Final Report

By: Joyce Tang, Max Kober, Drew Levine, and Mark Qian

Executive summary

Financial markets are believed to experience volatility when fear is felt throughout the market and investors rotate their portfolios into a risk-off strategy. When designing the algorithmic trading strategies we had one key principle in mind, a low risk. We believe that by separating the human element from our trading strategy we could create a low-risk portfolio that still achieves returns above the benchmark. Also, since our group is focusing on consumer staples, we believe our clients' objective is to minimize risks as much as possible. As a result, our group decided to focus on two metrics, beta, and sharp ratio. We accomplished this goal by creating 5 different trading models using core principles in statists, machine learning, and artificial intelligence.

Our next step was creating a scoring system so we could create a system to rank each model. With this ideology in mind, we set different weight to different metrics. Our model weighed Beta by sixty percent and sharp ratio by 10 percent which agrees with our methodology. We also used 1 month, 3 month, and twelve month draw down as a total of fifteen percent of the overall weighing of the model as we believe risk is always the priority we need to take care of. We put in total return and R-squared as well with a lower weight since we think the return is less important than the riskl. As a result, we got that Neural Networks created the best model for our desired needs. This weighing system allowed us to look at more aspects of the model then just total return. While conducting our training and pilot testing our group did its best to focus on year that did not have any anomalies such as 2020. Therefore, our model is best for an "average" year.

Our portfolio provides the best holdings for low-risk yet high predicted returns for the future. For a table with our recommended stocks and their returns, refer to Appendix Q.

Methodology

2.1 Objective

Before moving forward with any predictive modeling at all, we first collectively decided on our objective. Of course, returns are always important, but how we get those returns and other characteristics of our portfolio is equally as essential. We looked at some of the overarching themes of the class and eventually decided upon a primary and secondary goal. First, we wanted to **minimize risk as much as possible**. This decision largely can be attributed to our industry: consumer staples. A typical investor looking to hold a portfolio of consumer staples is likely doing so to **avoid the risk of the broader market**, **staying away from highly volatile growth stocks**. To achieve this, we specifically focused on **getting a beta as close to zero, essentially removing market risk**. Thus, our selection of results had this as our primary goal.

For our secondary goal, we wanted to **maximize returns**, **but still with risk in mind**. An investor looking to hold a portfolio of consumer staples would likely still want the best return for a given risk profile. Thus, our secondary metric was to **maximize sharpe ratio**. Sharpe ratio provided this exact objective, as it essentially describes the risk-adjusted return, or how impressive a return is for its given risk profile. A portfolio high in sharpe ratio should then provide good returns given its risk. In conjunction with our primary objective, using these two metrics would result in a low-risk portfolio with higher than expected returns given that low risk, which is exactly what our investor would likely want.

2.2 Discussion of our approaches

2.21 Linear Regression

Linear regression is the most basic and simplest approach to model the relationships between the independent variables (x) and dependent variable (y). In our model, we use signals collected from FactSet, such as book-to-market ratio and dividend yield as our independent variables (x) and portfolio return as our dependent variable (y). This approach finds the line of the best fit between the independent and dependent variable, creating the smallest root mean squared error (difference between the predicted y value and true y value) possible.

2.22 Lasso Penalized Regression

Lasso penalized regression is a type of linear regression that selects important signals and uses shrinkage. After selecting important signals, the model shrinks the coefficients of signals that are unimportant to 0. Hence, the sum of absolute values of the weights given by the signals are penalized. Having said that, Lasso penalized regression also models the relationships between the independent and dependent variable (y) with the line of best fit.

2.23 Random Forests

A Random forest is a type of model that consists of decision trees. These trees classify data through a series of conditional statements. Decision trees alone are extremely sensitive to the training data used to build them.

To improve the accuracy of the model a random forest uses a process known as bootstrapping. Bootstrapping randomly takes data from a dataset to create new unique datasets. Next random individual trees are made by selecting different elements. Finally, a tree is created using the variables selected and the new datasets from the bootstrapping process. Each tree also only focuses on a subset of variables from the bootstrapping process. This process is then repeated over and over to multiple trees. We then run data through each individual tree and receive an output prediction. These predictions are averaged from all the trees to get the final result.

2.24 Gradient Boosting Regression

Different from Random Forest that each tree is built independently, Gradient Boosting Regression randomly generates a number of trees in a way that each tree is built off of the residuals from trees before. Gradient Boosting Regression usually starts with a leaf that is the average value of the variable to predict, and add a tree based on the residuals - the difference between the observed values and the predicted values. In order to avoid that the model fits the training data too well so that there is a very high variance despite the low bias, Gradient Boosting Regression takes advantage of the learning rate to scale the contribution from the new tree, which results in a small step in the right direction instead of a big step reaching the destination directly. Then Grident Boosting Regression will add another tree based on the rew residuals, and keep adding trees based on the errors made by the previous tree until reaching the maximum specified. In this way, Gradient Boosting Regression will get a final answer of the value to predict by adding all tree's predictions after adjustment together and is able to address the overfitting problem. However, this ensemble method will also bring up huge computational costs.

2.25 Neural Networks

When many people are referring to "artificial intelligence", they are likely referring to something utilizing neural networks. Although it utilizes some similarities from models discussed above, it functions very differently. The name neural network is based on human neurons, and the model's processing mimics a human brain.

It works by having different layers, some hidden, of neurons. These neurons create their own weightings of each of the signals, and works similarly to linear

regressions. These weightings aggregate across all the neurons to create a comprehensive prediction of return. Each of these neurons begins with a random weighting, which typically produces garbage output on the first attempt. However, it utilizes “loss” (or in this case Mean Squared Error) to determine how far off the prediction was from the actual value. As these neurons repeat the process with new weightings, they learn which values are more effective at predicting the actual result. This learning is referred to as backpropogation. Each of these “trials” is called an epoch, and is one of the hyperparameters we were able to tune to improve the model.

In addition to different numbers of epochs, the layers also make a large impact on the results of the model. When creating a neural network, you can have any number of layers containing different numbers of neurons. As you add more hidden layers, there are more neurons working towards a prediction, but this also can cause overfitting and increased complexity needed to tune the weights of these neurons. Each layer of the model is also guided by an activation function, which determines whether a neuron’s prediction will be used. In our case, we use “ReLU” which essentially drops the neurons that have negative predictions from the overall prediction.

There are also some other parameters that change the way that the model processes, learns, and adapts. The first is batching, which functions exactly how it seems. Each epoch will process one “batch” of data rather than processing every single prediction fed. While this mostly serves to improve processing time, it also can reduce overfitting the data. Another hyperparameter is early stopping. Each epoch has the intention of reducing loss and improving the model, but sometimes new epochs don’t actually improve the model, so it is best to save time and reduce overtraining. The number of epochs without improvement before stopping is called the model’s patience, and is another hyperparameter to tune.

Another parameter is the model’s learning rate, which is essentially the magnitude at which each of the neurons can adjust their weights after each epoch. A higher learning rate can cause the model to hit a “local maximum/minimum” rather than finding the global maximum/minimum, or essentially finding what seems to be the ideal solution and sticking with it rather than the overall ideal solution. A lower learning rate can cause the model to fail to find an ideal solution at all. Lastly, we can add a penalty on parameters in a similar way to the lasso, penalizing signals that seem to not be beneficial to the overall model. Each of these parameters make neural networks one of the more complex models for predictions, however can produce great results if tuned well.

2.3 Selection & tuning of hyperprarmeters

For Linear and Lasso penalized regression, we have made two key selection decisions. Firstly, we have to decide what is the optimal split of the sets of data for training, validation and testing. Based on our research, we discovered that one of the common way for data scientists to split the data is using the 60-20-20 rule. This essentially refers to using 60% of the data for training, 20% for validation and the remaining 20% for testing. Since we have 20 years of data in total, the best split that we can do is using 5-year training (~56%), 3-year validation (~33%), and 1-year testing (~11%).

The second selection decision is about the type of window that shall be adopted to train our data sets. The main difference between rolling and expanding window is that the former have a fixed window size (5 years in our model), while the latter started at a fixed point and incorporate new data as it becomes available. Based on research, we found that sample size has a huge impact on the results generated by expanding window. In contrast to that, since rolling window maintains the same sample size in each iteration, its predictions are not affected by the sample size. Thus, we believe choosing rolling window would allow us to yield more accurate results than expanding window. Despite our preference of choosing rolling over expanding window, our group thinks that it is of equal importance to consider the final results of all possible models, Therefore, we tried running Linear and Lasso penalized regression using both rolling and expanding window. In the end, we have chosen Linear regression expanding window and Lasso regression rolling window as they were able to produce the best results (please refer to section 3. Results for details).

For Random Forest Regression and Gradient Boosting Regression, since we have a ton of hyperparameters to tune, the machine would require a large validation set to optimize the model performance. Therefore, in order to come to a split percentage that suits the requirements and meets the ensemble method's needs, we decided to adopt a common way in the data science area where we are going to use 60% of the data for training, 20% for validation and the remaining 20% for testing. Since we have 20 years in total, we would take 2001 to 2012 as the training set, 2013 to 2016 as the validation set and 2017 to 2020 as the testing set (Appendix A).

Fitting the training set too closely can lead to degradation of the Gradient Boosting Regression's generalization ability. Therefore, we need to be cautious about the hyperparameters we choose. The first hyperparameters we need to focus on is the number of the trees. Generally, increasing the number of trees will help reduce the error on training set, but setting it too high may lead to overfitting. Hence, we selected 4 numbers ranging from 300 to 1000 to represent different scenarios. By monitoring the prediction error on the validation set, we can then find the optimal value. Regarding how to decide the tree depth, trees are typically short when using Gradient Boosting Regression, as a large number of short trees may perform better according to

the history experience, so we limited the tree depth within 3 so that it would be a shallow learning model which has a higher probability to avoid overfitting. We need to pay attention to the learning rate as well. The predictions of each tree are added together sequentially and the contribution of each tree to this sum can be weighted to slow down the learning by the algorithm. Typically, a low learning rate would dramatically improve the performance because it makes each step small, although it will also drive up the training time as the model will require more iterations to converge to the final loss value. Thus, we picked three small learning rate within 0.1 so that we could get an accurate prediction. In addition to the above three hyperparameters, last but not least, we would also need to put max_features and subsample into consideration. Choosing max_features less than n_features will result in a a reduction of variance and an increase in bias, which agrees with our goal to restrict overfitting. Since there are 36 signals (or features in data science) in our dataset, we picked 5, 20, and 36 to represent the whole picture. The subsample refers to the fraction of samples to be used for fitting the individual base learners. Choosing subsample less than 1 leads to a reduction of variance and an increase in bias, which agrees again with our goal to avoid overfitting. Here we decided to choose the parameters of 0.5, 0.8, and 1 to see which subsample would work best. After putting all hyperparameters in a grid, we made use of the for loop to figure out the optimal combination of the hyperparameters to maximize the test sample R-squared (Appendix B). Random Forest Regression works almost the same as Gradient Boosting Regression, with less hyperparamters to tune.

For neural network hyperparameter tuning, we decided to take a different approach for testing. Because of the extreme complexity and computational demands that the model has, we decided to use a smaller dataset (which we called the “pilot testing”) to make the tuning process faster. In order to select which years in which the pilot testing would take place, we did some market research on the past 20 years (Appendix O). When using a smaller dataset, it was important to us to use years with as few anomalies as possible. For example, using 2020 as training data would skew a lot of different things due to the impacts of the pandemic. So, we analyzed market returns on the S&P and found that it has average annual returns of about 10% in the past century. We then selected years that had similar returns and did a deeper analysis, including looking at news events, interest rates, and other factors to verify that these years were as close to “normal” as possible. With these factors in mind, we selected 2004 for training, 2014 for validation, and 2016 for test data (although the 2016 election had an impact on markets, it was still largely normal from a data standpoint).

After these years were selected, it gave us the freedom to test many different hyperparameters that were tuned, included, and excluded. The goal with this pilot testing was to find the top 3 sets of hyperparameters and then compare those three with the full set of data to find the best model using neural networks. The process during the pilot testing first started with the basic model. We tested with different numbers of layers (from 2 to 5), each with different numbers of neurons (8, 16, 32, and 64 were common numbers used) but standardized at 100

epochs (this produced the best consistency). One of these produced the best results from the pilot testing, with a beta of 0.27633 and a sharpe ratio of -0.108613 (Appendix K). Following this testing, we experimented with batching, using different batch sizes while still using differing layer numbers and layers of different neurons. While batching completed its objective of improving performance time, it didn't have positive impact on the results, likely because our shortening of the dataset with our pilot testing already accomplished a form of batching. We attempted to remedy this by using different numbers of epochs, but batching still didn't produce attractive results, so we decided to remove it as a potential strategy.

Following this discovery, we experimented with early stopping, again differing layer numbers and sizes as well as the new hyperparameter of patience. Early stopping did improve certain sets of models, particularly when layers were smaller and epochs were larger. During this time, the number of epochs in conjunction with the patience had large impacts on results, so typical number of epochs were 100, 250, and 500. Shortly after this, we decided to add in a manual learning rate, which had a much larger effect on the results. Here, we produced our second best model during pilot testing with a beta of 0.419511 and sharpe ratio of -0.144684 (Appendix L). After adding learning rate, we spent time to adjust layer number and size, patience, learning rate, and epoch number. Observations from this portion of pilot testing were that learning rate greatly improved results for some models, but also could produce some really whacky numbers, including r-squared values even below -700. So certain models could be used, but others were somewhat garbage. Lastly, we added in penalties to our layers. This produced the third best model from the pilot testing with a beta of 0.502329 and a sharpe ratio of -0.093774 (Appendix M). Once penalties were added, we adjusted each of the previous hyperparameters again to see any improvement. Many of the models had good improvement with out-of-sample returns, but not with beta.

Once this pilot testing was completed, we took these top 3 models and input each of their hyperparameters, running it with the entire set of data. For splitting the entire dataset, we decided to follow common practice of data splits, which we found through online research (Appendix N). We did a split consisting of 80-10-10, meaning 80% training, 10% validation, and 10% testing. This came out to be 2001-2016 training, 2017-2018 validation, and 2019-2020 testing. Running each of the top 3 models gave us these results (Appendix K-M), leading to our final best neural net model, which will be discussed in the results section.

Results

3.1 Linear Regression

The table below (Appendix C) demonstrates that Linear regression expanding window is a better model than Linear regression rolling window as it produces a lower beta (1.115680).

Regression Method	a.Total return on $1 invested	b.Comparable return on a value-weighted benchmark of stocks in your industry	c.Test-sample R-squared	d.Geometric average return	e.Alpha	f.Beta	g.Maximum 1-month drawdowns	g.Maximum 3-month drawdown	g.Maximum 12-month drawdowns	h.Sharpe ratio
Linear Regression expanding window	1.0110965704067392	1.0099479907086333	-0.009015799338013553	0.10537989004516657	0.0009398668299851337	1.1156805273078354	-0.23104476312446787	-0.21597168925597177	-0.17892682695785334	0.02961138983654607
Linear Regression rolling window	1.0109906381416272	1.0099479907086333	-0.027305729221204444	0.10922781993816577	-0.014014263613907271	1.3239799259997354	-0.23104476312446787	-0.25275683931822857	-0.2585919370370492	0.031102615164556317

As for signals importance (Appendix D), we found that sales growth (+), EPS momentum (+), CFO/P (+), 52 Week High Price Relative (-) are the four most important signals as their p-value less than 0.10.

OLS Regression Results

Dep. Variable:	RET	R-squared:	0.004
Model:	OLS	Adj. R-squared:	0.003
Method:	Least Squares	F-statistic:	3.811
Date:	Tue, 05 Apr 2022	Prob (F-statistic):	1.12e-09
Time:	13:26:20	Log-Likelihood:	-1.1950e+05
No. Observations:	29157	AIC:	2.391e+05
Df Residuals:	29121	BIC:	2.394e+05
Df Model:	35		
Covariance Type:	cluster		

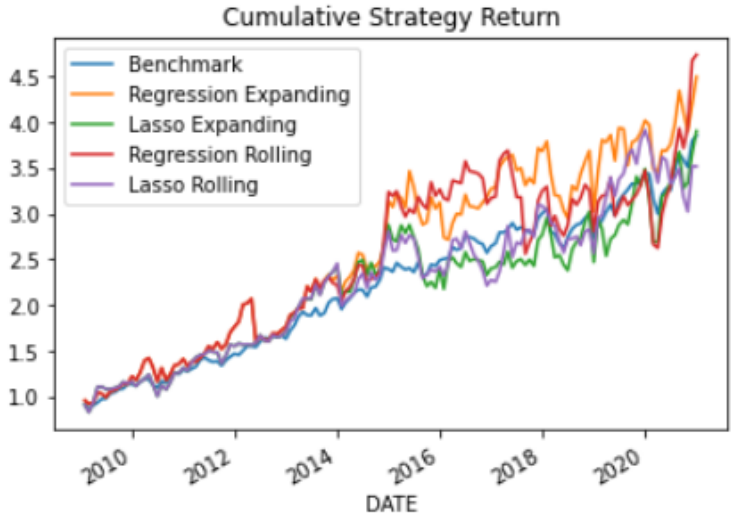
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.0258	0.911	1.127	0.261	-0.770	2.821
EP0	0.0118	0.048	0.244	0.808	-0.084	0.107
ROE	-0.0372	0.051	-0.722	0.471	-0.139	0.064
ROA	0.0439	0.087	0.506	0.613	-0.127	0.215
BM	0.0251	0.083	0.303	0.762	-0.138	0.189
DIVYIELD	0.0488	0.036	1.344	0.180	-0.023	0.120
EBIT_EV	-0.0374	0.052	-0.725	0.469	-0.139	0.064
SALE_EV	0.0558	0.051	1.092	0.276	-0.045	0.157
OCF_GROWTH	0.0384	0.032	1.217	0.225	-0.024	0.101
SALES_GROWTH	0.1420	0.054	2.622	0.009	0.035	0.249
AVG_SALES_GROWTH	0.0078	0.040	0.195	0.846	-0.071	0.087
EPS_MOMENTUM	0.1298	0.038	3.384	0.001	0.054	0.206
ANN_ACCRUAL	-0.1068	0.072	-1.484	0.139	-0.249	0.035
LTM_ACCRUAL	-0.0086	0.076	-0.114	0.909	-0.158	0.140
CHG_AT	-0.0229	0.039	-0.580	0.563	-0.101	0.055
LTM_AT	-0.0181	0.048	-0.375	0.708	-0.113	0.077
NOA_ASSETS	0.0042	0.057	0.073	0.942	-0.109	0.117
RD_ASSETS	0.0510	0.040	1.282	0.201	-0.027	0.129
EBITDA_DEBT	-0.0152	0.036	-0.417	0.677	-0.087	0.057
NET_EXT_FIN	-1.0477	1.862	-0.563	0.574	-4.720	2.624
AG	0.0313	0.075	0.415	0.678	-0.117	0.180
pastret	-0.0848	0.055	-1.533	0.127	-0.194	0.024
LT_PB	0.0547	0.035	1.552	0.122	-0.015	0.124
CF_PRC	0.1182	0.043	2.758	0.006	0.034	0.203
FCF_PRC	0.0101	0.071	0.142	0.887	-0.129	0.149
IVAL_PRC	-0.0384	0.086	-0.447	0.655	-0.208	0.131
REL_PRC	-0.2539	0.047	-5.365	0.000	-0.347	-0.161
EP1	-0.0503	0.078	-0.647	0.518	-0.204	0.103
EP2	-0.0411	0.057	-0.725	0.469	-0.153	0.071
EPS_LTG	-0.0288	0.057	-0.502	0.616	-0.142	0.084
PEG_VAL	-0.0103	0.047	-0.219	0.827	-0.103	0.082
SURP_MOM	-0.0281	0.038	-0.747	0.456	-0.102	0.046
FEPS_DELTA	0.0121	0.031	0.385	0.701	-0.050	0.074
REV_MOM	0.0284	0.030	0.948	0.344	-0.031	0.087
EPS_SURP	-0.0434	0.049	-0.889	0.375	-0.140	0.053
FDISP	-0.0114	0.042	-0.274	0.784	-0.094	0.071

3.2 Lasso Penalized Regression

Results (Appendix E) indicates that Lasso regression rolling window is a better model than Lasso

regression expanding window as it produces a lower beta (1.199194).

Regression Method	a.Total return on $1 invested	b.Comparable return on a value-weighted benchmark of stocks in your industry	c.Test-sample R-squared	d.Geometric average return	e.Alpha	f.Beta	g.Maximum 1-month drawdowns	g.Maximum 3-month drawdown	g.Maximum 12-month drawdowns	h.Sharpe ratio
Lasso Regression expanding window	1.0106343689694701	1.0099479907086333	-0.002445178027098205	0.0949920686858372	-0.01861737007514329	1.2172642745314584	-0.18261706826555302	-0.2258082957604033	-0.19950919252858912	0.03472701308519879
Lasso Regression rolling window	1.0105922432140053	1.0099479907086333	-0.024096178964223336	0.08740475285301996	-0.024573601818133478	1.199194242362682	-0.1904570881050941	-0.1702480383448941	-0.16815714170594975	0.03302200070970879



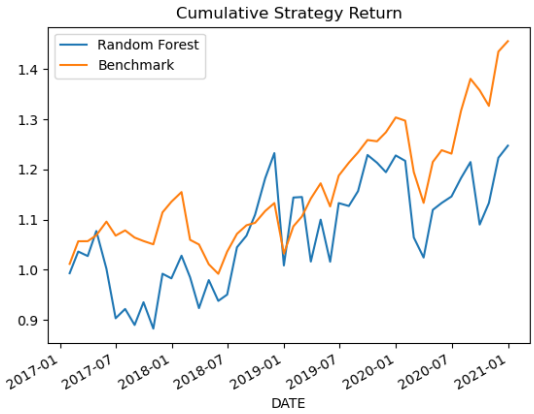
As for signals importance (Appendix F), we found that PE (+), Return on average total equity (-), CFO/P (-), FCF to Price(-), Forward 1YR EP (+), PEG Valuation (-) are the signals with importance a their coefficients have not been shrinked to 0.

```
[ 6.35614402e-07 -3.56257246e-05  0.00000000e+00 -0.00000000e+00
 -0.00000000e+00  0.00000000e+00 -0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00  0.00000000e+00
  0.00000000e+00 -0.00000000e+00  0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -8.96700775e-05 -2.23396201e-06
 -0.00000000e+00 -0.00000000e+00  1.51490777e-01  0.00000000e+00
  0.00000000e+00 -2.84199823e-02 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00]
```

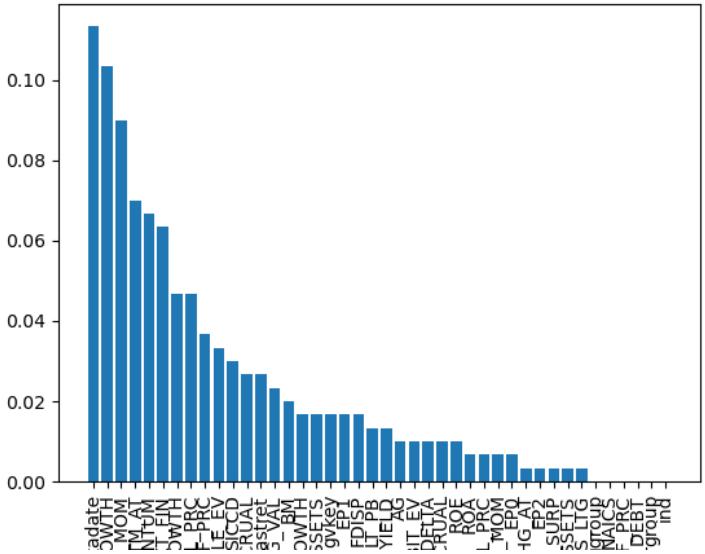
3.3 Random Forest

	Total return	Benchmark return	R-squared	Geometric average	Alpha	\
0	1.074885	1.159317	-0.000243	0.011122	-0.003357	

	Beta	1-month drawdowns	3-month drawdowns	12-month drawdowns	\
0	1.225609	-0.181828	-0.165853	-0.112937	



The result (Appendix G) indicted that if we use Random Forest Regression, its cumulative return would probably perform worse than the benchmark in the 4-year test set based on 2001-2013's historical market data.



```
[ 1 10 36 16 12 20  9 29 26  8 22 14 24 33  5 11 17  0 30 38 25  6 21  7
 35 13  3  4 28 34  2 15 31 37 18 32 39 23 27 19 40 41]
[0.01666667 0.11333333 0.00666667 0.01       0.00666667 0.02
 0.01333333 0.01       0.03333333 0.04666667 0.10333333 0.01666667
 0.06666667 0.01       0.02666667 0.00333333 0.07       0.01666667
 0.00333333 0.         0.06333333 0.01       0.03       0.
 0.02666667 0.01333333 0.03666667 0.         0.00666667 0.04666667
 0.01666667 0.00333333 0.00333333 0.02333333 0.00666667 0.01
 0.09       0.00333333 0.01666667 0.         0.         0.        ]
```

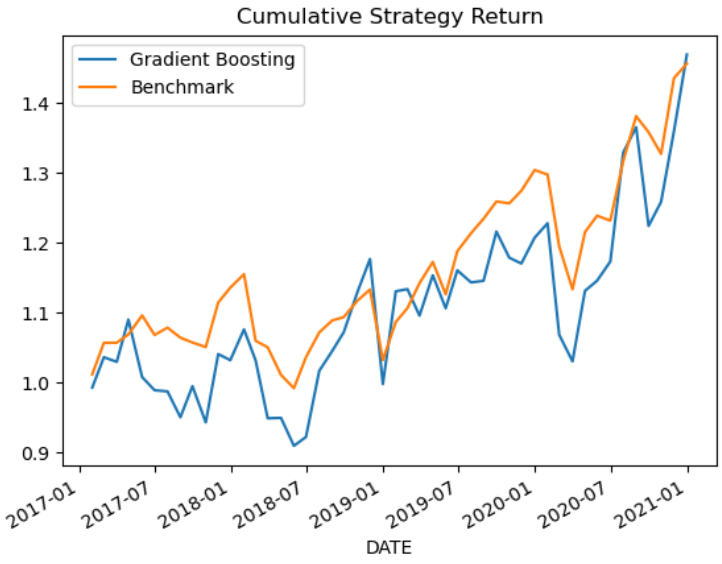
The data (Appendix H) we get here indicated that OCF_GROWTH, SURP_MOM, LTM_AT would be the top three signals which reduces the impurity most on average for the tuned model using Random Forest Regression. We skip the datadate as we believe it is an outlier.

3.4 Gradient Boosting Regression

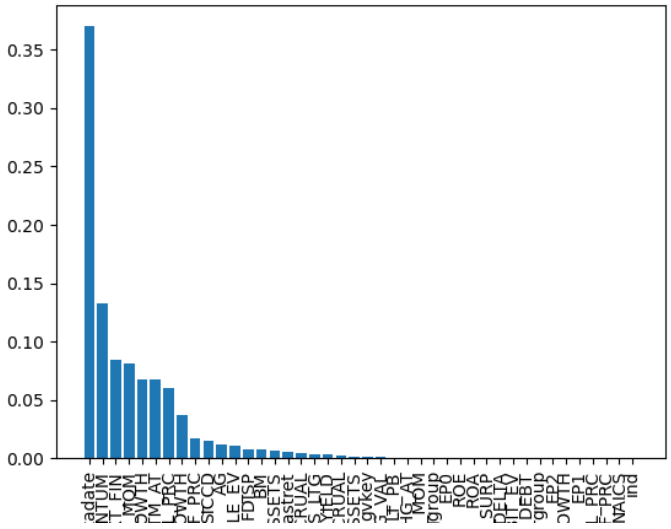
	Total return	Benchmark return	R-squared	Geometric average	Alpha	\
0	1.103992	1.159317	-0.000625	0.019416	-0.00032	

	Beta	1-month drawdowns	3-month drawdowns	12-month drawdowns	\
0	1.225056	-0.151953	-0.146534	-0.128976	

	Sharpe ratio
0	0.019646



The result (Appendix I) indicted that if we use Gradient Boosting Regression, its cumulative return would outperform the benchmark sometimes. However, for most of the time, the cumulative return would the competition against the benchmark.




```
[ 1 12 20 36 10 16 29  9 26 22 21  8 38  5 17 24 14 32  6 13 18  0 33 25
 15 34 39  2  3  4 37 35  7 19 40 31 11 30 28 27 23 41]
[0.00160228 0.37015642 0.         0.         0.         0.00756088
 0.00295682 0.         0.01026775 0.03745197 0.0676942  0.
 0.13285191 0.00244748 0.00389677 0.00048228 0.06725811 0.00662562
 0.0017871  0.         0.08403864 0.01188341 0.0150854  0.
 0.00504551 0.00058682 0.01731153 0.         0.         0.05997165
 0.         0.         0.00299214 0.00080847 0.         0.
 0.08116925 0.         0.00806758 0.         0.         0.        ]
```

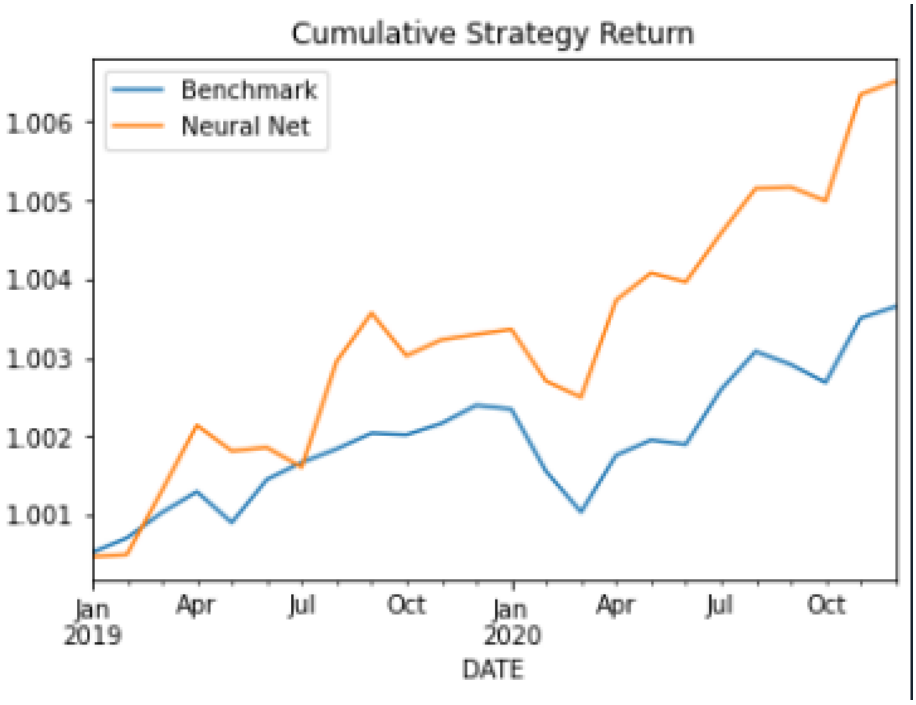
The data (Appendix J) we get here indicated that EPS_MOMENTUM, NET_EXT_FIN, SURP_MOM would be the top three signals which reduces the impurity most on average for the tuned model using Gradient Boosting Regression. We skip the datadate as we believe it is an outlier.

3.5 Neural Networks

	Total return	Benchmark return	R-squared	Geometric average	Alpha	\
0	1.003285	1.001959	-8.642964	0.000324	0.006718	

	Beta	1-month drawdowns	3-month drawdowns	12-month drawdowns	\
0	1.052571	-0.000657	-0.000796	0.001186	

	Sharpe ratio
0	-0.890965



Following the pilot testing and the subsequent testing of the top 3 models on the entire dataset, the above code output (Appendix K) demonstrates the best results from the neural network strategy based on our overarching objectives. Neural networks do not support signal importance because each of the neurons are hidden layers with different weightings that cannot be accessed.

Predictions

To determine the predictions in naming our winning stocks, we had to first collect new data on the signals using factset. In order to do this and have the file be compatible with our created models, we created a screener on factset that mimicked the data we received in the rank signals file. Utilizing the excel sheet containing the formulas of the US Master Screen, we created this screener and filtered it to only contain companies traded on the NASDAQ and NYSE. We then decided to export the file to excel, where we utilized functions such as VLOOKUP to trace the list of tickers we received to the corresponding signals from the screener. We then copied the values from this sheet onto a new file that we exported to a .csv file, which was ready to be utilized in python by our winning model. (File linked in submission)

To determine the winning model, we decided to use a scoring system that would provide a fair way of determining which results were better based on our overarching objective. To create these scores, we gave a weighting to each of the results that were output based on their importance to us. Then, we ranked each strategy 1 through 5 based on how it compared to each

other on that metric. For example, the highest geometric return would earn 5 points and the lowest would earn 1 point. We then averaged these points for each strategy to determine the winner. The weightings of these results were based on our overall strategy, placing beta at the highest followed by sharpe ratio. The actual number weightings were just overall feelings of what we wanted and come out to be somewhat arbitrary, but still within the main objective. Through this scoring system (Appendix P), neural network became the winning strategy.

With neural network being the best model based on its results matching with our overarching objective, we now had to adjust the initial code to be adaptable to this new dataset. This involved importing it as a new dataframe, adjusting the columns of both dataframes to match each other, adjusting the training/validation/test data sets to the appropriate data files, and running the model. In the end, we produced 16 winning stocks that passed the model and output their signals and predicted returns. (Appendix Q) (file linked in submission)

Error Analysis

Because of the complexity of this project and how the data worked out, we decided to discuss some of our errors that occurred and how we might mitigate those errors if this analysis were to be used in a professional environment.

Firstly, our neural network hyperparameter tuning strategy utilizing a smaller dataset of pilot years yielded a less-than-ideal model for certain metrics. Although it ended up producing the best output based on our objective, it had an extremely negative R-squared as well as a negative sharpe ratio. To mitigate this in the future, utilizing a batching strategy but over the entire dataset likely would have still saved time computationally without harming the integrity of the pilot testing. We quickly found that success during the pilot test did not correlate very well at all with success during full testing.

Secondly, our data manipulation strategy for turning the data from factset into usable signals that integrated with our models produced stunted results. Unfortunately, our model only produced 16 stocks total. Although we are unsure of the exact origin of why this occurred, we believe that the data we received from factset (which will be contained in a file within the submission) created many #N/A values for the signals. We quickly removed the stocks that were not traded on the NYSE or NASDAQ exchanges as well as dropped “nan” values in our code, but doing so left us with very few tickers remaining, likely resulting in a dataset that wasn’t well suited for our model. In the future, we would likely look at finding some of those #N/A values from sources outside of factset or removing signals that didn’t contain enough data. While some of these choices could have been made within the bounds of the project, we decided that because our testing had been done with all of the signals, it wouldn’t be with integrity to remove many of the signals simply because they didn’t fit our dataset. Thus, our output produced less-than-ideal results, although we are confident that under different circumstances, our methodology was sound.

Code appendix

Appendix A: Random Forest and Gradient Boosting Data Splits

```
#Train, validation, and test samples.  Train:  2001-2012, Valid:  2013-2016,
#Test:  2017-2020
#Since RFR has a lot of hyperparameters to tune, I select the split percetage 60% 20% 20%
X_train = df[df['year']<2013][column_name_list]
y_train = df[df['year']<2013]['RET']
X_valid = df[(df['year']<2017) & (df['year']>=2013)][column_name_list]
y_valid = df[(df['year']<2017) & (df['year']>=2013)]['RET']
X_test = df[(df['year']<2021) & (df['year']>=2017)][column_name_list]
y_test = df[(df['year']<2021) & (df['year']>=2017)]['RET']
test_data = df[df['year']>=2017]
```

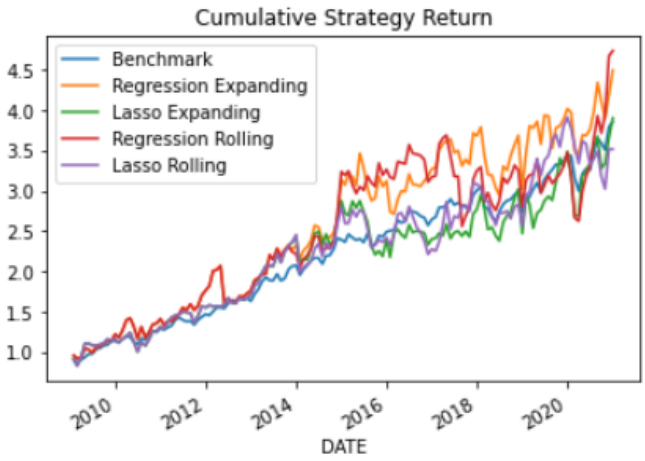
Appendix B: Hyperparameter Decisions in Random Forest and Gradient Boosting

```
#There are 36 features(signals) in the ranks_signal7 table
grid = {'n_estimators': [300,500,1000], 'max_depth': [1,3,5],
        'max_features': [3,5,10,15,20,36], 'random_state': [7]}
```

```
#Since shallow tree works better in GBR, I only use the tree depth within 3
#There are 36 features(signals) in the ranks_signal7 table
grid = {'n_estimators': [300,500,800,1000], 'max_depth': [1,2,3],
        'max_features': [5,20,36], 'learning_rate': [0.01,0.05,0.10],
        'subsample': [0.5,0.8,1.0], 'random_state': [7]}
```

Appendix C: Results from Linear regression

Regression Method	a.Total return on $1 invested	b.Comparable return on a value-weighted benchmark of stocks in your industry	c.Test-sample R-squared	d.Geometric average return	e.Alpha	f.Beta	g.Maximum 1-month drawdowns	g.Maximum 3-month drawdown	g.Maximum 12-month drawdowns	h.Sharpe ratio
Linear Regression expanding window	1.0110965704067392	1.0099479907086333	-0.009015799338013553	0.10537989004516657	0.0009398668299851337	1.1156805273078354	-0.23104476312446787	-0.21597168925597177	-0.17892682695785334	0.02961138983654607
Linear Regression rolling window	1.0109906381416272	1.0099479907086333	-0.027305729221204444	0.10922781993816577	-0.014014263613907271	1.3239799259997354	-0.23104476312446787	-0.25275683931822857	-0.2585919370370492	0.031102615164556317



Appendix D: Linear regression signal importance

OLS Regression Results

Dep. Variable:	RET	R-squared:	0.004
Model:	OLS	Adj. R-squared:	0.003
Method:	Least Squares	F-statistic:	3.811
Date:	Tue, 05 Apr 2022	Prob (F-statistic):	1.12e-09
Time:	13:26:20	Log-Likelihood:	-1.1950e+05
No. Observations:	29157	AIC:	2.391e+05
Df Residuals:	29121	BIC:	2.394e+05
Df Model:	35		
Covariance Type:	cluster		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.0258	0.911	1.127	0.261	-0.770	2.821
EP0	0.0118	0.048	0.244	0.808	-0.084	0.107
ROE	-0.0372	0.051	-0.722	0.471	-0.139	0.064
ROA	0.0439	0.087	0.506	0.613	-0.127	0.215
BM	0.0251	0.083	0.303	0.762	-0.138	0.189
DIVYIELD	0.0488	0.036	1.344	0.180	-0.023	0.120
EBIT_EV	-0.0374	0.052	-0.725	0.469	-0.139	0.064
SALE_EV	0.0558	0.051	1.092	0.276	-0.045	0.157
OCF_GROWTH	0.0384	0.032	1.217	0.225	-0.024	0.101
SALES_GROWTH	0.1420	0.054	2.622	0.009	0.035	0.249
AVG_SALES_GROWTH	0.0078	0.040	0.195	0.846	-0.071	0.087
EPS_MOMENTUM	0.1298	0.038	3.384	0.001	0.054	0.206
ANN_ACCRUAL	-0.1068	0.072	-1.484	0.139	-0.249	0.035
LTM_ACCRUAL	-0.0086	0.076	-0.114	0.909	-0.158	0.140
CHG_AT	-0.0229	0.039	-0.580	0.563	-0.101	0.055
LTM_AT	-0.0181	0.048	-0.375	0.708	-0.113	0.077
NOA_ASSETS	0.0042	0.057	0.073	0.942	-0.109	0.117
RD_ASSETS	0.0510	0.040	1.282	0.201	-0.027	0.129
EBITDA_DEBT	-0.0152	0.036	-0.417	0.677	-0.087	0.057
NET_EXT_FIN	-1.0477	1.862	-0.563	0.574	-4.720	2.624
AG	0.0313	0.075	0.415	0.678	-0.117	0.180
pastret	-0.0848	0.055	-1.533	0.127	-0.194	0.024
LT_PB	0.0547	0.035	1.552	0.122	-0.015	0.124
CF_PRC	0.1182	0.043	2.758	0.006	0.034	0.203
FCF_PRC	0.0101	0.071	0.142	0.887	-0.129	0.149
IVAL_PRC	-0.0384	0.086	-0.447	0.655	-0.208	0.131
REL_PRC	-0.2539	0.047	-5.365	0.000	-0.347	-0.161
EP1	-0.0503	0.078	-0.647	0.518	-0.204	0.103
EP2	-0.0411	0.057	-0.725	0.469	-0.153	0.071
EPS_LTG	-0.0288	0.057	-0.502	0.616	-0.142	0.084
PEG_VAL	-0.0103	0.047	-0.219	0.827	-0.103	0.082
SURP_MOM	-0.0281	0.038	-0.747	0.456	-0.102	0.046
FEPS_DELTA	0.0121	0.031	0.385	0.701	-0.050	0.074
REV_MOM	0.0284	0.030	0.948	0.344	-0.031	0.087
EPS_SURP	-0.0434	0.049	-0.889	0.375	-0.140	0.053
FDISP	-0.0114	0.042	-0.274	0.784	-0.094	0.071

Appendix E: Results from Lasso regression

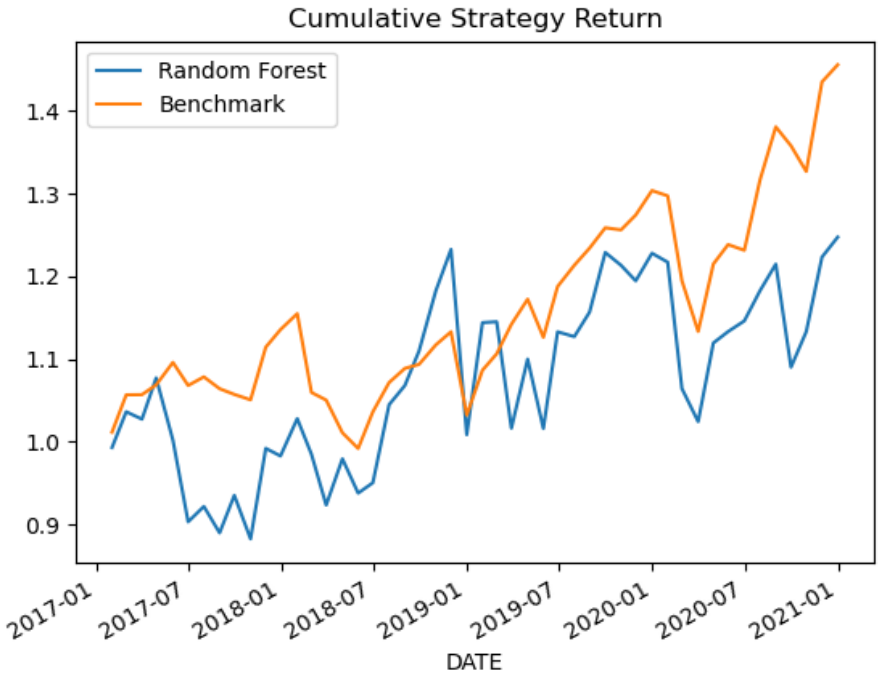
Regression Method	a.Total return on $1 invested	b.Comparable return on a value-weighted benchmark of stocks in your industry	c.Test-sample R-squared	d.Geometric average return	e.Alpha	f.Beta	g.Maximum 1-month drawdowns	g.Maximum 3-month drawdown	g.Maximum 12-month drawdowns	h.Sharpe ratio
Lasso Regression expanding window	1.0106343689694701	1.0099479907086333	-0.002445178027098205	0.0949920686858372	-0.01861737007514329	1.2172642745314584	-0.18261706826555302	-0.2258082957604033	-0.19950919252858912	0.03472701308519879
Lasso Regression rolling window	1.0105922432140053	1.0099479907086333	-0.024096178964223336	0.08740475285301996	-0.024573601818133478	1.199194242362682	-0.1904570881050941	-0.1702480383448941	-0.16815714170594975	0.03302200070970879

Appendix F: Lasso regression signal importance

```
[ 6.35614402e-07 -3.56257246e-05  0.00000000e+00 -0.00000000e+00
 -0.00000000e+00  0.00000000e+00 -0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00  0.00000000e+00
  0.00000000e+00 -0.00000000e+00  0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -8.96700775e-05 -2.23396201e-06
 -0.00000000e+00 -0.00000000e+00  1.51490777e-01  0.00000000e+00
  0.00000000e+00 -2.84199823e-02 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00]
```

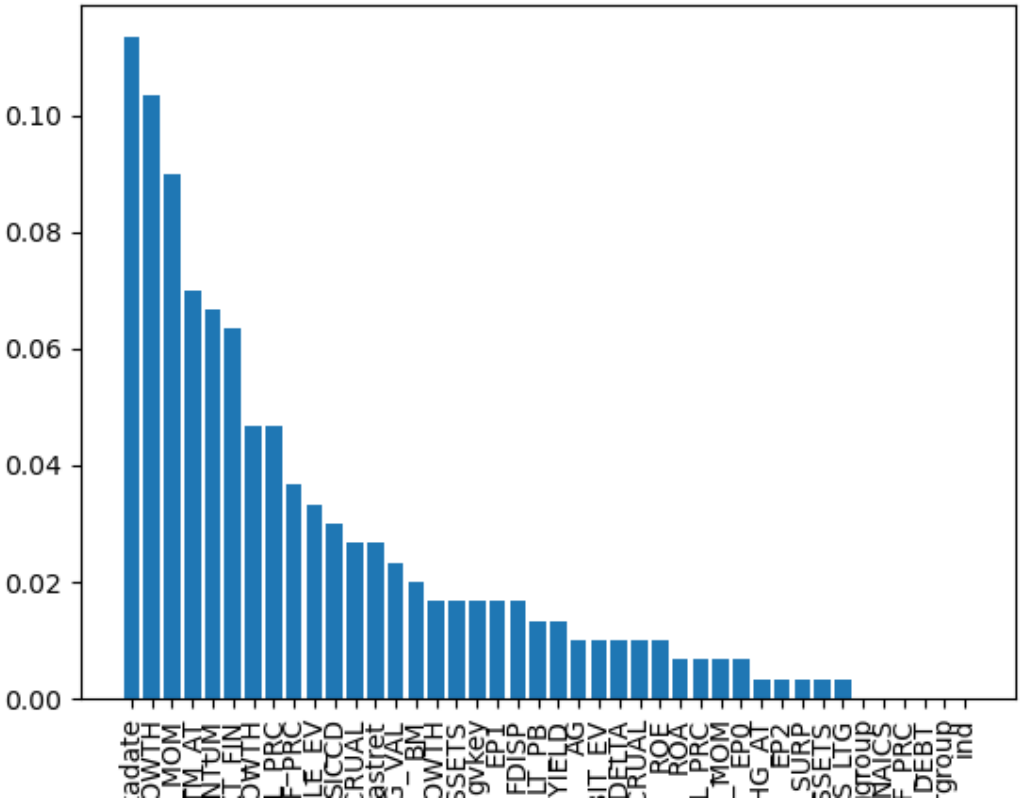
Appendix G: Results from Random forest regression

	Total return	Benchmark return	R-squared	Geometric average	Alpha	Beta	1-month drawdowns	3-month drawdowns	12-month drawdowns
0	1.074885	1.159317	-0.000243	0.011122	-0.003357	1.225609	-0.181828	-0.165853	-0.112937



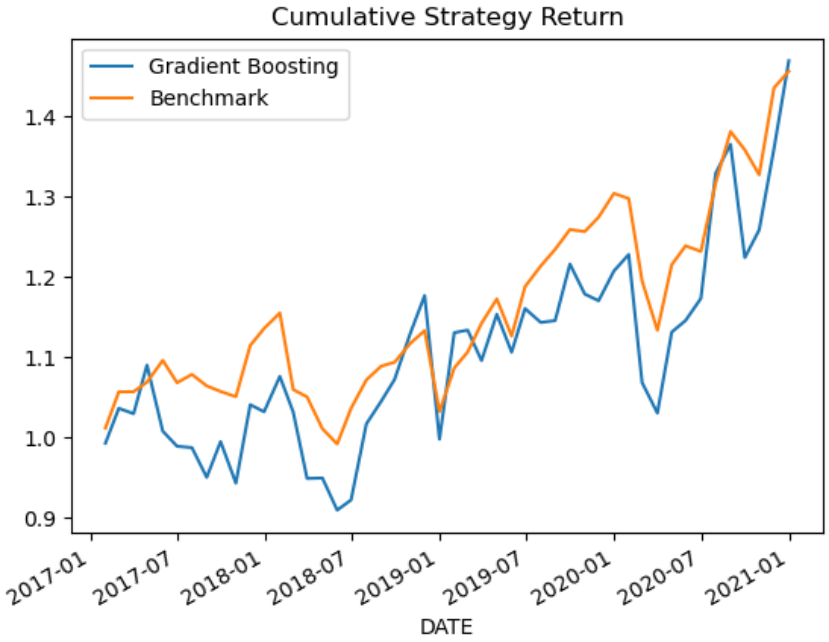
Appendix H: Random forest regression signal importance

```
[ 1 10 36 16 12 20  9 29 26  8 22 14 24 33  5 11 17  0 30 38 25  6 21  7
 35 13  3  4 28 34  2 15 31 37 18 32 39 23 27 19 40 41]
[0.01666667 0.11333333 0.00666667 0.01       0.00666667 0.02
 0.01333333 0.01       0.03333333 0.04666667 0.10333333 0.01666667
 0.06666667 0.01       0.02666667 0.00333333 0.07       0.01666667
 0.00333333 0.         0.06333333 0.01       0.03       0.
 0.02666667 0.01333333 0.03666667 0.         0.00666667 0.04666667
 0.01666667 0.00333333 0.00333333 0.02333333 0.00666667 0.01
 0.09       0.00333333 0.01666667 0.         0.         0.        ]
```



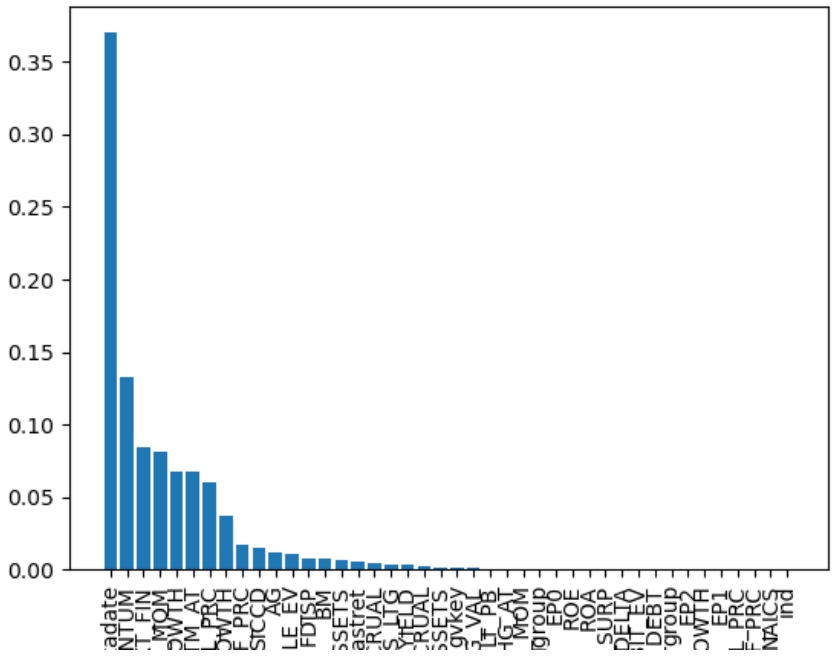
Appendix I: Results from Gradient boosting regression

```
   Total return  Benchmark return  R-squared  Geometric average    Alpha  \
0      1.103992          1.159317  -0.000625           0.019416 -0.00032

       Beta  1-month drawdowns  3-month drawdowns  12-month drawdowns  \
0  1.225056          -0.151953          -0.146534           -0.128976

   Sharpe ratio
0      0.019646
```



Appendix J: Gradient boosting regression signal importance

```
[ 1 12 20 36 10 16 29  9 26 22 21  8 38  5 17 24 14 32  6 13 18  0 33 25
 15 34 39  2  3  4 37 35  7 19 40 31 11 30 28 27 23 41]
[0.00160228 0.37015642 0.         0.         0.         0.00756088
 0.00295682 0.         0.01026775 0.03745197 0.0676942  0.
 0.13285191 0.00244748 0.00389677 0.00048228 0.06725811 0.00662562
 0.0017871  0.         0.08403864 0.01188341 0.0150854  0.
 0.00504551 0.00058682 0.01731153 0.         0.         0.05997165
 0.         0.         0.00299214 0.00080847 0.         0.
 0.08116925 0.         0.00806758 0.         0.         0.        ]
```

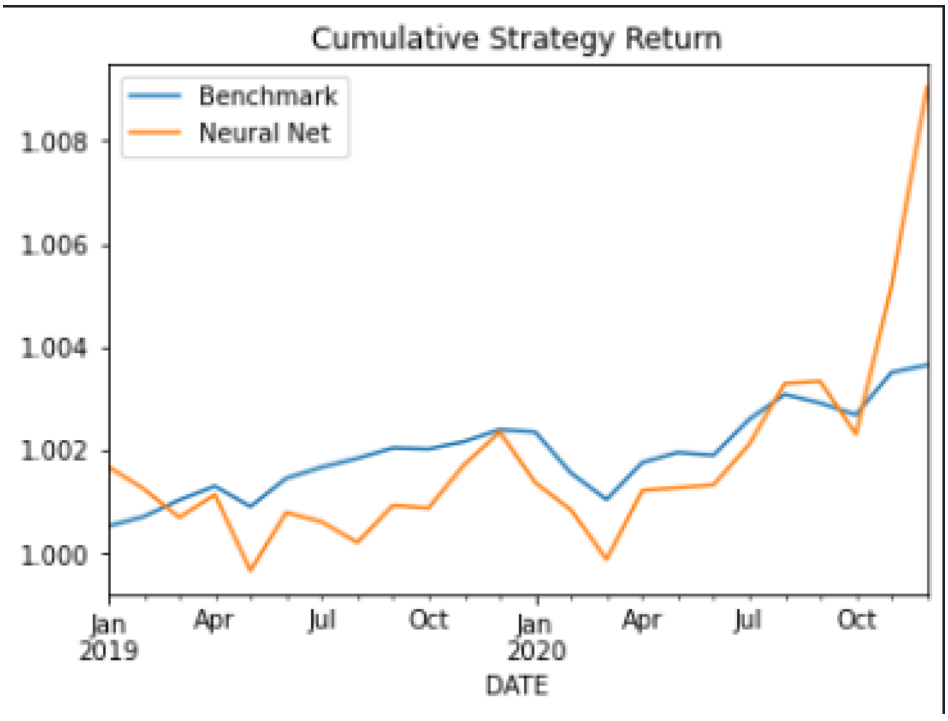


Appendix K: Results from best Neural Net pilot test on full data set

	Total return	Benchmark return	R-squared	Geometric average	Alpha	\
0	1.001793	1.001959	-0.009333	0.00045	0.125925	

	Beta	1-month drawdowns	3-month drawdowns	12-month drawdowns	\
0	2.00461	-0.001021	-0.002468	-0.000816	

	Sharpe ratio
0	-0.815376

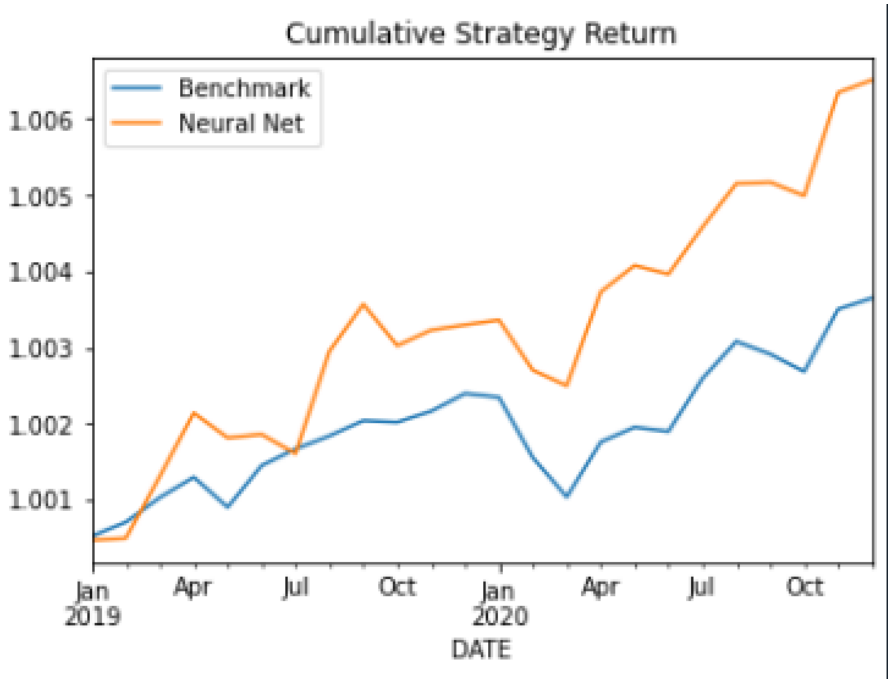


Appendix L: Results from second best Neural Net pilot test on full data set

	Total return	Benchmark return	R-squared	Geometric average	Alpha	\
0	1.003285	1.001959	-8.642964	0.000324	0.006718	

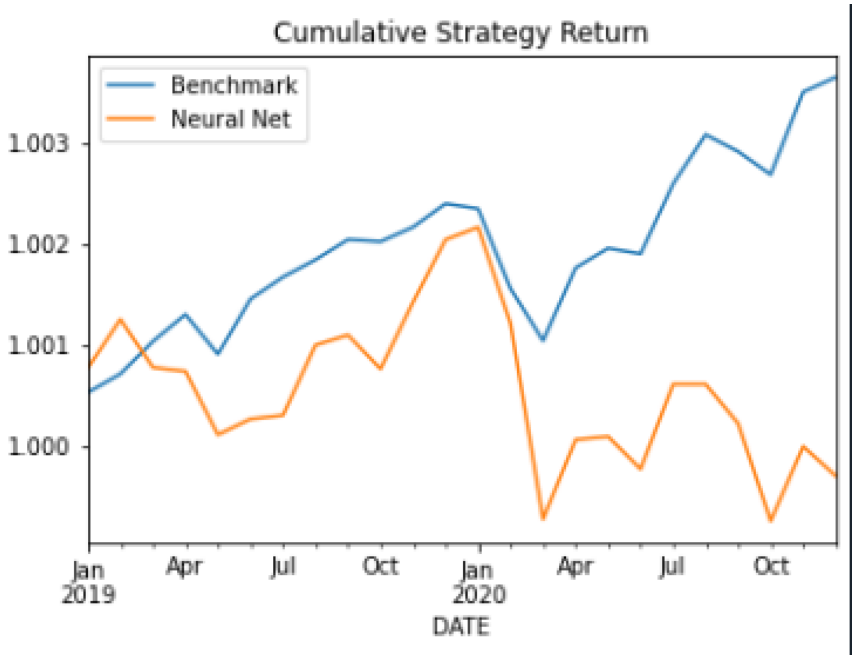
	Beta	1-month drawdowns	3-month drawdowns	12-month drawdowns	\
0	1.052571	-0.000657	-0.000796	0.001186	

	Sharpe ratio
0	-0.890965

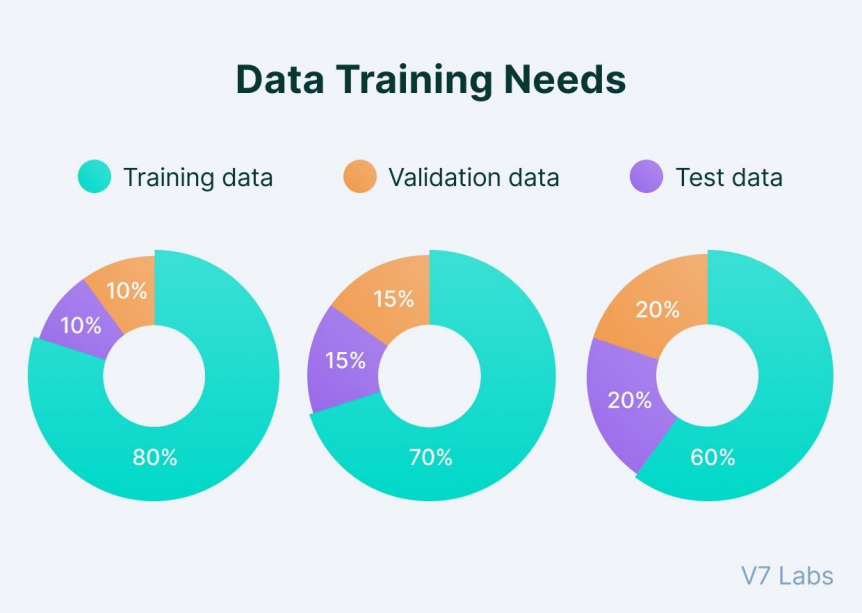


Appendix M: Results from third best Neural Net pilot test on full data set

```
   Total return  Benchmark return  R-squared  Geometric average     Alpha  \
0       1.00056          1.001959  -0.002162          -0.000015  0.047907

       Beta  1-month drawdowns  3-month drawdowns  12-month drawdowns  \
0  1.384584          -0.001936          -0.002764           -0.002335

   Sharpe ratio
0     -1.317644
```



Appendix N: Research regarding data splits



Appendix O: Analysis used to determine pilot years

Year	S&P Return (%)	1 + S&P Return	Mutiple	Salary(USD K)	Sanyang	Chinfon Bank	LICO	Total(USD K)	Life Expense	Investment (USD K)	Accumulated Wealth1(USD K)	Accumulated Wealth2(USD K)	Accumulated Wealth2(USD K)
1992	7.62	1.0762	1.0762	120				120	78	42	45.20	45.20	45.20
1993	10.08	1.1008	1.184681	120				120	78	42	95.99	91.56	94.18
1994	1.32	1.0132	1.2003187	120				120	78	42	139.81	140.24	147.07
1995	37.58	1.3758	1.6513985	120				120	78	42	250.14	191.35	204.20
1996	22.96	1.2296	2.0305596	0				0	0	0	307.57	200.92	220.53
1997	33.36	1.3336	2.7079543	286	64.3			350.3	227.695	122.605	573.68	339.70	370.59
1998	28.58	1.2858	3.4818877	286	64.3			350.3	227.695	122.605	895.28	485.42	532.65
1999	21.04	1.2104	4.2144768	286	64.3			350.3	227.695	122.605	1232.05	638.42	707.67
2000	-9.1	0.909	3.8309595	286	64.3			350.3	227.695	122.605	1231.38	799.08	896.70
2001	-11.89	0.8811	3.3754584	286	64.3			350.3	227.695	122.605	1193.00	967.77	1100.85
2002	-22.1	0.779	2.6294821	286	64.3			350.3	227.695	122.605	1024.85	1144.89	1321.33
2003	28.68	1.2868	3.3836175	286	64.3			350.3	227.695	122.605	1476.55	1330.87	1559.45
2004	10.88	1.1088	3.7517551	321	64.3			385.3	250.445	134.855	1786.73	1539.02	1829.85
2005	4.91	1.0491	3.9359663	321	64.3			385.3	250.445	134.855	2015.93	1757.56	2121.88
2006	15.79	1.1579	4.5574554	321	64.3			385.3	250.445	134.855	2490.39	1987.04	2437.28
2007	5.49	1.0549	4.8076597	321	64.3			385.3	250.445	134.855	2769.38	2227.99	2777.90
2008	-37	0.63	3.0288256	321	64.3			385.3	250.445	134.855	1829.67	2480.99	3145.78
2009	26.46	1.2646	3.8302529	321	64.3			385.3	250.445	134.855	2484.33	2746.63	3543.08
2010	15.06	1.1506	4.4070889	321	64.3		64.3	449.6	292.24	157.36	3039.53	3049.19	3996.48
2011	2.11	1.0211	4.5000785	321	64.3		64.3	449.6	292.24	157.36	3264.35	3366.88	4486.14
2012	16	1.16	5.2200911	321	64.3			385.3	250.445	134.855	3943.07	3676.82	4990.68
2013	32.39	1.3239	6.9108786	321	64.3			385.3	250.445	134.855	5398.77	4002.26	5535.58
2014	13.69	1.1369	7.8569778	321	64.3			385.3	250.445	134.855	6291.18	4343.97	6124.07
2015	1.38	1.0138	7.9654041	321	64.3			385.3	250.445	134.855	6514.71	4702.77	6759.64
2016	11.96	1.1196	8.9180665	0				0	0	0	7293.87	4937.91	7300.41
2017	21.83	1.2183	10.86488	0				0	0	0	8886.12	5184.80	7884.44
2018	-4.38	0.9562	10.388999	0				0	0	0	8496.91	5444.04	8515.19
2019	31.49	1.3149	13.660494	0				0	0	0	11172.59	5716.25	9196.41
2020	18.4	1.184	16.174025	0				0	0	0	13228.34	6002.06	9932.12
2021	28.71	1.2871	20.817588	0				0	0	0	17026.20	6302.16	10726.69

Appendix P: Scoring System

Data

Models	Total Return	Benchmark Return	R-squared	Geometirc Average	Alpha	Beta	1M Drawdowns	3M Drawdowns	12M Drawdowns	Sharpe Ratio	
Linear Expand	1.011097	1.009948	-0.009016	0.105380	0.000940	1.115681	-0.231045	-0.215972	-0.178927	0.029611	
Lasso rolling	1.010592	1.009948	-0.024096	0.087405	-0.024574	1.199194	-0.190457	-0.170248	-0.168157	0.033022	
Random Forest	1.074885	1.159317	-0.000243	0.011122	-0.003357	1.225609	-0.181828	-0.165853	-0.112937	0.009570	
Gradient Boosting	1.103992	1.159317	-0.000625	0.019416	-0.000320	1.225056	-0.151953	-0.146534	-0.128976	0.019646	
Neural	1.003285	1.001959	-8.642964	0.000324	0.006718	1.052571	-0.000657	-0.000796	0.001186	-0.890965	
Weight	0.03	0	0.03	0.03	0.05	0.6	0.05	0.05	0.05	0.1	1

Ranking

Models	Total Return	Benchmark Return	R-squared	Geometirc Average	Alpha	Beta	1M Drawdowns	3M Drawdowns	12M Drawdowns	Sharpe Ratio	Rank Scores
Linear Expand	3	3	3	5	4	4	1	1	1	4	3.05
Lasso rolling	2	2	2	4	1	3	2	2	2	5	2.55
Random Forest	4	4	5	2	2	1	3	3	4	2	1.95
Gradient Boosting	5	4	4	3	3	2	4	4	3	3	2.60
Neural	1	1	1	1	5	5	5	5	5	1	3.30

Appendix Q: Prediction results

	tic	NAME	EP0	ROE
87	PFGC	Performance Food Group Company	202.796700	1.417105
9	CLX	Clorox Company	73.482110	32.999330
7	CASY	Casey's General Stores, Inc.	25.062420	15.773310
100	BJ	BJ's Wholesale Club Holdings, Inc.	22.596820	88.225050
90	COST	Costco Wholesale Corporation	48.661180	31.439980
8	CHD	Church & Dwight Co., Inc.	31.092210	26.464760
45	SPB	Spectrum Brands Holdings, Inc.	29.299900	3.559078
142	PSMT	PriceSmart, Inc.	24.433270	11.226860
138	UNFI	United Natural Foods, Inc.	12.138760	15.709540
5	CPB	Campbell Soup Company	15.542370	29.559950
35	SYY	Sysco Corporation	58.468400	59.328720
147	EPC	Edgewell Personal Care Co.	19.025220	7.323944
66	SFM	Sprouts Farmers Markets, Inc.	15.455930	26.521950
167	GO	Grocery Outlet Holding Corp.	54.227830	6.451717
43	WBA	Walgreens Boots Alliance Inc	6.419842	24.791900
23	KR	Kroger Co.	26.544600	17.237750

	BM	DIVYIELD	EBIT_EV	SALE_EV	OCF_GROWTH	SALES_GROWTH
87	0.380528	0.000000	0.022462	2.997777	2.485153	0.489948
9	0.017398	2.495692	0.048419	0.333882	1.839431	-0.059011
7	0.272867	0.594086	0.049092	1.191413	-0.554999	0.461398
100	0.068593	0.000000	0.072810	1.347635	0.456415	0.080187
90	0.072591	2.882267	0.037053	0.793258	5.214775	0.176867
8	0.129240	0.985366	0.035562	0.188660	1.435428	0.060113
45	0.361608	1.756036	0.031416	0.586150	-1.716127	-0.096819
142	0.376186	0.827325	0.065348	1.457866	0.146084	0.114806
138	0.621022	0.000000	0.078827	4.436000	0.531174	0.029087
5	0.235545	3.339433	0.067392	0.434039	1.384385	-0.073954
35	0.027007	2.376289	0.033537	1.090158	4.926302	0.323226
147	0.755031	1.652893	0.099560	0.629962	-0.061568	0.078436
66	0.267313	0.000000	0.097936	1.257129	-0.735684	-0.057026
167	0.309190	0.000000	0.028466	0.671151	-0.292866	-0.017564
43	0.664229	3.704433	0.075227	1.664206	-0.894036	-0.018156
23	0.226477	1.863354	0.086615	2.327268	0.340374	0.040680

	LT_PB	CF_PRC	FCF_PRC	IVAL_PRC	REL_PRC	EP1	EP2
87	2.218558	0.029230	-0.142031	0.574451	0.904280	0.047549	0.062301
9	11.711700	0.048302	0.061308	0.129468	0.743453	0.028014	0.037563
7	2.048633	0.084590	-0.045670	0.436052	0.941749	0.040796	0.043645
100	3.236219	0.088018	0.055587	0.254623	0.942639	0.046535	0.050830
90	9.910463	0.037129	0.011211	0.159295	0.985725	0.021678	0.023785
8	31.470340	0.039725	0.010165	0.252719	0.980314	0.030850	0.033231
45	5.285128	0.034372	-0.057871	0.494502	0.847790	0.033718	0.051045
142	2.470293	0.035304	0.030397	0.550090	0.860270	0.043468	0.047973
138	0.486827	0.165951	0.047843	0.977540	0.803939	0.088956	0.095617
5	5.692417	0.083671	0.113624	0.471207	0.927405	0.058973	0.060670
35	8.381405	0.032915	0.103402	0.163948	0.994420	0.034231	0.046240
147	3.823285	0.113087	-0.041635	1.050577	0.733128	0.074303	0.082194
66	1.989927	0.135145	0.085591	0.537186	0.919921	0.067866	0.071805
167	2.158557	0.044866	-0.007665	0.419114	0.803736	0.027569	0.032362
43	1.146572	0.128955	0.135622	1.097404	0.831665	0.108294	0.106790
23	1.487765	0.162813	0.097377	0.488881	0.919083	0.065889	0.069757

	AVG_SALES_GROWTH	EPS_MOMENTUM	ANN_ACCRUAL	LTM_ACCRUAL	CHG_AT
87	89.062480	3.0	0.009428	0.274603	-0.164425
9	6.326143	0.0	-0.042401	-0.125412	-0.016352
7	283.646600	3.0	0.057639	0.160295	-0.314511
100	13.848100	6.0	0.009518	0.009518	0.119261
90	33.072850	6.0	-0.005330	0.047883	0.109047
8	32.818220	6.0	0.076504	0.076504	-0.022269
45	6.715765	2.0	0.008404	0.057879	-0.193046
142	9.051962	5.0	0.012374	0.049825	-0.101594
138	105.427800	5.0	0.017768	0.061617	-0.131426
5	202.032800	1.0	-0.047291	-0.046633	0.022115
35	217.750800	6.0	-0.129999	-0.130939	-0.276409
147	21.925790	3.0	0.046372	0.081252	0.018446
66	304.245500	2.0	0.037432	0.037432	-0.276337
167	182.153700	0.0	0.062362	0.062362	-0.152476
43	9.672335	3.0	-0.052594	0.071851	-0.229940
23	58.859790	3.0	-0.005180	-0.005180	0.003025

	LTM_AT	NOA_ASSETS	RD_ASSETS	EBITDA_DEBT	NET_EXT_FIN	pastret
87	0.133819	1.322058	0.000000	0.138932	0.213355	15.174920
9	-0.136550	0.681260	0.022974	0.344371	-0.632970	-7.954115
7	0.456674	1.030932	0.000000	0.414081	0.553597	13.786820
100	0.119261	0.685551	0.000000	1.067709	-0.346214	19.609520
90	0.118102	0.599032	0.000000	1.454040	-0.086135	28.875260
8	-0.022269	0.896311	0.013967	0.466347	-0.204394	23.980330
45	-0.117034	0.866253	0.007530	0.136431	0.070964	-2.974421
142	0.054158	0.832584	0.000000	1.410917	-0.022232	3.989434
138	0.075075	0.686892	0.000000	0.272465	-0.033517	5.532885
5	-0.064363	0.738974	0.007323	0.282362	-0.816162	16.099070
35	0.573067	0.676071	0.000000	0.258981	-0.461457	12.931010
147	0.002306	0.929069	0.015375	0.242860	0.098873	9.204304
66	-0.276337	0.881769	0.000000	2.097889	-0.541364	46.046710
167	-0.152476	1.008099	0.000000	0.347963	0.056509	47.462050
43	-0.049727	0.780174	0.000000	0.486828	-0.327627	-1.905537
23	0.003025	0.623583	0.000000	0.536740	-0.333744	48.148740

	EPS_LTG	PEG_VAL	SURP_MOM	FEPS_DELTA	REV_MOM	EPS_SURP	FDISP
87	23.337580	0.538974	0.005916	-0.000235	0.857143	0.002315	0.000612
9	-3.850108	0.012885	-0.002831	-0.029280	-1.437500	0.004690	0.001142
7	10.045460	0.235719	0.000898	0.002837	0.583333	0.003609	0.001019
100	13.581760	0.279210	0.007594	0.001825	-0.391304	0.005956	0.000713
90	13.562210	0.152192	0.002330	0.005279	0.935484	0.001654	0.000278
8	7.645863	0.159649	0.003617	-0.006344	-0.368421	0.001665	0.000195
45	21.095160	0.351955	-0.012899	-0.119136	-0.166667	0.012039	0.002342
142	8.000000	0.226032	0.001018	-0.020576	-0.500000	0.002817	0.000268
138	9.195983	0.505126	0.024111	-0.000375	-0.538462	0.017794	0.001444
5	1.639772	0.156628	0.004886	-0.000458	-0.111111	0.005178	0.000348
35	-1.500000	0.047924	-0.004052	-0.006107	-1.117647	0.001770	0.000595
147	4.000000	0.267491	0.011565	-0.012247	-1.200000	0.009287	0.000870
66	10.442570	0.332939	0.014255	-0.000764	1.000000	0.010247	0.000656
167	14.828320	0.158317	0.000835	0.000000	-0.928571	0.000937	0.000367
43	1.504898	0.281776	0.029300	0.004658	0.210526	0.014391	0.001502
23	9.688519	0.392091	0.015547	0.004460	0.909091	0.007867	0.001263

	pred_neural
87	-15.603782
9	-8.766266
7	-6.224753
100	-5.991754
90	-5.862285
8	-4.444230
45	-4.095740
142	-3.211796
138	-0.981041
5	-0.356756
35	-0.072876
147	0.640425
66	1.217233
167	1.588327
43	2.240764
23	4.205029