AFM 423 – Topics in Financial Econometrics

GPR #2 Final Report

Application of Machine Learning Models in Factor Investing to Improve Equity Return Prediction and Portfolio Construction

By: Samuel Benedict

Winter 2023

Abstract

The emergence of new return factors in explaining the variability of asset returns prompted us to check if various machine learning models can be applied to improve the prediction of returns and construct a profitable investment strategy. We gathered stock-specific factors alongside some macro factors and performed necessary transformations to ensure that the data is non-stationary. The return factors are then trained on models that can accommodate non-linearity or parameter regularization by using a rolling train/test window. The predictions from these models were also used to construct a portfolio that can be compared against benchmarks such as the market portfolio and equal-weighted portfolio. We assessed the viability of the various models using accuracy parameters and the risk/return profile of its generated portfolio. We found that the autoencoder was the best model to minimize the errors of the return prediction. Meanwhile, the random forest generated the highest portfolio returns and Sharpe ratio while XGBoost delivered the lowest portfolio volatilities. Several portfolios generated by random forest, neural network, and autoencoders were able to beat the market portfolio. We also found that a leveraged long/short strategy can outperform a long/only strategy provided that the predictor is consistently right in classifying the best stocks to long and short.

Introduction

Since the discovery of the three-factor model by Fama and French in 1993, factor investing with multiple linear regression (OLS) has become a prominent trend in asset management. However, new financial research papers have proven that conducting linear regression on these three factors (market risk premium, size, and value) is insufficient to explain asset returns, hence, the need to incorporate additional factors that can capture more variability in asset returns (Fama & French, 2015). Some potential factors that can explain variability in asset returns include fundamental factors such as profitability and investment (Fama & French, 2015), technical factors such as momentum (Carhart, 1997), and macro factors such as inflation (Feldstein, 1983). As discussed in the literature review of the previous report (GPR #1), these additional factors will present multicollinearity, where the factors might be correlated to each other and would violate the assumptions of a linear regression model. Moreover, these factors might not impact the returns linearly as assumed by linear regression.

With new advancements in statistics, novel machine learning models can now be used as a method to predict asset returns. For example, regularization can perform feature selection that minimizes the collinearity between various factors. Tree-based methods can display the feature importance, giving insight into important factors in the prediction. Neural networks can induce non-linear terms on the factors used to predict stock returns. Finally, autoencoders can be used to detect factor anomalies and reduce noise in the data.

Additionally, it would be interesting to see how these factor models can be used in constructing portfolios that would bring excess return compared to the market. Using the portfolio sorting as conducted by Fama and French (1993), we can test whether the machine learning predictions can be used to construct a profitable long-only, long-short, or leveraged long-short portfolio strategy.

Research Question

In this report, we are going to address two main questions relating to the implementation of ML models in factor investing:

1. Does the implementation of machine learning models (LASSO, Ridge Regression, Random Forest, XGBoost, Neural Networks, or Autoencoders) improve the prediction of equity returns compared to using standard multiple linear regression?
2. From these machine learning predictions, are there any portfolio construction strategies (Long-Only, Long-Short, or Leveraged Long-Short) that can achieve superior returns compared to the market?

Variables and Measures

To minimize the influence of time in the predictions, we need to make the monthly returns stationary across time by doing transformations such as taking a natural log of the return. Hence, our response variable is the monthly log return, i.e., Log(R1M_USD + 1) , where

$$\log(R1M_{USD}+1)=\log\left(\frac{P_{i+1}-P_i}{P_i}+1\right)=\ \log\left(\frac{P_{i+1}}{P_i}\right)$$

Based on our literature, the factors that might explain anomalies in equity returns are:

1. Market Risk Premium (Rm – Rf: Market – Risk-Free Return)
2. Market Capitalization (SMB: Small – Big Market Cap)
3. Book-to-Market Value (HML: High – Low Book-to-Market)
4. Momentum (WML: Winners – Losers)
5. Profitability (RMW: Robust – Weak Profits)
6. Capital Investment (CMA: Conservative – Aggressive Investments)
7. Inflation (1-Month Growth in Core CPI reading

Thus, we will be using these internal and external factors to predict future equity returns:

1. <u>Mkt.RF</u>: Market Risk Premium
   Market return – Risk-Free return, a macro factor obtained from the Kenneth R. French data library.
2. <u>Mkt_Cap_6M_USD</u>: 6-Month Average Market Capitalization
   Average of market capitalization (stock price * shares outstanding) over 6 months.
3. <u>Pb</u>: Price-to-Book Ratio
   The inverse of the Book-to-Market ratio, calculated as stock price per book value.
4. <u>Mom_Sharp_5M_USD</u>: 5-Month Sharpe-Adjusted Price Momentum
   The difference between 1-month ago to 6-month ago stock price, divided by its 5-month price volatility. This adjustment works so that each stock's price momentum (trend) can be compared relative to its volatility, which serves as a better comparison to other stocks.
5. <u>Ebitda_Margin</u>: EBITDA Margin
   Percentage difference between a company's EBITDA to its revenue. A higher EBITDA margin indicates that the company earns more profit per unit of sales.
6. <u>Capex_Sales</u>: Capital Expenditure per Sales
   The ratio between a company's capital expenses and its sales revenue. A higher Capex/Sales indicates that the company is spending more aggressively to push its sales.
7. <u>R1M_log_inflation</u>: 1-Month Log-Inflation
   Log-growth of monthly CPI reading obtained from the US Bureau of Labor Statistics. This macro factor is log-transformed to make the inflation data stationary.

The resulting prediction is also a log-return, so we need to convert back to the original return as exp(predicted log return) – 1, i.e.

$$\exp\left(\log\left(\widehat{R1M_{USD}}+1\right)\right)-1=\ \log\left(\widehat{R1M_{USD}}+1\right)\ -1=\ \widehat{R1M_{USD}}$$

To assess if our model is better than the benchmark (standard linear regression), we are using these prediction metrics:

1. Mean-Squared Error (MSE): $\frac{1}{n}\sum\left(\widehat{Y_\iota}-Y_i\right)^2$
2. Mean Absolute Error (MAE): $\frac{1}{n}\sum|\widehat{Y_\iota}-Y_i|$
3. Hit Ratio: $\frac{1}{n}\sum(\widehat{Y_\iota}*Y_i\geq 0)$

Where $\widehat{Y_\iota}=predicted\ return\ \ and\ \ Y_i=actual\ return$

Meanwhile, we can use the following metrics to determine the risk and reward features of long-only and long-short portfolios from the prediction:

1. (Geometric) Annualized Return / CAGR: $\bar{R} = \Pi(1+R_i)^{\frac{1}{t}} - 1$
2. Annualized Volatility (Standard Deviation): $\sigma_r = \sigma_{monthly\ ret} * \sqrt{12}$
3. Sharpe Ratio: $(\bar{R} - r_f)/\sigma_r$

Where $R_i = return\ for\ month\ i \quad and \quad r_f = annualized\ riskfree\ return$

Application of the Machine Learning Approach to Factor Investing

The machine learning predictors that we will be considering for return prediction are as follows: (* all formulas are obtained from the RM files in the class lectures)

1. Linear Regression

$$r_{i,t} = \alpha_i + \sum_{k=1}^{K} \beta_{i,k} f_{k,t} + \varepsilon_{i,t}$$

   Standard Ordinary Least-Squares Regression is the baseline (simplest) model to output a prediction of a return given this month's factors. This model assumes that all factors linearly influence asset returns to a degree as big as its respective coefficients. Also, it assumes that the errors follow a normal distribution with a constant variance. This would create an unbiased estimator of the return statistically.

   Aside from the abundance of assumptions, the downside of this model is that as more predictors are included, we are faced with the curse of dimensionality. Specifically, with higher number of dimensions (features), the distance between datapoints is similarly further apart and the estimate would then be less accurate. Also, models with higher number of predictors are usually prone to overfitting of the training data.

2. Linear Regression with LASSO (L1-Regularization)

$$y_t = \beta_0 + \beta_1 x_{1t} + \cdots + \beta_p x_{pt} + \varepsilon_t, \quad t = 1, \ldots, T \text{ subject to } \sum_{j=1}^{p} |\beta_j| < \delta$$

   The LASSO-induced regression introduces a penalization constraint on the coefficients, which shrinks some factor coefficients to 0 and reduces the dimensionality of the model. The parameter that tunes this penalization constraint is lambda, and we can tune the model to use the lambda that minimizes the errors by using cross validation.

3. Ridge Regression (L2- Regularization)

$$y_t = \beta_0 + \beta_1 x_{1t} + \cdots + \beta_p x_{pt} + \varepsilon_t, \quad t = 1, \ldots, T \text{ subject to } \sum_{j=1}^{p} \beta_j^2 < \delta$$

   Similar to LASSO, ridge regression is also built on linear regression with a penalization constraint on the coefficients. The difference is that while the coefficients are squared here (on an L2 norm), the coefficients in LASSO are on absolute value (L1 norm). This makes the model able to shrink the coefficients to be close to 0 but never completely zero. As in LASSO, we can also tune ridge regression by selecting the best lambda.

4. <u>Random Forest</u>
   One of the ways to introduce non-linearity in prediction is by using decision trees, where it classifies different values by its factor characteristics in form of a decision (is this factor bigger than or less than a certain threshold). It aims to create clusters that are homogeneous, meaning the values within a group are kept so that they are closely related to one another (reduce variability within cluster). However, decision trees prone to overfitting on the training data, especially if the trees are deep and more specific.

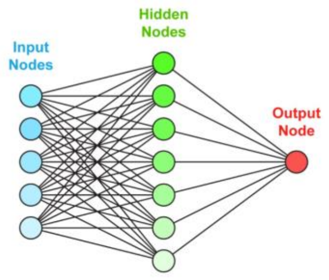
   A way to tackle this is to create simpler and shallower trees. Although these trees are not a good predictor individually, we can create many of these trees (through bootstrapping) to form a weighted prediction which is a better "wisdom of the crowd" prediction. Moreover, tree-based methods like random forest can also tell us the variable importance of each factor. To tune the random forest, we can tune some of its parameters such as maxnodes, maxdepth, number of trees, cp, and subsamples.

5. <u>XG Boost (Extreme Gradient Boosting)</u>

$$O = \underbrace{\sum_{i=1}^{I}[y_i - m_{J-1}(x_i) - T_J(x_i)]^2}_{error\ term} + \underbrace{\sum_{j=1}^{J}\Omega(T_j)}_{\substack{regularization \\ term}}$$

   Another way to tackle some of the problems in decision trees is by boosting the tree in each iteration, in which the tree is optimized with respect to its gradient until the iteration stops or converges. An extreme gradient boosting extends the above approach by also introducing a regularization term which limits the coefficients assigned to each factor. This would help the model to minimize the loss in a way that does not overfit the training data. We can tune on some of its parameters such as maxdepth, eta (learning rate / step size), gamma (minimum loss required to split), lambda (regularization parameter), and number of rounds (maximum iterations).

6. <u>Neural Network (Multi-Layer Perceptron)</u>

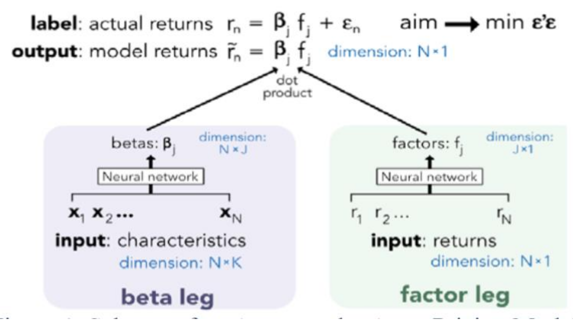


   In an MLP, we have input layer consisting of our factor values (nodes), the hidden layers consisting of weights which try to model the prediction, and the output layer which spits out the prediction. An activation function is required to transform the values from one layer to another until it reaches the output. Once the output is crosschecked with the actual validation data, the weights in the hidden layer are re-adjusted via backpropagation to make a better prediction at the next iteration. This process continues until the specified number of iterations. This will then minimize the specified loss function for the training and validation set.

Aside from cross-validating, one of the ways to minimize overfitting is by introducing dropout, in which some nodes are randomly dropped out of the model to reduce the model complexity. There are several parameters that we can tune for MLP, including the number of layers, the number of nodes on each layer, the activation function on each layer, the dropout layers, the optimizer, loss function, and number of epochs (iterations).

7. Autoencoder

$$r_{i,t} = \boldsymbol{NN}_{beta}(\boldsymbol{x}_{i,t-1}) + \boldsymbol{NN}_{factor}(\boldsymbol{r}_t) + e_{i,t}$$



As proposed by Gu, Kelly, and Xiu in their 2021 paper, the autoencoder asset pricing model is constructed with 2 neural networks, namely the beta side which captures the linear relationship of factors and the factor side which captures the nonlinear relationship between the factors themselves. More specifically, the beta side acts like the coefficients in linear regression, which aims to capture the systemic risk (macro risk) of the returns. On the other hand, the factor side captures the nonlinear relationship using an autoencoder which aims to capture idiosyncratic risk (stock-specific risk) among the stocks. By combining the two sides, the model can predict the asset returns nonlinearly. As with the standard MLP, we can tune its nodes and layers on each side as well as the number of epochs (iterations).

For each month in the testing period, the prediction results from the above model would then be used to construct various portfolio strategies as follows:

1. Long Only
   Buy the stocks that are in the top 20% of predicted return for the next month, equal-weighted. This would be rebalanced each month, we assume no transaction cost.

2. Long/Short
   Buy the stocks that are in the top 20% of predicted return for the next month and short sell the stocks that are in the bottom 20% of predicted return for the next month, all equal-weighted. This would be rebalanced each month, we assume no transaction cost and no interest for shorting.

3. Leveraged Long/Short
   Buy the stocks that are in the top 20% of predicted return for the next month, scaling it by 150%. Short sell 50% of the stocks that are in the bottom 20% of predicted return for the next month, scaling it by 50%. The net investment here is still 100%, which satisfies the budget (wealth) constraint. This would be rebalanced each month, we assume no transaction cost, no interest for leverage, and no interest for shorting.

The metrics to assess prediction accuracy and portfolio profitability are inspired from Nakagawa et al.'s 2019 research paper on deep recurrent factor model to predict stock returns based on non-linear multi-factor models.

Experimental Methodology

Data Overview

In this report, we used data_ml.xlsx, which is metadata of monthly stock returns from 1207 US stocks recorded from November 1998 to March 2019. For each stock and month, we have 93 different fundamental and technical characteristics (factors) of the stock, alongside its monthly returns. Of the 1207 distinct stocks, only 363 of them are recorded in all possible dates (245 consecutive months). Moreover, the factors in the metadata are already uniformized, ranging from 0 (lowest) to 1 (highest).

Data Preprocessing

1. From the data_ml metadata, keep the monthly return column (R1M_USD) and columns whose features are used to construct our factors (Mkt_Cap_6M_USD, Pb, Mom_Sharp_5M_USD, Ebitda_Margin, Capex_Sales).
2. Delete outliers (stocks with return above 500% on an occasion) and only select the stocks that are recorded for the whole 245 months.
3. Join the Ken French factor library and CPI dataset to obtain the 1-Month inflation and market risk premium.
4. Log transform the response variable (R1M_USD) and 1-Month inflation to minimize the non-stationarity of the data.
5. Re-uniformize the factors again for each month to make the data smoother, except for market risk premium and inflation.

Factor Diagnostics with Factor Regression (not directly used for prediction)

1. For each month and each factor (except for market risk premium and inflation), sort the stocks by each factor values. Then, group them into 5 quintiles. Alternatively, we can use the FF factor data from the Ken French library.
2. Take the average monthly log returns from the top and bottom quintiles (best 20% and worst 20% of factor value). Then, take the differences as applicable.
3. Do this for all the available dates, then join the data alongside market risk premium and 1-month log inflation.
4. Check which factors are significant in explaining the returns. This acts as a sanity check before we put the factors for prediction, as well as telling us which factors we should expect to be highly important for prediction.

Prediction of Monthly Returns

1. Split the portfolio into sliding train/validation and testing sets, as well as features (x) and labels (y). We have 245 months, so we use months 1 to 180 to predict returns on month 181. Next, use months 2 to 181 to predict month 182, and months k to k+179 to predict month k+180. We do this until we can predict month 245. In total, we will have 65 monthly predictions per predictor.
2. For each predictor other than linear regression, we use cross-validation on the train/validation portion (180-month sliding window) to tune its parameters.
   - For LASSO and Ridge Regression, tune for lambda before each prediction.
   - For Random Forest and XGBoost, spot check on some training and testing dates, then perform tuning with gridsearch. Set the tuned parameters for all predictions.
   - For Neural Networks and Autoencoders, spot check on some training and testing dates and set the tuned model (layers, nodes), optimizer, and epoch for all predictions.
3. Fit in the training features and training labels to each predictor (ML model). Then, predict the log return using the testing features. Iterate this step through all 65 months.
4. Convert the predicted log return back to the monthly return for actual comparison.
5. The predicted return is then compared to the actual return for each month using the accuracy metrics (MSE, MAE, Hit Ratio). Aggregate the results for each predictor to get an understanding of the predictive accuracy.

Portfolio Construction

1. For each predictor and each month (prediction), sort the stocks by predicted return and group them into 5 quintiles.
2. Then, construct the following portfolio strategies:
   - Long-only: average the actual return of the stocks that are on the top quintile of predicted returns.
   - Long/short: apply long-only strategy and subtract the average of actual return of the stocks that are on the bottom quintile of predicted returns.
   - Leveraged long/short: apply 1.5x long-only strategy and subtract 0.5x short-only strategy.
3. Record the portfolio return for each month and store it in an array of cumulative returns. Rebalance this strategy every month.
4. We also consider the following benchmark portfolios:
   - Market portfolio: use monthly return from the market risk premium + risk-free return from the Ken French factor library, let it compound for 65 months.
   - Risk-free portfolio: use monthly risk-free return from the Ken French factor library, let it compound for 65 months.
   - Equal-weight (1/N) portfolio: use the average monthly return of our whole stock universe (363 stocks) assuming that each stock is equally invested. Let it compound for 65 months.
5. Aggregate the risk and return of the various portfolio strategies using the portfolio metrics (annual return, annual volatility, Sharpe Ratio) to get an understanding of the portfolio characteristics and profitability.

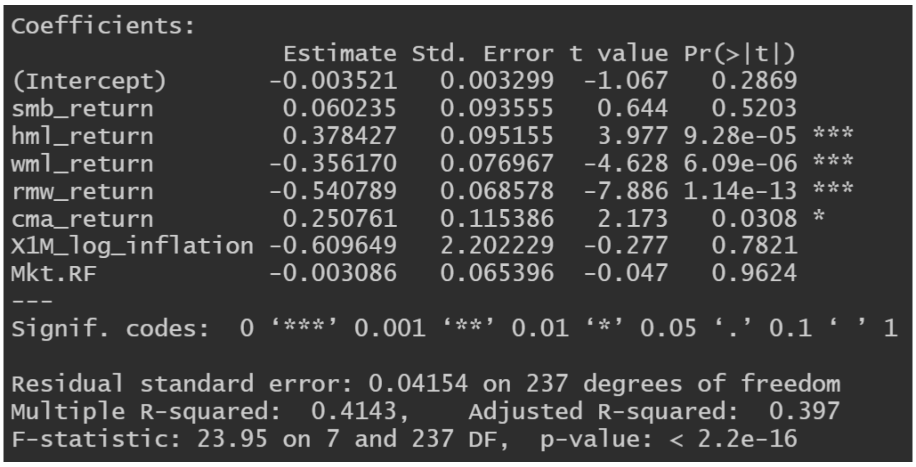
Related Work

GPR #2 follows the main points and guidelines as the proposed GPR #1, with some minor tweaks and changes. We would not proceed with incorporating stock beta and other value factors into consideration, and we have replaced the recurrent neural network predictor with autoencoders. We would also not strictly discarding the factors that are not significant in the factor diagnostics out of the predictor since the current data does not necessarily reflect the findings of the factor anomalies as laid out by Fama & French in their 1993 and 2015 paper.

Results and Discussion

Factor Diagnostics (not directly used for prediction)

The result of our Fama-French factor regression (manually getting the average return between quintiles then regressing it) are as follows:



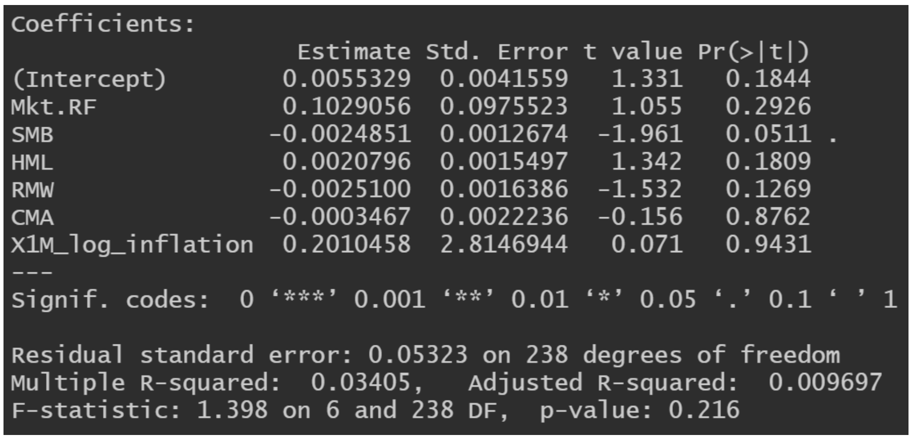
```
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       -0.003521   0.003299  -1.067   0.2869
smb_return         0.060235   0.093555   0.644   0.5203
hml_return         0.378427   0.095155   3.977 9.28e-05 ***
wml_return        -0.356170   0.076967  -4.628 6.09e-06 ***
rmw_return        -0.540789   0.068578  -7.886 1.14e-13 ***
cma_return         0.250761   0.115386   2.173   0.0308 *
X1M_log_inflation -0.609649   2.202229  -0.277   0.7821
Mkt.RF            -0.003086   0.065396  -0.047   0.9624
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.04154 on 237 degrees of freedom
Multiple R-squared:  0.4143,    Adjusted R-squared:  0.397
F-statistic: 23.95 on 7 and 237 DF,  p-value: < 2.2e-16
```

Figure 1: Factor Regression with quintiles from Metadata fitted to our log-return.

From the summary above, we can see that HML (value), WML (momentum), RMW (profitability), and CMA (investment) are all significant up to 5% significance as its p-value is below 0.05. We further confirm this by performing backward selection until all the factors are significant. This is a contrasting result compared to the Fama & French result in their 2015 paper, where they stated that SMB (size) and Market Risk Premium should be a significant factor but not HML as it should be a redundant factor with the addition of RMW and CMA (Fama & French, 2015). This suggests that despite the FF factors are significant when applied to the broad market, it might not be significant on a select number of stocks within a specific time frame (1998 to 2019). We also found that the 1-month log inflation is not significant in explaining the log returns of our selected stocks.

Another interesting finding is that while SMB’s, HML’s, and CMA’s coefficient estimate are as expected (positive), WML and RMW both have negative coefficient estimates at a significant level. This indicates that the stock log returns are indeed negatively affected by momentum and profitability. This might make sense for the following reasons. As our time frame is from 1998 to 2019, we see a lot of unprofitable high-growth companies that are very overvalued during market peaks in the dotcom bubble, housing bubble, and during 2017-2019. However, as they are continuing their win streaks, various macro factors start to crack and the market falls. This is especially unfavorable for the high-growth winners with higher momentum, as those stocks would be the first to see a big market correction. This diagnostic provides some insights to how the models would be structuring the prediction for future returns.

We also attempted to use the Ken French factor library and fit it directly with the metadata log returns. However, this regression is very weak overall compared to the manual regression, with an adjusted R-squared of only 0.0097 compared to the previous regression's 0.397.



```
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)        0.0055329  0.0041559   1.331   0.1844
Mkt.RF             0.1029056  0.0975523   1.055   0.2926
SMB               -0.0024851  0.0012674  -1.961   0.0511 .
HML                0.0020796  0.0015497   1.342   0.1809
RMW               -0.0025100  0.0016386  -1.532   0.1269
CMA               -0.0003467  0.0022236  -0.156   0.8762
X1M_log_inflation  0.2010458  2.8146944   0.071   0.9431
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.05323 on 238 degrees of freedom
Multiple R-squared:  0.03405,   Adjusted R-squared:  0.009697
F-statistic: 1.398 on 6 and 238 DF,  p-value: 0.216
```

Figure 2: Factor Regression with Ken French's factor library fitted to our log-return.

We see that none of the factors are significant at the 5% significance.

Model Tuning

After understanding our data and factors better, we proceed to tune our potential ML predictors by cross-validating on the sliding training sets. The tuning summary are provided as below:

Predictor	**Tuning summary**
LASSO Regression	Lambda chosen per iteration
Ridge Regression	Lambda chosen per iteration
Random Forest	Maxdepth = 5, mtry = 3, ntree = 2000
XGBoost	Eta = 0.5, maxdepth = 3, lambda = 1, gamma = 0.1, nrounds = 75
Neural Networks (MLP)	Layer 1: 16 units with tanh, alongside 25% dropout Layer 2: 4 units with tanh Output layer: 1 unit Epochs = 10, batch_size = 363 (number of stocks), optimizer = sgd
Autoencoders	For both beta side and factor side: Layer 1: 16 units with tanh, alongside 25% dropout Layer 2: 4 units with tanh Output layer: 1 unit Epochs = 25, batch_size = 363 (number of stocks), optimizer = sgd

Table 1: Tuning summary of various ML predictors.

We try to minimize overfitting by reducing the complexity of the each model such as reducing maxdepth, learning rate (eta), iterations (nrounds and epochs), layers, and nodes per layer.

Model Prediction Accuracy and Portfolio Return Analysis

Afterwards, the model is prepared for 65 rounds of prediction using the training/testing splits. The predicted log return is then converted to predicted return before being compared to the actual return as below:

Model	Mean Squared Error	Mean Absolute Error	Hit Ratio
Linear Regression	0.01294	0.08066	0.45158
LASSO Regression	0.01280	0.08001	0.44683
Ridge Regression	0.01278	0.07999	0.44717
Random Forest	0.01386	0.08374	0.45697
XGBoost	0.01301	0.07989	0.44569
Neural Networks	0.06476	0.19059	0.49964
Autoencoder	**0.01094**	**0.07265**	**0.59135**

Table 2: Average prediction accuracy of each ML model.

We can see that linear regression is a solid benchmark in terms of minimizing average MSE and MAE. The introduction of regularization models on top of the regression such as LASSO and Ridge does not significantly reduce the MSE and MAE. Furthermore, it has a worse hit ratio compared to the linear regression. Meanwhile, Random Forest and XGBoost also did not provide a significant improvement on the prediction accuracy, with both tree-based methods scoring a higher average MSE and MAE than linear regression. We observe that XGBoost is better than Random Forest in terms of aligning the predictions closer to the actual returns.

Two things stand out on this table. First, apart from autoencoder, we observe that most of the models stated above do not have a good hit ratio (worse than 0.5, which implies that it does no better than a coin-toss in terms of predicting return signs). Autoencoder is proven to be best model in terms of pointwise prediction as it has notably lower MSE and MAE than the other predictors. It also has an outstanding hit ratio of 59.1%, indicating that it has the potential to form a profitable portfolio strategy. Another interesting discovery is that our neural networks model has significantly higher MSE and MAE compared to the rest of the predictors, despite a slightly higher hit ratio. This might be caused by too much complexity induced on the data such that it becomes overfit to the trained log returns.

That being said, pointwise prediction alone might not address the whole concern on whether we can construct a profitable investment strategy. Hence, we apply various portfolio strategies and put them into test as below:

Model	Portfolio Strategy	Annual Return (5.5-yr CAGR)	Return Volatility	Sharpe Ratio
Market (Benchmark)	Long only	11.036 %	11.525 %	0.904
Risk-Free (Benchmark)	Long only	0.618 %	0.228 %	0.000
Equal-Weight (Benchmark)	Long only	7.868 %	18.290 %	0.396
Linear Regression	Long only	8.889 %	16.657 %	0.497
	Long/Short	-2.170 %	10.829 %	-0.257
	Leveraged L/S	7.776%	18.117 %	0.394

LASSO Regression	Long only	7.630 %	17.210 %	0.407
	Long/Short	-3.224 %	10.849 %	-0.354
	Leveraged L/S	5.895 %	18.907 %	0.279
Ridge Regression	Long only	7.957 %	16.400 %	0.447
	Long/Short	-2.741 %	10.902 %	-0.308
	Leveraged L/S	6.538 %	17.818 %	0.332
Random Forest	Long only	**12.199 %**	15.787 %	**0.734**
	Long/Short	**3.733 %**	10.720 %	**0.291**
	Leveraged L/S	**14.343 %**	17.096 %	**0.803**
XGBoost	Long only	9.515 %	**15.694 %**	0.567
	Long/Short	0.301 %	**5.628 %**	-0.056
	Leveraged L/S	9.674 %	**16.239 %**	0.558
Neural Networks (MLP)	Long only	10.322 %	15.298 %	0.634
	Long/Short	3.259 %	9.147 %	0.288
	Leveraged L/S	12.157 %	16.268 %	0.709
Autoencoder	Long only	10.518 %	16.034 %	0.617
	Long/Short	2.301 %	9.332 %	0.180
	Leveraged L/S	11.700 %	17.710 %	0.626

Table 3: Characteristics of portfolio formed by each ML model.

We first observe that the market portfolio obtained from Ken French factor library is performing well, delivering a sizeable 11% annual return over 5.5 years with a Sharpe Ratio of 0.9. From a risk-reward optimization standpoint, it is very hard to beat the market, with none of the strategies from any predictors delivering a better Sharpe ratio than the market portfolio. We also notice that the annual risk-free return is around 0.62%, signifying that the testing frame is at a low interest rate environment. This is significant information as companies would be able to get more aggressive and finance debt used for expansion at a low rate. Finally, if we are putting an equal-weight strategy (long all stocks in the observed stock universe), then we would be getting an annual return of 7.87% and a Sharpe ratio of 0.396 over the 5.5 years. For a strategy to be considered successful, it must beat the annual return and Sharpe ratio of the equal-weight strategy and market portfolio to some extent. Otherwise, investors would be better off spreading out their investments to all stocks or just follow the market (i.e., our ML strategy would not generate enough alpha that is more advantageous compared to the market).

Overall, we observe that the long-only strategy is a safe strategy considering the risk and returns. Also, since we know that the stock drift (average return) is generally positive, then any strategies that involves shorting would only work if we correctly identified the stocks that are providing significantly less return than the top quintile. Another noticeable observation is that the leveraged long/short strategy typically has a higher volatility compared to the other two strategies, which is expected as the effect of long stocks is amplified and further added with some shorts. The long/short strategy has a net investment of 0, meaning that it could finance the long portion of the portfolio with the short portion. This means that its comparable benchmark would be the risk-free return. Similarly, since the leveraged long/short has a net investment of 1, we would compare this strategy with its own long strategy, market portfolio, and equal-weighted portfolio.

Starting with the linear regression, we observe that the long-only strategy is better in terms of return and Sharpe ratio as compared to the long-short and leveraged long-short. Although the long strategy beats the equal-weight portfolio, the other strategies failed to do so, and it is most likely due to a bad shorting strategy (i.e., incorrectly classifying which stocks are in the bottom 20%). In this case, linear regression does not provide a good prediction on the bottom quintile, hence the failure in the shorting strategies. The regularization strategies also did not do well, as they have lower returns and Sharpe ratio than their corresponding linear regression strategies. The ridge regression strategies appear to do better than LASSO with noticeably higher returns and lower volatility, suggesting that it might be beneficial to consider ridge regression in portfolio construction rather than LASSO.

On the other hand, the tree-based method strategies look very solid in terms of portfolio risk, volatility, and Sharpe ratio. Specifically, the random forest leveraged strategy paid off well, delivering a 14.34% annual return which beats the market and a solid 0.8 Sharpe ratio. This is primarily fueled by a good selection of longs and shorts, which allow the strategy to take scale advantage of the levered longs and compensate it with low-return shorts. This success is also reflected in their long/short strategy that beats the risk-free return. While random forest dominates the return and Sharpe ratio, XGBoost proved that it is better at minimizing volatility, with lower standard deviations of returns in all three strategies. This suggests that XGBoost can be used to construct a more conservative portfolio while random forest can be used to construct a more aggressive portfolio.

Both neural network and autoencoder also did considerably well in the portfolio construction, with each leverage strategy able to beat the market return. Despite the failure in point prediction relative to other predictors, the multi-layer perceptron was doing well in classifying the top and bottom predictions, showcased by its positive long/short strategy and a good use of leverage. MLP also did marginally better than the autoencoder model in terms of higher return, higher Sharpe ratio, and lower volatility.

The prediction from standard linear regression and XGBoost are stable when being re-run as it is less subject to differences attributed on starting points or seedings. Next, the LASSO and ridge regression would have a changing prediction for different runs, but it would not be far-off from the previous prediction as it only changes due to different tuning from the randomness of the cross-validation. Random forest also does not vary widely as there are a lot of trees considered in the prediction (much like doing Monte Carlo simulation). On the flip side, predictions from neural networks and autoencoders are subject to change from different starting points and different directions in its gradient descent.

The detailed results of these investment strategies and portfolio characteristics are as follows:

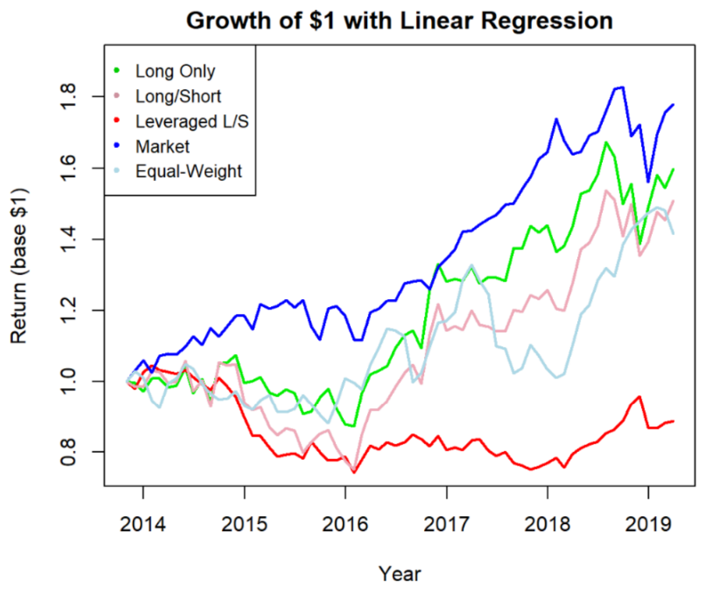


Figure 3: Cumulative growth of portfolios constructed with linear regression.

We can see that the linear regression strategies all underperformed the market portfolio for 5.5 years. It also does not have a good short selection as shown by the negative long-short strategy. Its long-only strategy performs slightly better compared to the equal-weight (1/N) strategy and its corresponding leveraged portfolio.

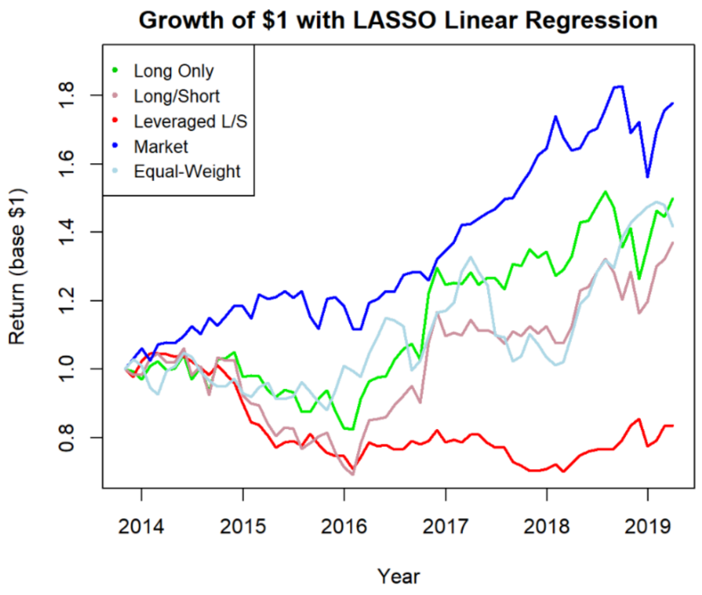


Figure 4: Cumulative growth of portfolios constructed with LASSO.

Here, we see that the LASSO strategies are even further than the ones from linear regression. Moreover, the leveraged strategy performs worse than the 1/N strategy, which suggests that it is not better than random portfolio construction.

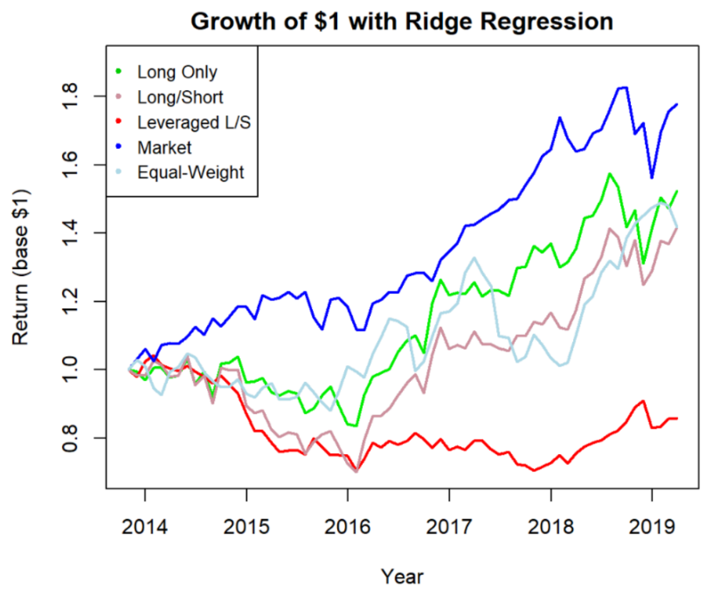


Figure 5: Cumulative growth of portfolios constructed with Ridge Regression.

Although the ridge regression portfolio performs slightly better than the LASSO, these strategies still trail far from the market portfolio and mimic the 1/N strategy which is sub-optimal.

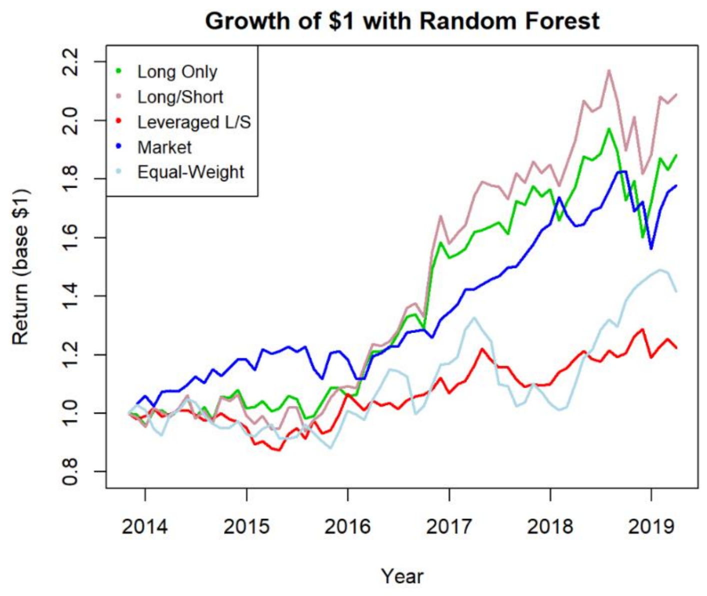


Figure 6: Cumulative growth of portfolios constructed with Random Forest.

The random forest strategies show significant success in beating the market, especially after mid-2016. It seems that from that period onwards, the predictions are more on-point and the portfolio made good bets on the longs and shorts. We also see how a good short portfolio can amplify the success of a long portfolio by allowing for more leverage and aggressive strategy which also performs better in terms of Sharpe ratio (risk/return optimization). The cumulative returns amplified the market portfolio more as time went on, suggesting that it has an increasingly volatile portfolio as the market swings wider in 2017 to 2019. It also beats simple regression models and shows that this predictor can be used to construct a profitable investment strategy.

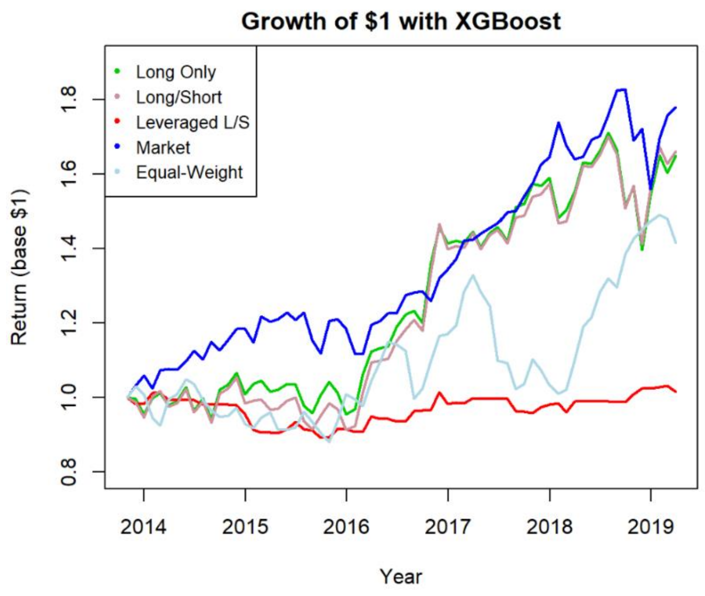


Figure 7: Cumulative growth of portfolios constructed with XGBoost.

The long and leveraged strategies above perform sufficiently well against the equal-weighted portfolio but fell short at the end against the market portfolio. Had the short portfolio performed well, we would achieve higher returns. We also see that both strategies are relatively less volatile compared to their corresponding strategies used by other predictors. This suggests that XGBoost has a solid predictive power on each month of the testing period.

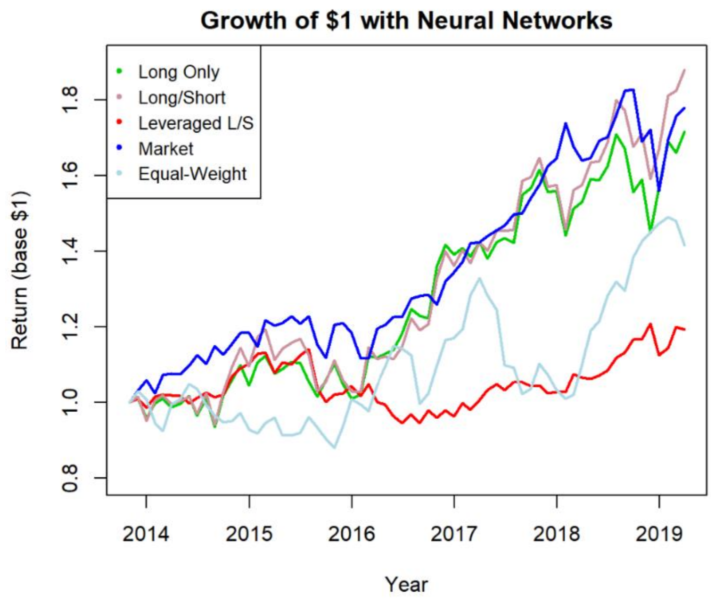


Figure 8: Cumulative growth of portfolios constructed with Neural Networks.

The neural network strategies performed nicely, in which its long and leveraged strategies returned at a similar level to the market return. It also follows the market return pretty well and it has a successful shorting strategy near the end of the testing period.

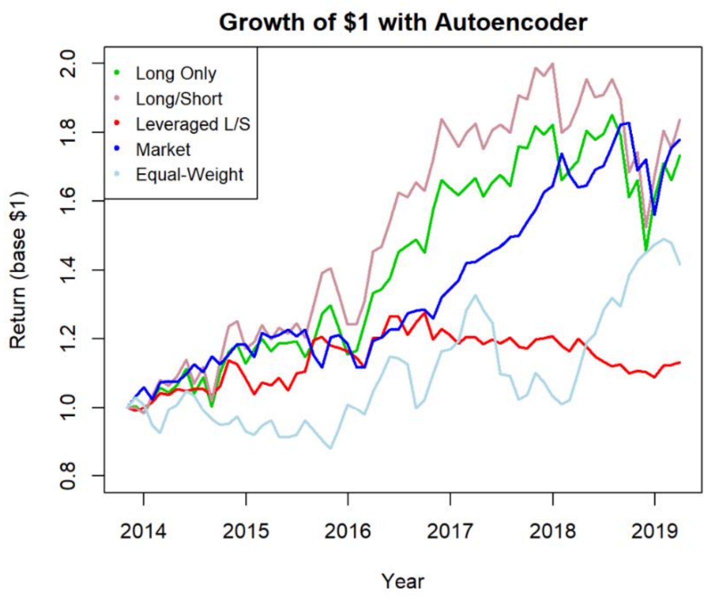


Figure 8: Cumulative growth of portfolios constructed with Autoencoders.

We can see that the long and leveraged strategies clearly outperforms the market during 2016 to mid-2018 but dip way more than the market portfolio later on. Despite its earlier success up to 2018, it visualizes the problem with this Autoencoder model, which is its relatively high volatility compared to other predictors. When the predictors do well, it would gain a huge advantage, but when the predictor did not do as expected, the portfolio downturn is equally amplified.

Conclusion

From our experiment, we can conclude that machine learning models can improve equity return predictions and portfolio construction compared to the market portfolio and the equal-weight portfolio. Out of the 5 internal and 2 macro factors, 4 of them are significant in explaining the variability in stock returns. The linear regression model serves as a decent benchmark model in predicting future returns and constructing profitable investment strategy without needing any regularization. The autoencoder is the best model for point prediction of the returns, scoring the lowest MSE and MAE while recording the highest hit ratio. Tree-based models provide proper stock classifications for portfolio construction, with random forest delivering the highest return and XGBoost providing the lowest volatility. Leveraged long/short strategies can deliver a better return and Sharpe ratio than other strategies as long as the predictor has the capacity to consistently be correct in classifying which stocks to long and which ones to short.

Remarks

We can improve this research by trying out different machine learning models or training them on factors that are not uniformized. We can also perform mean-variance optimization and other portfolio strategies that may potentially outperform the leveraged long/short strategy. Moreover, there are room for improvements in terms of model tuning (tune complex models on each iteration) and data preprocessing (using more stocks that do not appear in each observation). Finally, we believed that this research can provide a solid foundation for those who tried to explore if various machine learning methods can be employed to improve return prediction on construct a better portfolio than the market.

References

Fama, E. F., & French, K. R. (1993). Common Risk Factors in the Returns of Stocks and Bonds. *Journal of Financial Economics*, 3-56.

Fama, E. F., & French, K. R. (2015). A Five-Factor Asset Pricing Model. *Journal of Financial Economics*, 1-22.

Feldstein, M. (1983). Inflation and the Stock Market. *National Bureau of Economic Research: Inflation, Tax Rules, and Capital Formation*, 186 - 198.

Gu, S., Kelly, B., & Xiu, D. (2021). Autoencoder asset pricing models. *Journal of Econometrics*, 429-450.

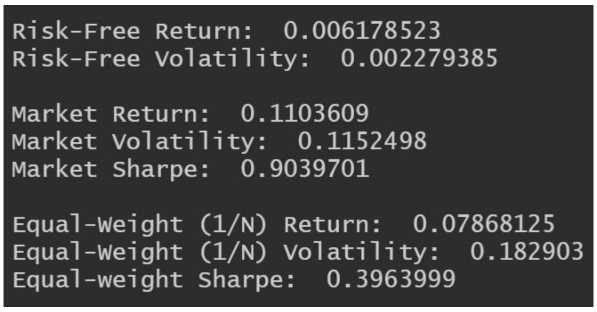
Nakagawa, K., Ito, T., Abe, M., & Izumi, K. (2019). Deep Recurrent Factor Model: Interpretable Non-Linear and Time-Varying Multi-Factor Model. *arXiv*, 1-6.

Appendix

The full implementation code is as attached in **GPR2-Samuel-Benedict.Rmd.**

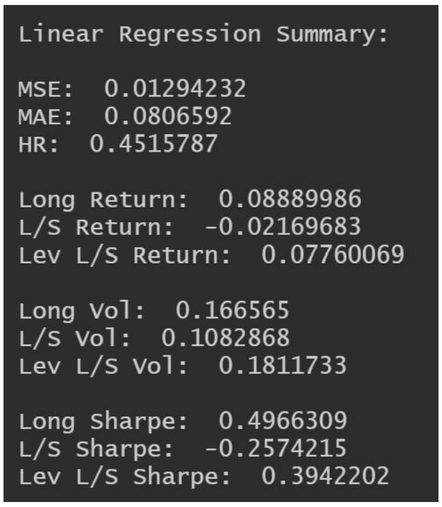
You can also see this code in https://github.com/notwhammy/AFM-423-ML-for-Factor-Investing

Attached below are the output summary from the market data:

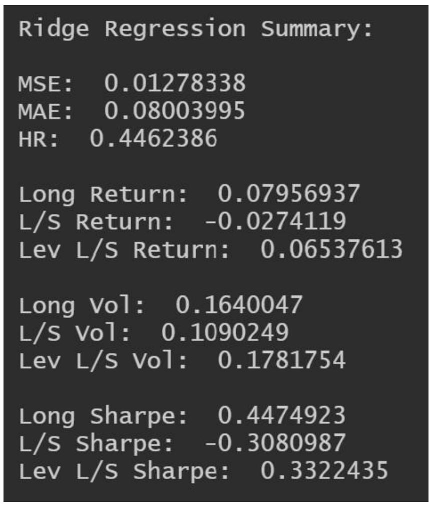


```
Risk-Free Return:  0.006178523
Risk-Free Volatility:  0.002279385

Market Return:  0.1103609
Market Volatility:  0.1152498
Market Sharpe:  0.9039701

Equal-Weight (1/N) Return:  0.07868125
Equal-Weight (1/N) Volatility:  0.182903
Equal-weight Sharpe:  0.3963999
```

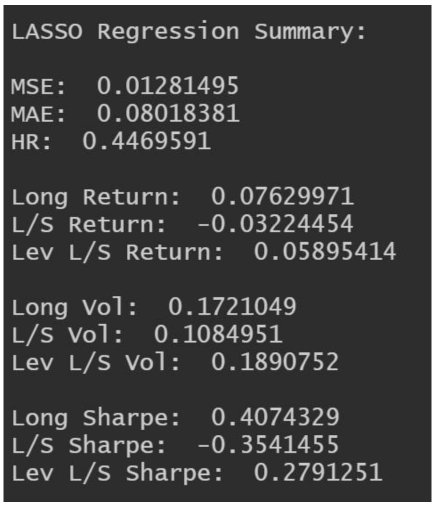
Attached below are the output summary from each predictor:



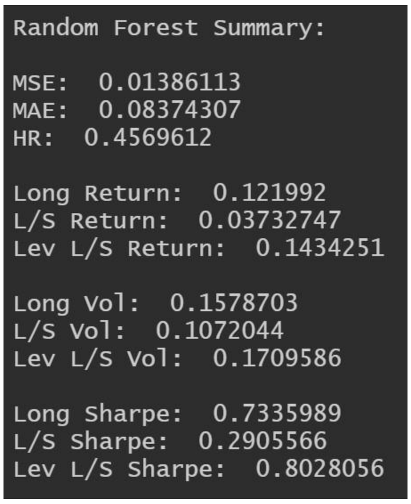
```
Linear Regression Summary:

MSE:  0.01294232
MAE:  0.0806592
HR:  0.4515787

Long Return:  0.08889986
L/S Return:  -0.02169683
Lev L/S Return:  0.07760069

Long Vol:  0.166565
L/S Vol:  0.1082868
Lev L/S Vol:  0.1811733

Long Sharpe:  0.4966309
L/S Sharpe:  -0.2574215
Lev L/S Sharpe:  0.3942202
```



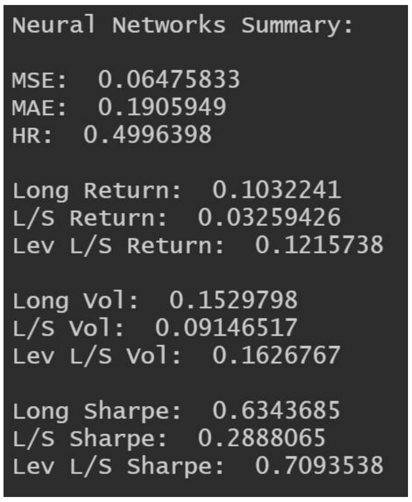
```
Ridge Regression Summary:

MSE:  0.01278338
MAE:  0.08003995
HR:  0.4462386

Long Return:  0.07956937
L/S Return:  -0.0274119
Lev L/S Return:  0.06537613

Long Vol:  0.1640047
L/S Vol:  0.1090249
Lev L/S Vol:  0.1781754

Long Sharpe:  0.4474923
L/S Sharpe:  -0.3080987
Lev L/S Sharpe:  0.3322435
```



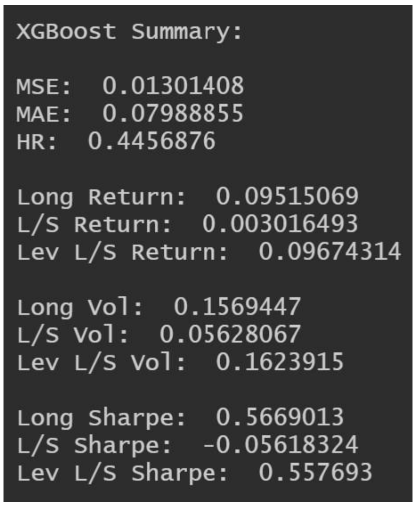
```
LASSO Regression Summary:

MSE:  0.01281495
MAE:  0.08018381
HR:  0.4469591

Long Return:  0.07629971
L/S Return:  -0.03224454
Lev L/S Return:  0.05895414

Long Vol:  0.1721049
L/S Vol:  0.1084951
Lev L/S Vol:  0.1890752

Long Sharpe:  0.4074329
L/S Sharpe:  -0.3541455
Lev L/S Sharpe:  0.2791251
```



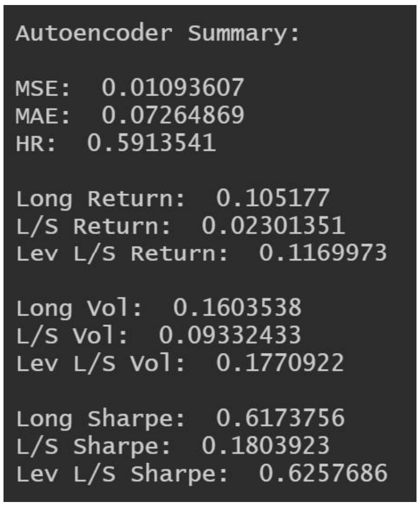
```
Random Forest Summary:

MSE:  0.01386113
MAE:  0.08374307
HR:  0.4569612

Long Return:  0.121992
L/S Return:  0.03732747
Lev L/S Return:  0.1434251

Long Vol:  0.1578703
L/S Vol:  0.1072044
Lev L/S Vol:  0.1709586

Long Sharpe:  0.7335989
L/S Sharpe:  0.2905566
Lev L/S Sharpe:  0.8028056
```



```
Neural Networks Summary:

MSE:  0.06475833
MAE:  0.1905949
HR:  0.4996398

Long Return:  0.1032241
L/S Return:  0.03259426
Lev L/S Return:  0.1215738

Long Vol:  0.1529798
L/S Vol:  0.09146517
Lev L/S Vol:  0.1626767

Long Sharpe:  0.6343685
L/S Sharpe:  0.2888065
Lev L/S Sharpe:  0.7093538
```



```
XGBoost Summary:

MSE:  0.01301408
MAE:  0.07988855
HR:  0.4456876

Long Return:  0.09515069
L/S Return:  0.003016493
Lev L/S Return:  0.09674314

Long Vol:  0.1569447
L/S Vol:  0.05628067
Lev L/S Vol:  0.1623915

Long Sharpe:  0.5669013
L/S Sharpe:  -0.05618324
Lev L/S Sharpe:  0.557693
```



```
Autoencoder Summary:

MSE:  0.01093607
MAE:  0.07264869
HR:  0.5913541

Long Return:  0.105177
L/S Return:  0.02301351
Lev L/S Return:  0.1169973

Long Vol:  0.1603538
L/S Vol:  0.09332433
Lev L/S Vol:  0.1770922

Long Sharpe:  0.6173756
L/S Sharpe:  0.1803923
Lev L/S Sharpe:  0.6257686
```