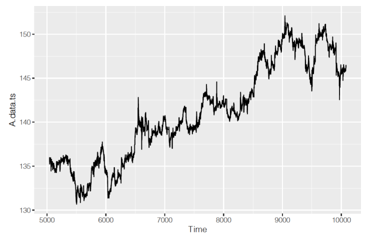
STATS202 Final Project

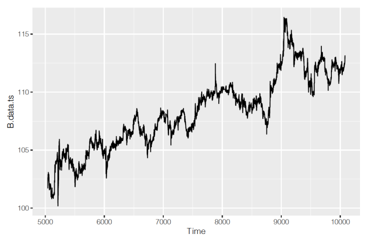
Name(s): Anna Lan Mingruo Shen David Yin
Emails(s): annalan@stanford.edu mingruos@stanford.edu yindavid@stanford.edu

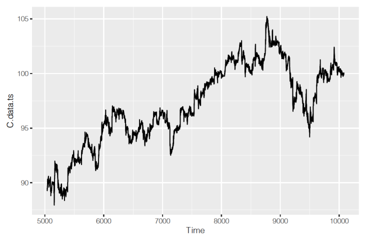
1 Selection

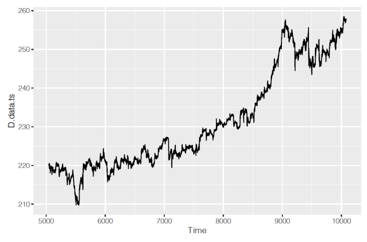
1.1 Exploratory Data Analysis

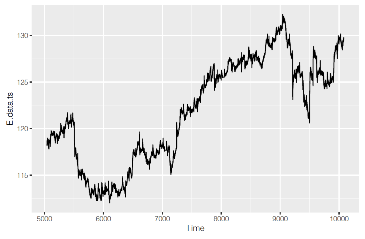
In our initial exploratory data analysis, we wanted to first visually investigate the 10 different stock symbols.

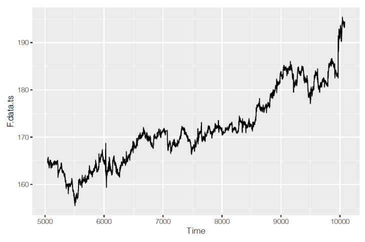


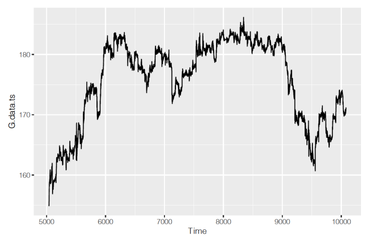


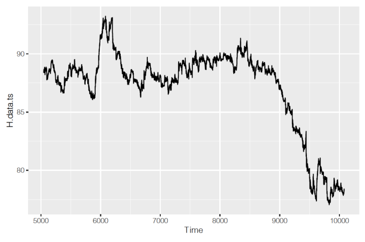


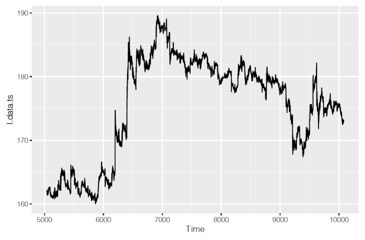


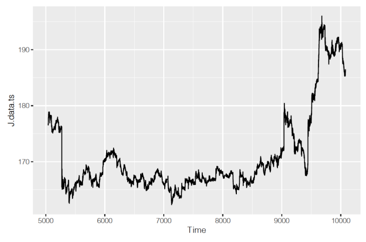








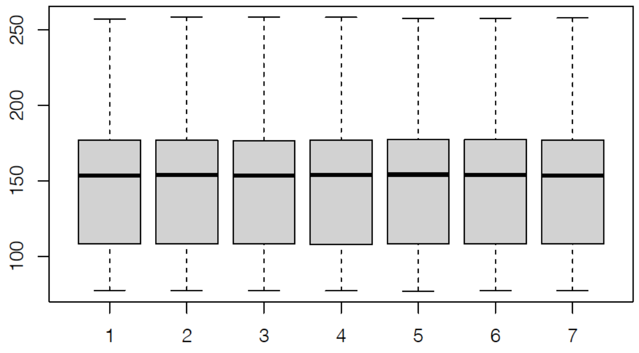




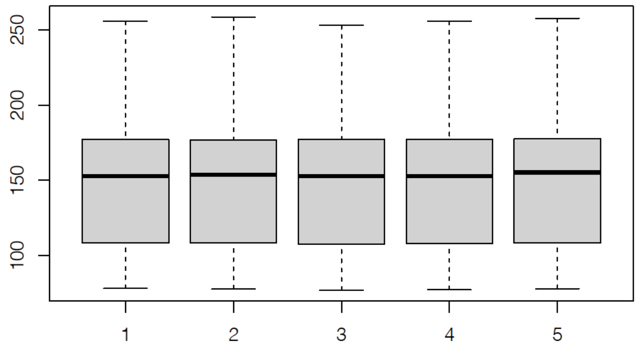
The first thing that we notice is that all 10 stocks seem to behave differently. Through visual inspection, we believe that a single model can not adequately capture the nuances of each stock and perhaps 10 separate models will need to be created.

Next we want to investigate for potential seasonality associated with the hour of the day and the day of the week.

Below is a boxplot with the open prices group by hour of the day (hour = 6, 7, 8, 9, 10, 11, and 12 respectively).



Below is a boxplot with the open prices grouped by the day of the week (day of week = day mod 5 = 0, 1, 2, 3, 4 respectively)



From visual inspection, there does not appear to seasonality either by hour of the day or by day of the week.

Lastly, we noticed that we should have a data entry for every 5 seconds from 6 a.m. through 1 p.m. for all 10 stock symbols. However, we quickly noticed that we were missing some columns. This will be addressed later.

2 Preprocessing

We start by looking at if the data is suitable for further development. Upon inspecting the training data, it appears that across the 10 instruments from A to J, as well as across each of the days from 0 to 86, we see that the number of timestamps each day are inconsistent. Across all of the 10*87 = 870 days, only less than half had 5040 timestamps which is the full count of 5 second intervals from 6:00 to 13:00 PT.

To handle this inconsistency for easier training and data wrangling purposes, we decide to impute the missing timestamps with the immediate next available values in the day, using a backward-fill method to replace the missing values. For example, if the first available open price of the day was at 6:56:00 for 123.45, all opening prices from 6:30:00 to 6:55:55 will be filled with the same value. We recognize the limitation that it might not reflect the actual fair value of the instrument at the time, and that the closing price of each interval may be different than the opening price of the next when they should be approximately the same. During feature selection, we will take these limitations into account, and also recognize that the imputed values are only a small subset of the total data.

3 Transformation, Feature Selection and Generation

The selection of the features depends on the methodology of the data modeling. For a random forest or other tree-based methods, generating multiple features is desirable. For this purpose, we decide to include all features important for the data modeling, including various past changes in price and volatility.

3.1 Past Price Changes

For feature generation, we want to generate an abundant number of features for our tree-based methods. One desirable set of features may be to include price changes over past days. We will leave it to the algorithm to decide which time intervals of change are most important, so we create percentage changes of prices over various past intervals, including:

- past 5s
- past 1m
- past 15m
- past 1h
- past 1-9 days

3.2 Past Volatility

Another quantity of interest would be past volatility of the stocks. It makes sense to think that if a stock fluctuates a lot the day before, it is likely to have similar fluctuations going forward. This was evident in practice over multiple past market events. For example, during the GameStop frenzy, periods of high volatility were followed by further near-term high volatility. By including historical standard deviations in price as features, we believe we might be able to capture some autocorrelation in volatility or price fluctuations in our models. We proceed to generate this by taking the standard deviation of the open price over the same intervals mentioned in the previous section.

3.3 Feature Generation and Data Selection

Due to memory limits on our machines, we cannot possibly generate the above features for each of open, close, high, low, and average prices. Thus, we decided to generate these features all based on the open price only, while keeping the other original close, high, low, and average price columns as features without further feature developments.

Since the first 9 days of derived features will have incomplete cells due to the nature of the feature generation dependent on past prices for up to 9 days, we exclude these days of data for training purposes.

4 Data Mining and Evaluation

We explore data mining on the future prices using 2 ways: a weighted moving average model and random forest.

4.1 Simple Linear Model

4.1.1 Approach

From the data exploration, we concluded that each stock symbol should have its own model. We chose to combine the principles of weighted moving average and simple linear regression in creating our 10 separate models to predict 9 sequential days.

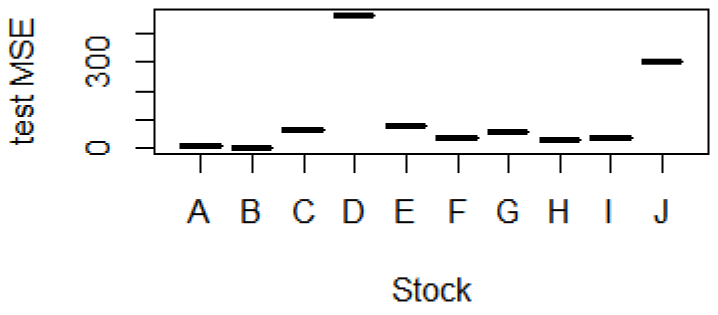
For each stock symbol, we split the data by day 9 to 77 as training set and day 78 to 86 (9 days) as test set. We exclude the data from day 0 to 8, since there are features containing NA in the first 9 days. In order to make predictions depending on the existing predictors, we decide to shift the open price by i days early, so we have 9 independent models with ith model predicts next i days.

We apply weighted moving average to the predictions of the 9 models and result in the ensemble of 9 models as the final stock open price forecasts.

Example: day 87 at time 7:00:00 = 9/(sum(1:9))*x(model 1 using features at day 86 7:00:00) + 8/(sum(1:9))*x(model 2 using features at day 85 7:00:00) + 7/(sum(1:9))*x(model 3 using features at day 84 7:00:00) + ... + 1/(sum(1:9))*x(model 9 using features at day 78 7:00:00)

The reason we choose to use weight moving average is that it is a common approach for decision-making in stock market. Also, it puts more weight on recent data, which is normally more accurate than simple moving average.

4.1.2 Performance on Test Data



From the plot, most of the symbols have relatively small test MSE, except for D and J. This makes sense as there could be relatively different trends in test set from the train set.

4.2 Random Forest

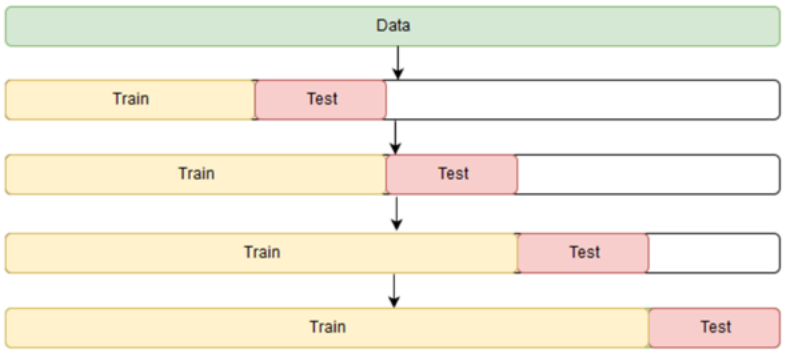
4.2.1 Approach

Similar to the simple linear model, we first try by having an individual model for each of the 10 tickers. Also, we will have 9 independent models, where the ith model forecasts ahead i days. The reasons for this approach are: first, it makes data training cleaner; second, we are able to independently develop a better model for the next-day prediction because we will have more proximal data to use, whereas developing a single 9-day-ahead model using data 8 days ago will likely not perform as well for the 1-day-ahead prediction.

We initially believe that a decision-tree-based model like random forest would outperform the linear regression due to its versatility on all forms of predictors, being able to choose many different splits at each node. Via bagging and pruning, the random forest should be able to select the best decisions over a relatively parsimonious amount of features. We will also search for the optimal number of features to consider for the splits, which we will mention later.

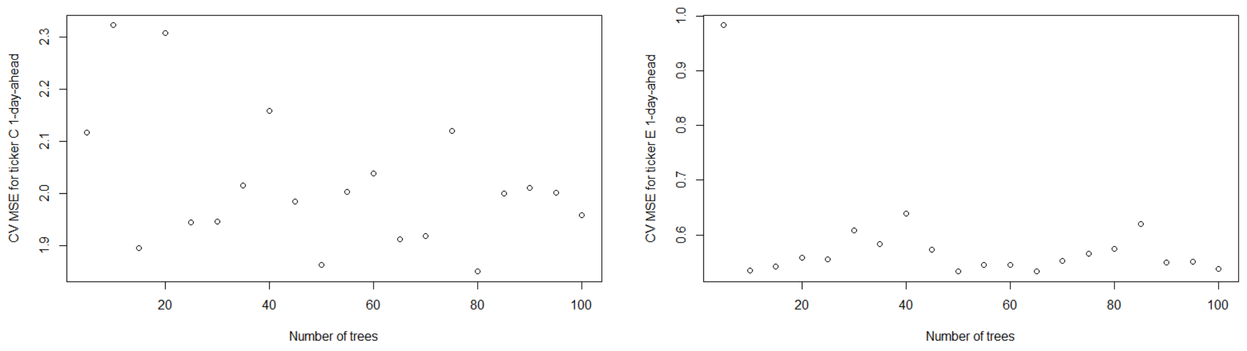
4.2.2 Model Selection and Hyperparameter Search

The cross validation for the hyperparameter search is done by using a sliding window for the holdout set, and using all training data prior to the holdout window at each fold. See the following illustration for the details:



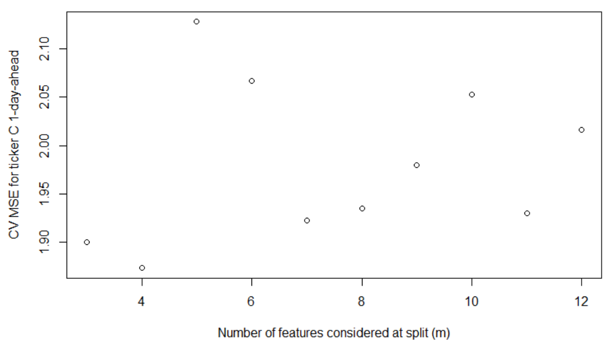
We decide to do a 5-fold cross validation, starting with the first 9 days of data as the first fold's training data, and incrementing by 1 day per sequential fold.

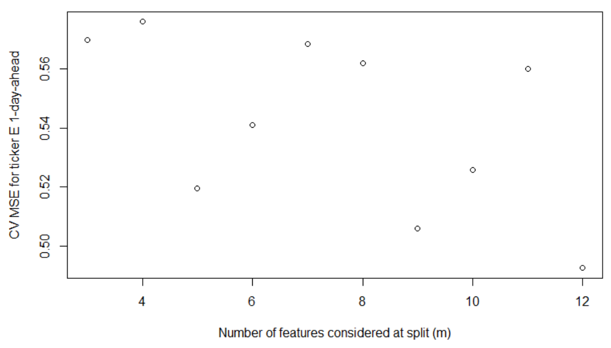
Due to computational run time and memory limits and, we will limit our search to two randomly chosen tickers (C and E) to do our searches for the best hyperparameters. First, due to memory constraints, we want to see if we can use a relatively small number of trees for our random forest model. We use the ranger library in R, taking a subtask of predicting for prices one day ahead, to perform cross validations within a grid search over trees from 5 to 100 in increments of 5 trees.



We see that the results generally indicate that with more trees, the CV MSE decreases as we originally expected. We see that this plateaus really fast with E, with anything more than 10 trees achieving a similar MSE, whereas for C, this is not too clear but there seems to be an overall downward trend with higher number of trees. This being said, for consistency we decide to choose 50 trees for all of our random forest models based on the plots to perform our model selection, as it is unlikely that more trees would benefit us greatly from these two grid search results.

We would also like to see if there is an optimal number of features considered at each split that could give us a better result. We consider number of features from 3 to 12 (out of 28), running a cross validation on the same task of predicting one day ahead, for the two tickers C and E.





Unfortunately we don't see a noticeable effect of number of features used per split vs the CV MSE. From here onwards, we will leave the number of features used as the default number, which is the square root of the number of features.

4.2.3 Model Evaluation via Cross Validation

For the model evaluation, we will use our random forest approach with the chosen hyperparameters (50 trees, default number of features tried per split) and compare it to performance of a linear regression model as well as a baseline model. The baseline model is chosen to be simply the last open price of the stock's training data.

For simplicity, we will do the model evaluation just for ticker A, for the 9 models where the ith model does an i-day-ahead prediction task. This will be done via a 10-fold cross validation. We decide to arbitrarily choose the first 30 days as the starting point of the first fold's training data, with the next day as testing window at each fold.

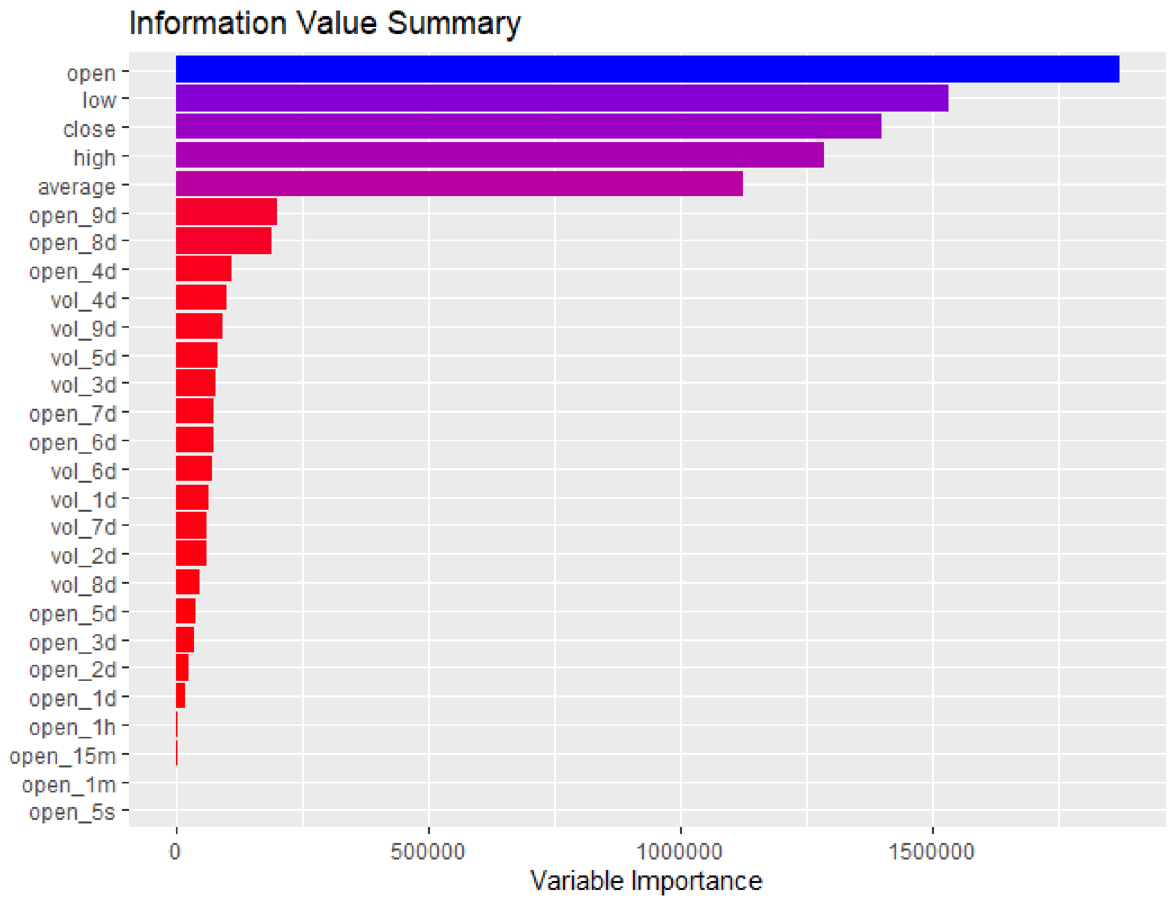
The full cross validation results on ticker A of all 9 models (one-day-ahead to nine-days-ahead) are shown in Table 1 in the Appendix.

We see that for the one-day-ahead model, we obtain a CV MSE of 3.27 for the random forest approach, 24.19 for the linear regression, and 1.96 for the baseline. Although both approaches unfortunately failed to beat the CV MSE of the baseline, the linear regression performed very poorly in comparison to the random forest. Also, the good performance of the baseline model (just taking the latest price in training set) makes sense as this is only a one-day-ahead prediction, and the price might not move much in a day.

For the multiple-days-ahead models, especially with days 6-9, we see that the random forest model was able to outperform the baseline. This might be due to prices moving farther away when time passes, and that an independent random forest model predicting a specified number of days ahead might be somewhat robust to that.

4.2.4 Feature Importance

We can look at variable importance after training the random forest model on our full training data (day 9 to 77) for stock A. The plot below shows the variable importance.



The original (current) open price seems to be the most important factor, closely followed by low, close, high, and average prices. Unfortunately, the features we generated and believed to possibly have a high impact on the model, such as historical returns and historical volatilities, did not nearly have high enough importance as the original prices themselves. This might also explain why the baseline model of choosing the last open price managed to perform our experimented models.

4.2.5 Performance on Test Data

We proceed to build the following 9 models, where the ith model does the i-days-forward prediction for each ticker, resulting in a total of 90 models. See results in Table 2 in the Appendix. We do the same for the benchmark model of taking the last available open price. See results in Table 3. The test MSEs range from stock to stock, mostly could be due to extreme price movements of certain stocks during the test period (see stock J's plot earlier for example). Overall, we see the random forest model outperforming the baseline in approximately half of the stocks, and underperforming for the other half. Also, there is no noticeable over- or underperformance associated with the number of days ahead prediction.

5 Reflection

From the two approaches we explored, a weighted moving average using simple linear models and a random forest, we had chosen to use the predictions of 9 sequential stock days from our linear model for submission, which links to the submission from team name "The Miner League".

When evaluating the performance of weighted moving average, we observed a relatively low test MSE. Our decision to select the weighted moving average model over the random forest can be summarized by the tradeoff between the use of trees compared to other parametric models. The appeal of using a random forest include its non-parametric ability to model complex data as well as its ability to be easily explained. However, tree based methods generally produce less accurate predictions as compared to some other regression and classification approaches, as mentioned in the ISL textbook.

After we received the results, we re-evaluated our models and predictions. Most importantly, by taking a closer look at the detailed predictions of the submission, we realized that how we sorted the final submission data is wrong and thus the wrong order led to a large error. Furthermore, we find that the unchosen random forest model produces more accurate predictions compared to the linear model predictions we submitted. Some possible reasons are: first, we didn't include open price itself as a predictor in the linear model, which is supposed to be the most important feature; second, the CV MSE from the random forest model is relatively smaller than that from the linear model. If we considered the above factors more important, We should have had more accurate predictions and chosen the random forest predictions for submission.

6 Appendix

Link to code: https://github.com/dydx314/stats202

Table 1: Random forest performance comparison via CV for stock A

Days-ahead	Random Forest	Linear Regression	Baseline
1	3.2718832	24.19474	1.957672
2	3.3520145	23.11883	3.312812
3	3.6350160	21.58401	4.440492
4	4.1528311	20.70494	4.832609
5	3.1602076	19.56742	4.624025
6	1.8430666	16.80402	4.224092
7	1.0337454	15.29169	3.915538
8	1.0996609	13.43730	3.793485
9	0.9689866	12.04944	3.272198

Table 2: Random forest test set performance for all stocks

Days-ahead	A	B	C	D	E
1	2.787701	2.429432	4.835544	12.838701	4.903170
2	5.389552	3.664984	9.369396	10.583000	9.484623
3	7.022167	4.247388	12.008682	8.605175	9.802538
4	7.852410	4.393205	14.506374	9.819748	9.365568
5	5.022377	3.227511	13.540116	7.560772	10.559608
6	4.684759	1.906849	15.179554	5.999064	6.986389
7	3.838531	1.929061	11.466223	12.866409	3.112404
8	4.480238	3.043228	6.075466	10.808740	5.583621
9	5.077855	3.991943	4.421077	27.072727	14.296734

Days-ahead	F	G	H	I	J
1	8.663518	80.41109	40.01852	32.347103	31.69005
2	5.075077	99.92025	61.77086	59.453649	84.46581
3	2.946538	130.02380	82.60969	61.546362	148.51898
4	2.823493	122.98178	76.19911	31.497456	203.08836
5	5.577207	110.38778	82.08147	25.458638	245.27323
6	6.453040	159.05063	77.65365	16.737733	306.73105
7	6.348772	124.89548	72.73556	9.689356	357.72199
8	24.697876	92.49206	65.68617	3.478483	372.29638
9	53.742187	68.25682	60.74643	6.242889	363.96770

Table 3: Baseline test set performance for all stocks

Days-ahead	A	B	C	D	E
1	5.857138	2.630450	14.683164	23.79377	30.16167
2	5.852458	2.512868	14.963018	28.91193	32.54857
3	5.816932	2.362151	14.737423	30.94504	34.52371
4	5.773429	2.642764	13.211297	29.85197	35.29997
5	5.932088	3.287554	12.735975	28.15289	36.71554
6	5.988634	3.385327	11.663481	27.57051	37.30306
7	6.292972	3.178365	9.462506	27.02049	31.67943
8	6.074584	2.772741	5.995889	20.56802	22.82470
9	6.411039	1.891630	2.940778	15.07014	19.94964

Days-ahead	F	G	H	I	J
1	3.705536	155.4527	25.57048	36.890082	31.51227
2	3.870606	165.5157	30.09020	37.557539	71.35720
3	3.702021	173.9741	34.66127	36.159894	119.91799
4	3.613937	180.8828	39.71980	29.245016	154.95226
5	4.045981	191.7384	47.85910	24.649870	185.54851
6	4.774748	191.8025	54.38502	18.379494	219.24526
7	4.876902	178.1730	59.01120	10.369413	257.09626
8	11.492431	155.2289	61.77256	6.090735	291.93209
9	23.742110	130.3777	62.13445	7.007034	307.73924