# 3244-2010-0005 - Project Final Report

Authors: Darren Ong, Lim Jun Kuang, Lionel, Neo Wei Hong, Nicholas Alexander, Ong Jing Long, Teo Jun Xiong

Emails: [e0310137@u.nus.edu](mailto:e0310137@u.nus.edu), [e0310450@u.nus.edu](mailto:e0310450@u.nus.edu), [e0310224@u.nus.edu](mailto:e0310224@u.nus.edu), [e0201064@u.nus.edu](mailto:e0201064@u.nus.edu), [e0426065@u.nus.edu](mailto:e0426065@u.nus.edu), [e0310647@u.nus.edu](mailto:e0310647@u.nus.edu)

### 1 Abstract

The project explores the extent to which future stock prices can be predicted using supervised machine learning models with fundamental and technical analysis data as inputs.

The project aims to determine which supervised machine learning model, from Time Series Multi-Linear Regression (TS-MLR), Recurrent Neural Networks (RNN), to Long Short-Term Memory (LSTM), can predict future stock prices with the lowest Root Mean Square Error (RMSE). In doing so, we conducted dimensionality reduction as well as feature selection, providing an insight into the categories of fundamental and technical analysis data that are particularly significant in predicting future stock prices. This insight may be integrated into stock-picking strategies and provide a benchmark on the ideal timing to buy or sell stocks.

The project singles out LSTM as the best performing machine learning model, with an average RMSE of 8.03 in predicting the closing price one month ahead, and an average RMSE of 13.45 in predicting the closing price six months ahead. Our source codes written for this project may be found in the footnotes listed at the bottom of this page[[1]](#footnote-0) [[2]](#footnote-1) [[3]](#footnote-2).

### 2 Introduction

Investing in the stock market tends to be the most volatile type of investment. As such, our project explores one of the ways to minimize such volatilities - analyzing company data to uncover possible trends in price changes of stocks. In doing so, our project hopes that these trends will be able to help increase certainty for investors. Idealistically, the best (least RMSE) model will allow investors to profit from investments and “beat the market”.

The significance of our project is two-fold. Firstly, it provides insight about the more significant factors that affect the prices of stocks, which in turn allows investors to narrow down the scope of research and analysis. Secondly, it provides evidence demonstrating the presence of market inefficiencies, which is one that does not incorporate all available information into a true reflection of an asset’s fair price successfully.

The problem statement our project intends to solve entails the question of the extent to which it is possible to observe patterns or relationships in stock price changes. On a micro level, we aim to determine which predictor(s) would serve as the most relevant ones in determining stock price changes as well as the model(s) that can enable investors to best minimize risks involved.

Some questions our project wishes to answer include:

1. Which predictor(s) are the most relevant in predicting stock price changes?
2. Which machine learning model is the most capable in predicting stock price changes?
3. Would common fundamental and technical analysis features be sufficient to make a useful or insightful prediction?

### 3 Related Work

This sets the foundation of our project which aims to differ and further explore other machine learning models that were not covered in the above study. In particular, the 3 models in our project are purposely chosen to be different so that we may analyze the effect of different machine learning models and their outcomes.

Furthermore, our project utilized fundamental data such as financial ratios as well as considered the effects of time on top of technical data, which was only used in the mentioned study. In particular, We want to explore whether the use of companies’ performance through their major economic indicators can play a part in yielding a more promising result as compared to just using merely historical stock prices. Similarly, features of time were included in our models as well for the same rationale.

### 4 Methodology

#### 4.1 Data Collection and Preprocessing

Data collection was conducted by using API calls to scrape New York Stock Exchange (NYSE) tickers from Yahoo Finance [2]. For the companies that offer two classes of stocks, class A and class B stocks, there was a need to remove such stocks because the API used to scrape fundamental and technical analysis data from is not able to return the corresponding data. Thereafter, the data for the remaining set of tickers was scraped from Financial Modelling Prep API [3] as CSV files, with a time frame between 1st January 2008 to 1st January 2020. Since the CSV files obtained contained separate categories of fundamental and technical analysis data, they are then concatenated according to whether it falls under fundamental or technical analysis.

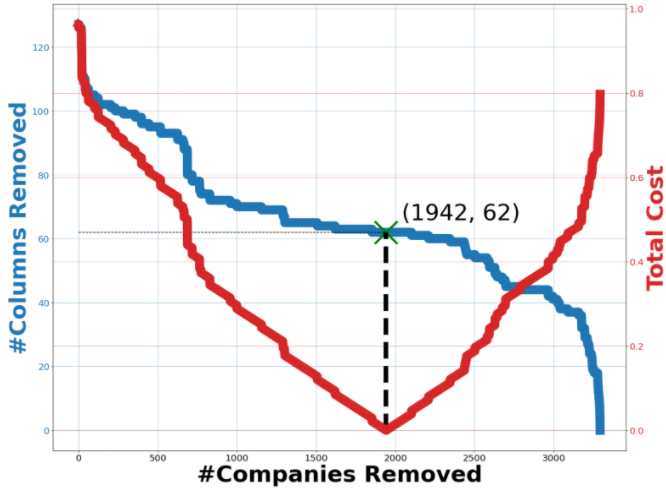
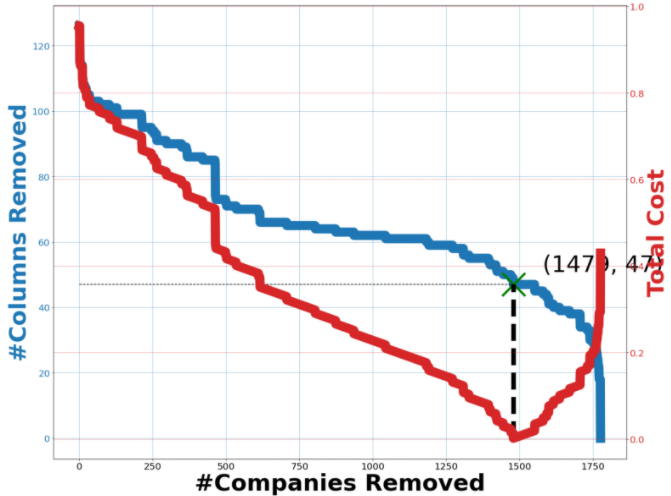
For each company’s technical analysis data, the closing price 1 month and 6 months ahead was appended to every valid row. This implied that for every row (), where there exists data for 30 days ahead (), its closing price was appended to the former, i.e. append to the row where . This procedure was repeated for rows where there exists data 180 days ahead (6 months).

After cleaning up the fundamental and technical analysis data, the two data sets were concatenated. As fundamental analysis data were obtained from quarterly reports (each row contains the date of the first day of the quarter), for each quarter, the corresponding fundamental analysis data of a company was appended to the technical analysis data which falls within the quarter. This was approximately 90 rows of technical analysis data that were appended with the same set of quarterly fundamental analysis data. This concatenation was done by using a sliding window algorithm: for each row in the fundamental analysis data with , append it to all rows in the technical analysis data with .

Data set wais obtained, and was splitted into and . contained the rows for where was not ‘NaN’, and contained the rows for which was not ‘NaN’.

As certain columns in the data set contained NaN or (entirely) 0 values, it posed a problem, since ratios could not be 0. Additionally, RNN and LSTM models are unable to predict where values are NaN. This is due to the mathematical nature of ‘NaN’. In python, any arithmetic operation with ‘NaN’ will be computed as ‘NaN’. Hence, the forward propagation, back propagation and weight updating will be regarded as ‘NaN’ which does not train the model at all.

We thereafter proceeded to examine the minimum number of companies and columns (features) required to be removed in order to eliminate such companies and / or columns. This was done using a greedy algorithm and heuristic such that at each step, the least common necessary column would be removed. The cost of the removal of a column was computed with the absolute difference between the total percentage of companies and features removed.

**

*Figures 1, 2: Trade-offs for removal of features (columns) and companies in the 1-month data set (left) and 6-month data set (right) respectively.*

Figures 1 and 2 visualize the trade-off in cost between the removal of companies and the removal of columns. For the 1-month data set, removing 1942 companies and 62 columns resulted in the least cost, leaving us with 1352 companies and 71 features. On the other hand, for the 6-month data set, removing 1479 companies and 47 columns resulted in the least cost, leaving us with 297 companies and 86 features.

The reason why we were able to have different numbers of companies and columns for 1 month and 6-month data was due to the fact that the factors we used on stocks were inevitably time dependent. Thus, the companies and features that were relevant in the prediction of prices for the 1-month data set would have likely and expectedly been different from the 6-month data set. Additionally, given the large volume of data and number of predictors available for both 1-month and 6-month data, we assumed that the set of predictors selected for each time period would likely be able to capture sufficient relevant information in the data set that would be comparable between 1-month and 6-month data. Nevertheless, to minimize the effect of any errors in comparisons when concluding that the prediction for 1-month data tends to be better than that for 6-months data or vice versa, we prioritized the use of heuristics that would favor the removal of companies as opposed to the removal of features. This would be consistent with our initial assumption that the companies that we were using behave homogeneously, would be influenced to similar degrees by the selected features and respond to relative feature changes in indistinguishable ways.

More details on other combinations of heuristics we used before arriving at the final set of heuristics used for the removal of companies and features may be found under Appendix A.

As a final step to preprocessing, we performed an 80:20 split on the data to produce the training-validation set and test set respectively. This was done separately for each company as combining the companies would certainly have led to data snooping effects where we could possibly have ended up predicting past data using future data, since the companies have different numbers of training examples. This was also performed independently for 1-month and 6-month labels as the same original set data would undoubtedly allow us to be able to derive less future 6-months stock prices than future 1-month stock prices.

To further standardize data for all models and minimize differences in further processing of data between models, we only kept companies with more than 60 and 365 rows of data for 1-month and 6-month data respectively. All models we used in this project thereafter made use of the same training-validation and test sets. This ultimately ensured that the results amongst all models were comparable.

#### 4.2 Model Training, Validation and Testing

For our project, 3 supervised machine learning models are used: time-series multi-linear regression model (TS-MLR), traditional recurrent neural network (RNN) and long short-term memory (LSTM).

TS-MLR

The first model is TS-MLR. We first began by further cleaning up our data. In our data set, we observed 2 types of outliers, the first of which is obvious data points that are extremely large or small relative to the data set for all companies (difference in order of magnitude by at least 16) and second of which was data that was exactly the same across consecutive quarters for some companies (which is highly unlikely for a company to have same data across the years). We then removed these 2 outliers from our data.

In developing the model, we made use of the “statsmodel” library [4], specifically training and validating our data using OLS regression [5]. Using the cleaned up training-validation data (80% of full data set), for each company, we *further* split it into training and validation sets and both splits are done with a ratio of 80:20. In both splits, we used the earlier set of data for training and the later for validation and testing respectively to avoid data snooping because our data depends on time.

In addition, we added features of time t, t2, t3 and quarters as we believe that the effects of time and the observation of any cyclic effects by using categorical variables for quarters might be useful information for MLR [6]. We also did not normalize the values of our features as some of the features do not have a maximum value.

For each iteration, we started with all features (71 for 1-month data and 47 for 6-month data). In every iteration, for all companies, we trained the model for each company using OLS regression on the first 80% of the training-validation set corresponding to the company. Thereafter, to perform validation, we predicted the remaining 20% of the data using the multi-dimensional line generated by the model, and computed RMSE for the company, by calculating the square root of the mean squared difference between the points in the original validation data set (20%) and the respective points predicted by the model. In addition, we also stored all f-stat values for the features that were used to produce the model for each company, which provided an indication of the importance of each feature.

At the end of each iteration where training and validation would have been performed for all companies (1352 for 1-month data and 6-month data), we stored the average of RMSE values for all companies, as well as the average of f-stat values for each feature used. The average RMSE value would be used to determine how well the models performed, whereas f-stat values helped us to understand which feature was most relevant and useful for prediction in the TS-MLR for the iteration. In particular, a lower f-stat value indicates that the overall model’s features are less likely to be useful if the feature is dropped. Thus, before the next iteration we removed the feature with the lowest f-stat value (which when removed would have led to the lowest probability of all features being irrelevant to the model).

Thereafter, we repeated the same process for all remaining iterations by performing feature selection, with one less feature in each iteration – going through all companies in each iteration to generate the corresponding average RMSE and average f-stat values of features – until no more features remain. Our final set of features selected was determined as the set of features having the lowest RMSE value from the training and validation process for all iterations. In the case where we have multiple sets of features having the same RMSE, we would have picked the set with the smallest number of features to minimize undesirable effects on prediction accuracy from the curse of dimensionality.

With our final set of features, we proceeded to train on the initial training set of each company (100% of the training-validation set and 80% of the full data set) and subsequently perform testing on the testing set to obtain the RMSE value for each company respectively. The process of testing is resembling to that of one iteration of the training and validation process, other than the difference in the data used and the fact that f-stat values for each feature is no longer tracked, since the selected features used would have already been determined to be the most useful from the training and validation process. Calculation of the RMSE was made in the same way as before by taking the root mean square difference in predicted values in the test set against actual values.

Finally, upon generating models for all companies during the testing process, the average RMSE of all companies is taken to be the RMSE for TS-MLR (for both 1-month or 6-month data).

The above steps were performed separately for both 1-month and 6-month closing price prediction models in TS-MLR. We note that we were able to take the average RMSE and f-stat values for all companies as we had a large number of companies available – even after preprocessing and further cleanup – and these companies had been taken to behave homogeneously in response to changes in the features selected. Furthermore, we did not take the average of RMSE values between the 1-month and 6-month model as uncaptured differences due to time would make any average RMSE difficult to generalize.

RNN

The second model we considered was RNN. As our project is a time-series forecasting project to predict stock prices, we decided to work with the RNN model which is frequently used for time-series forecasting due to its ability to make use of its internal state (memory) to process sequences of inputs. The connection between nodes also forms a direct graph along a sequence, allowing for the exhibition of temporal dynamic behavior for a time sequence [7].

In developing our model, we made use of keras’ RNN library [8]. Since our project aims to create 2 separate models to predict the closing stock price for a company 1 month and 6 months ahead, training of the 2 models was done separately. In this section, the focus will be on the methodology used in training the model in predicting the closing price 1 month ahead. A similar method can be applied in training the other model.

In order to train and tune the hyperparameters of our model, we made use of a 5-fold cross-validation on the training data set. The set of values for each hyperparameters we validated our model on includes the learning rate , batch size , momentum , dropout rate , and the number of epochs . By looping through each set of values for each hyperparameter, we have a total of 270 unique combinations of hyperparameters to validate.

We then ran the 5-fold cross validation for each unique combination. In each iteration of the 5-fold cross validation, we obtain a validation set which is 20% of the companies in the training data set and the remaining 80% will form the training set. Note that this 20% that forms the validation set will differ with each iteration of the cross validation, such that there would not be a company that ends up in the validation set more than once. Using MinMaxScaler [9], we normalize the training set and subsequently transform the training set from 2D vectors to 3D vectors, each containing 30 days of non-overlapping data.

These 3D vectors are then passed into the model which consists of 4 RNN layers, each with a dropout rate, which is according to the combination of hyperparameters being tested. The first layer of the model is the input layer and the last layer in the model is the dense layer [10] (i.e. the fully-connected layer connecting all the neurons from the previous RNN layer to it) which implements the output function [11] and stochastic gradient descent is used to iteratively update network weights. We also reset the state of the model between each company. In addition, we used the “tanh” activation function instead of “ReLU” to avoid the Dying ReLU problem. Instead of solving the vanishing gradient problem in RNN, ReLU returns 0 for all negative values. Consequently, a ReLU neuron could get stuck in the negative side and always output 0, where it is unlikely to recover.

We then validate this model using the validation set and obtain the average RMSE between the predicted output and the actual stock price 1 month for 1 company. Upon the completion of all iterations of the 5-fold cross validation, we then obtain the average RMSE per company per iteration for this particular combination of hyperparameters. The optimal combination of hyperparameters is then the combination that gives the lowest average RMSE.

Using this optimal combination of hyperparameters, we obtain our final model, which we will train using the entire training data set. Subsequently, we will test using the testing data set and obtain the average RMSE for a company to be used as the basis of comparison with the other 2 models.

Steps above were repeated for the model used in predicting the closing price for 6 months ahead, with the difference being input vectors of 180 rows used, as compared to the 30 rows used for 1-month prediction.

LSTM

The last model we considered was LSTM. As stock prices data are temporal and the data set is large, LSTM, a special type of RNN that overcomes the issue of vanishing gradient and long-term dependencies that traditional RNN faces, was chosen as the third model.

In developing our model, we made use of keras’ LSTM library [12]. As we wanted to predict the closing price for a company 1 and 6 months ahead, cross validation was done for the two data sets. However, the focus will be on the methodology for the model used in predicting the closing price for 1 month, as the methodology for the model used in predicting the closing price for 6 months is similar.

The training and testing data of all companies for 1-month prediction was first imported. 10-fold cross validation was used to tune the hyperparameters of the LSTM model. A set of values for each hyperparameter, the number of epochs , batch size , and dropout rate , were used to obtain a set of unique permutations [13]. For any fold (10% of the 1-month data set), the other 9 folds were used to train the model on a permutation of the hyperparameters, then validation is done on that fold and the validation cost for that permutation of hyperparameters is incremented. For this particular fold, the procedure repeats for all permutations of hyperparameters. After this is repeated for all 10 folds, the permutation with the least sum of validation cost (RMSE) is the set of hyperparameters used in training and testing the final model for the 1-month data set.

Referring to appendix B, the lowest validation RMSE for the 1-month data set is obtained when the model is trained with 1 epoch, batch size of 64, and a dropout rate of 0.2. For the 6-months data set, it is obtained when the model is trained with 1 epoch, batch size of 32, and a dropout rate of 0.3.

The LSTM model is built with 4 LSTM layers, each with a dropout rate after it. The first LSTM layer is the input layer and the last layer in the model is the dense [10] layer (i.e. the fully connected layer connecting all the neurons from the previous LSTM layer to it) which implements the output function [11]. The well-known adaptive moment (Adam) estimation was used to iteratively update network weights in place of standard gradient descent.

First, 10-fold cross validation was done using the training data. The training data is normalized using MinMaxScaler [9]. The input data points are 3D vectors generated by appending the past 30 days (from to ) for each row of data (with ) as shown in figure 3. The 3D vectors were reshaped. The 3D-input vectors were then fed into the LSTM model for training. Between the training of the model on each company, the state of the model was reset. 10-fold cross validation is conducted to calculate the average RMSE for each permutation of the hyperparameter and determine the set of hyperparameters with the lowest RMSE which was used for the actual training and testing.

The final model was initialized using the set of hyperparameters with the RMSE and trained using the training data of all companies. The state of the model is reset, and the input vectors from the testing set is fed to the model, and the model predicts the closing prices 1-month ahead using the input vectors. The average RMSE in predicting closing prices 1-month ahead was computed against the actual closing prices 1-month ahead. This average is used as a basis of comparison with the other 2 models.

Steps above were repeated for the model used in predicting the closing price for 6 months ahead, with the difference being 3D input vectors of 180 rows were used, as compared to the 30 for 1-month prediction.

### 5 Evaluation

Data set

Given the data set for our project, there are several aspects of data quality which we can evaluate. It is important to note that data quality encompasses multiple aspects and it may have a direct impact on the quality of our project.

After combining our technical and fundamental data, our data set faced the issues of missing column values due to incomplete information, specifically 0 or NaN values, as explained earlier in Section 4.1. Due to these missing values, we were forced to trim our original data set to remove columns consisting of such incomplete information. Every removed column corresponds to a unique feature and they could have potentially played an important role in improving the quality of our models. However, due to the limitation of the data set we obtained, we had to remove those features.

Another issue with our original data set was the inconsistent data for every company. For all our companies, their individual data may range from less than 1 year to longer than 15 years. For companies with very little data, we were then unable to use them effectively for our machine learning models, especially so in our 6-month prediction models as it required minimally 365 rows. Therefore, companies not meeting the minimum required number of rows of data were removed, further reducing our data set.

TS-MLR

In order to gain further insights on TS-MLR, we looked into two similar studies which employed linear regression models for the purpose of predicting stock prices and trends.

The first paper [14], titled Mobile App for Stock Prediction Using Improved Multiple Linear Regression was published in the 2017 International Conference on Sustainable Information Engineering and Technology (SIET)]. The paper made use of a hybrid Multiple Linear Regression (MLR) with Moving Average technique and historical stock prices to predict short-term stock prices of the Jakarta Stock Exchange (JKSE), ultimately yielding an RMSE of 122.83.

On the other hand, the second paper [15], titled Regression Techniques for the Prediction of Stock Price Trend and published in the 2012 International Conference on Statistics in Science, Business and Engineering (ICSSBE), made use of six selected features as measures to predict future stock price *trends* of stocks in Bursa Malaysia. Not only that, a data transformation process converted real-valued inputs into ordinal data. The resulting observation was an improvement in RMSE by more than twenty times, for linear regression, from 18.60 down to 0.84, which was astonishing as much as the prediction seemed to be for stock price trends.

Differences in stock exchanges aside, from the comparison of studies, we observed that multiple linear regression as a technique for stock price prediction could be more useful if real-valued data was transformed to ordinal data. This could be due to the inability of MLR to make use of past data – unlike RNN or LSTM – and thus making the weights of features extremely sensitive to movements in data (real values), as we have clearly observed in trial training runs of our MLR model, where the presence of unremoved outliers with an order of magnitude greater than 16 led to extremely large RMSE values, despite making up only less than 1% of the data. Additionally, MLR could be more useful if we redefined the predicted values to be categories of trends rather than actual stock prices, which are prone to high levels of fluctuations. This particular finding would show us that despite the fact that MLR is known to be relatively ineffective in predicting stock prices compared to other models such as RNN or LSTM, applications for MLR could still effectively help to solve a varied scope of problems in stock predictions.

LSTM & RNN

We first examine LSTM models used for stock market prediction. In a paper [16] by International Journal of Science and Research (IJSR), a LSTM model was used to predict stock prices. Only historical price data was used for their experimentation, and RMSE was used to analyze the loss of the model. From their experiment, we see that RMSE for testing (lowest RMSE is 0.00859) is much lower than what our LSTM model produced (8.03), significantly outperforming our LSTM model.

From what we have observed when comparing LSTM used in other experiments and the above 3 other models, the main difference is the data set used. In our experiment, we used fundamental analysis and technical analysis data, which has significantly more metrics (features) than what the other experiments have used. This is evident in the LSTM experiment conducted by IJSR, where they only used historical daily price data, and metrics used are the daily prices of high, low, open and close. The significantly lower number of features used in their experiment would explain why their RMSE is lower than our RMSE which could be due to the curse of dimensionality in our model.

The curse of dimensionality states that the number of samples needed to estimate an arbitrary function with a given level of accuracy grows exponentially with respect to the number of input variables (i.e., dimensionality) of the function. If there are more features than samples, then we run the risk of overfitting our model resulting in poor performance [17], [18]. After reducing the number of features (appendix A), the input data still has 69 features. Additionally, as explained above, not all companies have a full data set, i.e. they only contain a subset of dates between 1st January 2008 and 1st January 2020. This issue is exacerbated because each input vector contains 30 rows or 180 rows (depending on the data set). For example, given a company with 5 years of data in the window, there are initially approximately 1825 rows, however, after generating the non-overlapping input vectors, we are with just 60 and 10 data points for the 1-month and 6-months data set respectively. .

Additionally, the LSTM experiment conducted by IJSR uses technical indicators as features, which are directly correlated with the closing prices. For our LSTM model, since the intention was to unify the two schools of analysis (technical and fundamental), the inclusion of fundamental analysis data which may not be directly correlated with closing price may have caused the performance to deteriorate as well.

Finally, looking at the tuning of hyperparameters for LSTM, the number of LSTM layers was not chosen as a hyperparameter to be fine-tuned using cross validation due to the lack of time. For the same reason, the number of epochs was fixed at 1 for the cross validation on the 1-month data set because it has significantly more data than the 6-months data set, and it would take too much time. We recognized that there could have been more hyperparameters to be tuned, and more values could have been permuted.

Experimental Results

As discussed in the evaluation of the 3 models, the following table summarizes the experiment results with their corresponding average RMSE for 1 month and 6-month prediction.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | TS-MLR | RNN | LSTM |
| 1-month average RMSE | 44.18 | 47.02 | 8.03 |
| 6-months average RMSE | 81.62 | 49.83 | 13.45 |

*Table 1: Average RMSE for the 3 models*

### 6 Discussion

### 6.1 Most relevant predictors in predicting stock price changes

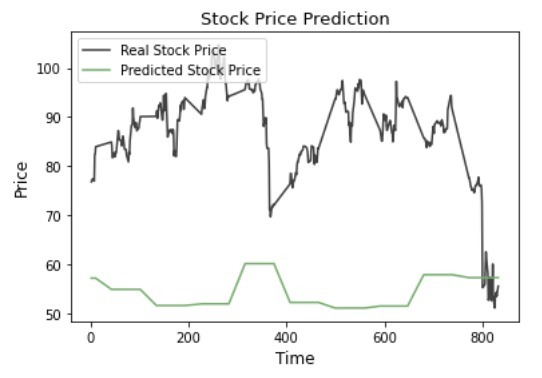
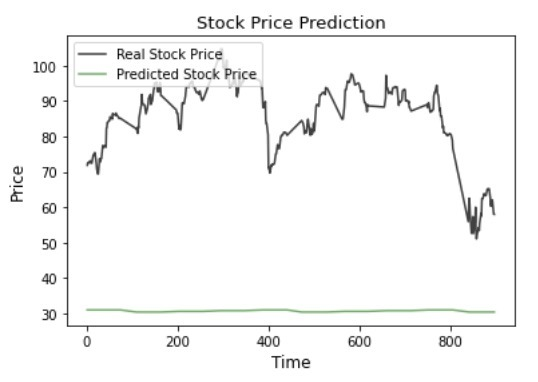
For our MLR model, using f-stat values as a basis for the usefulness of features and backward elimination of features (i.e. greedy removal of least useful features one by one), our feature selection process resulted in 1 useful feature for 1-month data and 8 useful features for 6-month data. The single feature that resulted from feature selection for 1-month data turned out to be Operating Income Growth, whereas that for 6-month data included: Net Income Growth, Operating Cash Flow Growth, Enterprise Value to Operating Cash Flow, Net Debt to EBITDA, Quick Ratio, Cash Conversion Cycle, Long Term Debt to Capitalization and Free Cash Flow to Operating Cash Flow Ratio.

As these final features were selected via the feature selection process, they would have yielded the sets of features for 1-month data and 6-month data respectively that result in the lowest corresponding RMSEs during the training-validation phase. These features would represent the sets of features that would most likely continue to yield low RMSEs for test data ,and could hence be said to be the most useful feature for predictions in our MLR model.

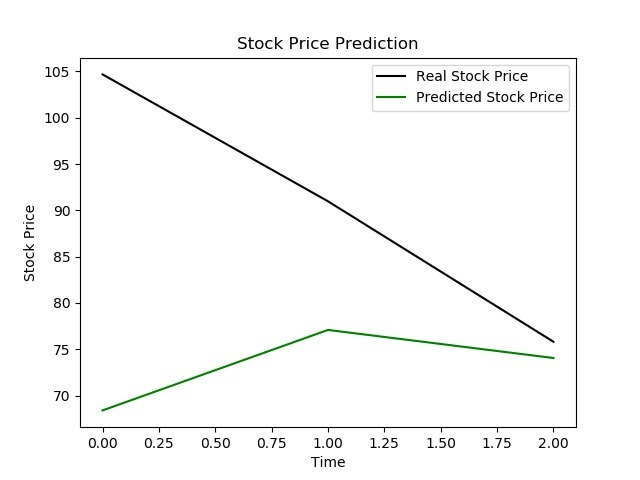
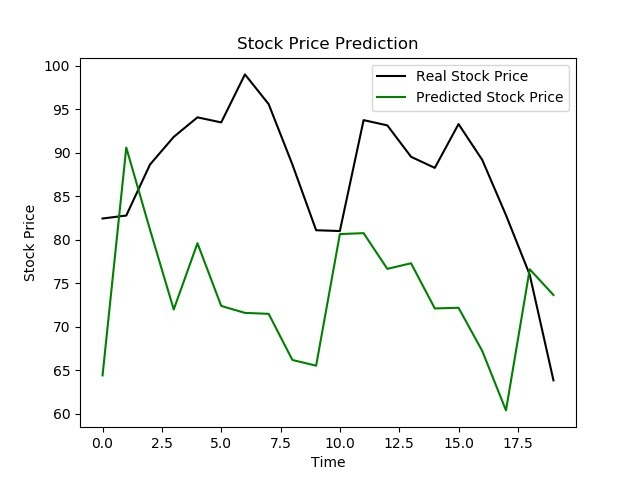
However, we believe that we should exercise caution in generalizing this observation, as it would be erroneous to conclude based on our feature selection process that the selected features would also be similarly relevant in other MLR models and / or data sets. This is due to factors including but not limited to the complexities of the models generated, behavior of companies not necessarily being homogenous and the bias-variance trade-off with the large number of initial dimensions. These reasons would also mean that as much as our selected features could be the most useful ones in our prediction, it would not have been possible to easily *explain* why these features out of the numerous initial features in particular have been selected.

For our deep learning models, there was no easy and accurate way to predict which variables were significant in predicting the stock price movement. The nature of the Neural Network model which sums the previous results from a node before being fed into an activation function , makes the contribution of a single input hard to detect. The contribution of a certain variable into one node may be different to the contribution of that variable to a different node. Furthermore, the sums of the values are fed to an activation function which is affected by the contributions from other predictors. Hence, the contributions of a specific variable is mixed with other variables such that it is hard to extract the specific contribution from that particular variable, especially in high dimensional data.

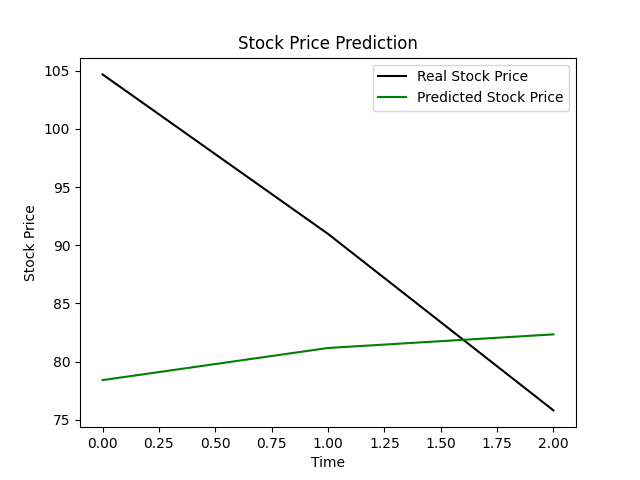
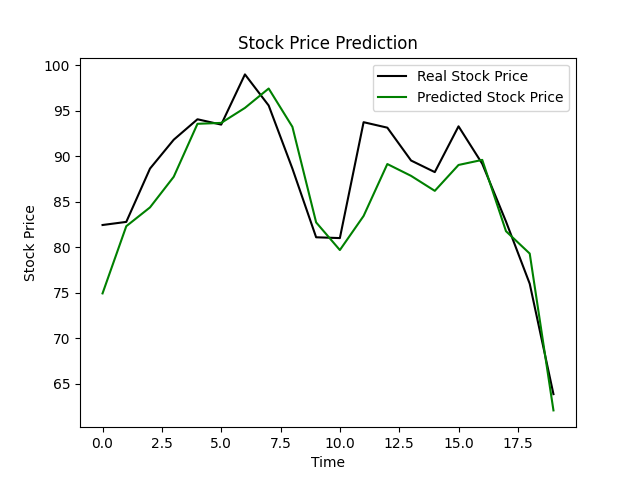
### 6.2 Best model for 1-month and 6-months prediction



*Figures 3, 4: Plots of closing price 1-month (left) and 6-months (right) ahead for CVGW using TS-MLR*



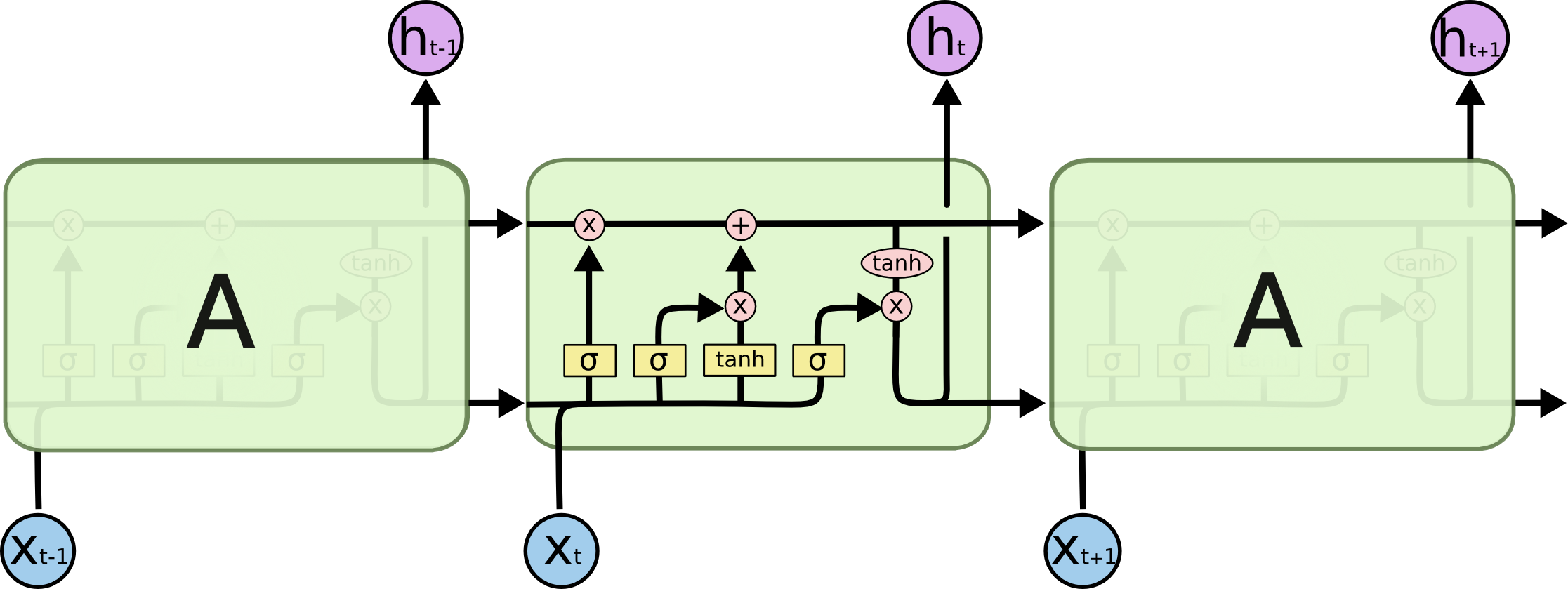
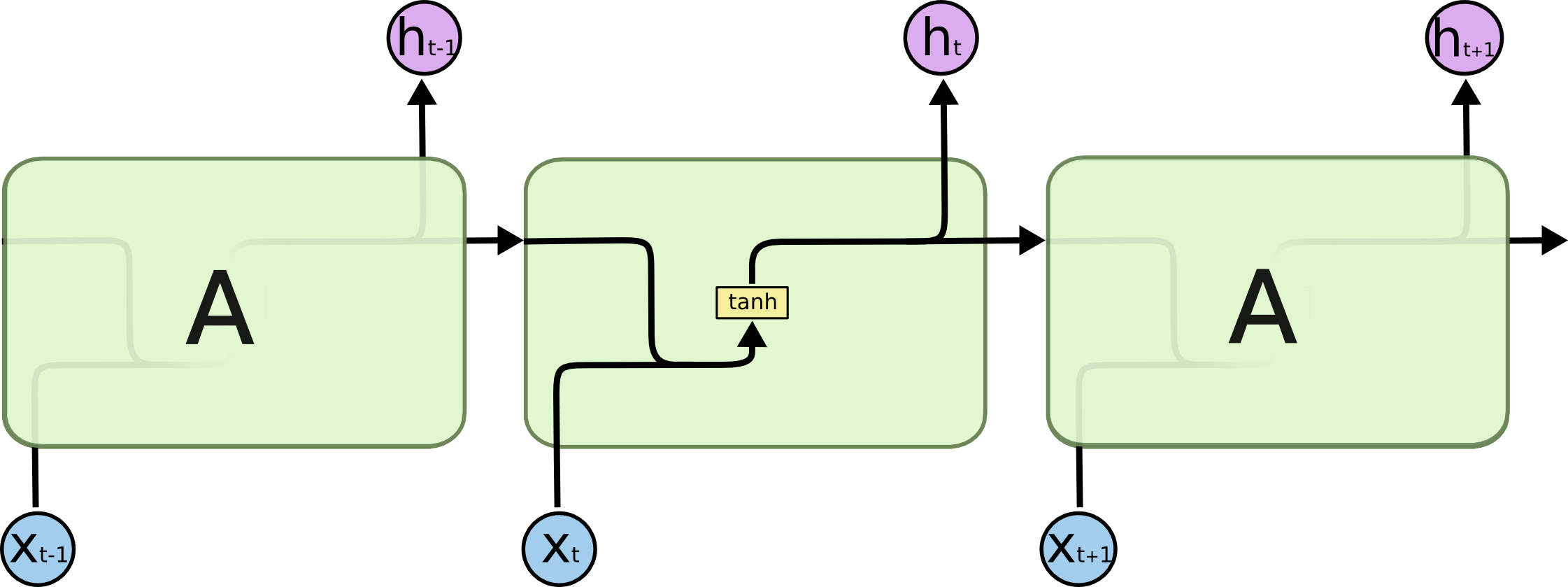
*Figures 5, 6: Plots of closing price 1-month (left) and 6-months (right) ahead for CVGW using RNN*



*Figures 7, 8: Plots of closing price 1-month (left) and 6-months (right) ahead for CVGW using LSTM*

Table 1 and figures 3, 4, 5, 6, 7, 8 supports our hypothesis and prior research that RNN outperforms TS-MLR in predicting future stock prices, and LSTM, a special type of RNN, outperforms RNN in the same task. The RNN (and LSTM) outperforms TS-MLR as a result of its ability to handle long term dependencies, i.e. connecting the previous data to the present input. However, RNN has a problem of vanishing gradients because of its single layer in a RNN cell. As a result it cannot handle long term dependencies well. LSTM outperforms RNN as it has four interacting layers in each LSTM cell, allowing it to model a cell state.

Using the leftmost gate, the cell determines which of its past information to “forget”. Using the middle gate and tanh activation function determines what new information from the present input to update the cell state with, then the cell state (top line) is updated. The last gate is the output gate, which is passed on to the next LSTM cell as an additional input. In contrast, for RNN, there is only a single activation function. Then, when the cost function is backpropagated to update the weights of the RNN and as the training progresses, information from the past gets “lost” due to vanishing gradient (weights become close to 0), preventing the model from learning long-term dependencies and making it ineffective.



*Figures 9,10 : RNN cell (left) and LSTM cell (right) respectively. Diagram credit: Christopher Olah @* [*coach.github.io*](https://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTM is one of the most successful RNNs architectures as it introduces the memory cell, a computational unit replacing traditional artificial neurons in the hidden layer of the network. With these memory cells, neural networks are able to “associate” past information and present input, therefore making it suitable to model the underlying structure of temporal stock price data with a high prediction accuracy.

### 6.3 Sufficiency of fundamental and technical analysis features for stock price predictions

It is apparent from the RMSE result that the three different models are able to capture the correlation between technical and fundamental data to the stock price movements to certain extent. In particular, the LSTM model is able to capture the time series information from the technical and fundamental data with a fairly high accuracy and low RMSE, for both 1-month data and 6-month data. However we acknowledge that for all models, there still exist relevant predictors that could have captured useful information but were not included in our training data, such as seasonality, industrial trends, sentiments, political information, natural disaster and others. These missing informational gaps that our models have assumed to be negligible renders the potential utility and applications of our models to be lower than they could possibly have been, thus limiting the sufficiency of fundamental and technical analysis features alone for predicting stock prices.

The impact of sentiments on stock price forecasting is evident in a study done by Keio University, titled “Combining Technical Analysis with Sentiment Analysis for Stock Price Prediction” [19]. In the study, it was reported that the inclusion of sentiment analysis of news and comments on top of the technical analysis gave a better prediction for stock price movements.

For MLR, attempts were made to include some information on seasonality by incorporating the categorical feature on quarters. However, during the feature selection process, this feature on quarters was interestingly dropped, along with the other features that were added to capture the relationship between time and stock prices (i.e. t, t2, t3). While this could indicate that feature selection had removed features which were not useful, the fact that the process had dropped features related to time and other key performance indicators could imply the presence of other significant factors. Such factors would include limitations to the assumptions that the features were all independent, homoscedastic and had linear relationships with stock prices, all of which would have been crucial to the use of OLS for regression. For instance, the features we used may not necessarily be independent as they were based on figures from the financial statements, which could be mathematically related to one another. As a result, we were unable to observe the effects of seasonality on stock prices using time-series MLR. To mitigate this issue, further improvements to the model could potentially be made by transforming the features to ensure a more uniform variance in residuals that could result in better performance of predictions.

### 

### 7 Conclusion

In conclusion, our LSTM model performed better than the traditional RNN and time series MLR in predicting stocks due to the ability of LSTM to memorize time related information. In contrast, time series MLR does not retain time related information and RNN has a short- term memory problem due to vanishing gradients. However, in predicting prices of stocks 6 times ahead, the RMSE was always higher for all models due to the lack of training data used in training the 6 months model. Since we require companies to have 180 priors data as one point, we have less data points for each company, resulting in more companies being filtered out.

We acknowledge that the assumption that companies having the same behavior in price movements might not hold. We also acknowledge that this model is only applicable to be used in the NYSE data set, or extrapolation might occur. We further acknowledge that we used one quarter lagging fundamental data in the training data, which might have different correlation for different companies. For instance, small cap companies may have unbalanced balance sheets but great performance, while big cap companies usually have balanced cash to debt ratio. In addition, due to the difficulties of obtaining fundamental data to certain companies – especially small ones –, those companies were filtered out from our training data, which could make our models biased towards middle sized to big cap companies. In addition, there are other aspects which influence the volatility and stochasticity of stock prices, which cannot be fully captured by our data, such as seasonality, emotional trading, and irrational human behavior. Nevertheless, our findings, in particular the ones from LSTM, suggest that the movement of stock prices across most companies can possibly be captured to some observable degree by a small number of fundamental and technical data, demonstrating evidence of existing market inefficiencies and certainly further potential applications of machine learning in the field of stock market predictions.

In the future, we would like to explore the possibility of fitting human behavior and sentiments into the model. This might be achieved by using the Natural language Machine Learning model to scan tweets and other information in the online media. In addition, we would like to explore other machine learning models such as Particle Swarm Optimization, least square support machine vector, and Convolutional Neural Network.

### 

### 8 Acknowledgements

We would like to share our appreciation for all of the following:

Our tutor in charge of our group project, Wen Bin, for his constant advice on who we should proceed on our project and answering all of our questions.

Our other grading tutors, Chris and Shreyas, for their insights on how we can better improve on our project and advice on the validation process.

Lastly, Prof Kan Min-Yen, for giving our group the opportunity to participate in STePS and the guidance in our project. Additionally, we are thankful for being introduced to the world of machine learning and how we can apply what we learn in this project.

### 9 References

[1] Stock Market Forecasting Using Machine Learning Algorithms <http://cs229.stanford.edu/proj2012/ShenJiangZhang-StockMarketForecastingusingMachineLearningAlgorithms.pdf>

[Accessed: 28-August-2020]

[2] R. Aroussi, “Yahoo! Finance Market Data Downloader,” GitHub. [Online]. Available: <https://github.com/ranaroussi/yfinance>. [Accessed: 12-Nov-2020].

[3] Financial Modeling Prep, “Financial Modeling Prep API Documentation,” Financial Modeling Prep. [Online]. Available: <https://financialmodelingprep.com/developer/docs/>. [Accessed: 12-Nov-2020].

[4] statsmodels, “API Reference,” *statsmodels*. [Online]. Available: <https://www.statsmodels.org/stable/api.html>. [Accessed: 12-Nov-2020].

[5] statsmodels, “statsmodels.regression.linear\_model.OLS,” statsmodels. [Online]. Available: <https://www.statsmodels.org/stable/generated/statsmodels.regression.linear_model.OLS.html>. [Accessed: 12-Nov-2020].

[6] BI 555, “Lecture 10. Time series forecasting with Multiple Linear Regression,” Youtube. [Online]. Available: <https://youtu.be/td9QLuijh9c>. [Accessed: 12-Nov-2020].

[7] M. Benkamoun, “Stock Price prediction using Recurrent Neural Networks,” Medium, 15-Feb-2019. [Online]. Available: <https://medium.com/@marianne.benkamoun/stock-price-prediction-using-recurrent-neural-networks-369c21817da8>. [Accessed:

12-Nov-2020].

[8] Keras, “Keras layers API”, Keras. [Online] <https://keras.io/api/layers/>

[9] sklearn, “sklearn.preprocessing.MinMaxScaler,” scikit. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>. [Accessed: 12-Nov-2020].

[10] Keras, “Keras documentation: Dense layer,” Keras. [Online]. Available: <https://keras.io/api/layers/core_layers/dense>. [Accessed: 12-Nov-2020].

[11] Novak, “What exactly is Dense in LSTM model description?,” Stack Overflow. [Online]. Available: <https://stackoverflow.com/questions/55613969/what-exactly-is-dense-in-lstm-model-description>. [Accessed: 12-Nov-2020].

[12] Keras, “Keras documentation: LSTM layer,” Keras. [Online]. Available: <https://keras.io/api/layers/recurrent_layers/lstm/>. [Accessed: 12-Nov-2020].

[13] “itertools - Functions creating iterators for efficient looping,” Python. [Online]. Available: <https://docs.python.org/3/library/itertools.html>. [Accessed: 12-Nov-2020].

[14] A. Izzah, Y. A. Sari, R. Widyastuti, and T. A. Cinderatama, “Mobile App for Stock Prediction Using Improved Multiple Linear Regression,” IEEE Xplore. [Online]. Available: <https://ieeexplore-ieee-org.libproxy1.nus.edu.sg/stamp/stamp.jsp?tp=&arnumber=8304126>. [Accessed: 12-Nov-2020].

[15] L. S. Han and M. J. Nordin, “Regression Techniques for the Prediction of Stock Price Trend,” IEEE Xplore. [Online]. Available: <https://ieeexplore-ieee-org.libproxy1.nus.edu.sg/stamp/stamp.jsp?tp=&arnumber=6396535>. [Accessed: 12-Nov-2020].

[16] M. Roondiwala, H. Patel, and S. Varma, “Predicting Stock Prices Using LSTM,” Semantic Scholar. [Online]. Available: <https://pdfs.semanticscholar.org/fb6b/c0c7bcfb7105ad3b9a66df2b1b63c664cb2f.pdf>. [Accessed: 13-Nov-2020].

[17] T. Yiu, “The Curse of Dimensionality,” Medium, 20-Jul-2019. [Online]. Available: <https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e>. [Accessed: 12-Nov-2020].

[18] A. Bilogur, “Curse of Dimensionality,” Kaggle, 07-Dec-2017. [Online]. Available: <https://www.kaggle.com/residentmario/curse-of-dimensionality>. [Accessed: 12-Nov-2020].

[19] S. Deng, T. Mitsubuchi, K. Shioda, T. Shimada and A. Sakurai, "Combining Technical Analysis with Sentiment Analysis for Stock Price Prediction," 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, Sydney, NSW, 2011, pp. 800-807, doi: 10.1109/DASC.2011.138.

### 

### 10 Appendices

### 

#### Appendix A: Heuristics for Minimising Number of Companies to Be Removed

Since trying out the removal of all possible combinations of companies and features using a dynamic programming algorithm would require factorial time complexity, we considered two main heuristics when minimising the number of columns, denoting them H1 and H2.

**Heuristic 1 (H1):** Remove company having the most number of nan/zero columns at each step (as suggested during the previous consultation)

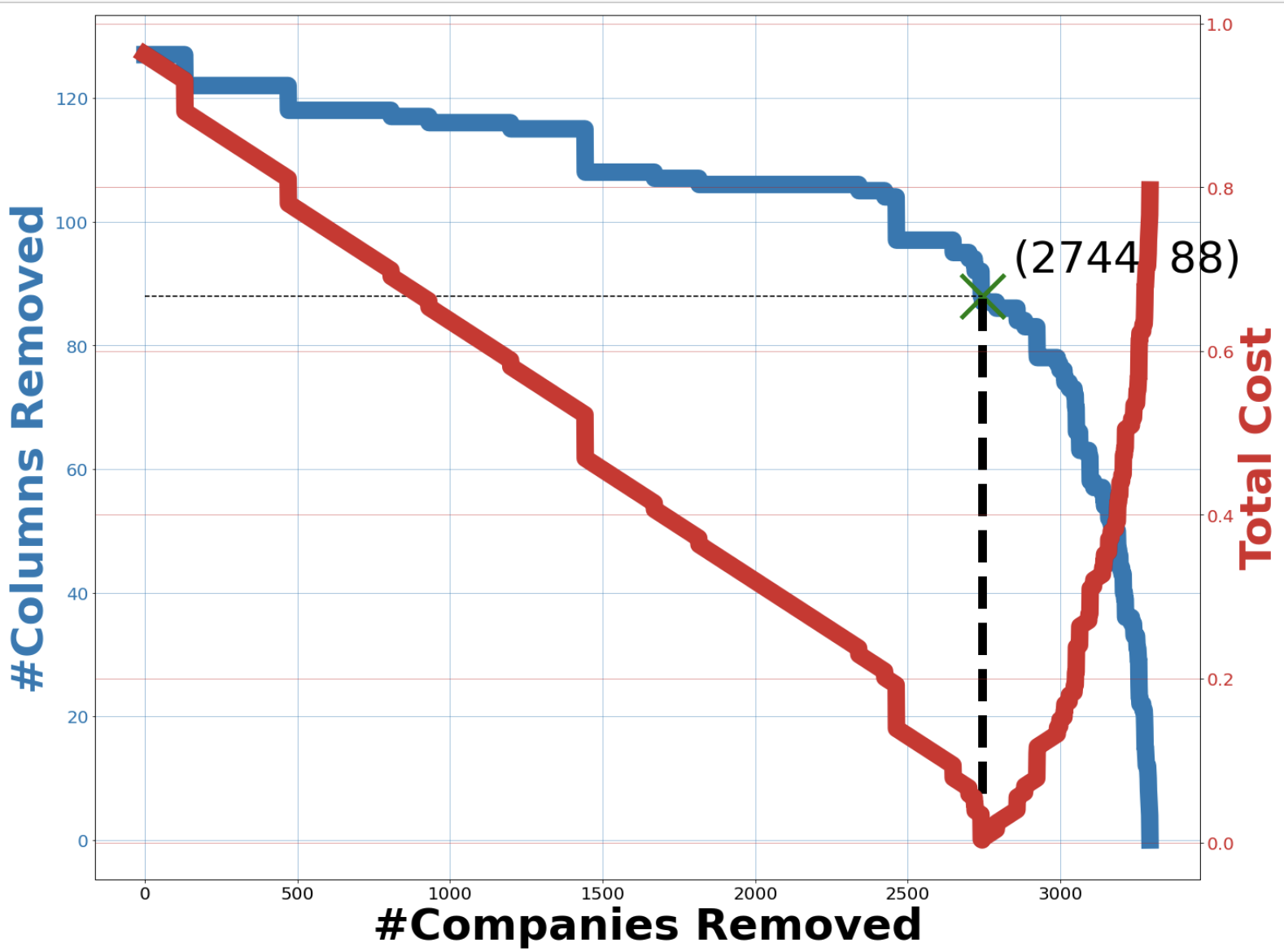
**Heuristic 2 (H1):** Remove companies in the order of the "rarest" non-zero column they have i.e. for each company, the nan/zero column of the company appearing the least number of times (amongst all companies) is counted, and compared with that of other companies; companies are removed in decreasing order of their rarest nan/zero column they have. The intuition behind this is that we want to remove as little companies as possible required to preserve each iterated column that is nan/zero in these companies.

Additionally, with regards to the definition of costs, we attempted the following cost functions, denoted C1 and C2.

**Cost function 1 (C1):** Absolute difference in percentage of companies removed and percentage of columns removed.

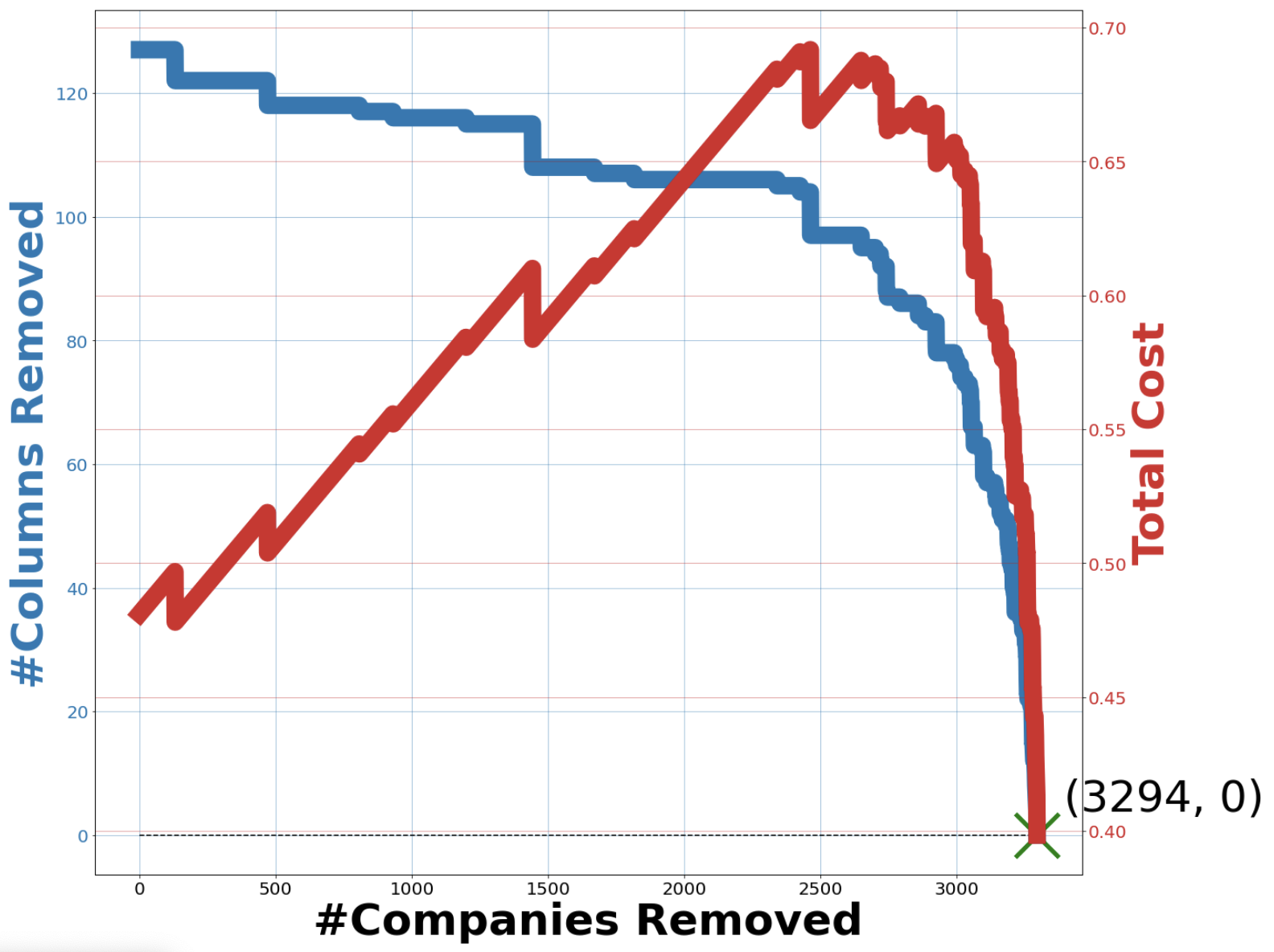
**Cost function 2 (C2):** Average percentage of companies removed and percentage of columns removed.

The combination of the different possibilities of heuristics and cost functions led to 4 possible ways to select the minimum number of companies and features for reduction, for each of the time periods of prediction (i.e. 1-month data and 6-months data), shown as follows.

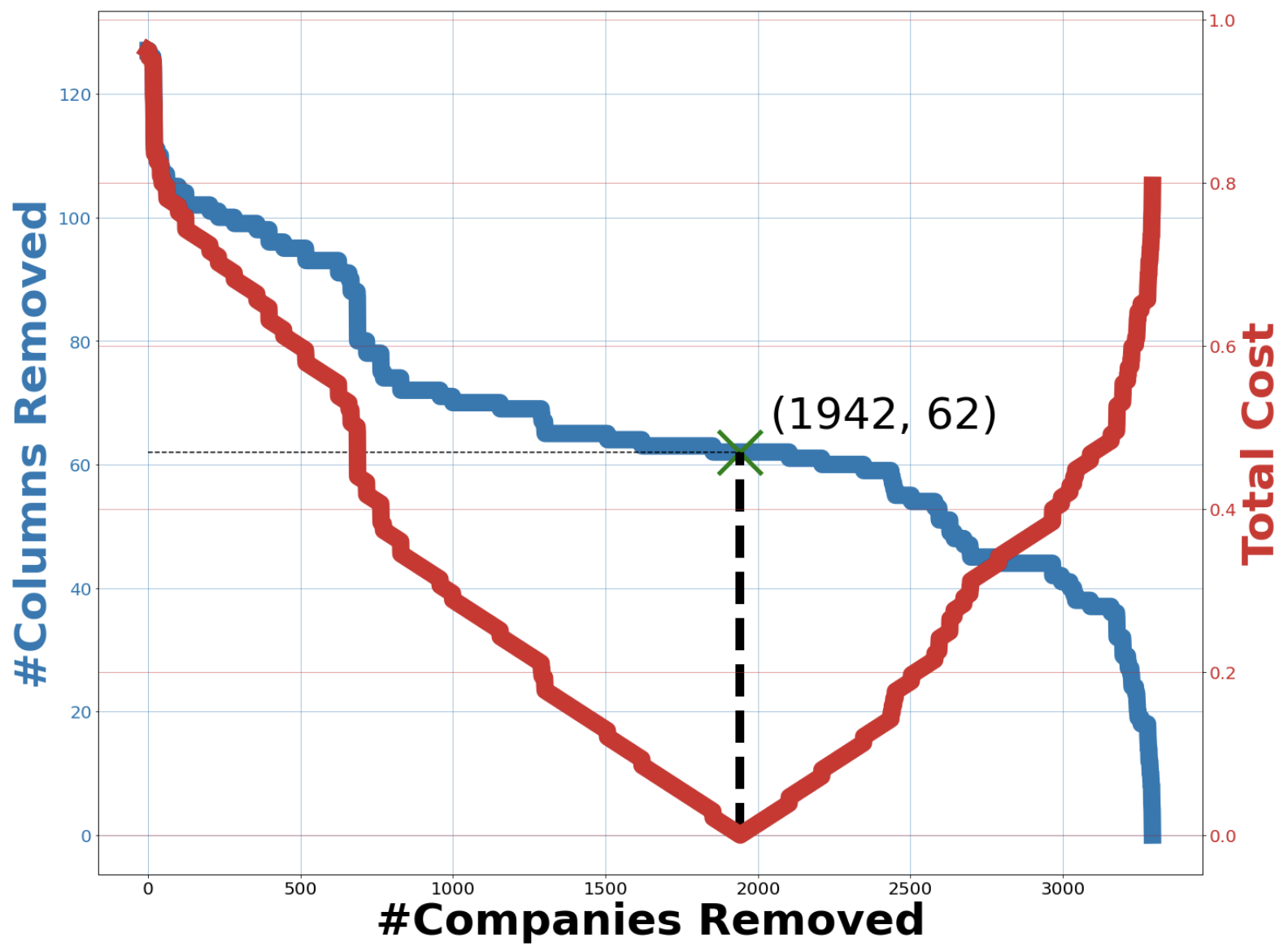


*Figure A1: H1 C1 for 1-month data*

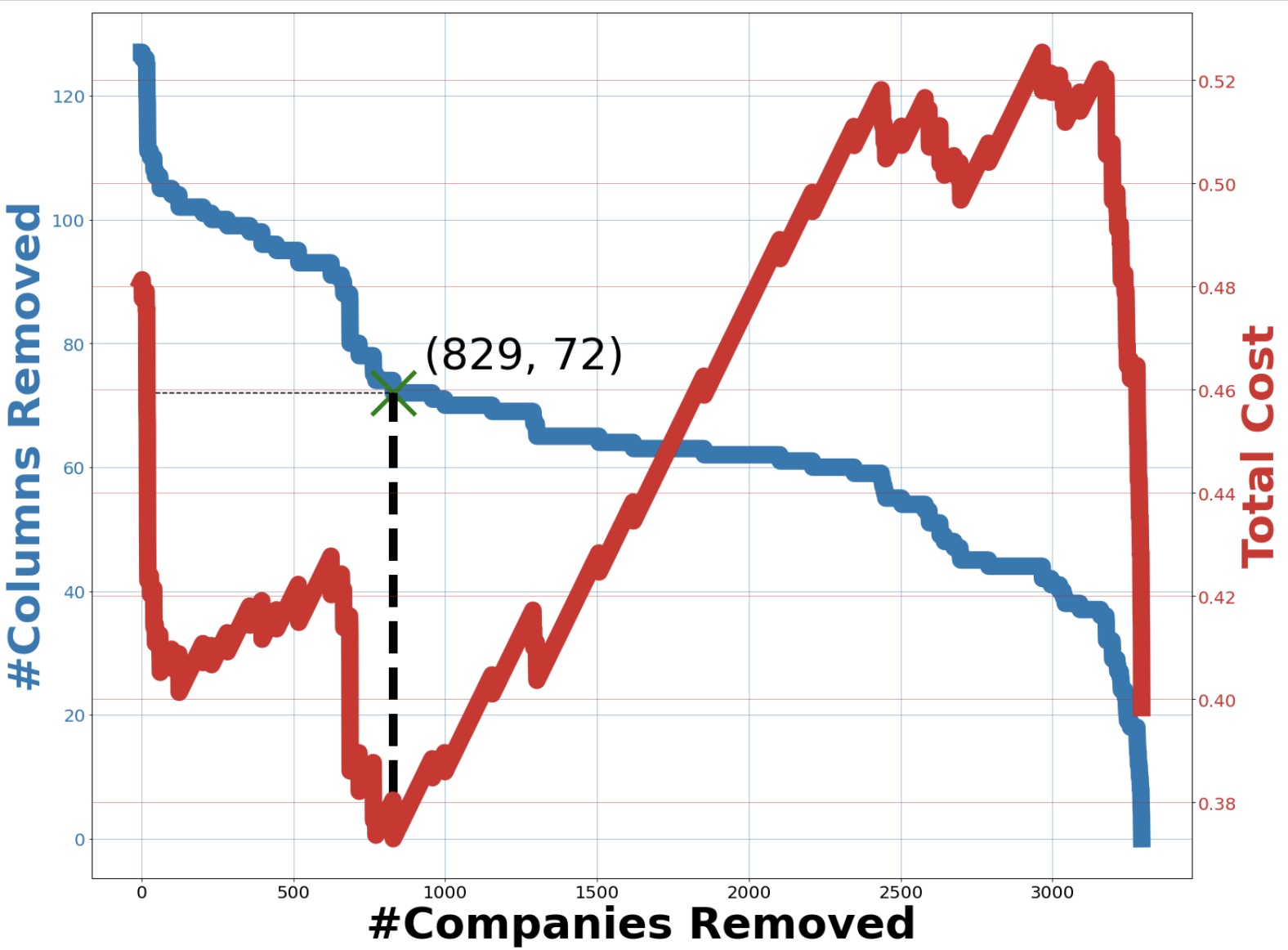
H1\_c2\_1mo



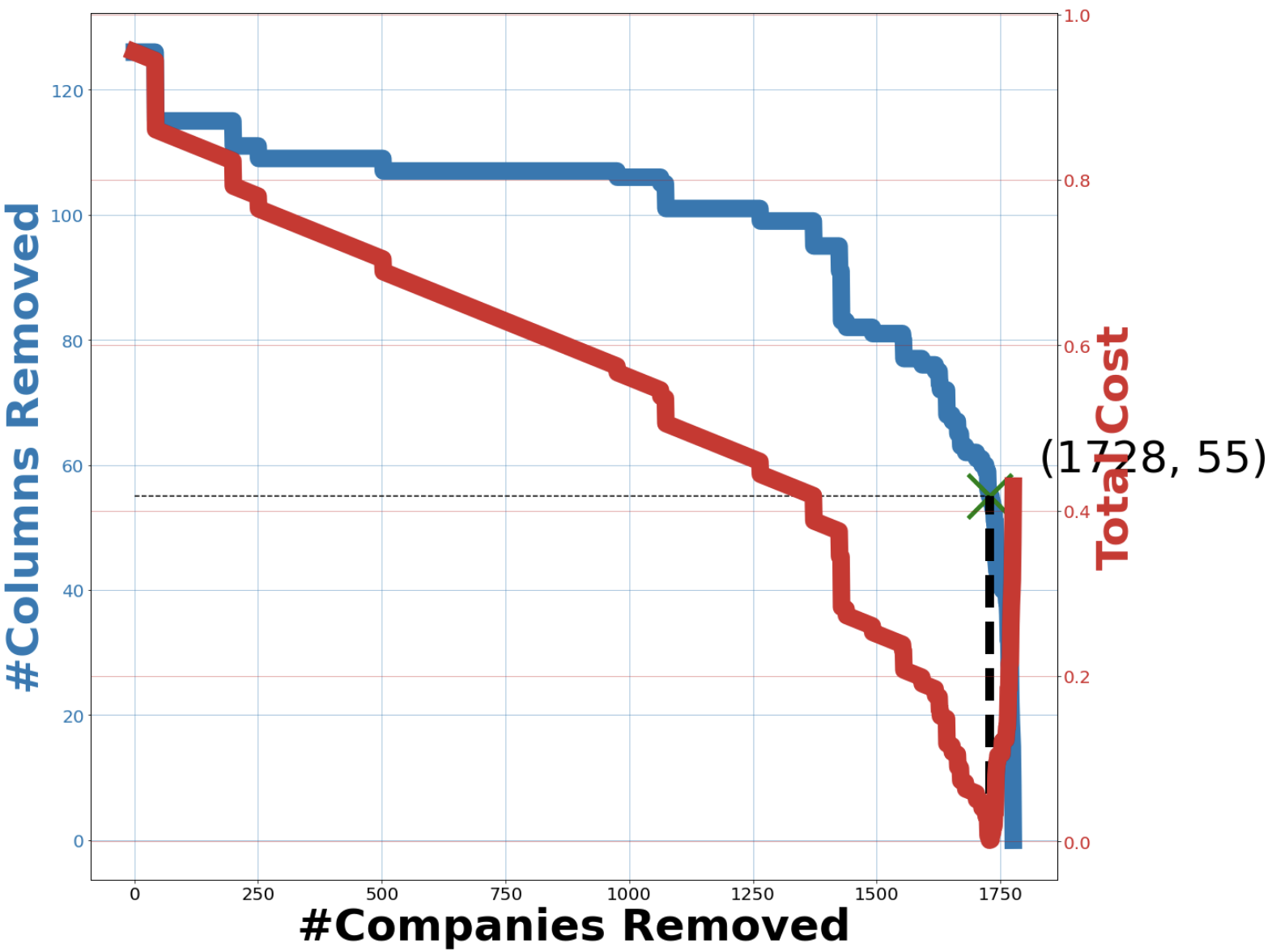
*Figure A2: H1 C2 for 1-month data*



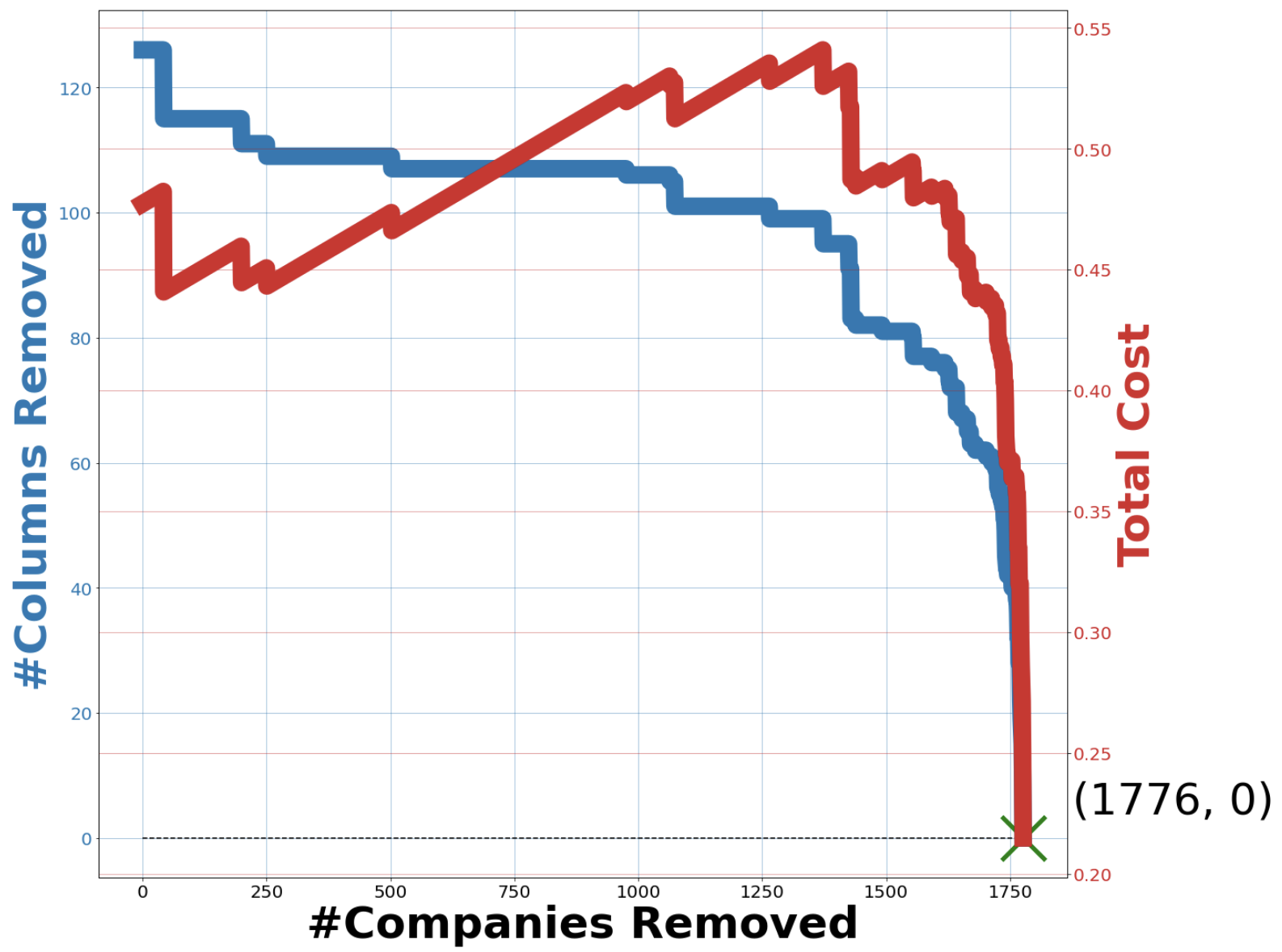
*Figure A3: H2 C1 for 1-month data*



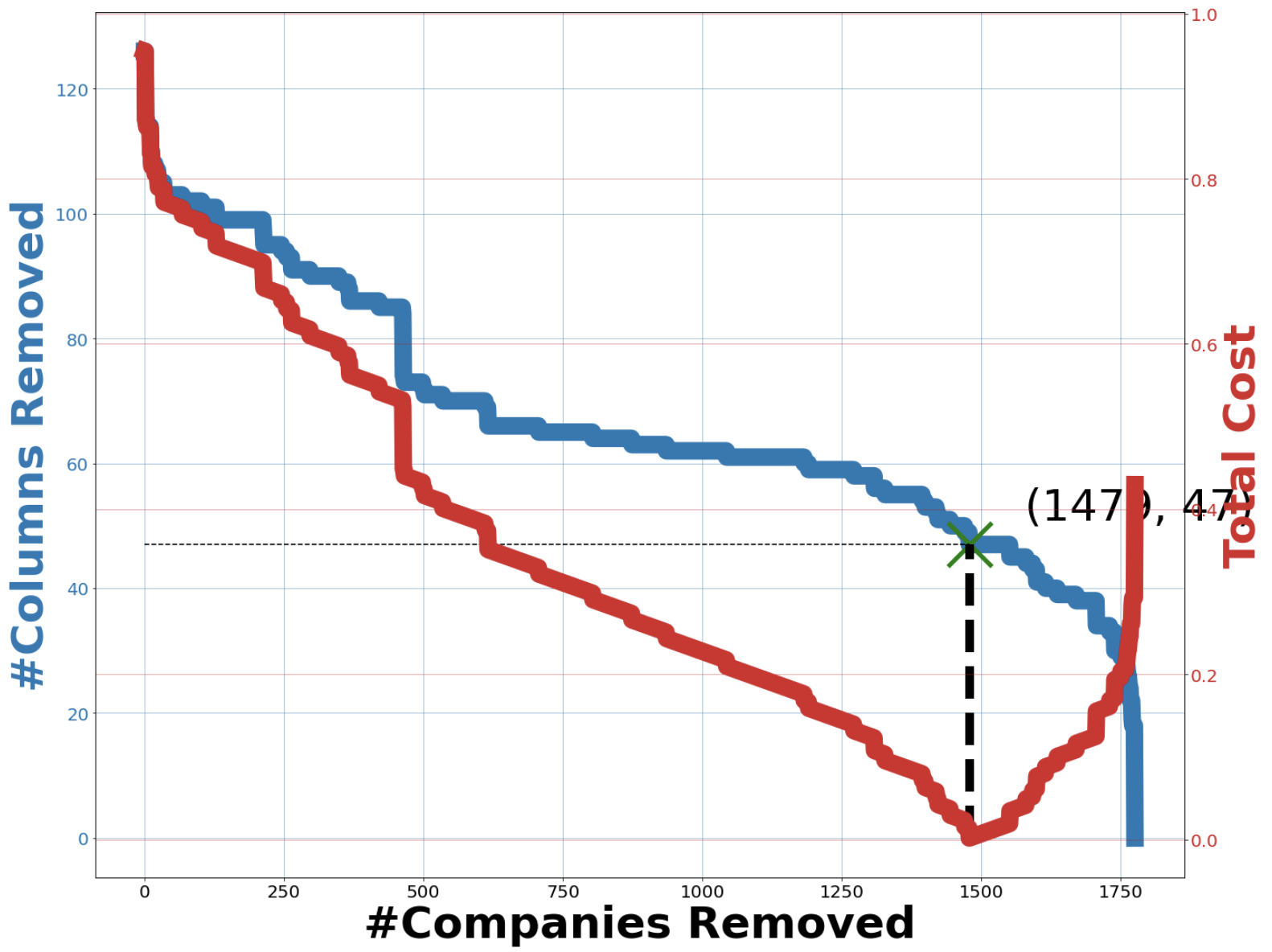
*Figure A4: H2 C2 for 1-month data*



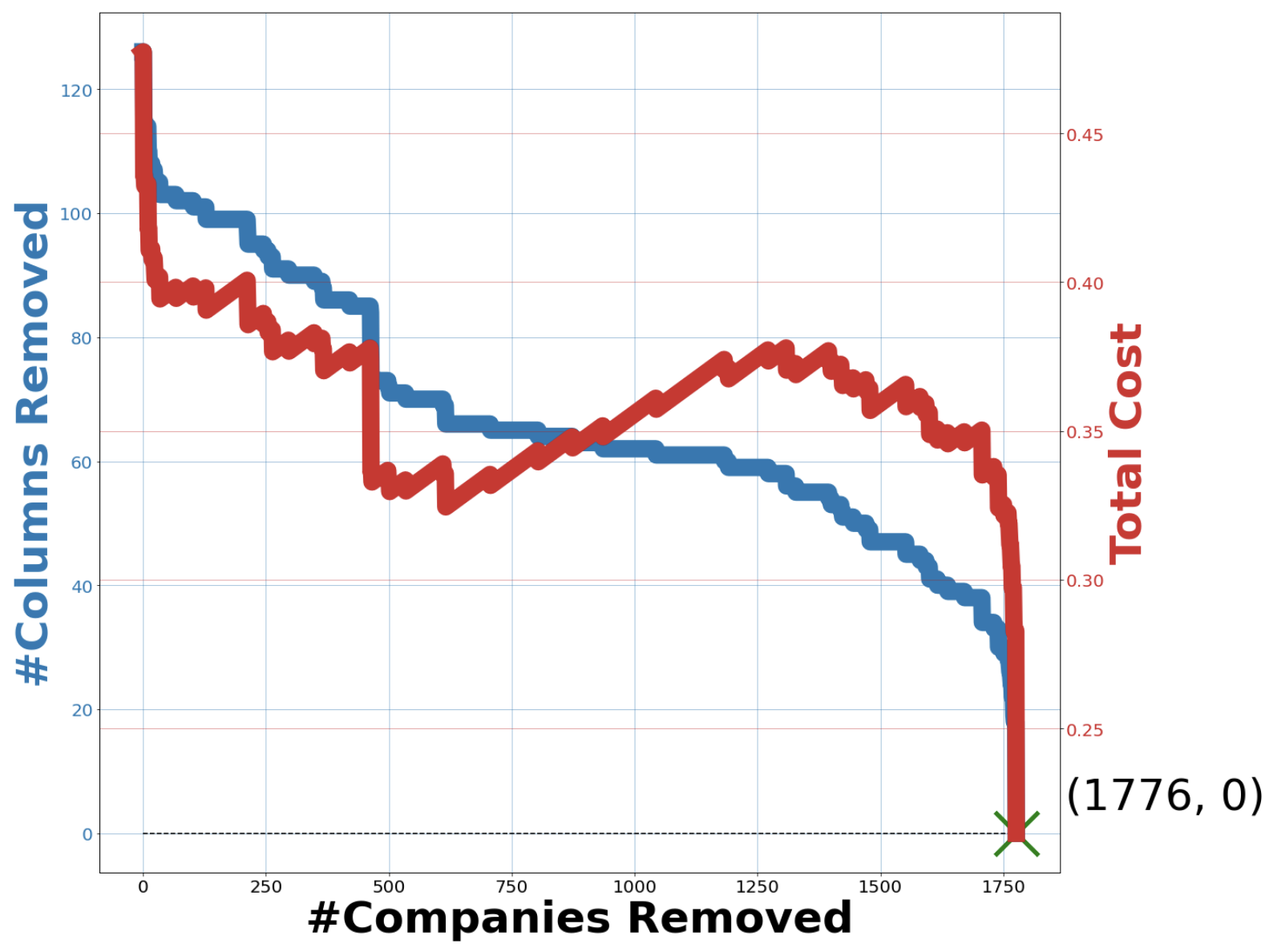
*Figure A5: H1 C1 for 6-month data*



*Figure A6: H1 C2 for 6-month data*



*Figure A7: H2 C1 for 6-month data*



*Figure A8: H2 C2 for 6-month data*

Ultimately we used the H2 C1 as the heuristic and cost function respectively, as it led to a reasonably low compromise in the minimum features required to be removed – in order to eliminate all companies and features having invalid data –, without having to remove an excessive number of companies, in both the cases for 1-month data and 6-month data.

### Appendix B: LSTM cross validation results

Rows highlighted in green are the best performing set of hyperparameters for each data set.

|  |  |  |  |
| --- | --- | --- | --- |
| Epochs | Batch size | Dropout rate | Sum of RMSE |
| 1 | 8 | 0.1 | 122.31 |
| 1 | 8 | 0.2 | 124.98 |
| 1 | 8 | 0.3 | 127.67 |
| 1 | 8 | 0.4 | 140.22 |
| 1 | 16 | 0.1 | 171.70 |
| 1 | 16 | 0.2 | 150.97 |
| 1 | 16 | 0.3 | 151.28 |
| 1 | 16 | 0.4 | 169.39 |
| 1 | 32 | 0.1 | 151.18 |
| 1 | 32 | 0.2 | 139.54 |
| 1 | 32 | 0.3 | 136.26 |
| 1 | 32 | 0.4 | 139.10 |
| 1 | 64 | 0.1 | 116.55 |
| 1 | 64 | 0.2 | 114.21 |
| 1 | 64 | 0.3 | 118.29 |
| 1 | 64 | 0.4 | 118.38 |

*Table B1. Average RMSE for hyperparameters in cross validation of LSTM for the 1-month data set*

|  |  |  |  |
| --- | --- | --- | --- |
| Epochs | Batch size | Dropout rate | Sum of RMSE |
| 1 | 8 | 0.1 | 45.97 |
| 1 | 8 | 0.2 | 42.50 |
| 1 | 8 | 0.3 | 43.80 |
| 1 | 8 | 0.4 | 43.64 |
| 1 | 16 | 0.1 | 40.64 |
| 1 | 16 | 0.2 | 40.24 |
| 1 | 16 | 0.3 | 40.42 |
| 1 | 16 | 0.4 | 39.90 |
| 1 | 32 | 0.1 | 40.96 |
| 1 | 32 | 0.2 | 39.79 |
| 1 | 32 | 0.3 | 40.05 |
| 1 | 32 | 0.4 | 40.05 |
| 1 | 64 | 0.1 | 42.12 |
| 1 | 64 | 0.2 | 40.09 |
| 1 | 64 | 0.3 | 39.64 |
| 1 | 64 | 0.4 | 40.03 |
| 2 | 8 | 0.1 | 66.50 |
| 2 | 8 | 0.2 | 65.78 |
| 2 | 8 | 0.3 | 68.51 |
| 2 | 8 | 0.4 | 63.42 |
| 2 | 16 | 0.1 | 45.17 |
| 2 | 16 | 0.2 | 42.30 |
| 2 | 16 | 0.3 | 42.80 |
| 2 | 16 | 0.4 | 43.03 |
| 2 | 32 | 0.1 | 43.90 |
| 2 | 32 | 0.2 | 43.27 |
| 2 | 32 | 0.3 | 43.77 |
| 2 | 32 | 0.4 | 43.07 |
| 2 | 64 | 0.1 | 42.81 |
| 2 | 64 | 0.2 | 45.46 |
| 2 | 64 | 0.3 | 42.68 |
| 2 | 64 | 0.4 | 43.54 |
| 3 | 8 | 0.1 | 64.84 |
| 3 | 8 | 0.2 | 63.15 |
| 3 | 8 | 0.3 | 63.16 |
| 3 | 8 | 0.4 | 64.25 |
| 3 | 16 | 0.1 | 54.93 |
| 3 | 16 | 0.2 | 57.22 |
| 3 | 16 | 0.3 | 54.59 |
| 3 | 16 | 0.4 | 56.20 |
| 3 | 32 | 0.1 | 60.41 |
| 3 | 32 | 0.2 | 56.47 |
| 3 | 32 | 0.3 | 55.44 |
| 3 | 32 | 0.4 | 55.04 |
| 3 | 64 | 0.1 | 58.86 |
| 3 | 64 | 0.2 | 59.34 |
| 3 | 64 | 0.3 | 56.68 |
| 3 | 64 | 0.4 | 57.26 |
| 4 | 8 | 0.1 | 47.00 |
| 4 | 8 | 0.2 | 47.93 |
| 4 | 8 | 0.3 | 50.45 |
| 4 | 8 | 0.4 | 51.52 |
| 4 | 16 | 0.1 | 68.76 |
| 4 | 16 | 0.2 | 70.12 |
| 4 | 16 | 0.3 | 69.27 |
| 4 | 16 | 0.4 | 67.74 |
| 4 | 32 | 0.1 | 71.89 |
| 4 | 32 | 0.2 | 71.03 |
| 4 | 32 | 0.3 | 67.71 |
| 4 | 32 | 0.4 | 66.84 |
| 4 | 64 | 0.1 | 69.92 |
| 4 | 64 | 0.2 | 69.09 |
| 4 | 64 | 0.3 | 67.42 |
| 4 | 64 | 0.4 | 68.62 |

*Table B2. Average RMSE for hyperparameters in cross validation of LSTM for the 6-months data set*

### 

1. MLR, preprocessing (financial ratios, labels, dates, quarters, removal of NaN / zeros, splitting of data into training-validation and test sets)):

   <https://github.com/whneo97/cs3244-2010-0005-mlr-stocks-prediction.git> [↑](#footnote-ref-0)
2. RNN: <https://github.com/ongjinglong/3244-2010-0005-RNN> [↑](#footnote-ref-1)
3. LSTM, preprocessing (labels, merging of fundamental and technical data): <https://github.com/teo-jun-xiong/lstm-stock-prediction> [↑](#footnote-ref-2)