Across all validation sets, only the temporally-aware comparisons between training and validation are purposefully informative. In other words, $[T=1]\ vs.\ [T=2]$ is informative for hyperparameter tuning, whereas $[T=5]\ vs.\ [T=2]$ or $[T=3]\ vs.\ [T=10]$ are likely uninformative. Instead of a brute force split over the entire series, a more appropriate method is one that — as mentioned before — is intentionally temporally-aware. Such a method is visualized in **Figure 5** below:

Figure 5. *Temporally-Aware CV Splitting of Time Series Data*[1]



Training-validation comparisons: $\begin{cases} [T = 1]\ vs.\ [T = 2] \\ [T = 1{,}2]\ vs.\ [T = 3] \\ \dots \\ [T = 1{,}2, \dots, 9]\ vs.\ [T = 10] \end{cases}$

In this manner, we see two distinct advantages of our time-series cross-validation here relative to the prior erroneous method. Not only are there fewer validation computations required, but these validations are also likely to be informative for our model construction. This is the method we will implement in the process of building our models; blocking time series split, or temporally-aware traditional k-folds, is also instituted as an experiment for both variants of the LSTM model.

Model Construction

Baseline (Strawman) Model[2]

As no hyperparameters were provided or pipeline construction was performed, this section will be left empty. Performance reports on this model will be included alongside metrics from all following models.

[1] Size of training and validation splits are not intended to be representative.

[2] Predicting the current period's log volume by the last period's log volume, or $\widehat{v_t} = v_{t-1}$.

Autoregressive Model with Elastic Net Regularization

Hyperparameters

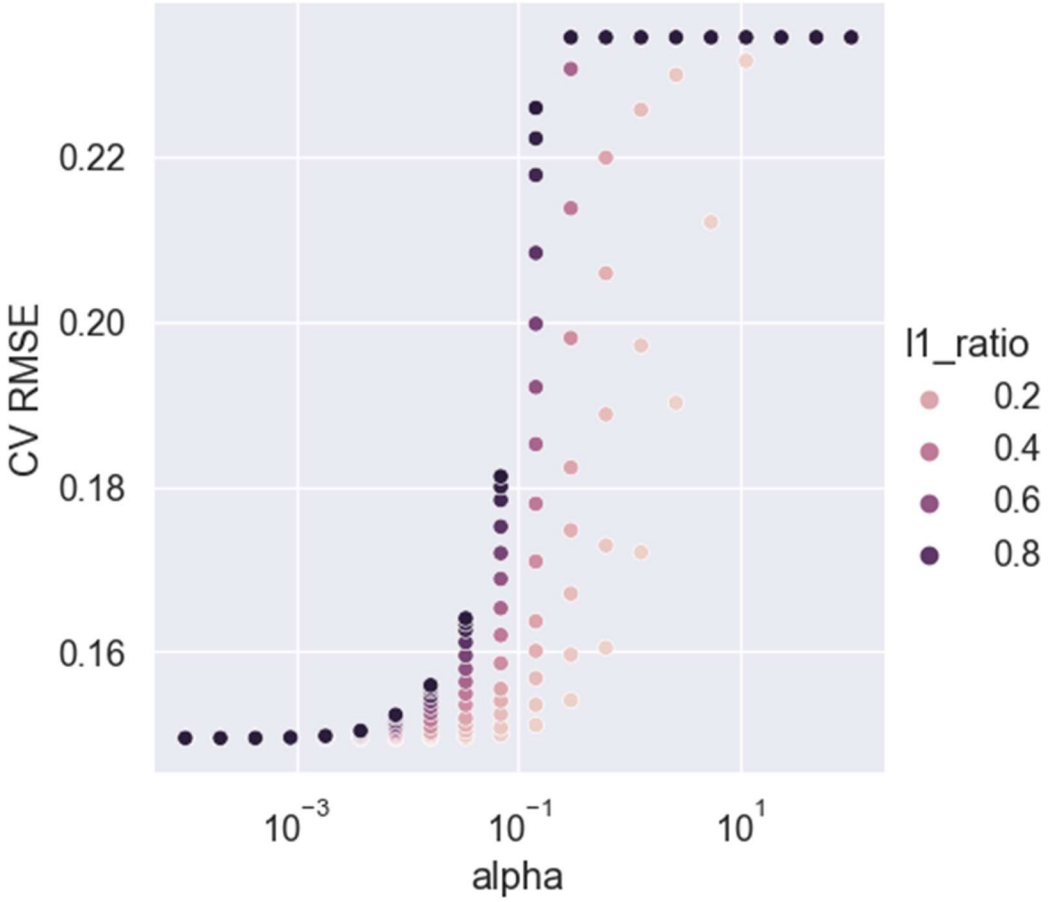
Fixed parameters: $\begin{cases} Lag = 5 \\ CV\ Folds = 10 \end{cases}$

Tuned hyperparameters: $\begin{cases} Alpha = [10^{-4}, \dots, 10^{2}] \\ L1\ Ratio = [0.01, 0.05, \dots, 0.99] \end{cases}$

CV Results

Optimal estimator: $ElasticNet\begin{pmatrix} Alpha \approx 0.01624 \\ L1\ Ratio = 0.01 \end{pmatrix}$

Figure 6. *CV and Search Fit Process of Autoregressive Model with Elastic Net Regularization*



Autoregressive Model with Multilayer Perceptron Tuning

Hyperparameters

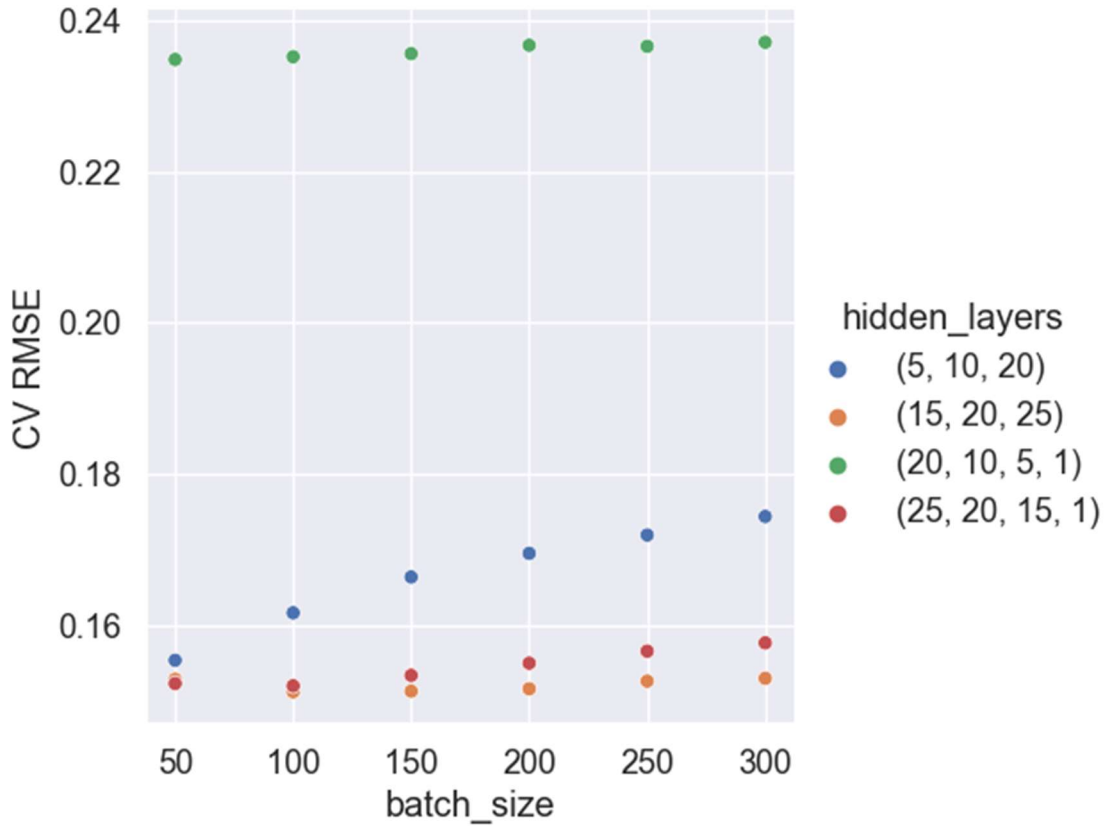
Fixed parameters: $\begin{cases} Lag = 5 \\ CV\ Folds = 10 \end{cases}$

Tuned hyperparameters: $\begin{cases} Hidden\ layer\ size = \begin{bmatrix} 1 & (1,1) & \dots & (1,10) \\ 2 & (2,1) & \dots & (2,10) \\ 3 & (3,1) & \dots & (3,10) \end{bmatrix} \\ Batch\ size = [1,2,\dots,10] \end{cases}$

CV Results

Optimal estimator: $MLPRegressor\begin{pmatrix} Hidden\ layer\ size = (15,20,25) \\ Batch\ size = 100 \end{pmatrix}$

Figure 6. *CV and Search Fit Process of Autoregressive Model with Multilayer Perceptron Tuning*



Random Forest Algorithm Model

Hyperparameters

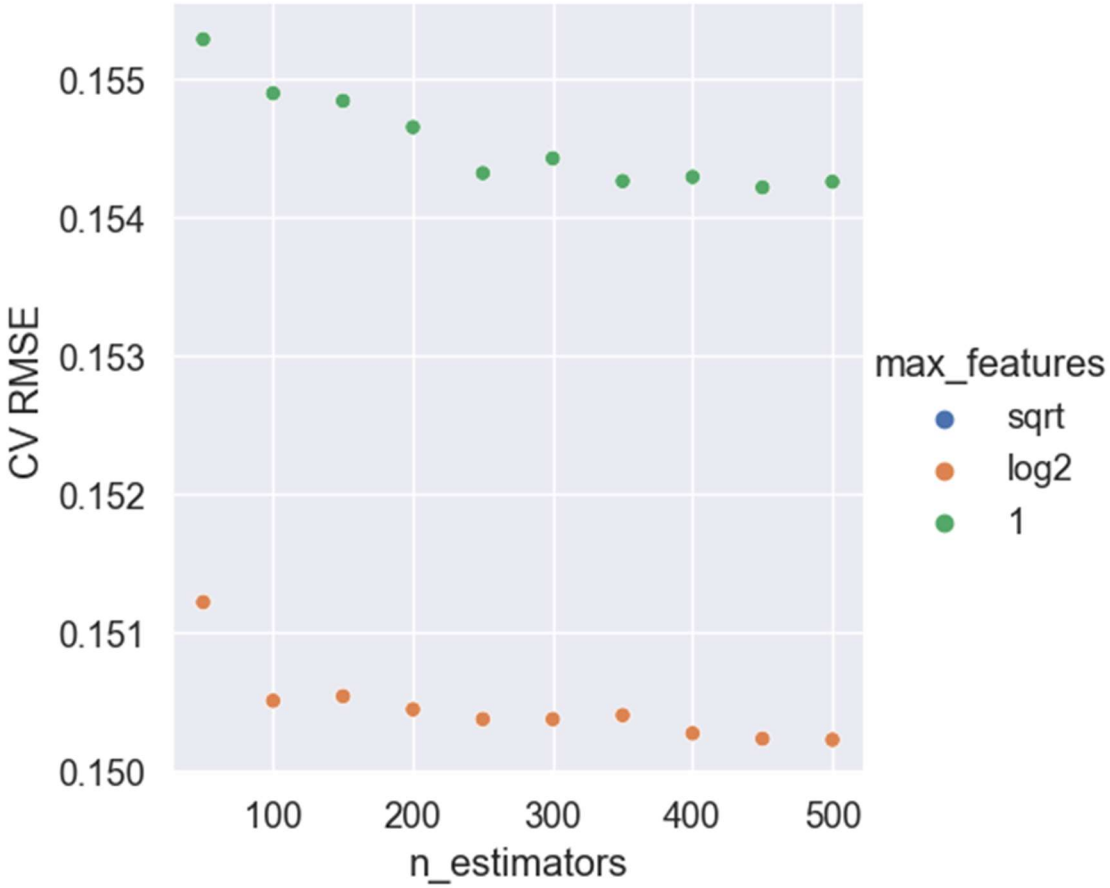
Fixed parameters: $\begin{cases} Lag = 5 \\ CV\ Folds = 10 \end{cases}$

Tuned hyperparameters: $\begin{cases} Number\ of\ estimators = [50, 100, \ldots, 500] \\ Max\ features = [sqrt,\ log_2,\ 1] \end{cases}$

CV Results

Optimal estimator[3]: $RandomForestRegressor\begin{pmatrix} Number\ of\ estimators = 500 \\ Max\ features = sqrt = log_2 \end{pmatrix}$

Figure 8. *CV and Search Fit Process of Random Forest Algorithm Model*



[3] Random forest regression max features tuning produces identical $CV\ RMSE$ values for $sqrt$ and log_2

Boosting (Scikit-learn) Algorithm Model

Hyperparameters

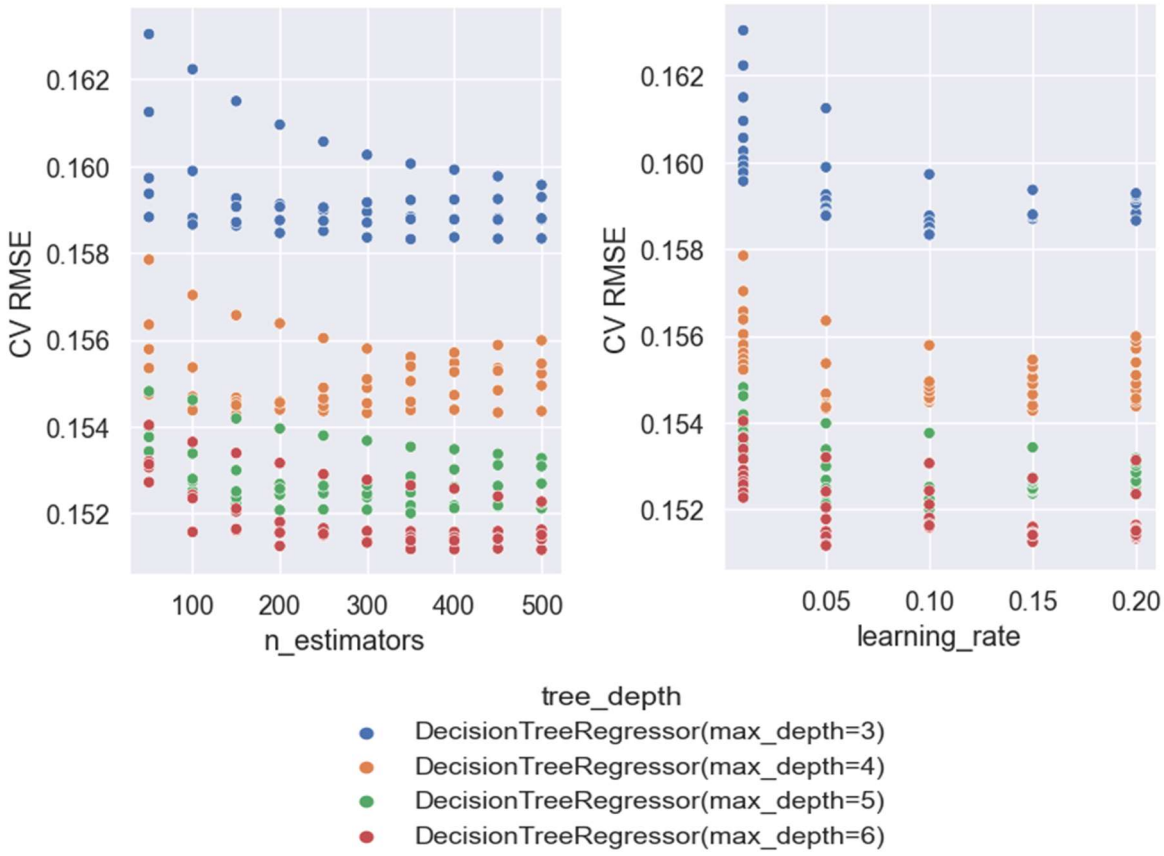
Fixed parameters: $\begin{cases} Lag = 5 \\ CV\ Folds = 10 \end{cases}$

Tuned hyperparameters: $\begin{cases} Number\ of\ estimators = [50, 100, \dots, 500] \\ Learning\ rate = [0.01, 0.05, 0.10, 0.15, 0.20] \\ Max\ depth = [3,4,5,6] \end{cases}$

CV Results

Optimal estimator: $AdaBoostRegressor \begin{pmatrix} Number\ of\ estimators = 500 \\ Learning\ rate = 0.05 \\ Max\ depth = 6 \end{pmatrix}$

Figure 9. *CV and Search Fit Process of Boosting (Scikit-learn) Algorithm Model*



Boosting (XGBoost) Algorithm Model

Hyperparameters

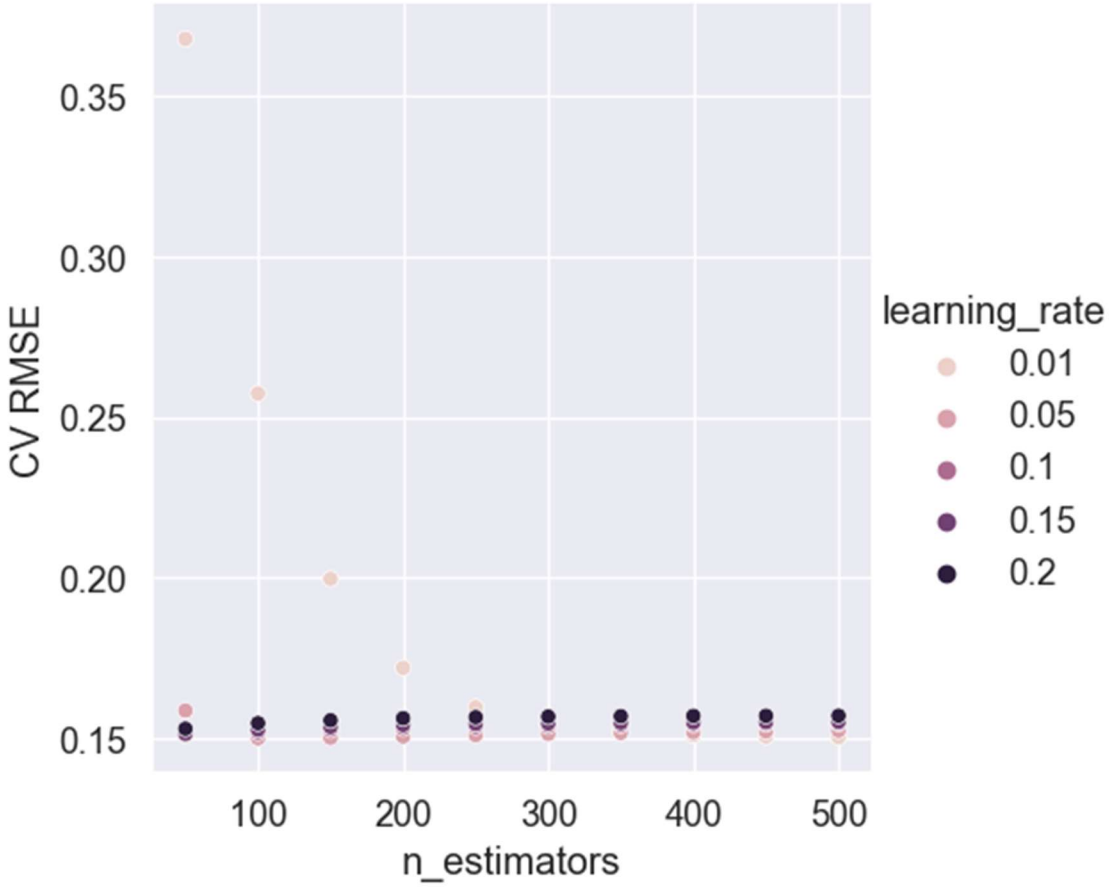
Fixed parameters: $\begin{cases} Lag = 5 \\ CV\ Folds = 10 \end{cases}$

Tuned hyperparameters: $\begin{cases} Number\ of\ estimators = [50, 100, \ldots, 500] \\ Learning\ rate = [0.01, 0.05, 0.10, 0.15, 0.20] \end{cases}$

CV Results

Optimal estimator: $XGBRegressor\begin{pmatrix} Number\ of\ estimators = 500 \\ Learning\ rate = 0.01 \end{pmatrix}$

Figure 10. *CV and Search Fit Process of Boosting (XGBoost) Algorithm*



Long Short-Term Memory Models[4]

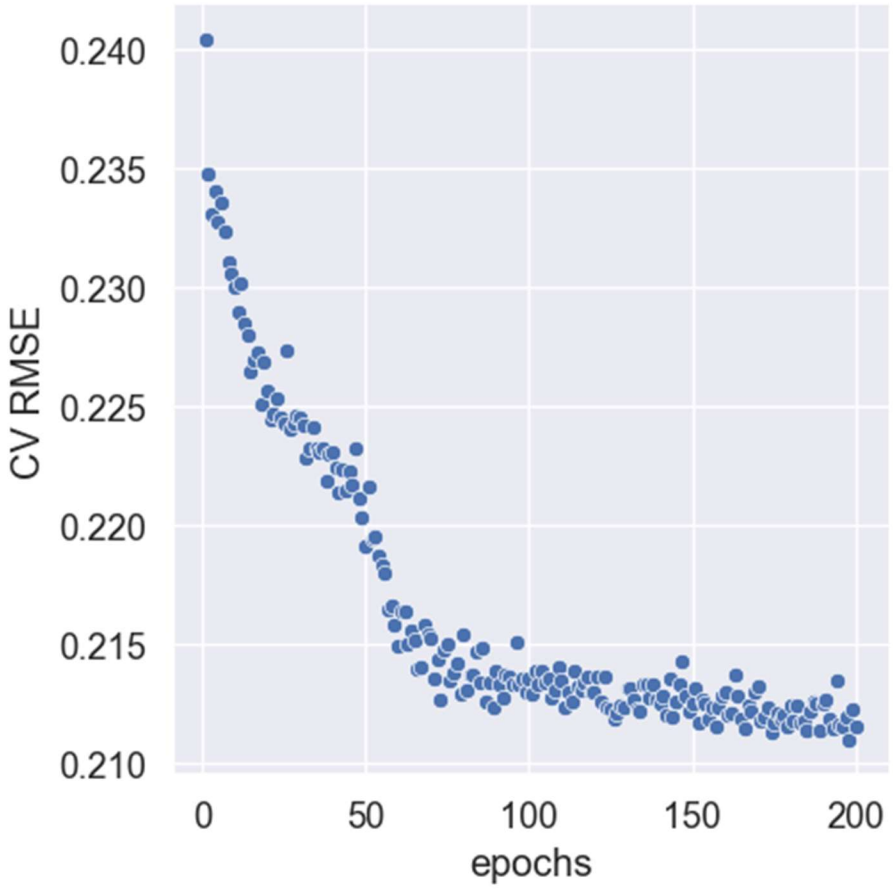
Hyperparameters

Fixed parameters: $\begin{cases} Lag = 5 \\ CV\ Folds = 5 \end{cases}$

Tuned hyperparameters: $\begin{cases} Dense\ units\ on\ inputs = [64, 80, \dots, 256] \\ Dense\ activation = [relu, tanh, softmax, sigmoid, \dots \\ \dots swish, gelu] \\ 50\%\ Dropout = [True, False] \end{cases}$

CV Results: LSTM w/ Time Series Split and Random Search

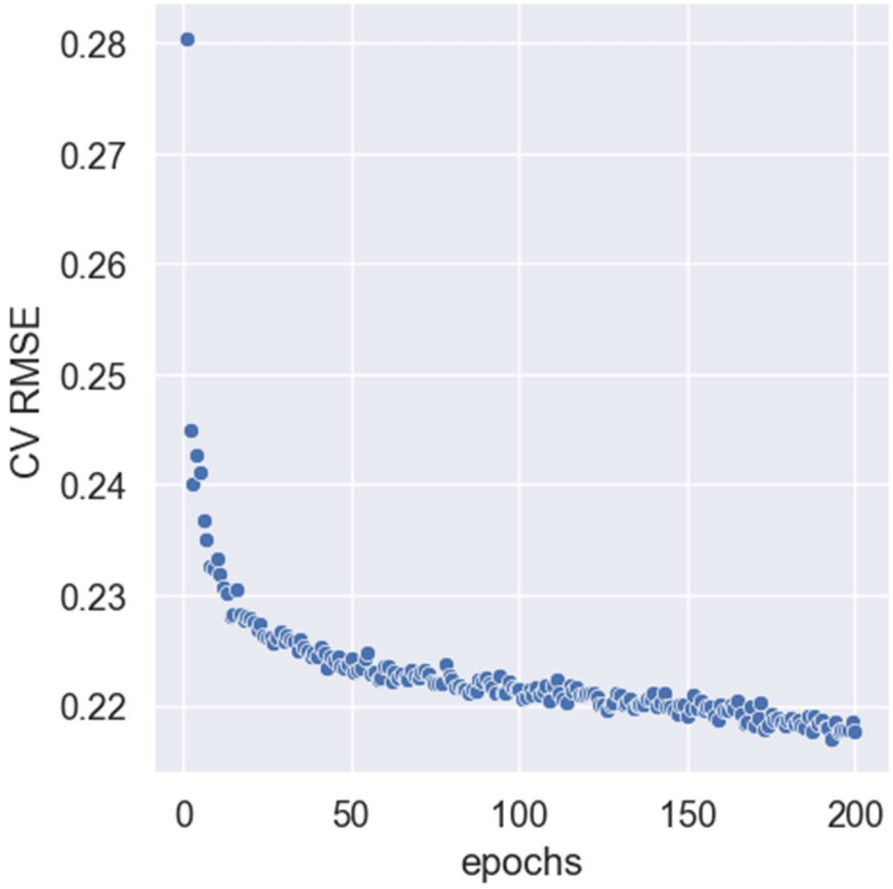
Figure 11. *CV and Search Fit Process of LSTM w/ Time Series Split and Random Search*



[4] Due to issues with json loading modules and Tensorflow, the most optimal hyperparameters and exact definitions of the most optimal estimator for each variant of LSTM model were unable to be acquired. However, the optimal hyperparameters were still able to be applied in a black box fashion.

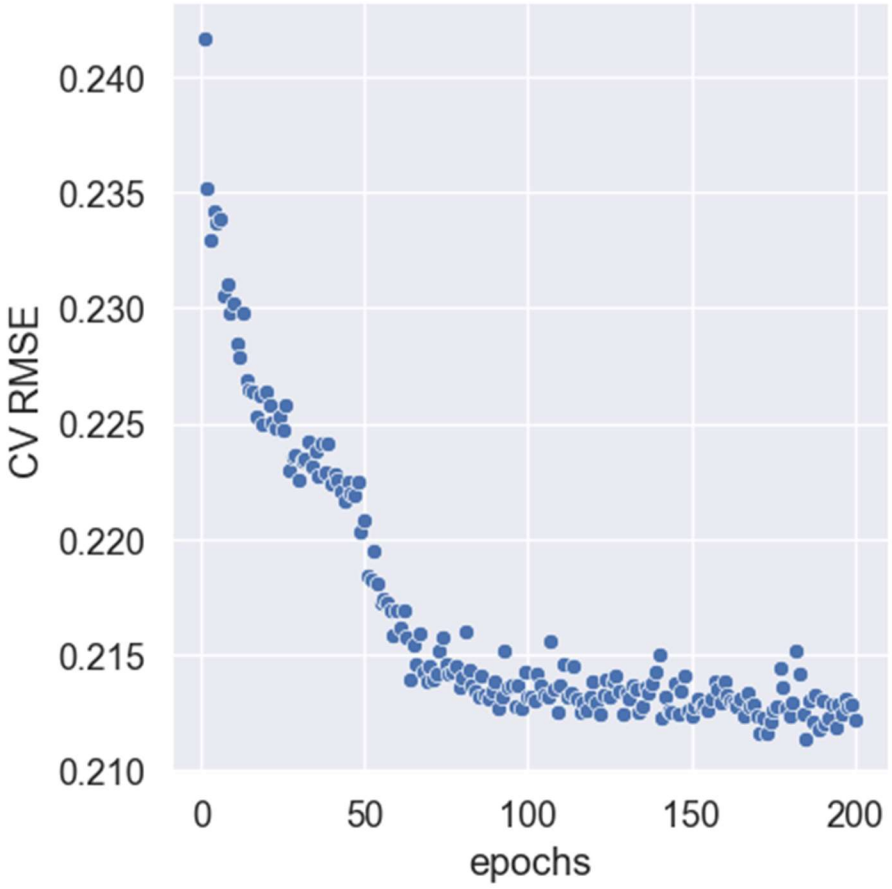
CV Results: LSTM w/ Blocking Time Series Split and Random Search

Figure 12. *CV and Search Fit Process of LSTM w/ Blocking Time Series Split and Random Search*



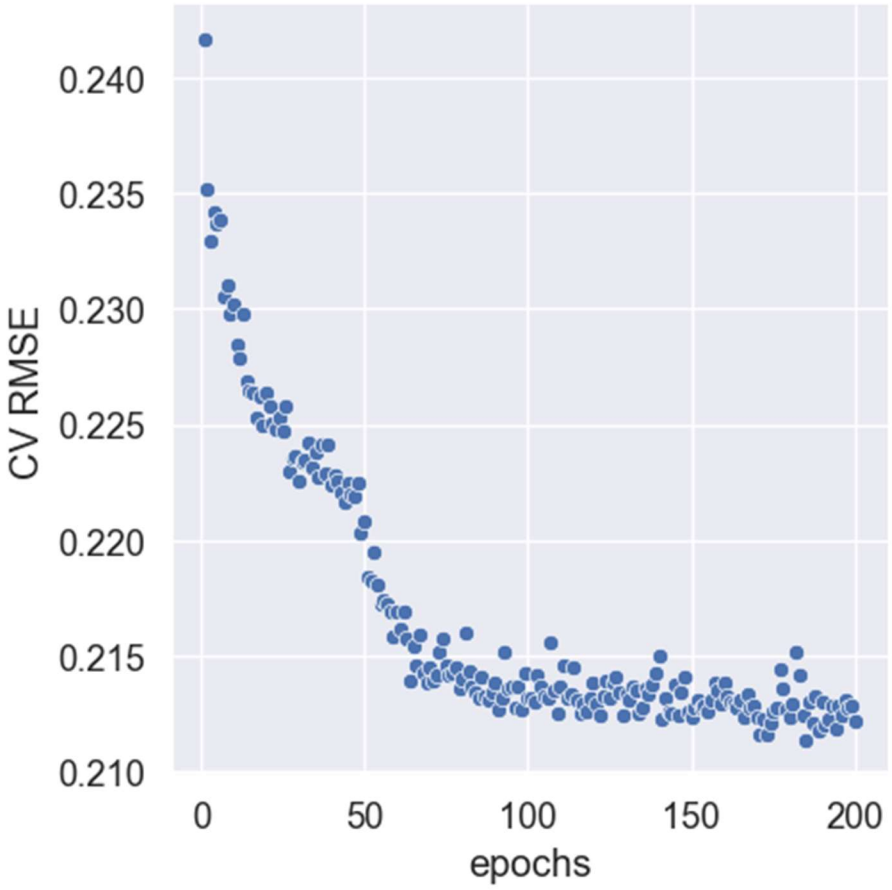
CV Results: LSTM w/ Time Series Split and Bayesian Optimization

Figure 13. *CV and Search Fit Process of LSTM w/ Time Series Split and Bayesian Optimization*



CV Results: LSTM w/ Blocking Time Series Split and Bayesian Optimization

Figure 14. *CV and Search Fit Process of LSTM w/ Blocking Time Series Split and Bayesian Optimization*



Model Performance Comparison

Below in **Table 1**, we list metric comparisons of model performance across all models. For visualizations of non-LSTM model performance in the form of predicted training and test values against their true values, please refer to **Figures 15-19** in the appendix.

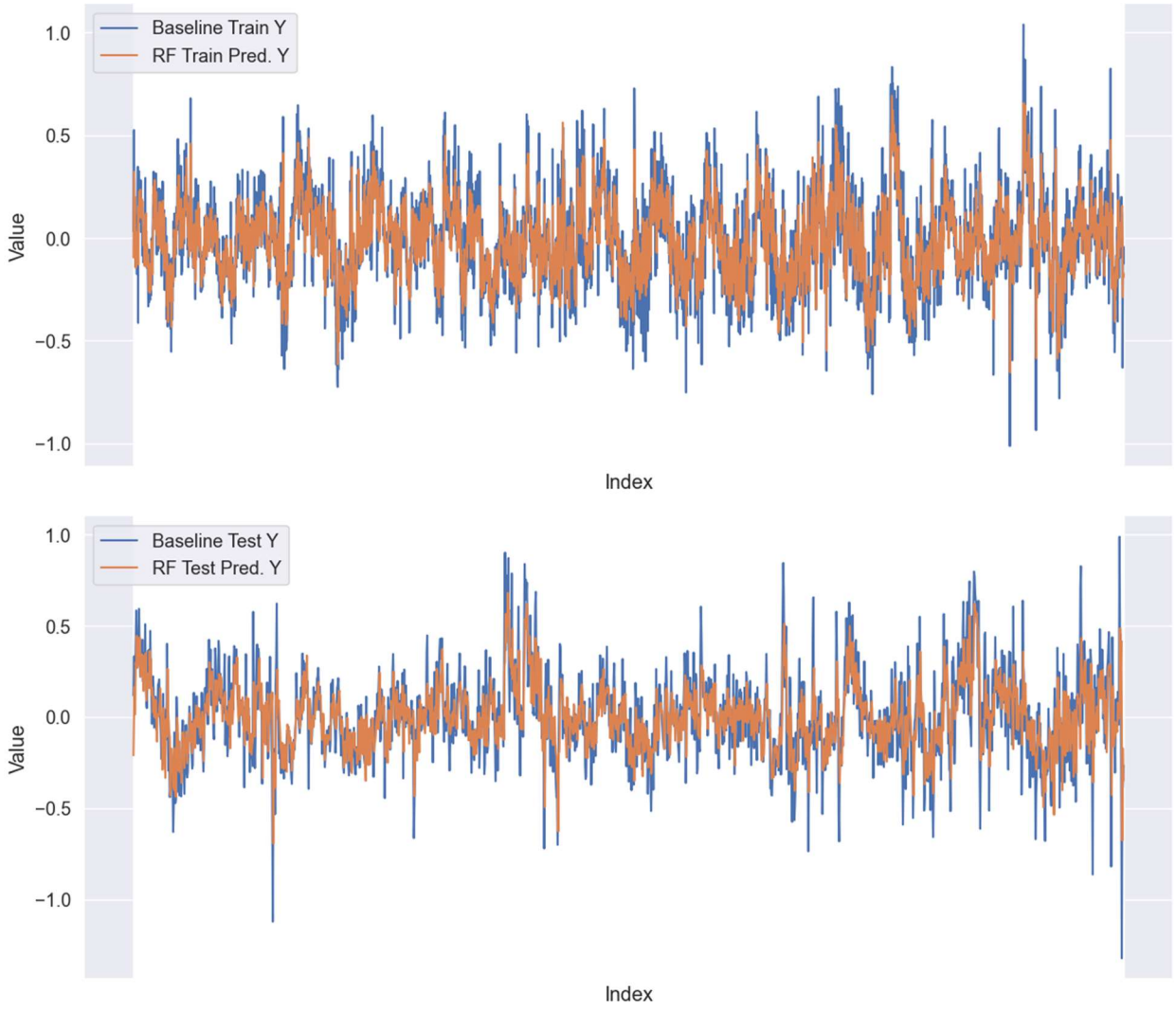
Table Z. *Comparison of Competing Supervised Learning Models*

Model	**Type**	**Cross-Validation**		**Testing**	
		R-Squared	*RMSE*	*R-Squared*	*RMSE*
Base. Strawman	—	0.48730	0.16509	0.33378	0.19577
AR w/ EN	—	0.59871	0.14946	0.44871	0.17809
AR w/ MLP	—	0.67708	0.15124	0.48622	0.17192
RF	—	0.94832	0.15022	0.46226	0.17589
Boost	Scikit-learn	0.72568	0.15116	0.45803	0.17658
	XGBoost	0.80079	0.14981	0.48098	0.17280
LSTM	TSS, Random	0.16545	0.21097	0.08504	0.22943
	BTSS, Random	0.11727	0.21697	0.14899	0.22126
	TSS, Bayesian	0.16230	0.21136	0.14932	0.22122
	BTSS, Bayesian	0.15465	0.21233	0.15142	0.22095

From the results above, we can determine that the most optimal model is the autoregressive model with elastic net regularization. We may also state that there appears to be some potential for overfitting within the random forest algorithm model and — to a lesser extent — both forms of boosting algorithm models. However, if overfitting is not present, then the random forest algorithm model is likely the most optimal for our forecasts. Whether there is overfitting present, and the relatively poor performance of the LSTM variants requires further investigation.

Appendix[5]

Figure 15. *Performance by Predicted vs. True Values for the Autoregressive Model with Elastic Net Regularization*



[5] The legend labels are erroneously named, but the figure titles are accurate; the author only noticed this immediately before submission.

Figure 16. *Performance by Predicted vs. True Values for the Autoregressive Model with Multilayer Perceptron Tuning*

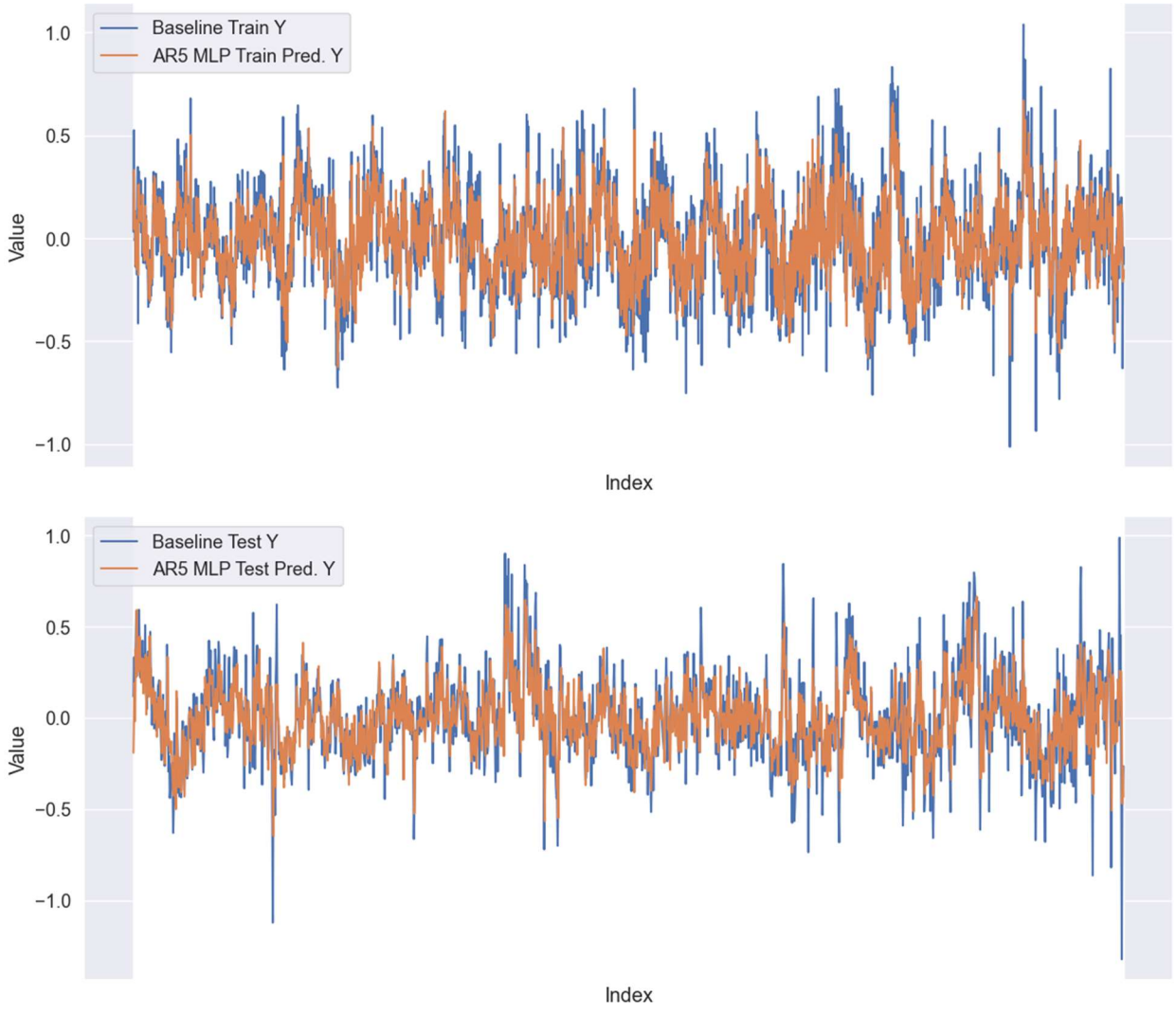


Figure 17. *Performance by Predicted vs. True Values for the Random Forest Algorithm Model*

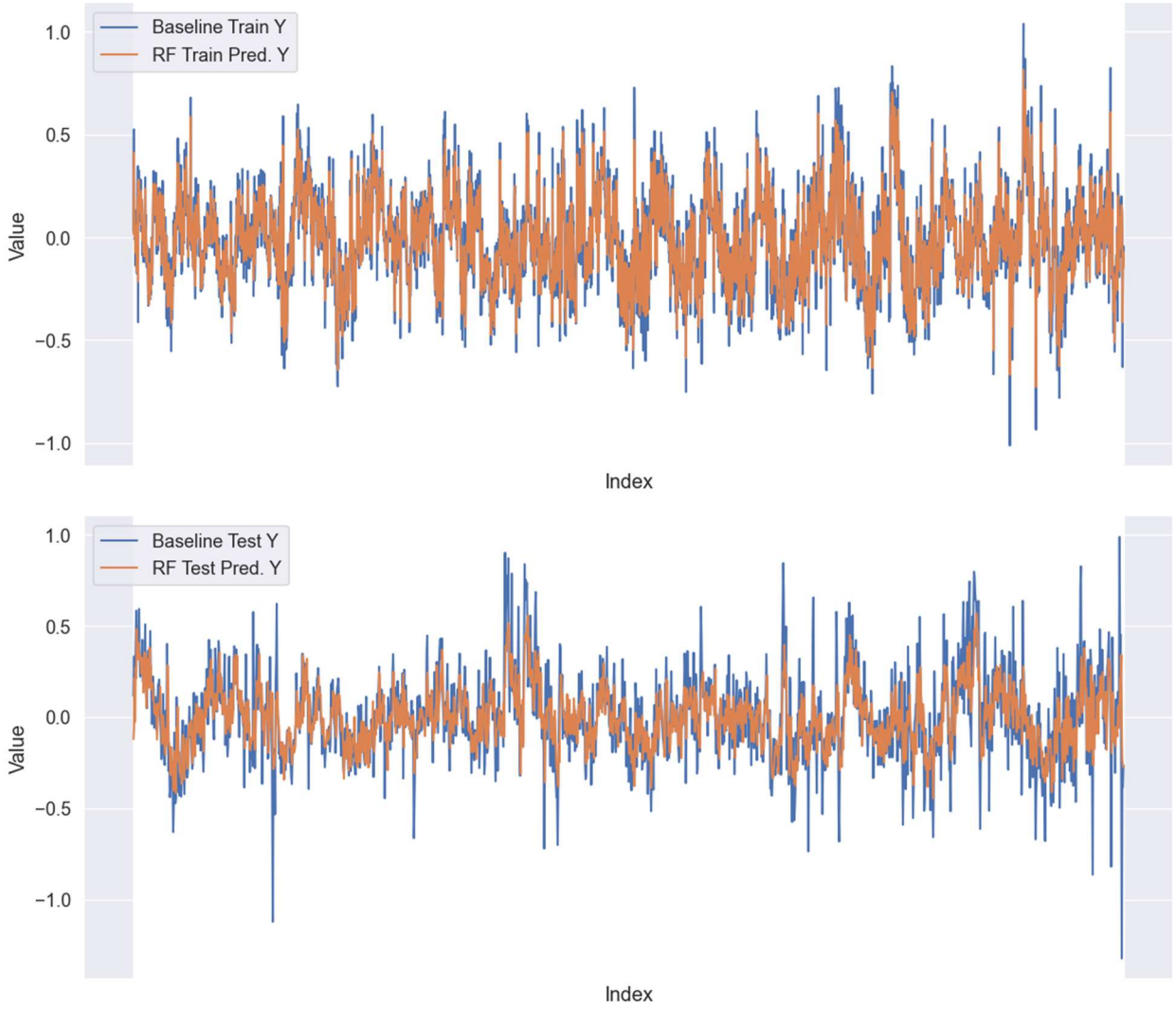


Figure 18. *Performance by Predicted vs. True Values for the Boosting (Scikit-learn) Algorithm Model*

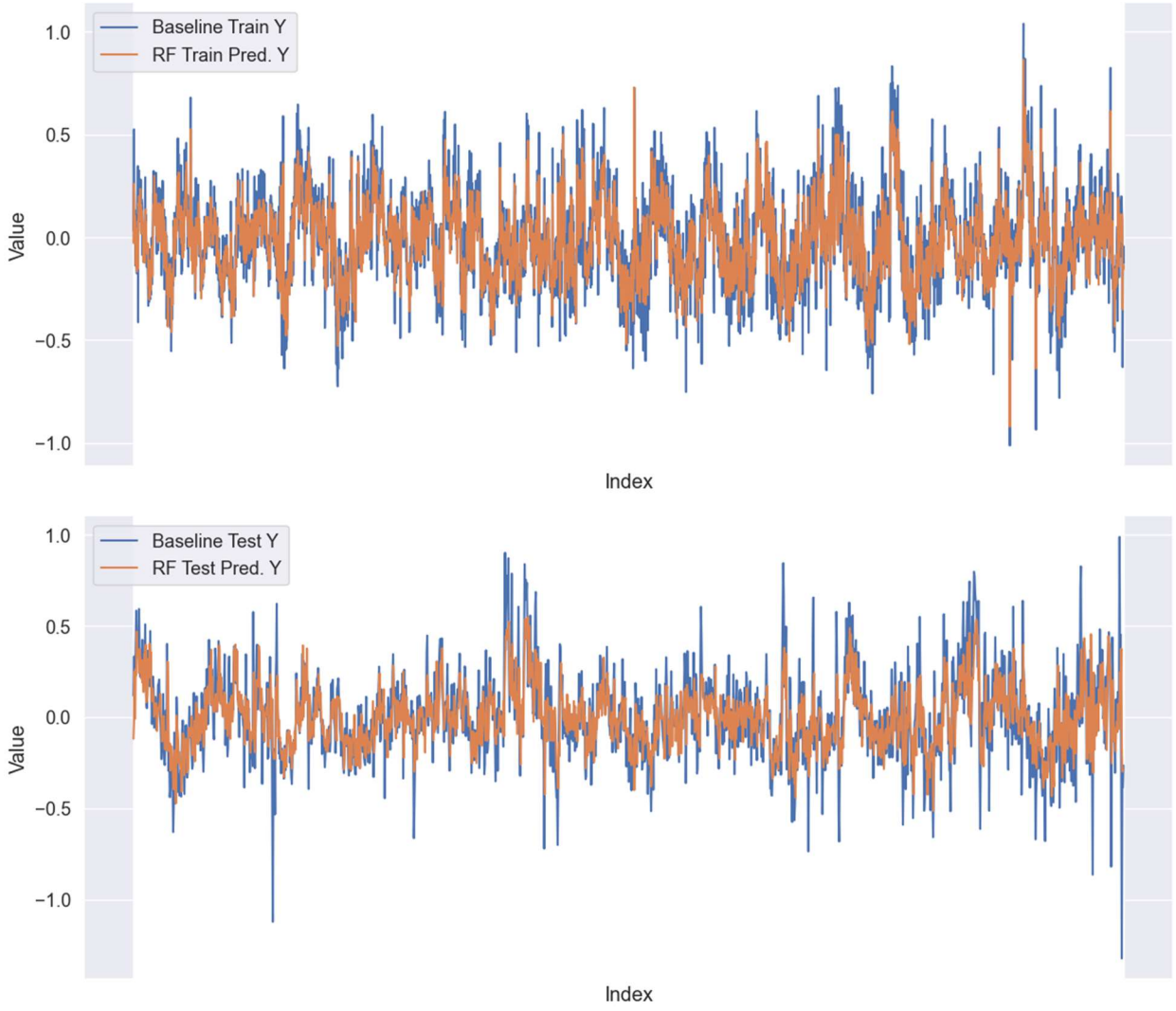


Figure 19. *Performance by Predicted vs. True Values for the Boosting (XGBoost) Algorithm Model*