FIE453

Big Data with Application in Finance

Final Project

"Measuring stock predictability: An investment decision tool"

Total number of pages: 21

Candidate numbers: 8, 15 and 54

Group number: 19

Autumn 2021

Table of contents

1. Introduction 3
2. Data 4
3. Pre-processing 4
4. Methods 6
    4.1 Sample splitting 6
    4.2 Performance metric MAE 7
    4.3 K-Nearest Neighbor (KNN) 7
    4.4 Bayesian ridge regression 8
    4.5 Stochastic Gradient Boosted Machine (GBM) 8
    4.6 Neural Networks 9
        4.6.1 Neural network training frameworks 10
        4.6.2 Optimizers 10
        4.6.3 Learning rate 12
        4.6.4 Batch size 12
        4.6.5 Epochs 12
        4.6.6 Hidden layers 12
        4.6.7 Measures against overfitting 13
    4.7 Reproducibility of results 13
5. Analysis and Findings 14
    5.1 Hyperparameters of models 14
    5.2 Feature importance 15
    5.3 Stock level performance 16
    5.4 Best performing neural network model 17
    5.5 Most predictable stocks 18
    5.6 Prediction performance of selected stocks in the evaluation period 19
    5.7 Framework use cases 21
6. Weaknesses 23
7. Conclusion 24
8. References 25
9. Appendix 29
    9.1 Feature Description 29
    9.2 Training process of two hidden layer neural network model 30
    9.3 Performance metrics of 30 stocks with highest predictability 31
    9.4 Financial indicators of 30 most predictable stocks 32
    9.5 Out-of-sample prediction of 30 most predictable stocks 33

1. Introduction

The efficient market hypothesis states that if all information is available, it should be priced in the stock (Fama, 1970). In reality the market response suffers from information inertia, due to information overload. It is almost impossible to retrieve all the useful information from available sources in the market and respond perfectly to it. Stock prices are commonly known to be hard to predict, since the price practically follows a random walk. However, it may be plausible that some stocks are easier to predict than others. Could identifying a pool of highly predictable stocks act as a tool for investors to construct portfolios, which subsequently enables them to perform well-informed short-term investment decisions? Following from this, we form the research question: *"Can a pool of highly predictable stocks serve as a basis for investment decisions?"*

To form this pool of stocks, we extract usable data from the merged data set of CRSP[1] and Compustat. Thereafter, we identify the most influential features on the dependent variable, RETX[2]. We try different machine learning models: K-Nearest Neighbor (KNN), Bayesian Ridge Regression, Stochastic Gradient Boosting Machine (GBM), as well as various neural network models. Using mean absolute error (MAE) as a performance metric, we find that the best performing model was the neural network, thus we opt to use this model for our RETX predictions. To establish an objective measure of our best performing model's performance, we compare it to a naive benchmark model that always predicts zero return. Our chosen neural network model significantly outperforms the benchmark model, thus it provides a good indication of the model's usability.

This paper will start with an overview of the initial data set, as well as a description of the reduced data after extensive pre-processing. We will give an explanation of our pre-processing approach and provide theory of models and metrics. Then, we will perform an analysis and provide findings of the most predictable stocks based on the best performing model. Finally, weaknesses of our choices will be discussed, before we wrap up with a brief conclusion at the end of the paper.

[1] CRSP - The Center for Research in Security Prices

[2] RETX - return without dividend which further in this paper will be referred to as 'returns'.

2. Data

We chose to use a merged data set containing both CRSP and Compustat data. CRSP consists of comprehensive proprietary stock market data, while Compustat contains financial, statistical and market information of publicly listed companies (CRSP, n.d.; Baker library, n.d). The data set has the following characteristics: 373 variables and 633,889 observations of 8,491 different companies identified as permno. It represents monthly observational data of companies' stock returns and other variables such as prices, market excess returns and other financial characteristics. The monthly observations range from 2011 to 2020. In this case study we want to predict RETX, which is monthly returns without dividend. This is due to the fact that the dividend pay-out date drives down the price, resulting in lower returns. Since the shareholder benefits from the dividend anyway, as long as the stock is bought before ex-dividend day, this should also be seen as a capital gain, we therefore use return without dividend RETX (Investor.gov, n.d).

3. Pre-processing

In order to predict the dependent variable, we need features that explain its variation. We will provide different pre-processing approaches used to retain features with higher quality in order to better explain the response variable and its implications. Features with missing data, near zero variance and high intercorrelation does not contribute to describing the dependent variable's variation and should be removed. Stock price and stock returns with dividends are causally related to the target feature and should be removed. Features comprising only missing data and zero values does not explain the dependent variable and we opt to remove it from the data set. As there are features containing both valuable information and different fractions of missing data, we introduce a missing data filtration retaining features that have less than 25 percent of missing data.

Two other commonly used feature reduction techniques are variance- and correlation-filtration. The purpose of the former filter removes features when the variance is close to zero. In other words, features containing only one unique value or features with an overrepresentation of a single value yields zero or near zero variance, respectively. In order to use the near zero variance filter we need to decide two parameters namely the frequency cut and unique cut point. Frequency cut is a cutoff ratio of the most common value relative to the second most common value (Kuhn,

2021). We chose to use 95/5, thus values with a higher frequency than this ratio will be removed. The second parameter that needs to be tuned is the unique cutoff, where we performed a 10 percent cutoff point. This means that features containing distinct values out of the number of total samples that yield a percentage above 10 percent are removed. The second filtration method, correlation filter, works by removing features that are highly correlated with other features (Kuhn et al., 2021). This is a pairwise comparison filter mechanism using the Pearson correlation method.

After performing extensive feature reduction, we need to consider the observations that have missing records[3]. Replacing missing feature-values could be done by simply using the mean or median of each observation series of each company, but this could probably introduce biases in the data. The missing value may be a simple error, a company's dissolution or the fact that some activa is traded less frequently. As we do not have a clear understanding of the cause of missing value, we opt to simply remove the observations containing any missing values in their recordings.

We would like our model to be used in any time period, thus we opt to remove features related to time, such as date and financial quarter. The feature, RET, shares the same property of calculating the change in return per month as our independent variable, RETX, thus we opt to remove this feature as well.

As there exists an infinite combination of filtration parameters the output of which features, and number of observations can vary greatly. Note that there may exist better options for feature reduction, but we choose to go further with these pre-preprocessing methods with our selection of parameters. Lastly, after conducting pre-processing and manually removing date features, stock returns and stock prices the dimensions are reduced to 29 features and 365,317 observations. A complete list of feature names can be found in the appendix 9.1

.

[3] Missing records denoted as NA (not available) in data

4. Methods

In this section we will provide a theoretical explanation of the chosen machine learning models: K-Nearest Neighbors, Bayesian Ridge Regression, Stochastic Gradient Boosting Machine, and Neural Network. The motivation for using neural networks models is based on Gu et al (2020) findings that neural networks outperform more traditional machine learning methods in predicting returns. With this in mind, it is interesting to examine if neural networks are able to outperform more simple models such as KNN and Bayesian ridge regression, but also more sophisticated models such as Stochastic Gradient Boosting Machines.

We will also present different parameter tuning options for each respective model. The problem at hand is to find a pool of stocks with the highest predictability. In order to do this, we have to separate the data in order to not introduce any information leakage between periods in time. We opt to split the data into a selection and evaluation period, where we pick the best performing model by training the models on a selected period from 2011 up until December 2017. We use 2018 and 2019 data as our designated evaluation period, where we will test the efficacy of our predictive model in a separate time period from which the model is trained. The general overview of data splits can be seen in figure 1.

The motivation behind this case of study is to find stocks with highest predictability, because predictable stocks can potentially introduce an advantage and guidance on where and how to put money in order to increase fortune. If it is possible to have a certain degree of confidence to correctly predict the outcome of stock return, an investor can utilize buy and short strategy if the stock moves in positive or negative direction, respectively. We have opted for a regression approach rather than a classifier predicting positive or negative returns in order to deduce the magnitude of the returns.

4.1 Sample splitting

In order to evaluate the performance of the models we split the data into train and test, where train data comprises both a train and validation set. We opt to use a fixed split of train, validation and test data of 60, 20 and 20 percent of the total dataset, respectively. This can be seen in figure 1.

Note that train, validation and test are not split in different time periods but consists of randomly sampled stocks within the selection period from 2011 to 2017. A cross validation scheme could have been used, but due to the size of the dataset it is likely that our validation estimates are reasonably unbiased using a fixed split. Furthermore, cross validation adds complexity to the training process as each model would have to be trained on a large number of training samples.

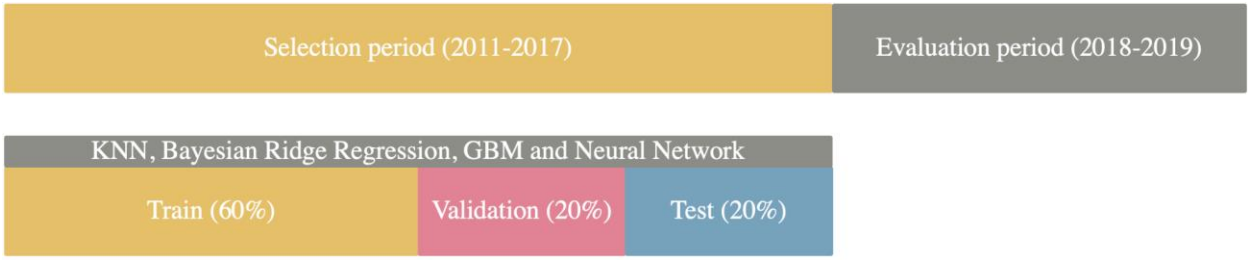


Figure 1: Data splits of selection and evaluation periods

4.2 Performance metric MAE

A common metric is needed to evaluate the predictive power of our respective machine learning models. There exists a vast variety of different performance metrics for this purpose, where the most common ones are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). We opt to go further with MAE, due to the fact that RMSE could be misleading and not easily interpreted because it is a function of 3 characteristics of error rather than the unambiguous measure of average error (Willmott and Matsuura, 2005). MAE is calculated by taking the average of the sum of absolute difference between the prediction and the true observed value, as shown in the following equation.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

Figure 2: Definition of mean absolute error

4.3 K-Nearest Neighbor (KNN)

K-nearest neighbors (KNN) is a supervised machine learning algorithm suitable for both classification and regression tasks (Christopher, 2021). In the case of regression, the KNN algorithm attempts to predict new data points by calculating the mean of the k closest surrounding

data points of the training set. The distance between the prediction point and the existing points is found by calculating the Euclidean distance, which is then ranked by increasing distance. Choosing the value of k is essential, as it heavily influences the prediction. A small k value will increase the chance of error and inaccuracy as the algorithm considers fewer data points. A higher k value will therefore decrease the risk of error as well as reducing the effect of potential outliers, but also increase the computational cost of the algorithm. Furthermore, the KNN-model is inherently sensitive to the quality and scale of the data, as all distances are weighed equally, which might reduce its predictive power.

We opt to use a tune length of 20 for our KNN-model. This parameter tells the algorithm to do a random search of 20 different values for k, then the value of k with the lowest value of validation MAE, is selected. By utilizing the tune length, we are able to tune the model's parameters to optimize the model's performance, as well as avoiding parameter-selection biases.

4.4 Bayesian ridge regression

Ridge regression is supervised machine learning through linear regression (Bhattacharyya, 2018). The ridge regression model utilizes regularization in order to manage the bias-variance trade-off which occurs in big data sets with multiple features (Pohl, 2021). This is done by introducing a penalty term to penalize high complexity. Ridge regression aims to minimize this penalty by pushing coefficients towards zero, commonly called shrinkage, in order to reduce the risk of overfitting which could in turn increase the model's predictive power. This introduces a higher bias, however, in machine learning a higher bias can be tolerated to minimize the variance. Contrary, Bayesian ridge regression, utilizes techniques to include regularization parameters in the estimation procedure (Pedregosa et al, 2011). Its uninformative priors are used to assign values to the hyperparameters based on random distributions, such as normal distribution.

4.5 Stochastic Gradient Boosted Machine (GBM)

The idea behind the boosting models is to improve the performance of weak learner models, which are models performing slightly better than by random chance, by adding new weak learners (Singh, 2018). Machine learning boosting is the method for creating such an ensemble, where Gradient

Boosting Machines (GBM) are one of the most renowned. GBM is based on random trees, where the initial decision tree assigns equal weight to each observation in the training set. The performance of the first decision tree is then evaluated. The weights of the observations on the next tree are then modified in accordance with what was found on the previous tree. Specifically, the weights of the observations which are difficult to classify are increased while the others are decreased. The gradient boosting machine evaluates the shortcomings of a weak learner by measuring the gradients in the loss function. The loss function gives an indication of how well the model's coefficients fit the underlying data. In order to minimize the identified loss of the model the GBM will continue adding weak learner models using the method of gradient descent. We opt to use a variation of gradient boosting called stochastic gradient boosting. This allows trees to be greedily created from subsamples of the training data, which can be utilized to reduce the correlation between the trees in the sequence (Brownlee, 2016).

We have introduced a tune length of 25, which randomly searches for 25 different values for our hyperparameters. These hyperparameters will for instance restrict the number of boosting iterations, which limits the amount of decision trees added to the model (Singh, 2018). By increasing the number of trees, the error on the training data set will be reduced, however the risk of overfitting increases as well. The interaction depth of the decision trees is also defined, which is the maximum number of splits that can be performed, thus limiting the number of nodes per tree. A shrinkage parameter is also defined. This parameter reduces the impact of each additional tree fitted to the model, thus reducing the importance of new trees per iteration. Lastly, we set a minimum terminal node size, controlling the minimum number of observations in the trees' terminal nodes. Gradient boosting machines are sensitive to its parameters, thus by fine-tuning the parameter we might be able to reduce the risk of overfitting and increase the overall predictive power of the model.

4.6 Neural Networks

Artificial neural network is a machine learning method that has been brought to particular attention in the last decades. Garnering much attention through papers such as Honrik's proof in 1989 that neural networks of only one hidden layer could be universal approximators (Honrik, 1989).

Universal approximation implies that the model is able to mimic any function, sementing a theoretical foundation for applying neural networks to a wide array of problems.

All neural networks used in this report is a feedforward type of neural network, where input features are passed to a number of neurons each containing a value which is a function of its input value and a neuron weight. In its most simple configuration, the weighted values are passed to an output layer and a prediction is made based on the sum product of all neurons over a threshold value. More complex networks pass their input through hidden layers, where the weights are recalculated based on backpropagation. The weights are updated "backwards" after calculating new weights based on the model's loss function. Gradient descent is a common method of recalculating weights of a neural network model after completing a single learning sequence. The weights of the neural network model are updated according to a loss function, initially based on the entire training sample. This approach is called Batch Gradient Descent (Ruder, 2016).

The number of configurations possibilities are vast, and an exhaustive trial and error of all possible hyperparameters is infeasible. As a result, we have implemented a grid search of a small subset of hyperparameters.

4.6.1 Neural network training frameworks

The Caret package provides the opportunity for automatic tuning through random search, which allows for a simple prototype of the neural network model. Providing a decent result, we further enhance the model performance by implementing a more comprehensive model from the Keras for package. This is built on Python's machine learning framework, Tensorflow, which is a deep learning model allowing for flexible tuning of size and depth. In addition, Tensorflow provides support for GPU-processing which greatly increases training speed. In practical terms, this added training speed allows us to test a greater set of hyperparameters, which increases the likelihood of picking a more generalizable model.

4.6.2 Optimizers

Adjusting all parameters on all samples of training data is computationally taxing and might even be prohibitive for large datasets. As a result, multiple approaches for reducing the search area of

the learning process have been developed. Learning after each instance of new training data is referred to as Stochastic Gradient Descent (SGD) (Ng, 2012), and greatly reduces workload. For greater datasets even this approach is too slow, and so-called adaptive optimizers have become the norm in recent years.

The approaches taken in these techniques vary greatly, but they have in common that they adjust learning rates according to each weight and the occurrence of the features associated with those weights. In essence this allows us to narrow down the search space, while minimizing the risk of getting stuck in local optima as is the case with SGD. In this report we have utilized a common adaptive learning technique ADAM. In the original paper by Kingma & Ba (2015), they show that ADAM is a good compromise between minimizing generalization error and maximizing training speed.

Several papers have highlighted that SGD, although generally slower, achieves better performance on unseen data. In particular, a paper by Keskar & Socher (2017) finds that although adaptive methods are quick to converge, in latter parts of training the feature related weights are overestimated and leads to worse generalization error when compared to SGD. Training for a total of at most 216 hyperparameter combinations for a variety of neural network models, makes the added training time posed by SGD an unreasonable tradeoff. We deem it more prudent to use the added training speed provided to us by adaptive methods such as ADAM to train models on a wider range of hyperparameters.

We adopt ADAM with different initial learning rates ranging from 0.1 to 0.4. The reader should note that these are regarded as very high initial learning rates. We implement a learning rate schedule that decreases learning rate after a certain number of epochs of no validation error improvement. The number of epochs the model waits before terminating is called *patience*. In addition to the natively adapting learning rate in ADAM, this approach minimizes the risk of getting stuck in local optima.

4.6.3 Learning rate

The learning rate controls how much the neural network model should respond to the estimated error when weights are updated (Brownlee, 2019). The difficult part is to find the optimal learning rate, because it is a trade-off between training process runtime and convergence probability. A high learning rate typically leads to faster learning, but risks missing local optimum. On the other hand, a small learning rate increases convergence time and increases the risk of the model getting stuck in a local optimum. This is widely considered to be the most important hyperparameter in the configuration of a neural network model.

4.6.4 Batch size

In the training of neural networks, especially on large datasets, large amounts of memory are needed to perform training on the entirety of our training dataset. Restricting the number of observations exposed to the model at a time reduces the severity of this problem. Lowering the batch size allows for greater specificity in our training, while sacrificing training time and incurring greater risks of overfitting. For all neural network models tested in this report, we tune our models based on a variety of batch sizes.

4.6.5 Epochs

In neural network terminology an epoch is denoted as the number of times the entire training dataset has been exposed to the neural network model. We adopt a learning rate schedule which decreases learning rate when the model is assumed to a local optimum, i.e, no improvement of validation error has been made for a number of epochs (patience). In addition, we adopt early stopping which terminates the training process after learning stalls. As a result, we frequently observe that learning stops anywhere between 20 and 100 epochs, depending on the complexity of the neural network. We opt to set the maximum number of epochs to 200 for all models, allowing the training to “automatically” determine the optimal number of epochs.

4.6.6 Hidden layers

A neural network consists of several layers where some or all of its neurons are connected. In this paper we will test four different neural network models ranging from one hidden layer to four hidden layers. The number of neurons in each layer follows the geometric pyramid rule, originally

proposed by Master (1993) and implemented by Gu et al (2020). For instance, the four-layered model has 32, 16, 8 and 4 neurons in the first, second, third and fourth hidden layer, respectively.

4.6.7 Measures against overfitting

As the complexity of neural networks increases, we risk overtraining or "overfitting" our models. This refers to the lack of generalizability predictive models suffer from when too closely mimicking the nuances found in the training set from which the model derives its parameters. To combat this, primarily two measures are tested in this report, namely early stopping and dropout layer. Early stopping entails limiting learning rate when the speed of the change in model weights slows down. If validation error does not decrease, the model terminates after a preset number of epochs (patience). Early stopping has been described as "Free Lunch", implying that its performance increase suffers no penalty (Hinton et al, 2015). A dropout layer is a more aggressive measure against overfitting. This probabilistic method checks the relevance of a neuron by randomly ignoring a share of previously active neurons. In practice this involves setting the elected neuron's weight to zero.

4.7 Reproducibility of results

The reader should note that GPU-processing cannot be exactly reproduced. To recreate our model outputs exactly, the models have to be preloaded. This is a drawback of using GPU-powered model training, but the difference in model results should be minimal. The reader should be aware that this is only the case for neural networks models using GPU-processing.

5. Analysis and Findings

In this section of our report, we will present the results of the models and the optimal parameters as determined by mean absolute error minimization. Then, we will choose the optimal model and most predictable stocks by MAE, which then can be used to construct a pool of predictable stocks that can serve as a basis for multiple possible portfolios. KNN, Bayesian Ridge Regression and GBM are optimized using random search, while the neural network model is tuned using grid search of a subset of hyperparameters.

5.1 Hyperparameters of models

The optimal KNN-model in the evaluation process yields a parameter k of 43 after tuning the model for 20 randomly selected k numbers of neighbors. Bayesian ridge regression is not explicitly tuned for any specific parameters, but uses a total of 1000 collected Markov Chain Monte Carlo (MCMC) samples and skips 29 MCMC samples referred to as thin. For the GBM-model we opt to use 25 random combinations of parameters in tuning, and the best performing parameter combination for this model is 50 numbers of trees, using 18 splits in the interaction depth, with 0.1 shrinkage and at least 10 observations in the terminal node.

The neural network models with one to four hidden layers and neurons that follow the geometric pyramid rule tested on unseen validation data yielding a total of 45 combinations (1 hidden layer), and 162 combinations (2, 3 and 4 hidden layers). The table in figure 3 shows the tested combinations of hyperparameters for each neural network model. We should again emphasize that this is not an exhaustive list of possible hyperparameters, which would be inconceivable, but rather a subset of hyperparameters deemed most relevant for achieving the best model performance. All tested models have their input normalized, and it should be emphasized that batch normalization in the table in figure 3 specifically refers to normalization performed between hidden layers.

Tested Hyperparameters of Neural Network models

Neural network models	Learning rates	Batch normalization	Dropout rates	Batch sizes	Maximum epochs	Patience
One hidden layer	0.005, 0.100, 0.400	No	0	300, 500, 1000	200	1, 2, 10, 5, 15
Two hidden layers	0.005, 0.100, 0.400	Yes, No	0.0, 0.2, 0.4	500, 1000, 2000	200	1, 5, 20
Three hidden layers	0.005, 0.100, 0.400	Yes, No	0.0, 0.2, 0.4	500, 1000, 2000	200	1, 5, 20
Four hiden layers	0.005, 0.100, 0.400	Yes, No	0.0, 0.2, 0.4	500, 1000, 2000	200	1, 5, 20

Figure 3: All tested hyperparameters of neural network models

5.2 Feature importance

Feature importance is a technique to rank all the features based on a score, where a high score refers to high importance on the dependent variable (Shin, 2021). This technique helps to improve the understanding of the data by identifying the features' concrete relationship with the dependent variable and which features are of lesser importance. To showcase the reduction in features from the initial dataset, we opt to include the identified feature selection from one of our models, namely the Stochastic Gradient Boosting Machine. This model identifies 29 features that have a significant relationship with the dependent variable, RETX. Figure 4 illustrates the first 20 of 29 features identified with the GBM model. It finds the features v*wretd*, *vol* and *shrout* to be of especially high importance, while the remaining identified features, albeit much lower than the aforementioned features, have a sufficiently high importance on RETX's variance. The amount of influence only a few features exert on the dependent variable, highlights the importance of feature selection. Given additional variables, the relative importance of each variable might change dramatically. A full list of descriptions of the abbreviations can be found in appendix 9.1**.**

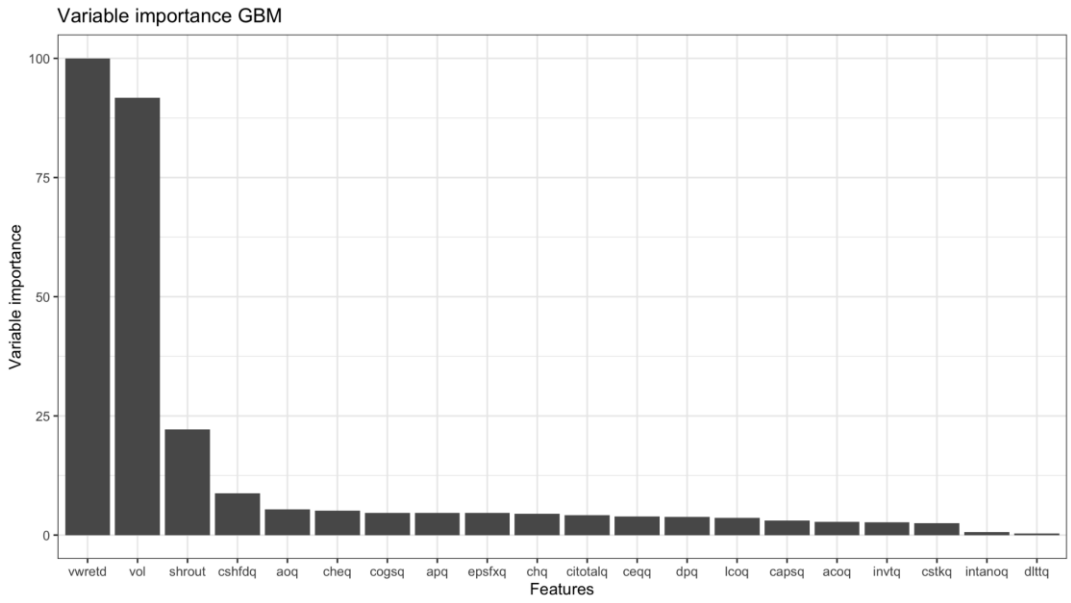


Figure 4:Influential variables as determined by the GBM model

5.3 Stock level performance

As the approach proposed in this report involves identifying a pool of the most predictable stocks, the predictive performance of our model should be evaluated based on error minimization of a pool of stocks rather than the test error of all observations. It is reasonable to assume that our models perform differently depending on the stock in question, and findings based on our models does suggest this. The reader can verify this by looking at the spread in the 50 most predictable stocks compared to the least predictable stocks as seen in the table in figure 5. To test our hypothesis an ensemble predictor is made from taking the average of test MAE scores for each stock. We find that the difference in MAE between the most predictable stocks and the least predictable stocks to be substantial.

Variation in Stock Predictability: Ensemble Model Predictions

Stocks	RMSE	MAE
Mean perfomance of 50 most predictable stocks	0.0568	0.0439
Mean perfomance of 50 least predictable stocks	0.3274	0.2092

Figure 5: Variation in stock predictability

Using our validated models, four Neural Network models, KNN, Bayesian Ridge Regression and Stochastic GBM we construct a ranked pool of predictable stocks. Our goal is to create a pool of readily predictable stocks for potential investors to construct portfolios. In the literature portfolio sizes vary vastly, but according to Tang (2013) variance reduction is greatest after 10 stocks. A selection of 30 stocks thus seems adequate to be able to construct a multitude of potential portfolios.

For each model we identify its 30 most predictable stocks and calculate the mean absolute error of the respective stock-level test-set predictions. Our preference should be reserved for the model which performs the best at predicting the returns of its most predictable stocks. From the table in figure 6 the reader will see comparisons of four neural network models, as well as KNN, GBM and bayesian ridge regression models. We compare it to a naïve predictor that always predicts zero return for every observation. As stock returns are stationary with a mean of zero, always predicting zero will likely yield good results.

Model performance on 30 most predictable stocks

Model name	RMSE top stocks	MAE top stocks
Neural Network 2 hidden layers	0.0405	0.0310
Neural Network 3 hidden layers	0.0411	0.0314
Neural Network 1 hidden layer	0.0410	0.0315
Neural Network 4 hidden layers	0.0454	0.0350
K-nearest neighbors	0.0447	0.0351
Gradient Boosting machine	0.0590	0.0459
Bayesian ridge regression	0.0673	0.0521
Zero prediction naive model	0.1300	0.0934

Figure 6: Model performance on 30 most predictable stocks as determined by each model

From table 6 it is evident that all tested models outperform the naïve predictor, providing ample support for our models. It surprised the authors of this report that a simple K-nearest neighbors model was able to outperform much more complex models. This highlights the perils of complexity, and that simpler models often outperform more parameter-rich counterparts. The neural networks performed well overall, with relatively small performance variations, and we observe that the two-layer model is the best performing in terms of stock-level predictions. The two-layer model allows for a trade-off between complexity and generalizability. We will use this model as a basis for our approach in the remainder of this report.

5.4 Best performing neural network model

The neural network models described in this report have been tested with a wide range of hyperparameter combinations. In figure 7 the reader will find the best performing hyperparameters of the neural network model which fared the best on stock-level test-set predictions.

Best Performing Neural Network Hyperparameters

Number of layers	2
Learning rates	0.005
Batch normalization	No
Dropout rates	0
Batch sizes	500
Patience	20

Figure 7: Hyperparameters of the best performing neural network model

The optimal learning rate is surprisingly low considering the aggressive learning rate decay imposed by ADAM and our implemented learning rate schedule. In the appendix 9.2 the reader will find a plot of the evolution of training and validation errors as learning proceeds with varying learning rates, as described in the methodology section of this report. We also observe that no batch normalization is performed after the activation function, as the reader can discern from figure 8. This means that the neuron weights have not strayed sufficiently from their initial distribution. Considering that this model only has two neural layers, it is not surprising that normalization between hidden layers is not necessary. No dropout layers are added to the model, which is sensible considering that the complexity of a two-layer model might not warrant aggressive regularization techniques. A relatively small batch size of 500 is found optimal.

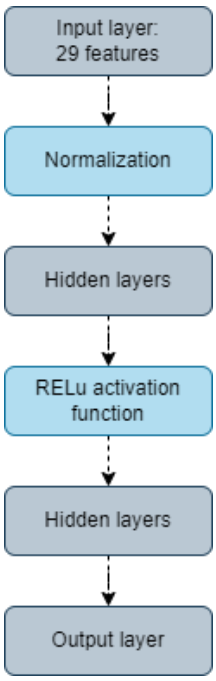


Figure 8: Visualization of layers of the best performing neural network model

5.5 Most predictable stocks

Using the two-layer neural network model we find the 30 most predictable stocks. From the tables in figures 9 and 10, it is evident that the most predictable stocks have some inherent similarities. For instance, we find that the average market capitalization is significantly higher than the average of all companies in the test set, which implies that our approach favors larger companies. Our model identifies companies with significantly higher trading volume, operating income and cash. The standard deviations of returns are calculated, which in this case is used as a proxy for a stock's volatility. It is apparent that the top 30 most predictable stocks have a notably lower dispersion of returns.

Mean financial indicators of all test companies

Mean market cap	Mean volume	Mean cash	Mean operating income	Standard deviation of returns
8139246	287530	1009	202	14.900%

Figure 9: Mean financial indicators of all test companies

Mean financial indicators of 30 most predictable stocks

Mean market cap	Mean volume	Mean cash	Mean operating income	Standard devation of returns
54284997	733817	5142	1255	5.134%

Figure 10: Mean financial indicators of most predictable stocks as determined by the best performing neural network model

A lower standard deviation of returns suggests a lower variability in returns, thus increasing their predictability. It makes sense that all else being equal, less variance yields easier predictions. This is likely another manifestation of our model's preference for large, high-grossing companies. These companies might be more predictable, but it restricts the possible scope of stocks that our model may provide insights into.

5.6 Prediction performance of selected stocks in the evaluation period

In this section of the report, we will evaluate our model based on out-of-sample predictions in a temporal distinct period from our training and validation period. A visualization of the evaluation period is shown in figure 1 in our paper. The goal is to ascertain whether the predictability of a group of stocks follows from one period to another. This is a key assumption in our approach and the evaluation period can thus be seen as a proving ground in this regard. The central question is: "*Will a stock that is characterized as predictable, remain predictable for the next two-year period?"* The evaluation period spans the whole period of 2018 and 2019, and predictions will be made using the best performing two hidden-layer neural network.

Performance Metrics in Evaluation Period of Two Hidden-Layer Neural Network

Model	Evaluation RMSE	Evaluaton MAE
Two hidden-layer neural network	0.0520	0.0393
Naive 0-benchmark	0.0652	0.0500

Figure 11: Performance metrics in evaluation period

Calculating MAE for our evaluation period compared to our zero-prediction benchmark, the results are evident: Our two hidden-layer neural network model provides more information than the zero-prediction benchmark. From the table in figure 11, the reader will note that performance has deteriorated when compared to selection period MAE. For out-of-sample predictions this is as expected, highlighting that predictability of stocks likely varies across different time periods. A MAE of 3.9 percent is still quite high considering that monthly stock returns rarely surpass 7

percent returns. However, as is frequently the case in finance, a model that is right in aggregate more than it is wrong may still be useful.

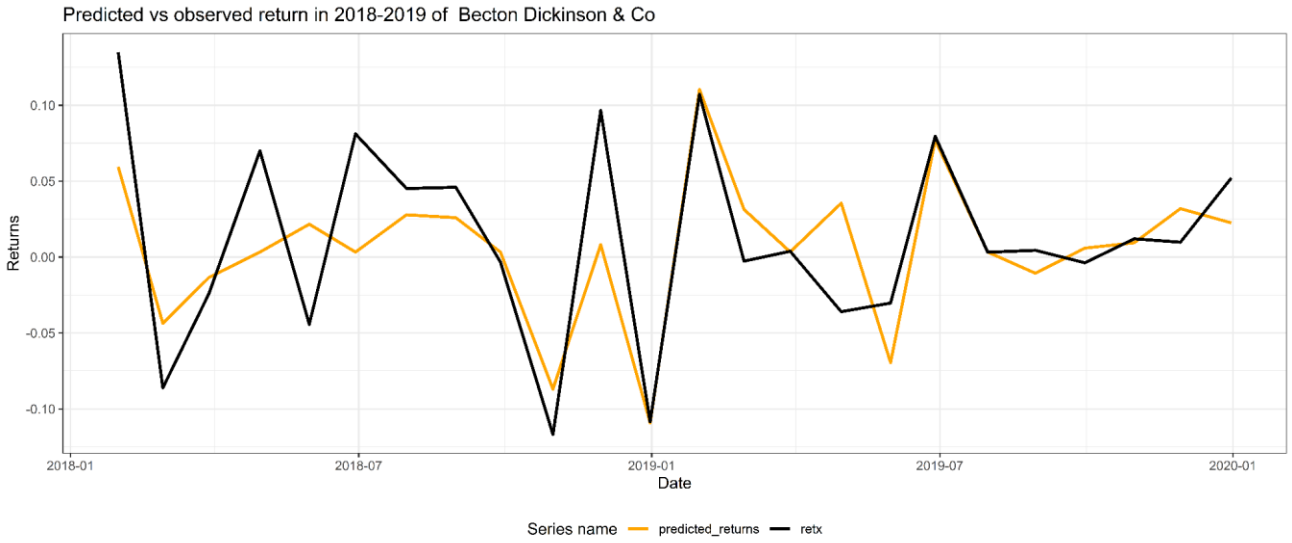


Figure 12: Plot of predicted versus observed returns of Becton Dickinson & Co in the evaluation period

From figure 12 the reader will find a plot of the most predictable company, Becton Dickinson & Co, as determined by our model. It is evident that the discrepancy between predicted returns and observed returns in the evaluation period is small. Notably, there are few months where predictions and observed returns have a sign-difference.

The same general trend can be seen for the next five most predictable stocks, as seen in figure 13. A full collection of evaluation data predictions can be found in appendix 9.5. The stocks deemed predictable in the selection period, are largely predictable in the evaluation period. For Johnson & Johnson, for instance, there seems to be a clear divergence, where the predictions do not match the sign of observed returns. The degree to which stock predictability remains after the training period will likely vary from stock to stock, as the figure 13 might suggest.

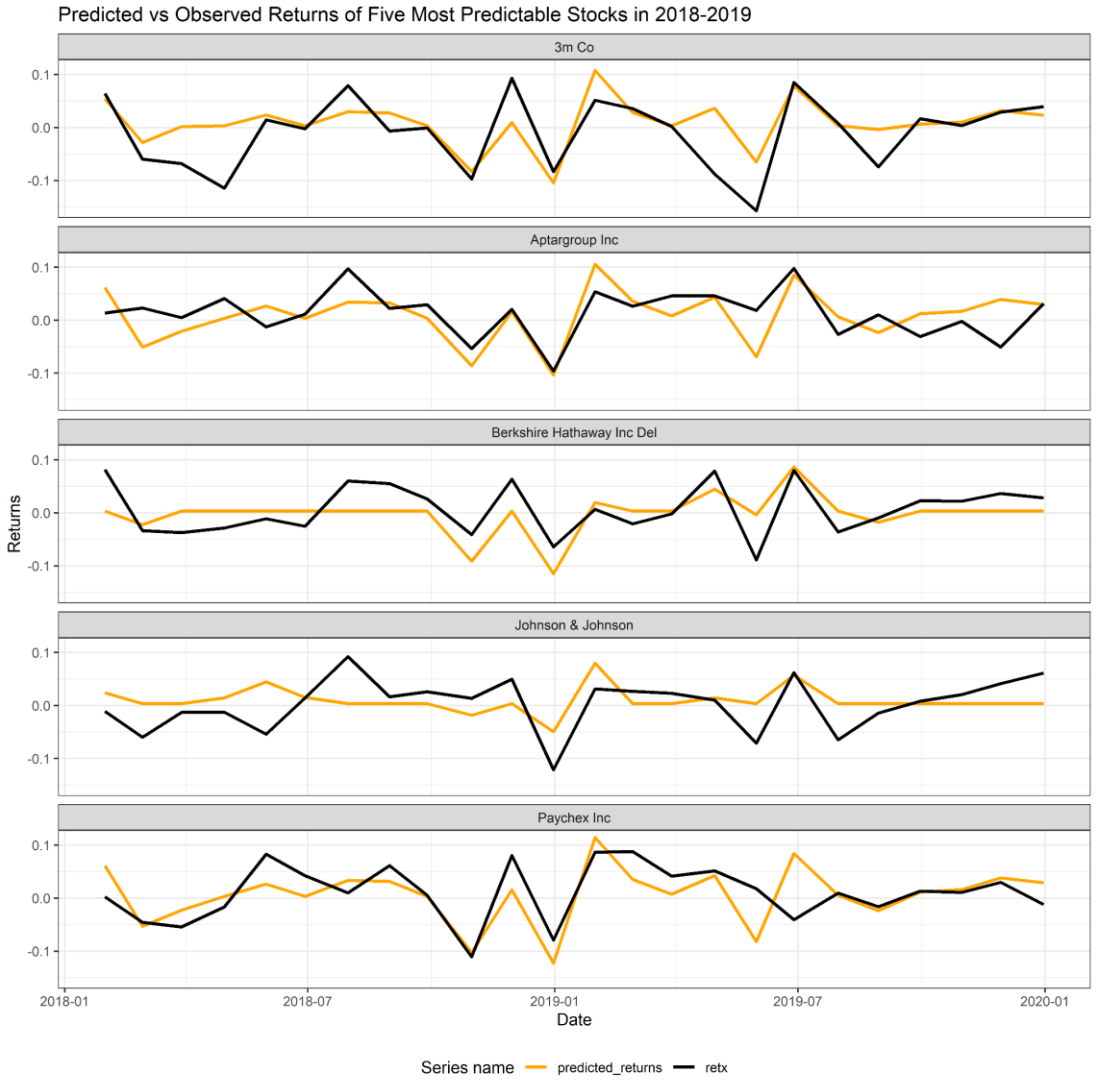


Figure 13: Predicted vs observed returns in evaluation period

Our predictions in the evaluation period are made without retraining the model. Using new information on a model trained in a different time period is likely to result in performance degradation, as we have seen. We still find that our model results transfer into a new period, highlighting the potential usefulness of our model in explaining returns. In the next section of this report, we will explore potential use cases of our framework.

5.7 Framework use cases

As we have seen, stocks carry their predictability when conducting out-of-sample predictions in a new time period. Given the availability of feature information that can be passed to our predictive

models, the framework presented in this report may provide value for investors as buying or selling signals. The predictions of our model can be thought of as indications of return given a set of observable traits which a stock displays at any given point.

The stocks identified as most predictable, in our selection period from 2011 to 2017, can be seen as a subset of stocks that may constitute a portfolio. Depending on the investor's risk-variance preference, as well as inclinations towards or against any economic sectors, stocks can be picked on either a long or short market exposure. Initial stock allocation may thus depend on either a predicted negative return, warranting a short position, or a predicted positive return, warranting a long position. When new stock information is available, new features provide new predictions from the model, allowing the investor to either remain with the initial portfolio or reallocate based on the pool of predictable stocks. Our framework can be seen as an information digesting tool that allows investors to extract buying or selling signals.

"Our framework can be seen as an information digesting tool that allows investors to extract buying or selling signals"

As our model is trained and evaluated on monthly data, it follows naturally that the updates of the models should occur on at least a near-monthly basis. Any deviation from the temporal span in which the model is trained, might weaken the trustworthiness of our model's predictions. This implies that the model cannot be used long-term without portfolio reallocation or changing the weights of individual stocks in the portfolio. As such our framework can either be used by short-term investors, or longer-term investors as a buying or selling signal for portfolio reweighting or reallocation.

An advantage of a regression approach rather than a classifier of returns is that investors can deduce the magnitude of returns. A prediction of slight positive returns might not justify the transaction costs or opportunity costs incurred by investing in the stock, while a regression approach thus might provide more information to the investor.

6. Weaknesses

The degree to which individual features explain changes in returns vary greatly. In section 5.2 we saw that the volume and market excess returns are among the most important features. More influential features however may be unaccounted for. Research by Han et al (2018), finds that stock price momentum is among the most important determinants of stock returns, a feature which we have not included in our models. It is likely that altering the composition of features in our models, might improve the performance of our predictions.

To be able to make predictions using our model, a total of 29 features needs to be available to the potential investor. For professional investors this seems a reasonable assumption, however some of the data represented in our features are updated infrequently. This poses a potential problem as any outdated feature values may give false predictions, leading to poor investment decisions.

The training and evaluation of machine learning models are a computationally expensive process, and the computational power available to us restricts both the scope of models and the range of hyperparameters tested in this report.

We have remarked, as the reader will know from section 5.6, that it is reasonable that the predictive models should be retrained when presented to new data. Due to computational resource constraints, we have trained our chosen model on the selection and assumed continued predictability of our selection of stocks. This may not be a realistic use of our model, and in a production setting the model could be retrained to better fit changes in the predictability of stocks after new data has emerged.

7. Conclusion

The large amounts of financial data available can make investment decisions difficult. Our goal has been to create an investment decision tool based on identifying the most predictable stocks. A wide range of machine learning models have been trained and evaluated for this purpose. We find that neural networks perform the best in predicting stock-level returns. Our two hidden-layer neural network is used to construct a pool of stocks deemed most predictable, from which portfolios can be constructed by potential investors with different risk profiles. These companies are generally large, profitable and less volatile than the average test company, leaving us with a somewhat homogenous pool of companies. Performing out-of-sample predictions in the evaluation period we find that our model is able to predict returns, although with a lesser degree of certainty. Stocks deemed predictable, generally stay predictable in our evaluation period, suggesting that our model predictions may be used as a buying or selling signal.

8. References

Bhattacharyya, S. (2018, September 26). *Ridge and Lasso Regression: L1 and L2 Regularization*. Towards Data Science. Retrieved December 11, 2021, from https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b

Brownlee, J. (2016, September 9). *A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning*. Machine Learning Mastery. Retrieved December 11, 2021, from https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

Brownlee, J. (2019, January 25). *Understand the Impact of Learning Rate on Neural Network Performance*. Machine Learning Mastery. Retrieved December 13, 2021, from https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

Christopher, A. (2021, February 2). *K-Nearest Neighbor. A complete explanation of K-NN*. Medium. Retrieved December 11, 2021, from https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4

CRSP (n.d.). *Why CRSP?*. Retrieved December 14, 2021, from https://www.crsp.org/main-menu/why-crsp

Baker Library. (n.d.). Retrieved December 14, 2021, from https://www.library.hbs.edu/find/databases/compustat

Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, *25*(2).

Gu, S., Kelly, B., & Xiu, D. (2020). *Empirical Asset Pricing via Machine Learning* (Vol. 33). The Review of Financial Studies. https://doi.org/10.1093/rfs/hhaa009

Han, Y., He, A., Rapach, D. E., & Zhou, G. (2018). What Firm Characteristics Drive US Stock Returns. https://economics.indiana.edu/documents/workshop-papers/fall2018/what-firm-characteristics-drive-us-stock-returns.pdf

Hinton, G., Bengio, Y., & LeCun, Y. (2015). *Canadian Institute for Advanced Research*. http://www.iro.umontreal.ca/~bengioy/talks/DL-Tutorial-NIPS2015.pdf

Honrik, K. (1989). Multilayer Feedforward Networks are Universal Approximators. *Neural Networks, 2*, 359-366. https://cognitivemedium.com/magic_paper/assets/Hornik.pdf

Investor.gov. (n.d.). *Ex-Dividend Dates: When Are You Entitled to Stock and Cash Dividends*. Investor.gov. Retrieved December 12, 2021, from https://www.investor.gov/introduction-investing/investing-basics/glossary/ex-dividend-dates-when-are-you-entitled-stock-and

Keskar, N. S., & Socher, R. (2017). Improving Generalization Performance by Switching from Adam to SGD. https://arxiv.org/pdf/1712.07628.pdf

Kingma, D. P., & Ba, J. L. (2015). Adam: A Method for Stochastic Optimization. https://arxiv.org/pdf/1412.6980.pdf

Kuhn, M. (2021). *nearZeroVar function*. RDocumentation. Retrieved December 13, 2021, from https://www.rdocumentation.org/packages/caret/versions/6.0-90/topics/nearZeroVar

Kuhn, M., Li, D., & Lescarbeau, R. (2021). *High Correlation Filter*. recipes. Retrieved December 13, 2021, from https://recipes.tidymodels.org/reference/step_corr.html

Masters, T. (1993). *Practical Neural Network Recipes in C++*. Academic Press.

Ng, A. (2012). *CS229 Lecture notes* [Supervised learning]. Stanford University. https://web.archive.org/web/20180618211933/http://cs229.stanford.edu/notes/cs229-notes1.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). *Scikit-learn: Machine Learning in Python* (Vol. 12). Journal of Machine Learning Research. https://scikit-learn.org/stable/about.html#citing-scikit-learn

Pohl, W. (2021, September 19). *Regularization [PowerPoint Slides]*. Canvas. https://nhh.instructure.com/courses/1477/files?preview=233505

Ruder, S. (2016). An overview of gradient descent optimization algorithms. https://doi.org/10.1016/j.omega.2003.10.002

Shin, T. (2021, February 26). *Understanding Feature Importance and How to Implement it in Python*. Towards Data Science. Retrieved December 12, 2021, from https://towardsdatascience.com/understanding-feature-importance-and-how-to-implement-it-in-python-ff0287b20285

Singh, H. (2018, November 3). *Understanding Gradient Boosting Machines*. Towards Data Science. Retrieved December 11, 2021, from https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab

Tang, G. Y.N. (2004). How efficient is naive portfolio diversification? *Omega, 32*(2). https://doi.org/10.1016/j.omega.2003.10.002

Willmott, C.J. and Matsuura, K. (2005). Advantages of the Mean Absolute Error (MAE) over the Root Mean Square Error (RMSE) in Assessing Average Model Performance. *Climate Research, 30,* 79-82.

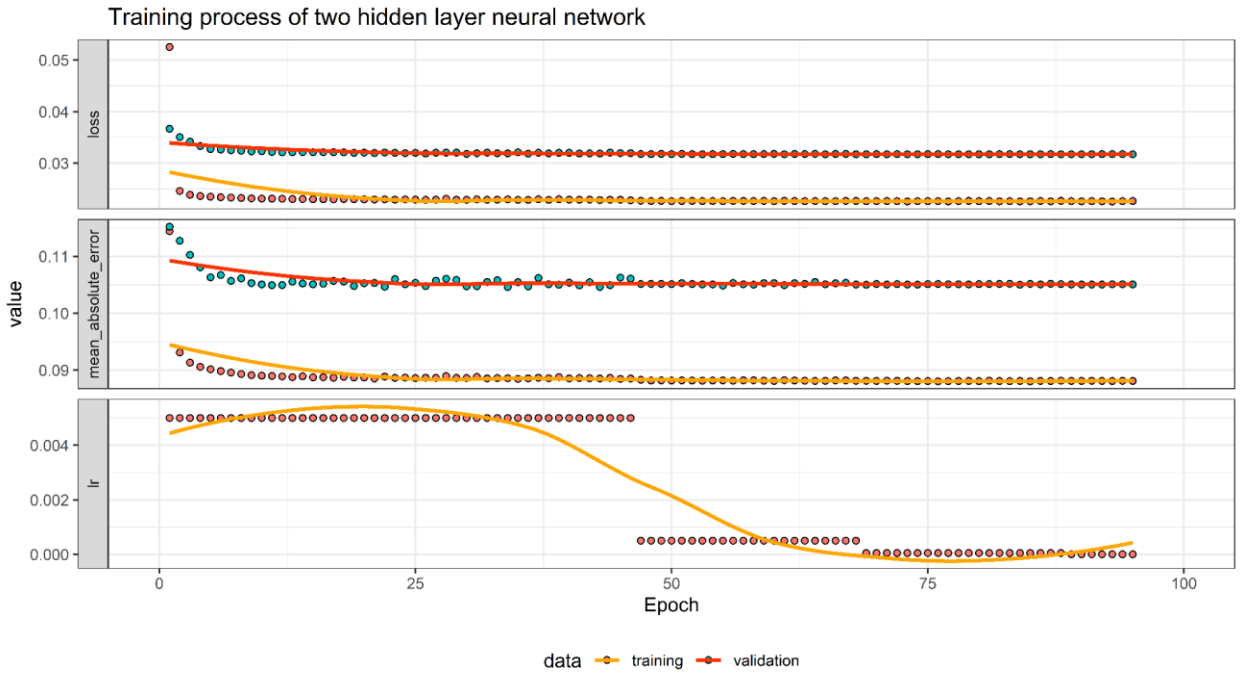
http://dx.doi.org/10.3354/cr030079

9. Appendix

9.1 Feature Description

Feature	Description
vwretd	Market Excess Return
vol	Volume
shrout	Shared Outstanding
cshfdq	Common Shared for Diluted EPS
aoq	Assets - Other - Total
cheq	Cash and Short-Term Investments
cogsq	Cost of Goods Sold
apq	Account Payable/Creditors - Trade
epsfxq	Earnings Per Share (Diluted) - Excluding Extraodinary Items
chq	Cash
citotalq	Comprehensive Income - Parent
ceqq	Common/Ordinary Equity - Total
dpq	Depreciation and Amortization - Total
lcoq	Current Liabilities - Other - Total
capsq	Capital Surplus/Share Premium Reserve
acoq	Current Asets - Other - Total
invtq	Inventories - Total
cstkq	Common/Ordinary Stock (Capital)
intanoq	Other Intangibles
dlttq	Long-Term Debt - Total
loq	Liabilities - Other
nopiq	Non-Operating Income (Expense) - Total
oeps12	Earnings Per Share from Operations - 12 Month Moving
oepsxq	Earnings Per Share - Diluted - from Operations
oiadpq	Operations Income After Depreciation - Quarterly
ppentq	Property Plant and Equipment - Total (Net)
reunaq	Unadjusted Retained Earnings
costat	Active/Inactive Status Marker
marketcap	Market Capitilization

9.2 Training process of two hidden layer neural network model



9.3 Performance metrics of 30 stocks with highest predictability

30 stocks of highest predictability

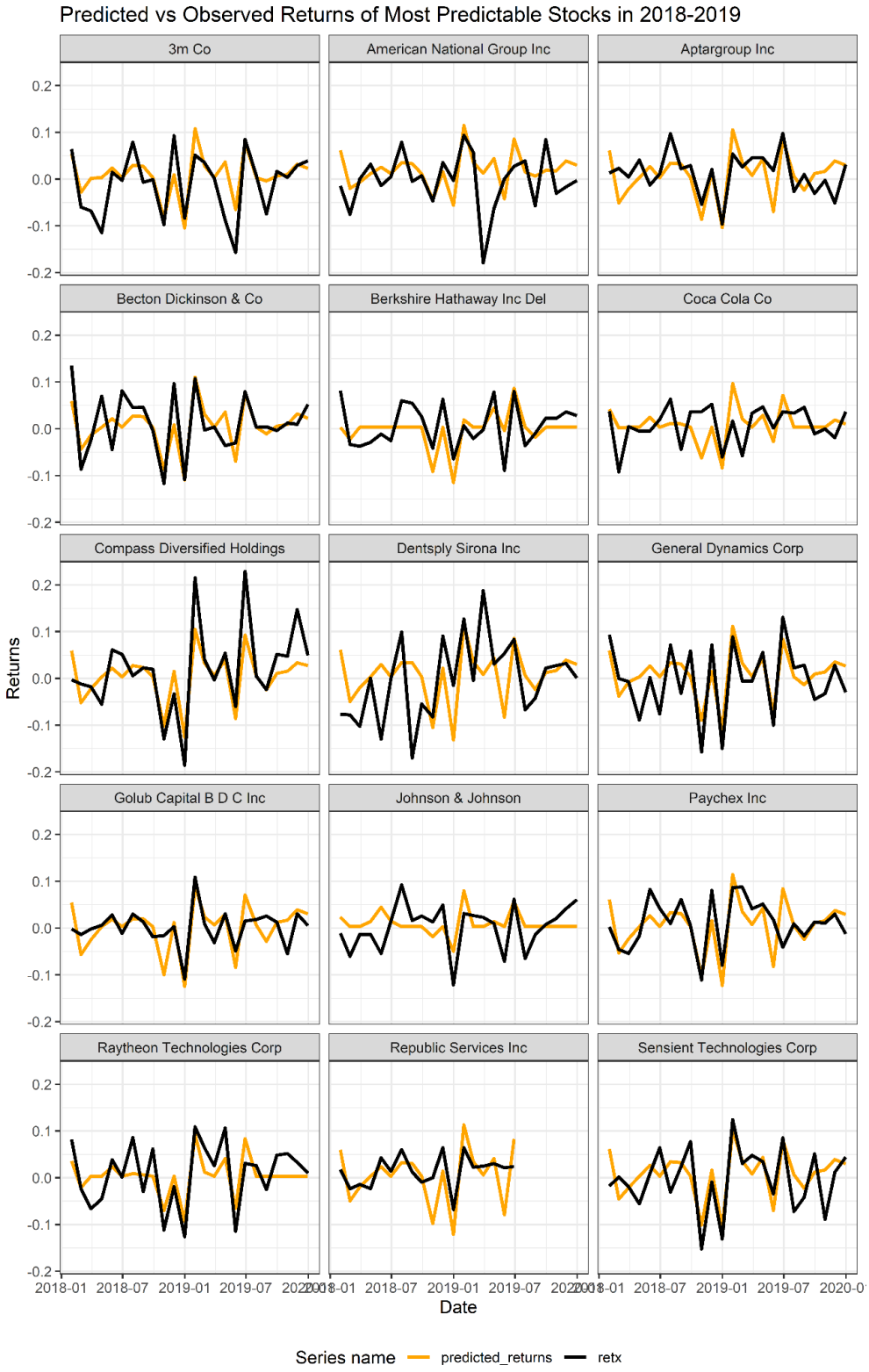
Company name	Company identifier	Test RMSE	Test MAE
Becton Dickinson & Co	39642	0.03282	0.02507
3m Co	22592	0.03124	0.02517
Aptargroup Inc	79133	0.03211	0.02575
Berkshire Hathaway Inc Del	83443	0.03513	0.02757
Johnson & Johnson	22111	0.03580	0.02842
Paychex Inc	61621	0.03670	0.02861
Raytheon Technologies Corp	17830	0.03588	0.02890
Dentsply Sirona Inc	11600	0.03838	0.02894
Compass Diversified Holdings	91271	0.03676	0.02908
Coca Cola Co	11308	0.03891	0.02929
General Dynamics Corp	12052	0.03940	0.02951
Republic Services Inc	86228	0.04082	0.02964
Golub Capital B D C Inc	93352	0.03958	0.02971
Sensient Technologies Corp	59619	0.04008	0.03030
American National Group Inc	13507	0.04139	0.03143
Procter & Gamble Co	18163	0.03970	0.03192
Student Transportation Inc	13015	0.05658	0.03201
Ametek Inc New	85257	0.03906	0.03204
Schein Henry Inc	82581	0.04208	0.03222
Texas Instruments Inc	15579	0.04088	0.03226
Franklin Resources Inc	37584	0.04240	0.03286
Emerson Electric Co	22103	0.04075	0.03300
Steris Plc New	77649	0.04327	0.03309
C I T Group Inc New	93150	0.04656	0.03409
Bank Of Nova Scotia	89428	0.04645	0.03411
Merck & Co Inc New	22752	0.04359	0.03432
Allison Transmission Hldgs Inc	13284	0.04453	0.03453
Oracle Corp	10104	0.04722	0.03493
International Paper Co	21573	0.04285	0.03509
Smith A O Corp	65402	0.04487	0.03566

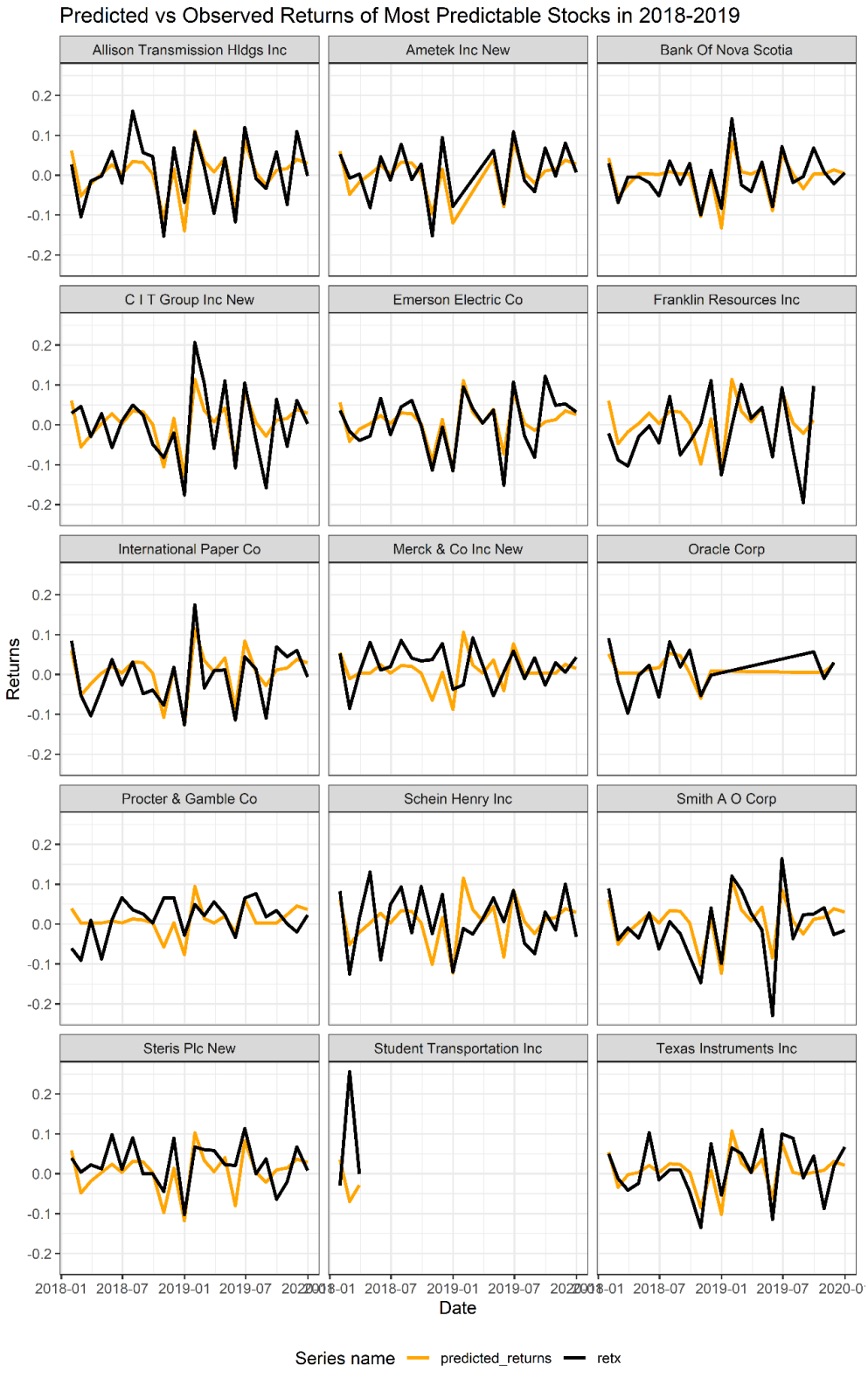
9.4 Financial indicators of 30 most predictable stocks

Financial indicators of 30 most predictable stocks

Company name	Mean market cap	Mean volume	Mean cash	Mean operating income
Becton Dickinson & Co	25937326	222863	2494	442
3m Co	86609476	554280	2409	1704
Aptargroup Inc	4083847	54220	341	76
Berkshire Hathaway Inc Del	151385414	767933	51314	7863
Johnson & Johnson	255526728	1825626	16845	4782
Paychex Inc	15623708	502170	193	253
Raytheon Technologies Corp	87358483	832909	5855	1975
Dentsply Sirona Inc	8048113	244610	191	110
Compass Diversified Holdings	836280	35742	51	16
Coca Cola Co	176042316	2724117	9997	2661
General Dynamics Corp	36930654	348854	3144	965
Republic Services Inc	13823842	367432	92	363
Golub Capital B D C Inc	711124	40547	16	17
Sensient Technologies Corp	2556276	42736	20	51
American National Group Inc	2632627	5659	167	69
Procter & Gamble Co	207719591	1971283	6292	3755
Student Transportation Inc	493207	29818	5	6
Ametek Inc New	10699029	252388	316	205
Schein Henry Inc	9945196	117778	92	177
Texas Instruments Inc	49814059	1471712	1391	926
Franklin Resources Inc	27450220	430526	6928	685
Emerson Electric Co	38657018	834629	2903	920
Steris Plc New	3612529	97300	187	69
C I T Group Inc New	8221793	344419	5899	472
Bank Of Nova Scotia	65601316	135647	6265	2375
Merck & Co Inc New	144689703	2540461	11441	2455
Allison Transmission Hldgs Inc	4881431	261229	183	120
Oracle Corp	166339309	4072269	17322	3505
International Paper Co	18194684	743801	1605	560
Smith A O Corp	4124611	141566	314	76

9.5 Out-of-sample prediction of 30 most predictable stocks




Predicted vs Observed Returns of Most Predictable Stocks in 2018-2019
Allison Transmission Hldgs Inc
Ametek Inc New
Bank Of Nova Scotia
C I T Group Inc New
Emerson Electric Co
Franklin Resources Inc
International Paper Co
Merck & Co Inc New
Oracle Corp
Procter & Gamble Co
Schein Henry Inc
Smith A O Corp
Steris Plc New
Student Transportation Inc
Texas Instruments Inc
Returns
0.2
0.1
0.0
-0.1
-0.2
2018-01
2018-07
2019-01
2019-07
2020-01
Date
Series name
predicted_returns
retx