Multi-Factor Modelling with Shapely Additive Explanations

Student: Thomas Frew
School of Computer Science
The University of Adelaide
thomas.frew@student.adelaide.edu.au

Supervisor: Jinan Zou
School of Computer Science
The University of Adelaide
jinan.zou@adelaide.edu.au

Supervisor: Haiyao Cao
School of Computer Science
The University of Adelaide
haiyao.cao@adelaide.edu.au

ABSTRACT

While machine learning models can effectively predict stock prices, their decision-making process is opaque and immutable. Hence, they are exceedingly difficult for traders to understand and control. Using SHAP for feature selection, we propose a novel technique for constructing multi-factor models based on machine learning algorithms. These models are interpretable, mutable and can be used to produce a consistent profit.

1 INTRODUCTION AND MOTIVATION

1.1 Equities

Equities — also known as stocks or shares — are financial instruments that represent partial ownership of a company. Traders can buy equities to receive a stake in their company and sell them to receive a payout at their current value.

The price of a stock is determined by the value traders are willing to exchange it for. This is influenced by the complex global market, making stock prices innately unpredictable.

1.2 Value Investing

Value investing is a well-established trading strategy that takes advantage of stock price fluctuations. As described by economist Benjamin Graham [1], the idea is to buy stocks that are priced below their "intrinsic value", what the stock is truly worth. We expect stocks to eventually return to this value, where they can be sold at a profit.

To determine a stock's intrinsic value, value investors several economic factors related to the stock and its company. These commonly include:

- **Fundamental factors**: Factors that reflect a company's financial position, operations, and development. These are published by companies in their balance sheets, income statements and cash flow statements.
- **Technical factors**: Factors that reflect a stock's price trends, stakeholder support and volatility. They are artificially constructed from a stock's historical price and volume data.

1.3 Multi-Factor Modelling

Multi-factor models (MFMs) are linear models that calculate a stock's growth (Y) based on a linear combination of its economic factors (F). Each factor is assigned a coefficient (β) that measures its relative impact on the result.

$$Y = \beta_1 F_1 + \beta_2 F_2 \ldots + \varepsilon$$

Equation 1: The linear structure of a multi-factor model.

Since multi-factor models predict a stock's growth, they can be used to implement a value investing strategy. Any stock expected to appreciate should be purchased and sold when its price is no longer expected to increase.

1.4 Transparency and Mutability

Multi-factor models are beneficial for traders because they are highly transparent and mutable:

- **Transparent**: Each factor's impact on the model's predictions is clear and quantified. Hence, it is easy to learn how the model works and diagnose any issues.
- **Mutable**: The model's factors and coefficients can be meaningfully changed to integrate new ideas and adapt to changing market conditions.

With hundreds of published economic factors [2], it can be challenging to determine the best combination to construct an accurate multi-factor model. This often involves extensive economic research and regression analysis.

1.5 Machine Learning for Trading

Machine learning (ML) is a form of artificial intelligence that can learn abstract, non-linear patterns in data without being explicitly programmed. This is ideal for effective stock price prediction, since the market exhibits complex interactions that humans are unable to comprehend.

Several studies have used ML models to predict stock prices or the direction in which their prices will move. These have been highly successful, with recent models predicting stock movements with up to 67% accuracy [3,4].

However, due to the self-learning nature of ML models, their transparency and mutability are often limited. Using them to make trading decisions can be risky, as their behaviour cannot be easily understood or corrected to avoid poor investments.

1.6 Shapely Additive Explanations

To address the issue of transparency in ML models, the field of explainable AI (XAI) has emerged. XAI refers to techniques that provide insights into an ML model's decision-making process.

Shapely Additive Explanations (SHAP) is an XAI tool designed to explain specific predictions of an ML model [5]. It does this by calculating scalar values (SHAP values) that quantify the effect of each feature on the result. These are often visualised using force plots, as depicted in Figure 1 [6].

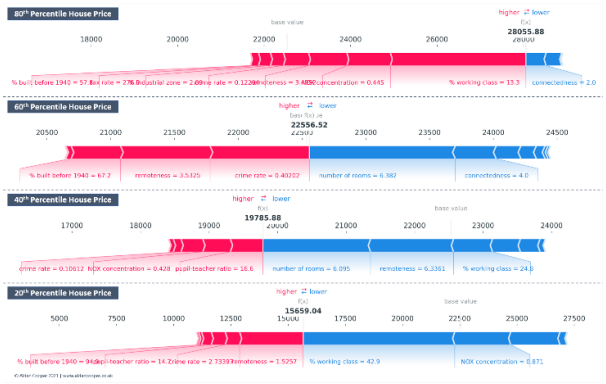

Figure 1: Force plots of SHAP values for house price prediction. Factors with positive (red) values have increased the prediction. Factors with negative (blue) factors have decreased the prediction.

SHAP values have two notable properties:

- **Sign**: Features with positive SHAP values have increased the model's prediction. Features with negative SHAP values have decreased the model's prediction.
- **Magnitude**: Features with larger absolute SHAP values have had a more significant impact on the model's prediction.

Since the magnitude of a feature's SHAP value measures its importance for a particular prediction, the average magnitude of a feature's SHAP value indicates its significance overall. This property is applicable to any ML architecture.

By calculating the average, absolute SHAP value for each feature in an ML model, we can identify its most important factors and use these to construct effective multi-factor models. Since ML models can understand patterns beyond human comprehension, this feature selection strategy could produce better models than those designed by professional economists.

1.7 Objective

In this paper, we propose and explore a novel method for constructing multi-factor models based on the SHAP values of ML algorithms.

First, we train an ML model to predict a stock's price change (over the next quarter) based on current values for its fundamental and technical factors. Next, we use our model's SHAP values to select a subset of its most important factors. Finally, we perform linear regression using this subset of factors, thereby constructing an effective multi-factor model.

We demonstrate that our generated multi-factor models:

- Can predict a stock's growth more effectively than standard linear regression models.
- Produce strong and consistent returns that can compete with published multi-factor models.

We also created MultiMax [7], a publicly available trading app that implements this modelling pipeline.

2 LITERATURE REVIEW

2.1 Price Prediction with Multi-Factor Modelling

In 1962, economist William Sharpe developed the first price-modelling function, the Capital Asset Pricing Model (CAPM) [8]. Sharpe claimed that a stock's "risk premium" provides sufficient information to explain its growth. However, this single-factor model can only explain up to 70% of a stock's returns [9], failing to model differences between shares with the same risk premium.

To address this issue, economists Eugene Fama and Kenneth French created the first multi-factor model [10] by considering a stock's risk premium alongside its company's size (small-minus-big) and growth objectives (high-minute-low). This model can explain up to 90% of a stock's returns, which is a significant improvement over CAPM.

As of 2023, researchers have published hundreds of economic factors and multi-factor models to improve upon the Fama-French model [2,11]. The most notable of these is Carhart's 4-factor model, which uses momentum, the first technical factor in multi-factor modelling [12].

The primary issue with multi-factor modelling is that it requires domain expertise. Given the massive range of available factors, it is challenging to select a small, relevant subset that can accurately model a stock's growth. Some researchers have attempted to do

this with statistical feature selection methods, like LASSO regression [13] and recursive feature elimination [14] However, these may not capture the insights of a professional economist or ML model.

2.2 Price Prediction with Machine Learning

The earliest use of ML for stock price prediction involved using deep neural networks (DNNs) [15] for technical analysis: predicting stock prices based on previous price and volume data [16,17]. These models were somewhat accurate but received opposition from Fama's Weak Efficient Market Hypothesis [18] and observations that they "behaved like an exponentially smoothing algorithm", with predictions lagging behind stock prices [16].

Further experiments trained price-predicting DNNs on fundamental and technical factors, helping generate more decisive forecasts [19,20]. Macroeconomic factors [21] and sentiment data [22] have also been shown to influence stock prices and are used frequently in state-of-the-art ML models.

Despite their early success in stock price prediction, DNNs are limited by their lack of memory. Past data cannot be used to influence current decisions, only its residual effect on the model's weights and biases. Hence, DNNs cannot understand temporal patterns in data, such as trends that indicate future stock growth.

To solve this issue, researchers moved their attention to RNNs and LSTMs, types of DNN that can remember short and long-term patterns [23 – 26]. Studies by Moro et al. and Dautel et al. have shown that these systems can predict the direction of a stock's growth 2% more accurately than static DNNs [27,28].

However, RNNs and LSTMs introduce their own issues. Due to their long, sequential nature, these models are prone to vanishing gradients and can sometimes prioritise irrelevant features [29], making them unsuitable for sustained, live trading.

Self-attention networks have been recently created to address these problems [30]. By assigning weight (attention) between input features, self-attention networks can effectively filter noise and identify complex relationships between input features. Models like Tuncer et al.'s DAPP and Liu et al.'s TeaNET have been shown to predict the direction of a stock's growth 4% more accurately than attention-less baselines [31,32].

Researchers have also predicted stock prices using random forests [33] and gradient-boosting machines [34], which make decisions based on a collection of decision trees. These models are typically less accurate than DNNs and cannot learn temporal dependencies, but they are transparent and can learn abstract, non-linear relationships in relatively few computations [35,36].

3 METHODOLOGY

3.1 Data Source

Our objective is to train ML and multi-factor models to predict a stock's percentage growth (over the next quarter) from the current values of their fundamental and technical factors. Since ML models are sensitive to the quality of their training data, we created a large dataset for this purpose.

Using the financial API FinancialToolkit, we downloaded the economic factors and quarterly growth for all equities in the Wilshire 5000: the 3,687 most actively traded stocks in the US [37]. We collected quarterly samples between January 2000 and January 2023, which is the complete period companies have been required to digitally publish their fundamental factors [38].

The FinanceToolkit provides 196 fundamental factors and 40 technical factors, which we downloaded for all our companies. We also created 196 additional features for the growth rates of fundamental factors. While fundamentals describe a company's current position, their growth rates describe how it is changing. Both are relevant in stock price prediction.

However, not all companies published data for every factor in our 23-year sample. 25.9% of data points are missing and cannot be interpreted by our models. To solve this problem, we removed all columns in the dataset that contained >30% missing values. Then, we removed all rows that still had missing values.

This strategy caused us to lose 76.2% of our dataset. The original dataset had 217,559 rows and 432 columns, while our new dataset had 60,559 rows and 369 columns. Nevertheless, this is still sufficiently large to train the simple ML and multi-factor models we explored in this paper.

Dataset	Rows	Fund. Factors	Fund. Growth	Tech. Factors
Original	217,559	196	196	40
New	60,559	192	140	35

3.2 Machine Learning Models

Using our cleaned dataset, we trained three ML architectures to predict a stock's growth. These served as the "base models" for our SHAP feature selection process, which we used to construct our multi-factor models.

3.3 Deep neural networks

Deep neural networks (DNNs) are a popular class of artificial neural networks famous for their ability to model complex patterns within data. DNNs consist of layers of biases (neurons)

and weights (synapses) connecting them, resembling a brain. DNNs take in data and process it through their layers, making it abstract and refined. The final layer outputs a prediction based on this information. Neural networks learn through backpropagation, a mathematical procedure that optimises its weights and biases.

3.4 Decision Trees

Two of our ML architectures, random forests and gradient boosting machines, use decision trees to predict a stock's growth.

Decision trees [39] are a machine learning algorithm that can classify data or make predictions based on a hierarchical tree of decisions. For regression tasks, their goal is to predict a continuous value for each point in a dataset. The entire dataset begins at the top (root) of the tree. Then, it is split recursively down the tree into smaller subsets based on a condition (decision) at each node. This process continues until every data point arrives at a leaf node, containing a prediction for its value. Ideally, the data is partitioned into leaf nodes such that the mean error of their predictions is minimised.

3.5 Random Forests

Random forests (RFs) [35] make predictions using an ensemble (forest) of many decision trees to help minimise overfitting. Each tree is trained in parallel on a random subset of the dataset's features. Typically, if the dataset has M features, $\sqrt{M}$ features are the ideal sample size [34]. To predict on a data point, we take the mean output of all trees in the forest.

3.6 Gradient Boosting Machines

Gradient boosting machines (GBMs) [36] also use an ensemble of decision trees. However, GBMs construct these sequentially, with each tree correcting the errors made by the previous one. This process continues until for a specified number of "boosting rounds". To predict on a data point, we take the sum of outputs from all trees in the forest.

3.7 Feature Selection with SHAP

The magnitude of a feature's SHAP value measures its importance for a particular prediction, so the average magnitude of a feature's SHAP value measures its overall significance.

We use this property on our ML models to select the N most important factors from our dataset, then use these to train multi-factor models. The choice of N is arbitrary and depends on how accurate (and complicated) we want our multi-factor models to be. We chose N = 10 as a reasonable balance between accuracy and complexity.

See Algorithm 1 for details on this feature selection process.

SelectFactors(N):

1. Train a machine learning model.
2. Calculate SHAP values for each of the model's predictions.
3. Calculate the mean absolute SHAP value for each factor.
4. Return the N factors with the largest mean absolute SHAP values.

Algorithm 1: Selecting the N most relevant factors for our multi-factor models, using mean absolute SHAP values.

3.8 Multi-Factor Modelling

After selecting each ML model's 10 most important factors, we performed linear regression on these features to develop our multi-factor models.

Linear regression [40] is an ML technique that finds the best-fitting linear relationship between a dependent variable and a set of independent variables. This is achieved by finding the linear equation with minimal error between its outputs and the dependent variable's actual values.

Multi-factor models are linear equations, so linear regression is the best way to construct them from our selected features.

3.9 Evaluation Metrics

Our objective is to train ML models and multi-factor models that can accurately predict stock prices.

If our ML models are accurate, we can use them to select appropriate features for our multi-factor models. If the multi-factor models that we construct are accurate, then our feature selection methodology is effective.

To evaluate the accuracy of our ML and multi-factor models, we measured the mean absolute error (MAE) of their predictions. Lower MAE indicates higher accuracy.

3.10 Backtesting

We also want our constructed multi-factor models to be useful in making profitable trades. For this purpose, we used our multi-factor models in the TopKDrop strategy to perform simulated trading (backtesting) between January 2000 and January 2023.

The TopKDrop strategy is a popular portfolio selection strategy in ML-based trading [41,42]. Every D days, this strategy sells all its stocks, predicts the K most profitable stocks currently are, and invests in those stocks equally.

The values for D and K are arbitrary and depend on the amount of diversification in our portfolio and depend on how we want our

portfolio to behave. We chose D = 300 and K = 10 to create a reasonably diverse portfolio held for relatively long periods.

See Algorithm 2 for details on the TopKDrop strategy.

TopKDrop (D, K):

1. If D days have not passed since the last portfolio selection, do nothing. Otherwise:
2. Sell all stocks.
3. Predict the growth of each stock from the Wilshire 5000.
4. Invest 1/K of our equity in the K stocks with the highest predicted growth.

Algorithm 2: The TopKDrop strategy for value investing using our multi-factor model.

To evaluate the profitability and risk of our multi-factor trading strategies, we used three standard economic metrics [43,44]:

Sharpe ratio: Measures a trading strategy's total return relative to its risk. Strategies with high Sharpe ratios are good at consistently producing high returns.

Higher Sharpe ratio indicates the trading strategy creates higher and less risky returns.

Drawdown: Measures the maximum portion of money a trading strategy loses between any two times. Strategies with high drawdowns are likely to lose lots of money at once.

Compounding annual returns (CAR): Measures a trading strategy's yearly return with re-investment. Strategies with high CARs are highly profitable.

See Appendix A for a flowchart of this paper's methodology.

4 EXPERIMENTAL SETUP

4.1 Machine Learning Models

To select the optimal parameters for our random forest and GBM, we five folds of cross-validation based on Oshiro et al.'s and Bentéjac et al.'s recommended parameter range [45,46].

Parameters	Random Forest	GBM
# of trees/boosting rounds	64	64
Max depth	7	5

We used Auto-Keras' model selection algorithm to select the optimal hyperparameters for our DNN (over 20 trials) [47].

Parameter	Random Forest
# of neurons in hidden layers	128, 256 and 512 neurons.
Learning rate	0.001
Batch size	32
Epochs	16

4.2 Evaluation Metrics

Mean absolute error (MAE): The average absolute error between a model's prediction for an input ($\hat{Y}_i$) and its actual output (Y_i) over n predictions.

$$\text{MAE} = \frac{1}{\text{n}} \sum_{i=1}^{n} |Y_i - \hat{Y}_i|$$

Sharpe Ratio (SR): A trading strategy's total return (R) relative to the return of a risk-free investment (R_f).

$$\text{SR} = \frac{R - R_f}{\sigma_R}$$

Drawdown (DD): The maximum % loss between a trading portfolio's peak (maximum) and trough (minimum) value.

$$\text{Drawdown} = \max\left[\frac{\text{peak} - \text{trough}}{\text{trough}}\right]$$

Compounding Annual Return (CAR): A trading strategy's % profit compounded annually.

4.3 Trading Environment

We performed our backtesting using QuantConnect [48], an online trading platform that supports custom data and machine learning. QuantConnect also simulates a $0.005/share transaction fee, which is like most live trading platforms [49].

4.4 Baselines (Stock Price Prediction)

We tested the accuracy of our ML and multi-factor models by testing them against the following baselines:

Mean Guessing: Guessing the mean stock growth for each input. We should expect all models to outperform this baseline.

Linear Regression: Training a multi-factor model using linear regression without any feature selection.

4.5 Baselines (Backtesting)

We tested the profitability and risk of our multi-factor models by backtesting them compared to the following baselines:

Buy-and-Hold: Buying all stocks from the Wilshire 5000 and holding onto them.

Published Multi-Factor Models: The Fama-French 3-Factor model [10] and Carhart 4-Factor model [12]: two influential multi-factor models in literature.

5 RESULTS

To ensure our results are significant, we performed 10 trials for each ML architecture: training an ML model, selecting its most important features and using these to construct and backtest a multi-factor model. We observed the following mean results:

5.1 Prediction Accuracy

Our ML models successfully predicted stock prices with relatively little error. Our DNN and GBM performed similarly, with MAEs of 20.34% and 20.41% respectively. Our random forest performed somewhat worse, with an MAE of 21.04%. Compared to standard linear regression, our ML models predicted stock prices with 3.24% to 3.94% less error. Hence, our ML models are successful at predicting stock prices and are appropriate for feature selection.

Like their base models, our DNN and GBM-based multi-factor models performed similarly, with MAEs of 23.48% and 23.51%, respectively. Our RF-based model performed somewhat worse, with an MAE of 23.93%. Relative to standard linear regression, our multi-factor models performed slightly better, achieving 0.38% to 0.8% less error. This indicates that our feature selection technique is effective in producing reliable multi-factor models.

However, our multi-factor models performed significantly worse than their base models. Our DNN gained 3.14% MAE, our GBM gained 3.10% MAE, and our random forest gained 2.89% MAE. This emphasises the restrictions imposed by the linearity of multi-factor modelling. Future research could explore the limitations of multi-factor modelling in greater depth.

Interestingly, feature selection from the more accurate ML models created more accurate multi-factor models. This is unsurprising since better models should select better factors. Future research could investigate the nature of this relationship and whether it holds for all ML models.

Model	**MAE (%)**
Guessing at the mean	28.56
Linear regression	24.28
Random forest	21.04
GBM	20.41
DNN	**20.34**
RF-based MFM	23.93
GBM-based MFM	23.51
DNN-based MFM	**23.48**

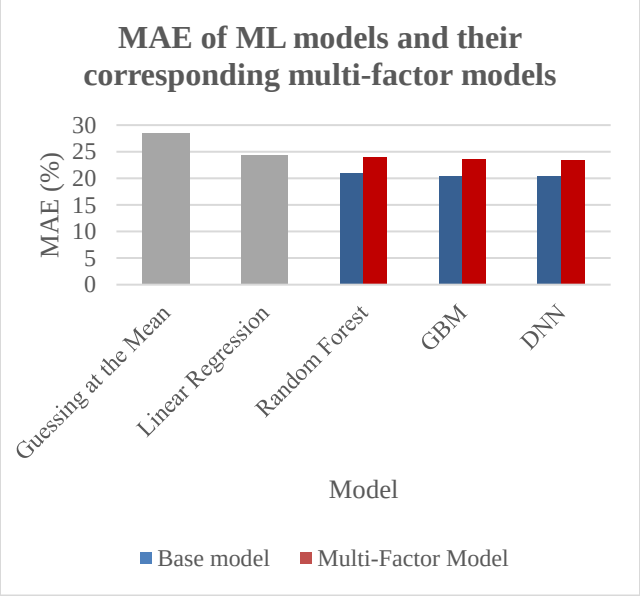

Figure 2: Accuracies of our ML models and their constructed, multi-factor models.

5.2 Feature Selection Behaviour

Compared to the composition of our dataset, all our ML models selected a smaller portion of fundamental factors and a larger portion of technical factors. This could be because technical factors correlate more directly with a stock's behaviour and are artificially constructed to reflect this. Conversely, while many fundamentals correlate with their stock's behaviour, they are not explicitly created for this purpose.

Interestingly, DNNs selected more fundamental factors and fundamental growth rates than tree-based models. This could be because DNNs are better suited to extracting the abstract insights hidden in fundamental factors.

Furthermore, all types of economic factor were selected at least 18% of the time. Therefore, fundamental factors, fundamental growth rates and technical factors all play a crucial role in effective stock price prediction and multi-factor modelling.

Model	Fundamental factors (%)	Fundamental growth (%)	Technical factors (%)
Dataset	52	38	10
RF	41	21	38
GBM	41	18	41
DNN	48	26	26

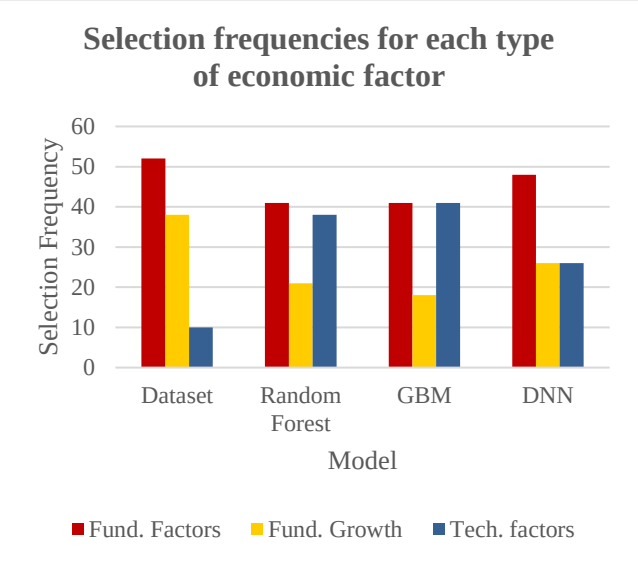

Figure 3: Selection frequencies for the different types of economic factors organised by ML architecture.

5.3 Backtesting

All our multi-factor models successfully produced a profit during backtesting. Our DNN-based model achieved the best annual return of 18.90%. Our RF and GBM-based models performed significantly worse, with annual returns of 10.73% and 10.18%, respectively.

Compared to the buy-and-hold strategy, all our multi-factor models produced a higher return and Sharpe ratio with a lower drawdown. Therefore, our constructed models are more profitable and less risky than buying and holding onto the Wilshire 5000.

However, compared to published multi-factor models, our RF and GBM-based models are not useful. They achieved between 10.73% and 12.94% smaller annual returns than published multi-factor models and significantly smaller Sharpe ratios. While our RF-based multi-factor model did achieve the lowest drawdown of all tested strategies (30.54%), this is not beneficial enough for such models to be considered viable.

Our DNN-based model produced far more promising results. Its annual return was between 2.46% and 4.22% less than published multi-factor models, which is reasonably competitive. Furthermore, its Sharpe ratio (0.779) was higher than all published multi-factor models by 0.136, which is a significant difference. Hence, DNN-based multi-factor models can provide a profitable, less risky alternative to published multi-factor models.

This is evident in Figure 4, where our DNN-based multi-factor model follows a smooth ascent in returns (compared to the chaotic behaviour of published multi-factor models).

Model	SR	DD (%)	CAR (%)
Buy and Hold	0.224	55.50	6.150
Fama-French	0.606	77.10	21.26
Carhart	0.643	75.60	**23.12**
RF-based MFM	0.538	**30.54**	10.73
GBM-based MFM	0.492	39.60	10.18
DNN-based MFM	**0.779**	39.16	18.90

6 MULTIMAX

Our model construction technique provides a valuable opportunity to educate traders on machine learning, value investing and multi-factor modelling. Hence, we have published MultiMax [48], a QuantConnect app that encapsulates our feature selection and backtesting methodology.

In MultiMax, users can:

- Establish a trading environment with their own dataset and parameters.
- Train a random forest, GBM or DNN to predict stock prices using their desired hyperparameters.
- Construct a multi-factor model using their ML model's N most important factors.
- Perform backtesting using their multi-factor model in the TopKDrop strategy.
- Save their models as pre-trained trading objects (PTOs) that can be shared and reused for live trading.
- Analyse and visualise the behaviour, accuracy, profitability, and risk of their ML and multi-factor models.

To access MultiMax, please visit our public GitHub release [7]. For information on this project's research and development, please see our private GitHub repository [50].

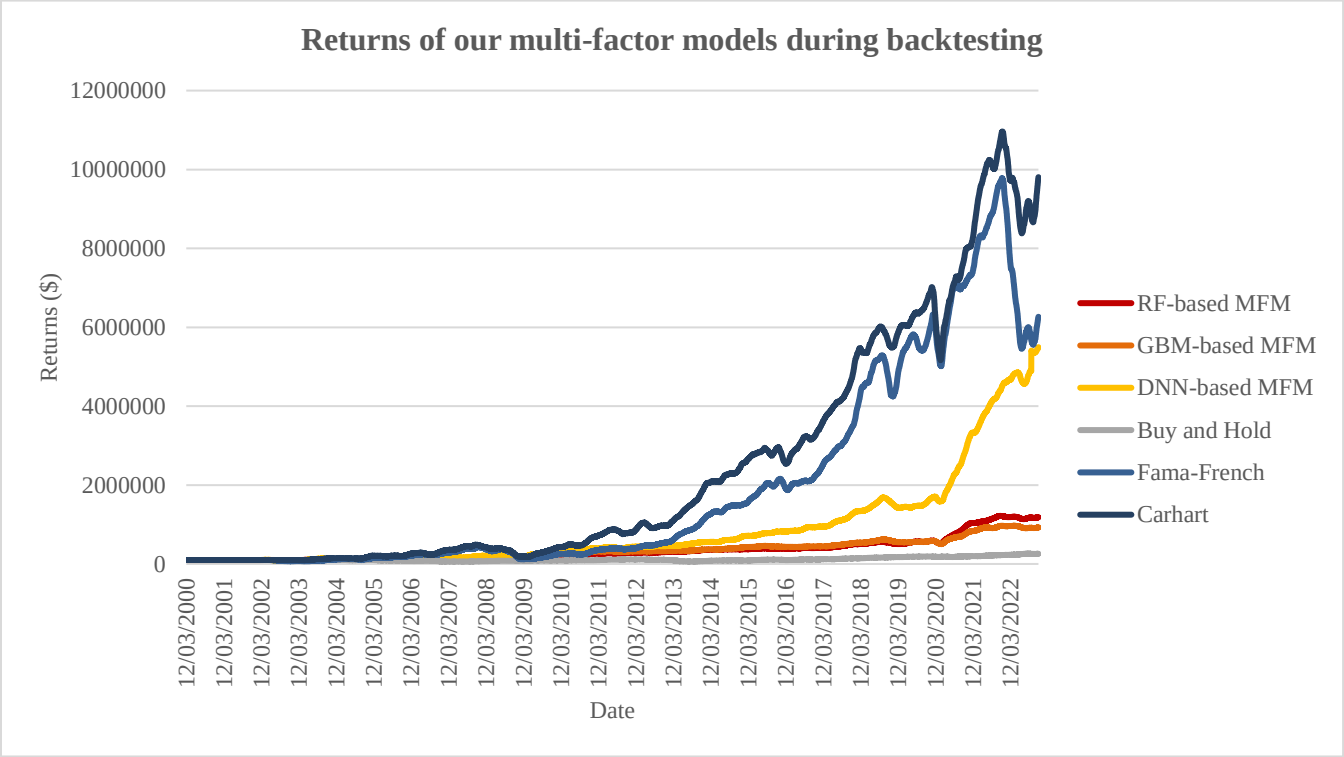

Figure 4: Returns of our multi-factor models, compared to various baselines, throughout the 23-year backtesting period.

See Appendix B for a showcase of screenshots from MultiMax.

Due to limitations imposed by QuantConnect, MultiMax can only be used in QuantConnect's online environment, with ML training times limited to 10 minutes and datasets limited to 100MB. Future work will involve publishing MultiMax as a standalone desktop app.

7 CONCLUSION

In this paper, we proposed a novel technique for constructing multi-factor models, using SHAP values for feature selection.

We have trained random forests, GBMs and DNNs to accurately predict a stock's growth using their fundamental and technical factors. We then used the SHAP values of these models to select important features and used these to train multi-factor models.

Our multi-factor models predict stock prices more accurately than standard linear regression, but with significant loss compared to their base models. DNN-based multi-factor models achieved the lowest prediction error. We also observed that the error of our multi-factor models correlates to the error of their base models.

We noticed that ML models consider technical factors to be more important than fundamental factors, although both were selected frequently. Interestingly, DNNs find fundamental factors more important than other tree-based models.

Finally, we performed a series of backtests using our constructed multi-factor models. All of our models were more profitable and less risky than the "Buy and Hold" strategy. However, our RF and GBM-based models performed significantly worse than existing multi-factor models in literature. Conversely, our DNN-based multi-factor models produced strong returns with less risk than published multi-factor models. Hence, our model construction technique is viable in creating effective multi-factor models from DNNs.

7.1 Future Work

The promising results of DNN-based multi-factor models present several opportunities for further research into our multi-factor modelling technique.

Further experiments could try constructing multi-factor models from state-of-the-art neural networks, like TEANet [X] and DAPP [X]. These could also be used to build temporal linear models (like VAR and VECM [50, 51]) or non-linear models (like non-linear MFMs and non-linear SVMs [52, 53]).

The nature of economic factors and their selection rates can also be investigated further. Future research could propose alternate metrics for a feature's importance or construct new factors based on those that are frequently selected.

BIBLIOGRAPHY

[1]: Graham, B. (1985). The intelligent investor: a book of practical counsel. New York, N.Y.: Harper & Row.

[2]: Feng, G., Giglio, S. and Xiu, D. (2020). Taming the Factor Zoo: A Test of New Factors. The Journal of Finance, 75(3). doi:https://doi.org/10.1111/jofi.12883.

[3]: Wang, H., Li, S., Wang, T. and Zheng, J. (2021). Hierarchical Adaptive Temporal-Relational Modeling for Stock Trend Prediction. doi:https://doi.org/10.24963/ijcai.2021/508.

[4]: Zhang, Q., Qin, C., Zhang, Y., Bao, F., Zhang, C. and Liu, P. (2022). Transformer-based attention network for stock movement prediction. Expert Systems with Applications, 202, p.117239. doi:https://doi.org/10.1016/j.eswa.2022.117239.

[5]: Lundberg, S. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, [online] 31, pp.4768–4777. doi:https://dl.acm.org/doi/10.5555/3295222.3295230.

[6]: Aidan Cooper. (2021). Explaining Machine Learning Models: A Non-Technical Guide to Interpreting SHAP Analyses. [online] Available at: https://www.aidancooper.co.uk/a-non-technical-guide-to-interpreting-shap-analyses/.

[7]: https://github.com/Thomas-Frew/MultiMax

[8]: Sharpe, W.F. (1964). Capital Asset Prices: A Theory of Market Equilibrium Under Conditions of Risk. The Journal of Finance, 19(3), pp.425–442. doi:https://doi.org/10.1111/j.1540-6261.1964.tb02865.x.

[9]: Li, B. and Zhong, M. (2010). Predictability of stock returns and consumption-based CAPM: Evidence from a small open market. 2010(22), pp.148–173.

[10]: Fama, E.F. and French, K.R. (1992). The Cross-Section of Expected Stock Returns. The Journal of Finance, 47(2), pp.427–465.

[11]: COCHRANE, J.H. (2011). Presidential Address: Discount Rates. The Journal of Finance, [online] 66(4), pp.1047–1108. doi:https://doi.org/10.1111/j.1540-6261.2011.01671.x.

[12]: Carhart, M.M. (1997). On Persistence in Mutual Fund Performance. The Journal of Finance, [online] 52(1), pp.57–82. Available at: https://doi.org/10.1111/j.1540-6261.1997.tb03808.x.

[13]: Roy, S.S., Mittal, D., Basu, A. and Abraham, A. (2015). Stock Market Forecasting Using LASSO Linear Regression Model. Advances in Intelligent Systems and Computing, pp.371–381. doi:https://doi.org/10.1007/978-3-319-13572-4_31.

[14]: Shen, J. and Shafiq, M.O. (2020). Short-term stock market price trend prediction using a comprehensive deep learning system. Journal of Big Data, [online] 7(1). doi:https://doi.org/10.1186/s40537-020-00333-6.

[15]: Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, pp.85–117. doi:https://doi.org/10.1016/j.neunet.2014.09.003.

[16]: Schöneburg, E. (1990). Stock price prediction using neural networks: A project report. Neurocomputing, 2(1), pp.17–27. doi:https://doi.org/10.1016/0925-2312(90)90013-h.

[17]: White (1988). Economic prediction using neural networks: the case of IBM daily stock returns. IEEE International Conference on Neural Networks. doi:https://doi.org/10.1109/icnn.1988.23959.

[18]: Fama, E.F. (1970). Efficient Capital Markets: a Review of Theory and Empirical Work. The Journal of Finance, [online] 25(2), pp.383–417. doi:https://doi.org/10.2307/2325486.

[19]: Lam, M. (2004). Neural network techniques for financial performance prediction: integrating fundamental and technical analysis. Decision Support Systems, 37(4), pp.567–581. doi:https://doi.org/10.1016/s0167-9236(03)00088-5.

[20]: Fang, Y. (2018). Feature Selection, Deep Neural Network and Trend Prediction. Journal of Shanghai Jiaotong University (Science), 23(2), pp.297–307. doi:https://doi.org/10.1007/s12204-018-1938-5.

[21]: Weng, B., Martinez, W., Tsai, Y.-T., Li, C., Lu, L., Barth, J.R. and Megahed, F.M. (2018). Macroeconomic indicators alone can predict the monthly closing price of major U.S. indices: Insights from artificial intelligence, time-series analysis and hybrid models. Applied Soft Computing, 71, pp.685–697. doi:https://doi.org/10.1016/j.asoc.2018.07.024.

[22]: Chen, C.-C., Huang, H.-H. and Chen, H.-H. (2019). Crowd View: Converting Investors' Opinions into Indicators. Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19). doi:https://doi.org/10.24963/ijcai.2019/936.

[23]: Soni, P., Tewari, Y. and Krishnan, D. (2022). Machine Learning Approaches in Stock Price Prediction: A Systematic Review. Journal of Physics: Conference Series, 2161(1), p.012065. doi:https://doi.org/10.1088/1742-6596/2161/1/012065.

[24] Wang, H., Wang, T. and Li, Y. (2020). Incorporating Expert-Based Investment Opinion Signals in Stock Prediction: A Deep Learning Framework. Proceedings of the AAAI Conference on Artificial Intelligence, 34(01), pp.971–978. doi:https://doi.org/10.1609/aaai.v34i01.5445.

[25] Wu, S., Liu, Y., Zou, Z. and Weng, T.-H. (2021). S_I_LSTM: stock price prediction based on multiple data sources and sentiment analysis. Connection Science, pp.1–19. doi:https://doi.org/10.1080/09540091.2021.1940101.

[26]: [7] Aslam, N., Rustam, F., Lee, E., Washington, P.B. and Ashraf, I. (2022). Sentiment Analysis and Emotion Detection on Cryptocurrency Related Tweets using Ensemble LSTM-GRU Model. IEEE Access, pp.1–1. doi:https://doi.org/10.1109/access.2022.3165621.

[27]: Dautel, A.J., Härdle, W.K., Lessmann, S. and Seow, H.-V. (2020). Forex exchange rate forecasting using deep recurrent neural networks. Digital Finance. doi:https://doi.org/10.1007/s42521-020-00019-x.

[28]: Fabbri, M. and Moro, G. (2018). Dow Jones Trading with Deep Learning: The Unreasonable Effectiveness of Recurrent Neural Networks. Proceedings of the 7th International Conference on Data Science, Technology and Applications. doi:https://doi.org/10.5220/0006922101420153.

[29]: Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), pp.1735–1780. doi:https://doi.org/10.1162/neco.1997.9.8.1735.

[30]: Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. (2017). Attention is All you Need. [online] Neural Information Processing Systems. Available at: https://dl.acm.org/doi/10.5555/3295222.3295349.

[31]: Tuna Tuncer, Kaya, U., Emre Sefer, Onur Alacam and Hoser, T. (2022). Asset Price and Direction Prediction via Deep 2D Transformer and Convolutional Neural Networks. doi:https://doi.org/10.1145/3533271.3561738.

[32]: Zhang, Q., Qin, C., Zhang, Y., Bao, F., Zhang, C. and Liu, P. (2022). Transformer-based attention network for stock movement prediction. Expert Systems with Applications, 202, p.117239. doi:https://doi.org/10.1016/j.eswa.2022.117239.

[33]: Subba , R.P., Kr, S. and Ar, K.M. (2019). Stock Market Prices Prediction using Random Forest and Extra Tree Regression. International Journal of Recent Technology and Engineering, 8(3), pp.1224–1228. doi:https://doi.org/10.35940/ijrte.c4314.098319.

[34]: Basak, S., Kar, S., Saha, S., Khaidem, L. and Dey, S.R. (2019). Predicting the direction of stock market prices using tree-based classifiers. The North American Journal of Economics and Finance, [online] 47, pp.552–567. doi:https://doi.org/10.1016/j.najef.2018.06.013.

[35]: Breiman, L. (2001). Random Forests. Machine Learning, 45(1), pp.5–32. doi:https://doi.org/10.1023/a:1010933404324.

[36]: Friedman, J.H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics, [online] 29(5), pp.1189–1232. Available at: https://www.jstor.org/stable/2699986.

[37]: Investopedia. (2022). Wilshire 5000 Total Market Index (TMWX) Definition. [online] Available at: https://www.investopedia.com/terms/w/wilshire5000equityindex.asp.

[38]: US Securities and Exchange Commission (2006). Electronic Filing and EDGAR (October 2006). [online] Available at: https://www.sec.gov/info/edgar/regoverview.htm.

[39]: Ali, M.A., Hickman, P.J. and Clementson, A.T. (1975). The Application of Automatic Interaction Detection (AID) in Operational Research. Operational Research Quarterly (1970-1977), 26(2), p.243. doi:https://doi.org/10.2307/3008458.

[40]: Hocking, R.R. (1983). Developments in Linear Regression Methodology: 1959-1982. Technometrics, 25(3), p.219. doi:https://doi.org/10.2307/1268603.

[41]: Duan, Y., Wang, L., Zhang, Q. and Li, J. (2022). FactorVAE: A Probabilistic Dynamic Factor Model Based on Variational Autoencoder for Predicting Cross-Sectional Stock Returns. Proceedings of the AAAI Conference on Artificial Intelligence, 36(4), pp.4468–4476. doi:https://doi.org/10.1609/aaai.v36i4.20369.

[42]: Hou, M., Xu, C., Liu, Y., Liu, W., Bian, J., Wu, L., Li, Z., Chen, E. and Liu, T.-Y. (2021). Stock Trend Prediction with Multi-granularity Data. Proceedings of the 30th ACM International Conference on Information & Knowledge Management. doi:https://doi.org/10.1145/3459637.3482483.

[43]: Wang, Z., Huang, B., Tu, S., Zhang, K. and Xu, L. (2021). DeepTrader: A Deep Reinforcement Learning Approach for Risk-Return Balanced Portfolio Management with Market Conditions Embedding. Proceedings of the AAAI Conference on Artificial Intelligence, 35(1), pp.643–650. doi:https://doi.org/10.1609/aaai.v35i1.16144.

[44]: Niu, H., Liu, S. and Li, J. (2022). MetaTrader. Proceedings of the 31st ACM International Conference on Information & Knowledge Management. doi:https://doi.org/10.1145/3511808.3557363.

[45]: Oshiro, T.M., Perez, P.S. and Baranauskas, J.A. (2012). How Many Trees in a Random Forest? Machine Learning and Data Mining in Pattern Recognition, 7376, pp.154–168. doi:https://doi.org/10.1007/978-3-642-31537-4_13.

[46]: Bentéjac, C., Csörgő, A. and Martínez-Muñoz, G. (2020). A comparative analysis of gradient boosting algorithms. Artificial

Intelligence Review, 54. doi:https://doi.org/10.1007/s10462-020-09896-5.

[47]: Jin, H., Song, Q. and Hu, X. (2019). Auto-Keras: An Efficient Neural Architecture Search System. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '19. [online] doi:https://doi.org/10.1145/3292500.3330648.

[48]: QuantConnect. (2023). QuantConnect. [online] Available at: https://www.quantconnect.com/.

[49]: Transaction Fees. [online] QuantConnect. Available at: https://www.quantconnect.com/docs/v2/writing-algorithms/reality-modeling/transaction-fees/key-concepts [Accessed 28 Oct. 2023].

[50]: https://github.com/Thomas-Frew/ML-Autotrader-23 (invitation only)

[51]: Sims, C.A. (1980). Macroeconomics and Reality. Econometrica, [online] 48(1), p.1. doi:https://doi.org/10.2307/1912017.

[52]: Granger, C.W.J. and Newbold, P. (1974). Spurious regressions in econometrics. Journal of Econometrics, 2(2), pp.111–120. doi:https://doi.org/10.1016/0304-4076(74)90034-7.

[53]: Chicheportiche, R. and Bouchaud, J.-P. . (2015). A nested factor model for non-linear dependencies in stock returns. Quantitative Finance, 15(11), pp.1789–1804. doi:https://doi.org/10.1080/14697688.2014.994668.

[54]: Suykens, J.A.K. (2001). Support Vector Machines: A Nonlinear Modelling and Control Perspective. European Journal of Control, 7(2-3), pp.311–327. doi:https://doi.org/10.3166/ejc.7.311-327.

APPENDIX A

A visualisation of our methodology, from dataset creation to backtesting.

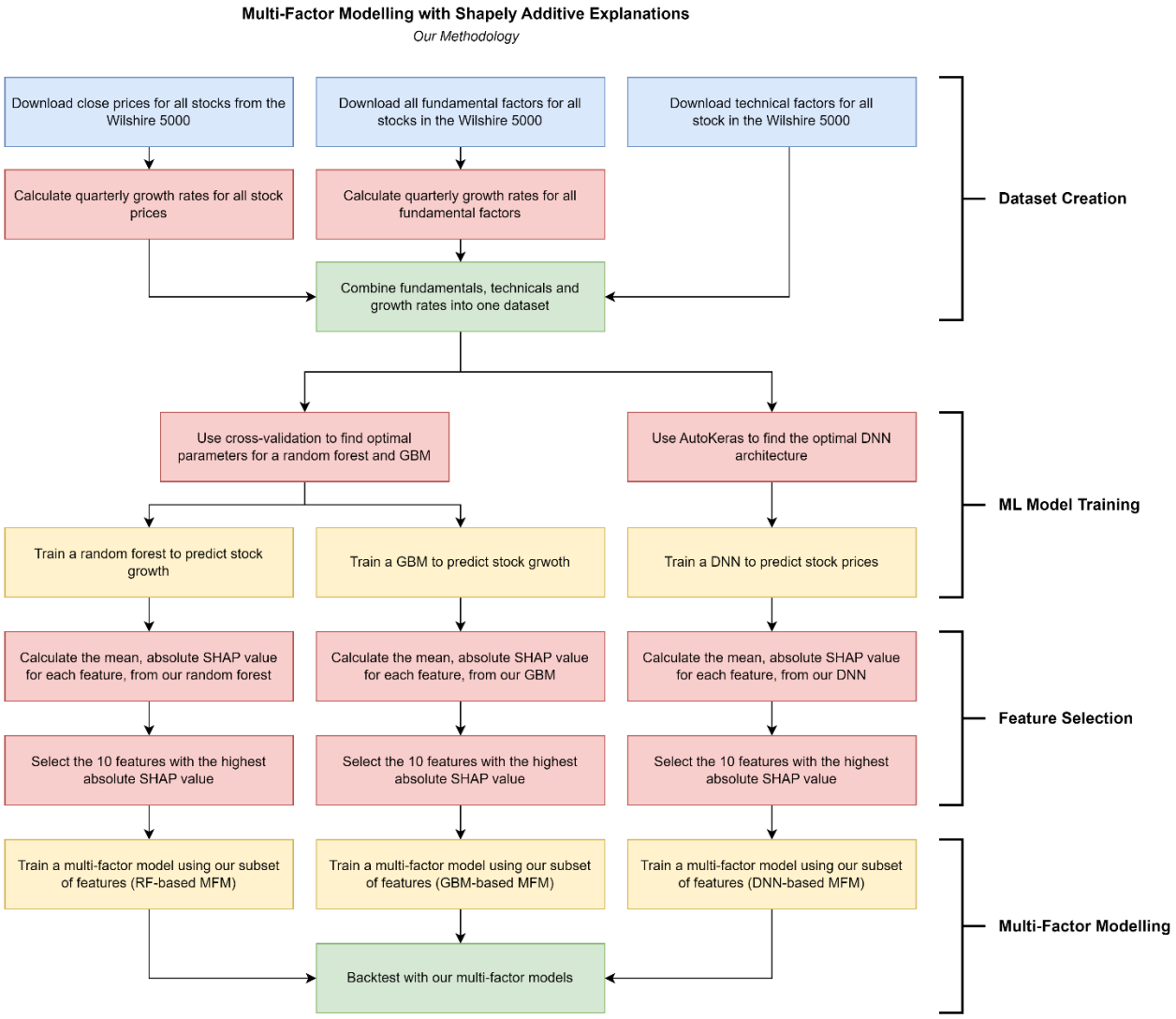

Figure 5: Our multi-factor modelling methodology.

APPENDIX B

Screenshots from MultiMax, demonstrating its features and backend. Graphing and backtesting statistics provided by QuantConnect.

```
# Trading method (manual/automatic/pretrained)
self.method = "Automatic"

# Trading parameters
self.source = "[SOURCE]"

self.start_date = datetime(2000, 1, 1)
self.end_date = datetime(2023, 1, 1)

self.start_cash = 100000
self.rebalance_days = 300
self.num_long = 10

# ML model parameters (RF, GBM, DNN)

self.num_trees = 32
self.max_depth = 5

self.batch_size = 32
self.epochs = 16

# Multi-factor model parameters
self.num_features = 10
```

Figure 6: Parameters and hyperparameters for training and backtesting

```
def TrainRandomForest(self, trees, depth, num_features):

    self.qc.Debug(f"Training Random Forest regressor with {trees} trees {depth} levels deep")

    # Create a Random Forest model and train it
    self.model = RandomForestRegressor(n_estimators=trees, max_depth=depth, random_state=self.seed_2)
    self.model.fit(self.X_train, self.y_train)

    # Test the base model
    self.ml_preds = self.model.predict(self.X_test)
    mae = mean_absolute_error(self.y_test,self.ml_preds)

    self.qc.Debug(f"Random Forest trained with MAE {mae}.")

    # Do SHAP on random forest
    explainer = shap.TreeExplainer(self.model)
    shap_values = explainer.shap_values(self.X_test)

    # Sort by SHAP values and extract the N most important features
    feature_importance = np.abs(shap_values).mean(axis=0)
    selected_features = self.X_train.columns[np.argsort(feature_importance)[::-1][:num_features]]

    # Prepare the data with the selected features
    X_train_selected = self.X_train[selected_features]
    X_test_selected = self.X_test[selected_features]

    # Train a linear regression model with the selected features
    self.lr_model = LinearRegression()
    self.lr_model.fit(X_train_selected, self.y_train)

    # Evaluate the performance of the linear regression model
    self.lr_preds = self.lr_model.predict(X_test_selected)
    mae = mean_absolute_error(self.lr_preds, self.y_test)
```

Figure 7: Part of the source code for training a random forest, performing feature selection, and training a multi-factor model on these features.

```
Launching analysis for 851ee4a58d589a5166dcccc67a315910 with LEAN Engine v2.5.0.0.15930
Successfully loaded dataset of shape (42330, 346)
Successfully loaded train set (38097, 345) and test set (4233, 345)
Model Generator created
Training Random Forest regressor with 16 trees 3 levels deep
Random Forest trained with MAE 20.539696352113793.
Linear Regression model distilled with MAE: 22.222711258478423
Ichimoku Leading Span A, -128.35003839683952
Market Cap (Growth), -107.20545748389479
Advancers - Decliners, 37.82057018988174
Interest Income, -36.45342990747671
Property Plant and Equipment_y, 15.461884694036295
Capital Expenditure, 15.461884694036229
Depreciation and Amortization_y, 8.546994845580413
Change in Working Capital, 6.005134153455608
Free Cash Flow to Operating Cash Flow Ratio, -2.6242021430770026
Free Cash Flow to Operating Cash Flow Ratio (Growth), -2.624202143076979
```

Figure 8: Output from training an ML model and using its top 10 features to construct a multi-factor model.

```
Algorithm complete with a total return of 22.58411197 and Sharpe Ratio 0.7217682759234555.
Top 5 Investments:
SPY: 5
CPT: 5
FANG: 5
FDS: 5
KEYS: 5
Packaged and Exported Data as Salt.pto.
Algorithm Id:(851ee4a58d589a5166dcccc67a315910) completed in 141.36 seconds at 218k data points per second. Processing total of 30,822,622 data points.
```

Figure 9: Output from a completed backtest.

Figure 10: QuantConnect's UI for visualising our multi-factor model's returns over time and invested assets.

PSR	17.664%	Sharpe Ratio	0.722
Total Trades	234	Average Win	4.58%
Average Loss	-1.54%	Compounding Annual Return	14.722%
Drawdown	34.700%	Expectancy	1.845
Net Profit	2258.411%	Loss Rate	28%
Win Rate	72%	Profit-Loss Ratio	2.96
Alpha	0.072	Beta	0.432
Annual Standard Deviation	0.122	Annual Variance	0.015
Information Ratio	0.38	Tracking Error	0.136
Treynor Ratio	0.204	Total Fees	$3580.27
Estimated Strategy Capacity	$2900000.00	Lowest Capacity Asset	CDAY WJR7CQEIKN29
Portfolio Turnover	0.29%		

Figure 11: QuantConnect's UI for visualising backtesting statistics.

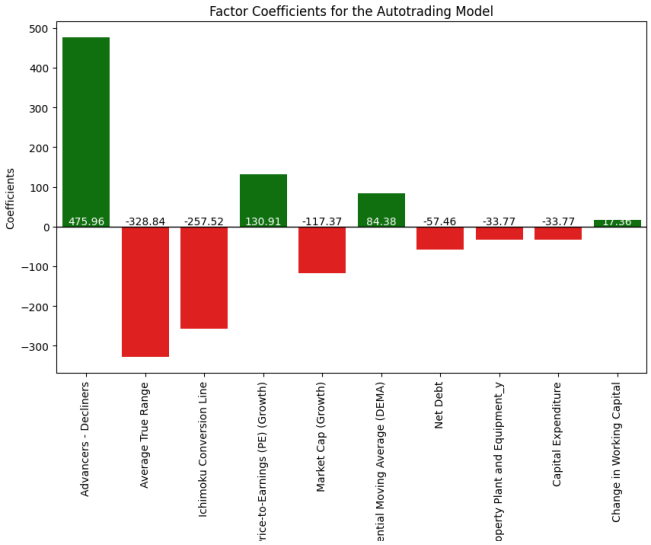

Figure 12: Visualising the coefficients of our multi-factor model (results tool).

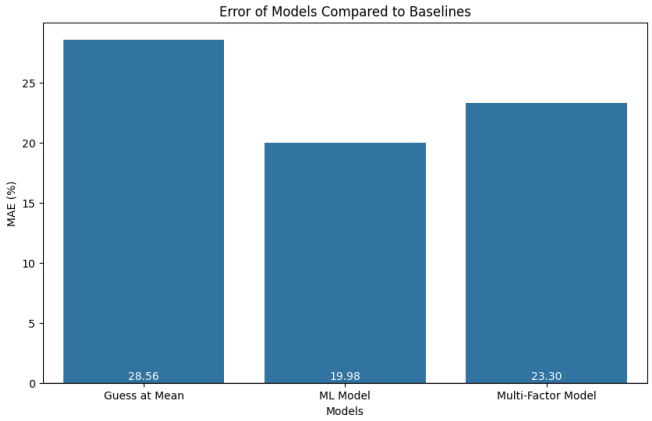

Figure 13: Visualising the error of our ML and multi-factor model, relative to simply guessing at the mean (results tool).

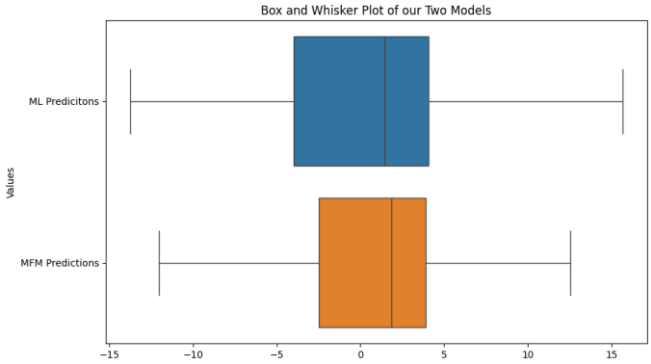

Figure 14: Visualising the predictions of our ML and multi-factor model (results tool).

Statistic	Value
Total Return	9.56373
Annualised Return	0.10788
Sharpe Ratio	0.552808
Max Drawdown	0.265
Alpha	0.0463114
Beta	0.375161
Win Rate	0.265
Information Ratio	0.173155

Figure 15: Visualising the statistics from our backtest (results tool).