Erasmus University Rotterdam, Erasmus School of Economics

Master Thesis Econometrics and Management Science

Master Specialisation in Quantitative Finance

Harnessing the Power of Neural Networks for Equity Premium Forecasting

Student: Christien van de Kerk *Supervisor:* DJC van Dijk

Student ID number: 448897 *Second assessor:* PV Vallarino

1st October 2023

Abstract

This study explores to what extent the long short-term memory (LSTM) neural network model can improve the performance of stock return forecasting models, with a focus on the U.S. market excess return. The LSTM architecture, introduced to enhance neural networks' performance when dealing with sequential time-series data by the use of an internal memory mechanism, is contrasted with the simpler yet highly effective feedforward neural network (FNN) model, and two additional linear benchmark models. The results show that, though the LSTM model's forecast demonstrates enhanced statistical accuracy compared to all benchmark models, the FNN model offers higher economic value to an investor. This can be attributed to the fact that certain information considered as outliers or noise by the LSTM model's memory mechanism, appears to be of significant value for forecasting equity returns. Furthermore, the analysis of features' contribution to the predictions shows that the individual feature importance ranking is relatively similar across the linear and non-linear forecasts. However, the neural networks models' ability to capture interaction effects among the features, yields contrasting results regarding the effect of the trading signals. Overall, this paper provides a nuanced understanding of the inner workings of neural networks and the contribution of individual features to their stock return predictions, hence should provide researchers and investors employing financial forecasting models valuable tools and insights.

The content of this thesis is the sole responsibility of the author and does not reflect the view of the supervisor, second assessor, Erasmus School of Economics or Erasmus University.

Table of Contents

List of Abbreviations

Adj-DM	Adjusted Diebold and Mariano or Clark and West test
CPI	Consumer price index
DGP	Data generating process
FNN	Feedforward neural network
LSTM	Long short-term memory
MA	Moving average
NBER	National bureau of economic research
MSFE	Mean squared forecast error
OBV	On-balance volume
OLS	Ordinary least squares
OOS	Out-of-sample
ReLU	Rectified linear unit
RMSFE	Root mean squared forecast error
RNN	Recurrent neural network
SELU	Scaled exponential linear unit
SGD	Stochastic gradient descent
SHAP	Shapley additive explanation

1 Introduction

The predictability of stock market returns has been a topic of great interest in the field of finance for numerous years. Researchers persistently strive to forecast the future value of financial instruments traded on exchanges. As one of the pioneers in this field, Cowles (1933) investigated the predictability of the equity premium and presented his findings in the article: 'Can Stock Market Forecasters Forecast?'. He primarily concluded that modelled portfolio returns do not significantly differ from what would be expected with pure luck, and long ago suggested a negative answer to the central question at hand.

Despite the modest empirical results, due to the inherent level of uncertainty and randomness in the behaviour of stock returns, the canonical problem of measuring the equity premium continues to attract considerable attention. This paper contributes to the ongoing discussion by re-examining the question and introduces a novel forecasting model within this research domain: the long short-term memory (LSTM) neural network. The performance of the LSTM neural network model in forecasting the equity premium will be evaluated and compared to that of the well-established feedforward neural network (FNN), and two additional benchmark models.

Since its inception, the LSTM model has emerged as a promising and successful architecture within the realm of neural networks. In particular, the LSTM architecture is known for its ability to model and recognise time dependencies in data. This capability enables the network to effectively capture temporal relationships, while handling noise and accurately predict continuous variables. The model's forecasting efficiency has been demonstrated across various research disciplines that deal with sequential data, where the order of data points matters, such as text, audio, and video analysis. Given the model's proven competence in handling noisy, time-series data, which are aspects of stock returns, the theoretical arguments, supported by Wüthrich and Merz (2023), Taddy (2018), Hochreiter and Schmidhuber (1997), and Yu et al. (2019), lean towards employing the LSTM model in forecasting equity returns. However, while the LSTM architecture possesses strong capabilities, its predictive power within the context of asset pricing remains relatively unexplored. Hence, this paper aims to provide a comprehensive evaluation of the LSTM network's efficiency in forecasting the equity premium, investigating the model's potential in this critical financial domain.

Despite the limited research specifically on the LSTM model in stock forecasting, studies by Rapach and Zhou (2020), Gu et al. (2020), and Ican et al. (2017), suggest that, when looking at the broader machine learning framework, the models can significantly enhance predictive accuracy for the equity premium problem, characterised by the large set of potential predictors and high degree of noise. Machine learning offers a diverse collection of high-dimensional models, which enhances flexibility in approximating the complex data generation process (DGP) underlying the equity premium. The models mitigate the risk of overfitting, which is critical for out-of-sample (OOS) forecasting performance when dealing with many input features, while maintaining a balance between capturing relevant patterns and avoiding noise in the data. Neural networks are the most powerful and versatile models in machine learning, and the architecture emerged as the preferred approach for modelling complex challenges. They have the theoretical foundation as 'universal approximators', and studies by Cybenko (1989), and Hornik et al. (1989) demonstrate that even shallow neural networks can effectively approximate any continuous function. Gu et al. (2020) evaluate and compare the predictive performance of top performing models in forecasting stock returns, and conclude that neural networks, specifically they evaluate the FNN architecture, consistently outperform other forecasting models. The researchers conclude that incorporating non-linearity significantly enhances the accuracy of predictions. The superior performance demonstrated by neural network models, when compared to other (non-linear) machine learning techniques including regression trees, suggests that their potential to capture even more complex, non-linear relationships and interactions in the data is a crucial component to effectively model the equity premium. The employed FNN model represents the most traditional form of neural network and since its inception, more sophisticated and advanced network architectures, like the LSTM model, have been developed. Incorporating the FNN in the analysis of this study facilitates a comprehensive comparison of the LSTM model's performance, to that of one of the most effective equity premium forecasting models applied currently.

Feedforward indicates that at each time step, the information in the input features is passed in a directed, acyclic path through the network to generate a forecast. A FNN lacks the ability to retain information from previous time steps in its network to predict the current output. An extension to this architecture, allowing the network to have cycles and designed to work with sequential data, is referred to as a recurrent neural network (RNN). A RNN model incorporates feedback loops that enables the network to use information from preceding time steps to predict the current output. However, Hochreiter and Schmidhuber (1997) show that estimating the parameters in a traditional RNN causes problems. As a solution to address the limitations of the conventional RNN model, they introduce the LSTM neural network, which quickly emerged as one of the most successful RNN architectures. The distinguishing feature of an LSTM model is its incorporation of a complex memory mechanism. This unique characteristic allows the model to store information, in order to effectively identify and integrate both short and long-time dependencies, thereby enhancing its predictive capabilities, particularly in noisy time-series data. Yet, existing literature that utilises neural networks in predicting stock returns, is predominantly focused on FNNs. In practice, complex models aren't always superior, and a simpler architecture like the FNN might capture the primary dynamics of the underlying DGP sufficiently well. It could be that the equity premium doesn't have sequential dependencies significant enough to be captured by the LSTM model. The challenge is to understand weather the LSTM model's memory mechanism adds significant value in forecasting the equity premium. In other words, does the added complexity due to the memory mechanism lead to better results than obtained with the simpler, yet already promising FNN model?

Building on the insights of Leung et al. (2000), which highlight that a prediction with a low forecast error does not necessarily guarantees high economical value, this research is conducted in two phases. Firstly, the LSTM model is employed to forecast the U.S. monthly equity premium with the aim of enhancing statistical accuracy, measured by the mean squared forecast error (MSFE). Secondly, the potential economic gains associated with using the LSTM architecture in forecasting stock returns are evaluated. The economic value is assessed based on the realised utility to an investor in a mean-variance frame-work. The difference in obtained utility from using distinct forecasting models when making investment decisions, can be regarded as a portfolio management fee an investor would be willing to pay in order to switch from one forecasting model to another. Overall, the objective of this study is to answer the following research question:

To what extent can an LSTM neural network model improve the performance of equity premium forecasting models?

To address this question, the study incorporates a comprehensive set of 29 predictive features into the forecasting models. As proposed by Brock et al. (1992) and Neely et al. (2014), these include both technical and fundamental variables, aiming to enhance the performance of the asset pricing models.

The evaluation of the forecasting performance of the LSTM model uses two additional benchmark models for comparison: a combination forecast and the historical average forecast. These models offer a comparison between linear and non-linear methods for processing input features and serve as a baseline in the analysis. The combination forecast model is introduced by Rapach et al. (2010), to address the limitations of conventional ordinary least squares (OLS) regression models, aiming to improve OOS forecasting power within the asset pricing context. Historically, there's been a longstanding and widespread belief suggesting that numerous economic variables exhibit predictive power in forecasting the equity premium. Summarised by Lettau and Ludvigson (2001, p. 842): 'It is now widely accepted that excess returns are predictable by variables such as dividend-price ratios, earnings-price ratios, dividend-earnings ratios, and an assortment of other financial indicators.' However, a re-examination of the empirical evidence by Welch and Goyal (2008), concluded that none of the predictive variables individually performs well in forecasting the equity premium, and lack potential for assisting investors in profitable market predictions. Moreover, while a multivariate regression model exhibits robust in-sample significance, it demonstrates poor OOS performance. The traditional OLS regression estimation seeks to maximise the model's explanatory power within the estimation sample and is therefore susceptible to overfitting, resulting in the poor OOS accuracy. The researchers' pioneering findings highlight the need for improved asset return forecasting models that have the ability to handle many input features, mitigate the risk of overfitting, and demonstrate good performance both in-sample and OOS.

Following these controversial findings on return predictability, Rapach et al. (2010) suggest a simple average of univariate OLS regression forecasts to generate a combined forecast. The proposed combination forecast model offers an effective shrinkage strategy, addressing the problem of multicollinearity present in the multivariate OLS regression forecast, whilst avoiding overfitting. The method demonstrates consistent and significant OOS gains relative to the historical average forecast. Therefore, the combination forecast is the first model to provided evidence that, when considered collectively, predictive features do hold substantial value and predictive power in forecasting the equity premium, both in-sample and OOS. The inclusion of the combination forecast, and historical average forecast as benchmark models in this study, not only enriches the analysis but also offers valuable insights into the historical context and the evolutionary path of stock returns forecasting models.

While the focus of this study lies on enhancing OOS forecasting accuracy, there is substantial value in comprehending the nuances of the forecasting models. In order to make informed investment decisions, particularly when dealing with numerous predictors, it is crucial for investors to understand which ones play a relevant role in the forecasts. In addition, the ability to accurately interpret a model's output, provides a foundation to enhance the forecasts, and gain insights into the underlying process. However, due to the complex and highly parameterised inner structure of the neural network architectures, they are less transparent and harder to interpret compared to simpler (linear) forecasting methods. The models are often referred to as 'black boxes', as understanding how they arrive at their predictions can be quite challenging.

To address this challenge and gain insights into the forecasting model's decision-making process, the Shapley additive explanations (SHAP) method, as proposed by Lundberg and Lee (2017), is integrated into the evaluation of the forecasting models. A recent literature review of existing machine learning interpretability methods by Linardatos et al. (2020), concludes that the SHAP method is the most comprehensive and general approach for assessing feature importance. The method is model-agnostic, and can be applied to any type of data, hence suitable in this study.

The forecasting performance of the employed models is evaluated over the OOS period, spanning from January 1990 to December 2022. The empirical results demonstrate superior forecasting performance for both neural network models compared to the linear combination forecast and historical average forecast. Based on economic value evaluation, a traditional FNN model outperforms the LSTM model. In contrast, the LSTM model shows small improvements in MSFE relative to the FNN model. The LSTM architecture's memory mechanism enables the model to recognise patterns in the input features over time. However, by doing so, the model considers valuable information in some of the extreme values as outliers or noise. As a result, the LSTM model shows a more robust and stable forecast, whereas the FNN model seems to stand out in capturing the extreme values in the equity premium, which leads to its enhanced economic value.

The study shows that the predictive features contain valuable information for forecasting the equity premium, and the individual feature importance ranking remains relatively similar across the distinct forecasting models. The neural network models have the capabilities to capture complex patterns and interactions among the features, leading to counter-intuitive feature effects on the forecasts, which can be of value to researchers and investors.

The remainder of this study is structured as follows: Section 2 outlines the data used in conducting the research, and the construction of predictive features from the raw data. Sections 3.1 through 3.5 elaborate on the forecasting models employed, namely the LSTM neural network, FNN, combination forecast, and historical average forecast. Subsequently, Sections 3.6 through 3.8, provide three evaluation methods to analyse the forecasts. The empirical results are discussed in Section 4, and Section 5 performs two robustness checks to validate and reinforce the results. The study concludes with Section 6, providing the answer to the research question.

2 Data

The objective of this study is to forecast the equity premium by employing an LSTM neural network model, and to compare the forecasting results to that of the benchmark models. The research question will be addressed in the context of the U.S. equity premium, specifically the monthly S&P 500 excess return. The excess return is defined, consistent with the widely employed definition by Welch and Goyal (2008), as the continuously compounded log return on a value-weighted S&P 500 market portfolio, in excess of a risk-free rate. The risk-free rate is based on the three-month Treasury bill rate. The sample period spans from January 1950 to December 2022, consisting of 876 monthly observations. The average monthly equity premium is 0.57%, with a standard deviation of 4.23%.

Forecasting financial time series data entails predicting the future value of the series, frequently based on various input features. The model's forecasting power largely depends on these input features. In stock analysis, these features are primarily classified into two categories: fundamental and technical features. Fundamental analysis considers intrinsic value factors like financial statements, earnings, and macroeconomic trends, while technical analysis uses statistical trends like price movements and trading volume for future performance insights. While many existing asset pricing models predominantly focus on fundamental features, studies by Brock et al. (1992) and Neely et al. (2014) suggest the importance of including technical features for comprehensive and accurate forecasts. In line with these findings, a review by Ican et al. (2017) indicates that the use of both fundamental and technical indicators can enhance the predictive performance of a neural network in stock market analysis. Accordingly, this study integrates both types of predictive features as inputs for the forecasting models. The set of considered monthly fundamental features consists of 14 widely recognised macroeconomic features addressed in the empirical literature, following amongst others, Welch and Goyal (2008), and Rapach and Zhou (2020):

1. **Log dividend-price ratio** (DP): the log of the 12-month moving sum of dividends paid on the S&P500 index minus the log of the S&P 500 index.

2. **Log earnings-price ratio** (EP): log of a 12-month moving sum of earnings on the S&P 500 index minus the log of the S&P 500 index.

3. **Log dividend yield** (DY): log of a 12-month moving sum of dividends minus the log of lagged S&P 500 index.

4. **Log dividend-payout ratio** (DE): log of a 12-month moving sum of dividends minus the log of a 12-month moving sum of earnings on the S&P 500 index.

5. **Stock variance** (SVAR): monthly sum of squared daily returns on the S&P 500 index.

6. **Book-to-market ratio** (BM): book-to-market value ratio for the Dow Jones Industrial Average.

7. **Net equity expansion** (NTIS): ratio of a 12-month moving sum of net equity issues by NYSE-listed stocks to the total end-of-year market capitalisation of NYSE-lited stocks.
8. **Treasury bill rate** (TBL): three-month Treasury bill yield (secondary market).
9. **Long-term yield** (LTY): long-term government bond yield.
10. **Long-term return** (LTR): return on long-term government bonds.
11. **Term spread** (TMS): long-term government bond yield minus the Treasury bill rate.
12. **Default yield spread** (DFY): difference between BAA- and AAA-rated corporate bond yields.
13. **Default return spread** (DFR): long-term corporate bond return minus the long-term government bond return.
14. **Inflation** (INFL): inflation calculated from the Consumer Price Index (CPI) (all urban consumers), lagged value to account for the delay in CPI releases.

The data required to construct these macroeconomic features are regularly updated and available through Amit Goyal's website[1]. Further summary statistics for the equity premium and the fundamental features are provided in Table 7, Appendix A.

Besides the 14 fundamental features, a recession dummy, and 14 technical features are used as input for the forecasting models. The recession dummy, derived from the National Bureau of Economic Research (NBER), adopts a value of 1 during recessions, which accounts for 11% of the months. Conversely, it takes on a value of 0 during business-cycle expansions, representing the remaining period. As presented in Table 7, Appendix A, several features demonstrate a high level of auto-correlation, suggesting a strong relationship between their values at different points in time. In order to detect these trends in the data and identify instances of changing trends, the technical input features are generated based on three widely employed trading strategies in line with Neely et al. (2014).

The first trading strategy revolves around the moving average (MA) rule. It generates a buy signal when the $\text{MA}_{s,t}$ price index over a short period, is greater than or equal to the $\text{MA}_{l,t}$ price index over a long period. Conversely, it generates a sell signal when the short $\text{MA}_{s,t}$ is below the long $\text{MA}_{l,t}$. A total of six monthly MA buy or sell signals are generated, denoted as $\text{MA(s,l)}_t = 1$ or $\text{MA(s,l)}_t = 0$, respectively. Where s = 1, 2 or 3, represents the short period, and l = 9 or 12, the long period. The second trading strategy is based on momentum. It

[1]The data are available from Amit Goyals's web page at https://sites.google.com/view/agoyal145

generates a buy signal when the current price exceeds the price of m months ago, and a sell signal when the current price is lower than the price of m months ago. Two monthly momentum signals are generated, denoted as $\text{MOM}(\text{m})_t$, for m = 9 or 12 months. The final trading strategy combines traded volume with prices to identify overall market trends. This strategy uses the on-balance volume (OBV), which is calculated by multiplying the total traded volume during a specific period by 1 in case of a price increase, and by -1 in case of a price decrease over that specific period. The trading signals are obtained by comparing the short $\text{MA}_{s,t}^{OBV}$ of the OBV, where s = 1, 2 or 3, represents the short period, with the long $\text{MA}_{l,t}^{OBV}$, where l = 9 or 12 denotes the long period. If the short $\text{MA}_{s,t}^{OBV}$ is greater than or equal to the long $\text{MA}_{l,t}^{OBV}$, a buy signal is generated, conversely, if $\text{MA}_{l,t}^{OBV}$ exceeds $\text{MA}_{s,t}^{OBV}$, a sell signal is generated. The motivation behind the OBV strategy is that a relatively high trading volume in recent months, combined with a recent price increase, implies a strong positive market trend. As a result, the short $\text{MA}_{s,t}^{OBV}$ exceeds the long $\text{MA}_{l,t}^{OBV}$, and a buy signal is generated. A total of six monthly signals, denoted as $\text{VOL}(\text{s,l})_t$, are generated. Further details and the equations related to the three trading strategies are provided in Appendix C.1. The technical indicators generate a buy signal for approximately 72% of the time. Among the indicators, MOM(12) generates the highest proportion of buy signals (74%), while VOL(2,9) generates the lowest proportion (69%).

In total, the forecasting models leverage a combination of 29 predictive features, encompassing both fundamental and technical aspects. The respective values of these fundamental and technical input features throughout the entire OOS forecast horizon are illustrated in Figures 8 and 9, Appendix B. Visually, the predictors represent a variety of information sources. Notably, all features, except for the TMS, demonstrate increased volatility during recession periods.

The OOS forecasting period for all forecasting models models includes the 33 most recent years of data, ranging from January 1990 until December 2022. Consequently, the initial in-sample period, which is used to estimate the parameters in the forecasting models, spans from January 1950 until December 1989. To avoid the look-ahead bias, the data considered within the in-sample period are limited to the available information up to time step t, in order to forecast the equity premium at t+1. All forecasting models employ an expanding estimation window, which means that additional observations are incorporated and added to the in-sample period as they become available. Thereafter, the model parameters are re-estimated, using the updated in-sample period, to forecast the subsequent value in the OOS period. Although a rolling window seems more capable of adapting to parameter changes over time, it comes with the drawback of a shorter estimation sample and consequently less precise parameter estimates.

Neural network models typically require a large in-sample data-set for parameter estimation, and in the context of OOS return prediction it is often observed that an expanding window performs better in practical applications (Rapach and Zhou, 2013; Gu et al., 2020).

3 Methodology

To evaluate the performance of the LSTM architecture in forecasting the equity premium, and to assess the added value of the model's complexity, its performance is compared with three benchmark models. In essence, the methodology section presents the four employed forecasting models and their optimisation processes, and then shifts to describing the three distinct methods to analyse the forecasts.

The discussion on the forecasting models begins with the FNN model, as understanding the FNN architecture is crucial for a better comprehension of the more advanced LSTM network. This is followed by an overview of the traditional RNN model, leading to a subsection dedicated to the parameter estimation process employed to optimise the neural network models. This subsection also explores the problems of the traditional RNN model, leading to the introduction of the LSTM model as a solution. The discussion on the neural network models concludes with their tuning and forecast specifications. Subsequently, the combination forecast and historical average forecast are explained.

The focus then moves to the methods employed to evaluate the forecasts. The first method is based on statistical accuracy, a standard approach to assess forecasting performance. However, since equity return predictions may require a different perspective, a second evaluation method based on potential economic value is incorporated to provide a comprehensive evaluation. The section ends with a method to analyse the contribution of the selected input features to the forecasts.

3.1 Feedforward neural network

The fundamental concept of a neural network is to derive a linear combination of the input features, and subsequently model the target variable as a non-linear function of these linear combined signals. The FNN model holds significant importance for machine learning practitioners. The model forms the basis of many applications, and serves as a conceptual stepping stone in the progression towards RNN architectures, including the LSTM (Goodfellow et al., 2016; Hastie et al., 2009). Therefore, employing the FNN in the analysis establishes a reference point for contrasting the characteristics of the LSTM layer, and provides insights into the potential value of the LSTM network's complexity.

An FNN processes the input information in x_t, through $q \geq 1$ layers $z^{(1)}, z^{(2)}, ..., z^{(q)}$, before reaching the output layer to predict the ultimate outcome $\hat{r}_{t+1}^{FNN}$. The input includes all predictive features $x_t = (x_{1,t}, x_{2,t}, ..., x_{K,t})$ at time step t, and is passed forward in a directed, acyclic path through the layers. Each layer $z^{(m)}$ contains neurons which transform the input features by multiplying them with a set of weights and adding a bias term. Hence, each neuron in the first layer takes all K input features and linearly transforms the input in x_t following equation (1):

$$a_i(x_t; \theta_t) = \beta_{i0}^0 + \sum_{j=1}^{K} \beta_{ij}^0 x_{j,t} \quad \text{for} \quad i = 1, ..., n. \tag{1}$$

Thereafter, an activation function ϕ, is element-wise applied to the transformed input $a_i(x_t; \theta_t)$. The activation function introduces non-linearity, allowing the network to learn more complex patterns. The activated output values from each neuron i, in layer m, denoted as $z_{i,t}^{(m)}$, are passed through as input values for the neurons in the next layer $z^{(m+1)}$. So for a deep neural network, the input for each neuron in layers $m > 1$, equals the activated output from the neurons in the previous layer $z_t^{(m-1)} = (z_{1,t}^{(m-1)}, z_{2,t}^{(m-1)}, ..., z_{n,t}^{(m-1)})$, instead of the input x_t in equation (1). Finally, the output layer aggregates the weighted signals into the prediction of the target variable. For example, a single layer FNN with n neurons aggregates the activated outputs of all neurons and forms a forecast according to equation (2):

$$\hat{r}_{t+1}^{FNN} = \beta_0^1 + \sum_{i=1}^{n} \beta_i^1 \phi(a_i(x_t; \theta_t)). \tag{2}$$

3.2 Recurrent neural network

RNNs have been introduced to process sequential time series data in a temporal causal way. The network architecture, similarly to the FNN, consists of several interconnected layers, where the first layer receives the input data, which are then processed through the network's layers, until the data reaches the output layer. However, this network has a cyclical way of handling the input data. The network incorporates feedback loops such that the model can retain previous information to predict the subsequent output. To visualise the difference, Figure 1 displays a FNN and a RNN layer. Consider again all input features at time step t as $x_t = (x_{1,t}, x_{2,t} ..., x_{K,t})$. RNNs differ from traditional FNNs by not only using the information available at time step t incorporated in x_t, to predict the target variable $\hat{r}_{t+1}^{RNN}$, but by leveraging the entire time-series $x_{1:t}$. The model accomplishes this by combining the individual component x_t of the series $x_{1:t}$, and the output $z_{t-1}^{(m)}$ from the previous time step as layer inputs for the current step. Where the variable $z_{t-1}^{(m)} = (z_{1,t-1}^{(m)}, z_{2,t-1}^{(m)}, ..., z_{n,t-1}^{(m)})$, contains the activated output from the neurons of the same layer during the previous time step. Therefore, carries information from past variables $x_{1:t-1}$ into the current prediction. This way, the model captures the temporal sequence and retains a memory of past observations.

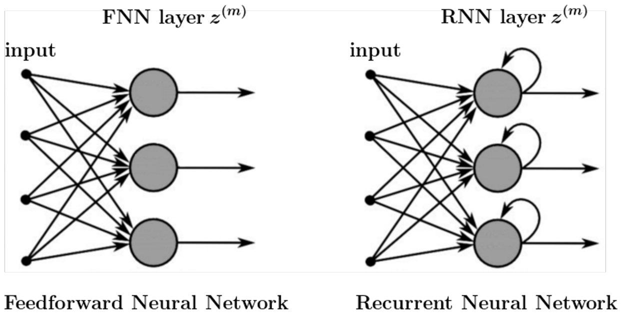


Figure 1: A FNN versus a RNN

Note: On the left, the structure of a FNN model, which takes the input (x_t for $m = 1$, and $z_t^{(m-1)}$ for $m > 1$), and passes it through layer $z^{(m)}$ in a direct path. The structure on the right represents a RNN model. This type of network takes the augmented input ($(x_t, z_{t-1}^{(1)})$ for $m = 1$, and $(z_t^{(m-1)}, z_{t-1}^{(m)})$ for $m > 1$), by incorporating feedback loops.

In case of a single layer RNN, the layer $z^{(1)}$ structurally resembles a standard FNN layer, except that the input for the FNN layer, denoted as x_t, is augmented to $(x_t, z_{t-1}^{(1)})$ to handle the time-series data effectively. Each neuron in the RNN layer takes the augmented input and linearly transforms the input following equation (3), where the upper index $^{(1)}$ indicates the fact that this is the first (and single) RNN layer in this example:

$$a_i(x_t, z_{t-1}^{(1)}; \theta_t) = \beta_{i0}^0 + \sum_{j=1}^{K} \beta_{ij}^0 x_{j,t} + \sum_{j=1}^{n} w_{ij} z_{j,t-1}^{(1)} \quad \text{for} \quad i = 1, ..., n. \tag{3}$$

Thereafter, an activation function ϕ is applied to the transformed input, and finally the activated output of all n neurons are aggregated according to equation (4):

$$\hat{r}_{t+1}^{RNN} = \beta_0^1 + \sum_{i=1}^{n} \beta_i^1 \phi(a_i(x_t, z_{t-1}^{(1)}; \theta_t)). \tag{4}$$

Similarly as with the FNN model, when extending this shallow network to a multi-layer architecture, the fundamental structure of the layers remains the same only the input changes. For deeper RNN layers, denoted as $z^{(m)}$ where $m > 1$, the input for each neuron $(x_t, z_{t-1}^{(1)})$ in equations (3), and (4) converts to $(z_t^{(m-1)}, z_{t-1}^{(m)})$.

3.2.1 Parameter estimation

Training neural network models to obtain parameter estimates for $\theta_t = (\beta^0, \beta^1)$ in an FNN, and $\theta_t = (\beta^0, w, \beta^1)$ in an RNN, is carried out by optimising a loss function, using the data in the in-sample period. In this study the MSFE (displayed in Section 3.6) is selected as loss function for the neural network models. This choice is primarily because employing the MSFE aligns the objective of the neural network models with that of the other OLS regression benchmark models, which aim to minimise the sum of the squared forecast errors. As mentioned in Section 1, finding the parameter values can be hard for traditional RNNs. This optimisation process operates in several steps, as presented in Taddy (2018), starting with the initialisation of the weight and bias terms with random values. A forward pass is then executed, which involves feeding the input through the network layers to generate an output for the target variable. Subsequently, the value of the loss function is calculated by comparing this output with the true target value. In the third step, the backpropagation algorithm is employed to compute the gradient of the loss function with respect to the parameters in θ_t. It is noteworthy that the computation of the gradient also follows a layered structure, proceeding in reverse order from the output back to the input. The chain rule of calculus is employed at each step to calculate the gradient. For example, the partial derivative of the loss function with respect to weight w_{ij} in equation (3), of neuron i in layer $z^{(m)}$, is given by equation (5):

$$\frac{\partial L_t}{\partial w_{ij}} = \frac{\partial L_t}{\partial z_{i,t}^{(m)}} \frac{\partial z_{i,t}^{(m)}}{\partial w_{ij}}, \tag{5}$$

where $z_{i,t}^{(m)}$ represent the activated output values from neuron i in layer $z^{(m)}$, and L_t the value of the loss function after processing the input at time step t. Another application of the chain rule can be used to expand $\frac{\partial L_t}{\partial z_{i,t}^{(m)}}$ as $\frac{\partial L_t}{\partial z_{i,t}^{(m+1)}} \frac{\partial z_{i,t}^{(m+1)}}{\partial z_{i,t}^{(m)}}$, and so on until the full gradient is written as a product of layer-specific operations. Parameter updates based on the gradient are then performed, utilising the Adam algorithm introduced by Kingma and Ba (2014), as defined in equation (6). Updates are made using the gradient from a single observation such that $N = 1$, according to the stochastic gradient descent algorithm (SGD). The mini-batch SGD algorithm is also considered, which differs in that it updates model parameters based on a batch of gradients with $N > 1$ instead of just one. However, the performance of the mini-batch SGD algorithm did not surpass that of the traditional SGD, which is the preferred method due to its faster computational speed.

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \eta_t \frac{1}{N} \sum_{i=1}^{N} \nabla L_t(\hat{\theta}_t), \tag{6}$$

$\nabla L_t(\hat{\theta}_t)$ is the gradient of the loss function evaluated at the current estimates $\hat{\theta}_t$, while processing input at time step t. The learning rate η_t determines the magnitude of the change in the parameter estimates. If η_t is set too high, overshooting the optimal solution is a risk, potentially causing the parameter estimates to diverge rather then converge. Conversely, an excessively low value for η_t can lead to a slow convergence, requiring numerous iterations and increasing the chance of becoming stuck in a local minimum. To mitigate these risks, the Adam algorithm computes individual adaptive learning rates for different parameters, based on the historical behaviour of the gradients.

During training, traditional RNNs face challenges caused by unstable gradients. In deeper networks, the model fails to accurately estimate the weight parameters tied to the output from the previous time step in early layers, represented as w_{ij} in equation (3). This hinders the model's ability to learn temporal dependencies in the data for which they were originally designed. The gradient of these weight parameters depends multiplicative on the partial derivatives of subsequent layers due to the chain rule of calculus, as outlined in equation (5). If some of these partial derivatives are less than one, the gradient diminishes exponentially. As a result, the parameter update, following equation (6), becomes negligible, preventing the parameter from reaching its optimal value. This is called the vanishing gradient problem, which slows down the training process significantly or may cause the network to fail in learning complex patterns and hence to produce inaccurate predictions. On the other hand, the exploding gradient problem, which might occur if some of the earlier gradients are greater than one, causes the weights in the network to become very large. As a result, the training becomes unstable, and difficult to converge Hochreiter and Schmidhuber (1997).

3.2.2 Long short-term memory neural network

The LSTM network is an advanced type of RNN, designed to address the challenges encountered when training traditional RNNs. The main characteristic of an LSTM layer is that it incorporates a so called cell state, resulting in a more complex inner structure compared to a standard RNN layer. The cell state acts as a conduit that stores information for later use, therefore serves as a memory mechanism. Figure 2 provides a visual representation of an LSTM layer $z_{t-1}^{(m)}$.

The cell state is regulated by three gates, each with their own set of parameters that are

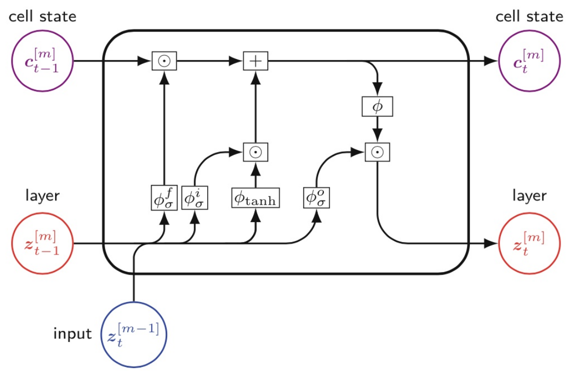


Figure 2: Diagram of an LSTM layer

Note: LSTM layer $z^{(m)}$, with its input $(z_t^{(m-1)}, z_{t-1}^{(m)})$. The forget gate decides what information from the previous cell state $c_{t-1}^{(m)}$ should be forgotten or retained, using ϕ_σ^f. The input gate decides what new information from the input should be added to the cell state $c_t^{(m)}$, using ϕ_σ^i and ϕ_{tanh}. The gates together regulate the update $(+)$ of the cell state. The output gate decides the new internal state $z_t^{(m)}$, using ϕ_σ^o, and ϕ. The Hadamard product $(\odot)$ is used to combine information.

learned during the training process. The gates linearly transform the input, thereafter use a non-linear activation function to create their activated output. Even in the presence of noisy data, the gating mechanisms control the flow of information into, within, and out of the LSTM layer. Therefore, the model can preserve the important information while disregarding the noise (Wüthrich and Merz, 2023; Hochreiter and Schmidhuber, 1997).

Hence, the input information flows through the layers, where each neuron within the LSTM layers processes the input information through the gates. After processing through all LSTM layers, the output layer aggregates the activated outputs of all neurons analogous to the traditional RNN model, following equation (4). Similar as for a standard RNN, in deeper LSTM layers ($z^{(m)}$ where $m > 1$) the input $(x_t, z_{t-1}^{(1)})$ converts to $(z_t^{(m-1)}, z_{t-1}^{(m)})$. For example, each neuron in the first layer ($z^{(1)}$) of an LSTM network processes the input information through its gates in the following step wise manner:

1. The forget gate decides what information from the previous cell state $c_{t-1}^{(m)}$ should be forgotten or retained. The gate activates the transformed input, including the input value of the current time step x_t and the output value of the previous time step $z_{t-1}^{(1)}$, using the sigmoid activation function ϕ_σ^f, following equation (7):

$$f_t = \phi_\sigma^f(w_f[x_t, z_{t-1}^{(1)}] + \beta_f), \tag{7}$$

   where the value of f_t lies between 0 (completely forget) and 1 (completely remember), w_f, and β_f respectively are the weight and bias terms of the forget gate.

2. The input gate decides what new information from the input should be added to the cell

state $c_t^{(m)}$. The sigmoid activation function ϕ_σ^i is used to activate the transformed input. Simultaneously, the hyperbolic tangent activation function ϕ_{tanh}, is used to decide how much new input information could potentially be added to the cell state $\hat{c}_t^{(m)}$. This is respectively shown in equations (8), and (9):

$$i_t = \phi_\sigma^i(w_i[x_t, z_{t-1}^{(1)}] + \beta_i), \tag{8}$$

$$\hat{c}_t^{(m)} = \phi_{tanh}(w_c[x_t, z_{t-1}^{(1)}] + \beta_c), \tag{9}$$

where the value of i_t lies between 0 and 1, and $\hat{c}_t^{(m)}$ between -1 and 1. This ensures that the updated cell state values are bounded, which helps prevent the cell state from growing too fast or diminishing too quickly. w_i, β_i, and w_c, β_i respectively are the sets of weight and bias terms of the input gate and the cell sate.

3. The current cell state is updated by combining the output of the forget gate and the input gate using the Hadamard product $\odot$, following equation (10):

$$c_t^{(m)} = f_t \odot c_{t-1}^{(m)} + i_t \odot \hat{c}_t^{(m)} \tag{10}$$

4. The output gate decides what new information should be outputted. The sigmoid activation function ϕ_σ^o is used to activate the transformed input information, following equation (11):

$$o_t = \phi_\sigma^o(w_o[x_t, z_{t-1}^{(1)}] + \beta_o), \tag{11}$$

where the value of o_t lies between 0 and 1, w_o, and β_o respectively are the weight and bias terms of the output gate.

5. Finally, the activated output of neuron i in the (first) LSTM layer $z_{i,t}^{(1)}$ is obtained after combining the output of the output gate with the information from the updated cell state by using the Hadamard product $\odot$. A general activation function ϕ is applied to decide what information from the updated cell state should be outputted, according to equation (12):

$$z_t^{(1)} = o_t \odot \phi(c_t^{(m)}). \tag{12}$$

A similar optimisation process as described in Section 3.2.1 is used to estimate the parameters within the LSTM model. This includes employing the backpropagation algorithm to compute the gradient of the loss function with respect to all weight and bias terms within the LSTM layer. The computation of these gradients proceeds in the reverse order of the above specified steps, working from the output backward to the input. This implies a flow of gradients through the gates, starting with the gradient of the loss function concerning the output in $z_t^{(1)}$. Subsequently, these gradients are used to compute the gradients with respect to the parameters within the output gate (as shown in equation (11)) and the cell state update (as shown in equation (10)). The gradients with respect to the cell state are then propagated further backward to calculate the gradients with respect to the parameters within the forget gate and the input gate (as shown in equations (7), (8), and (9)).

During the forward pass, input information flows through both the input and the forget gate. If, for instance, one of these gates determines that certain information is not relevant, it diminishes the contribution of that information to its output. As a result, the gradient associated with that specific information is reduced during backpropagation. However, since the updates to the cell state are additive, using information from both the input and forget gate, this does not necessarily lead to a reduced gradient concerning the cell state. The gating mechanisms deliberately regulate the flow of information and corresponding gradients, ensuring that relevant information is retained, and gradients are preserved. As a result, this mechanism allows the gradient to propagate backward through the model, learn temporal dependencies in the data, without vanishing or exploding. Consequently, the model is able to capture long time lags, while maintaining stable.

3.3 Hyperparameter tuning and forecast specifications

Neural network models contain not only model weights and bias terms, hereafter referred to as model parameters, but they also have certain hyperparameters. The hyperparameters, such as the number of layers or neurons per layer, define the network's structure and complexity, and effectively tuning them is essential for enhancing the model's predictive accuracy. The hyperparameter values for both the FNN and LSTM model are tuned separately following a commonly employed strategy in the academic literature, as outlined by Gu et al. (2020).

Accordingly, the in-sample period is divided into a training and validation set, that maintain the temporal ordering of the data. The first part of the data is considered the training set, which is used to optimise the weights and bias terms given a specific set of hyperparameter values, by minimising the MSFE loss function. This minimisation process is performed equally for both neural network models by employing the SGD method, incorporating backpropagation and the Adam algorithm, as described in Section 3.2.1. The resulting model parameter estimates, in conjunction with the specific set of hyperparameter values, are then used to forecast the target value within the validation set.

A distinct grid search is conducted for both neural network models, to systematically explore all possible combinations of hyperparameter values. Hence, for each combination of hyperparameter values, the MSFE loss function is minimised, and the resulting model parameter estimates are used to forecast the target value in the validation set. The optimal model structure is determined by selecting the combination of hyperparameter values that yields the lowest forecast error on the validation set. As the validation set is used in the optimisation process, it serves as a proxy for evaluating the model's performance on unseen data, but it is not truly OOS. The remaining part of the data-set, including the most recent 33 years of data, are considered the test set. The test set is used to evaluate the resulting model's true OOS predictive performance. The hyperparameter tuning is conducted independently for both the FNN and LSTM models using the initial in-sample period. The training set spans from January 1950 to December 1979, while the validation set includes the ten years prior to the OOS period, ranging from January 1980 to December 1989.

The hyperparameter tuning process to select the optimal network structure can be a challenging and time consuming task. It is unrealistic and unnecessary to search through an infinite number of hyperparameter values. Instead, the search space for each hyperparameter is limited to ranges that have shown promising results in the literature according to Gu et al. (2020). A grid search over the possible values for each hyperparameter is performed separately for the LSTM and FNN models. The search space and resulting optimal values are shown in Table 2 for the LSTM model, and in Table 8, Appendix A, for the FNN. The hyperparameters selected for tuning are the following Wüthrich and Merz (2023):

- **Number of hidden layers**: This hyperparameter controls the depth of the model architecture by specifying the number of layers stacked on top of each other. Increasing the depth of the model by adding more hidden layers can potentially improve its accuracy. However, deeper models also have a higher risk of overfitting, as they become overly specialised on the training data. On a relatively small data-set, shallow networks, with only a few layers, have proven to perform well. Therefore, the search space for the number of layers is constrained to a maximum of five.

- **Neurons per layer**: The number of neurons within each hidden layer impacts the model's expressive power. The neurons process the input information and each neuron provides a different representation of the data. By combining the outputs of multiple neurons, the layer can extract crucial feature information from the input data.

- **Activation function**: The activation function introduces non-linearity in a neural network model, allowing it to learn complex patterns. The activation function, denoted as ϕ, determines whether the output of a neuron should be activated or not. The gating mechanism in the LSTM layer contains multiple activation functions, of which only the general activation function in equation (12) requires tuning. The other activation functions, specifically the sigmoid and hyperbolic tangent, remain unchanged due to their design purposes, following the experiments in Hochreiter and Schmidhuber (1997). A grid search considering popular functions is performed, including the sigmoid, hyperbolic tangent, rectified linear unit (ReLU), and scaled exponential linear unit (SELU) activation functions.

Table 1: Search space of non-linear activation functions

Activation function		**Output range**
Sigmoid activation	$\phi(x) = (1 + e^{-x})^{-1}$	[0, 1]
Hyperbolic tangent activation	$\phi(x) = \dfrac{(e^x - e^{-x})}{(e^x + e^{-x})}$	[-1, 1]
ReLU activation	$\phi(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$	[0, ∞)
SELU activation	$\phi(x) = \begin{cases} \lambda x & \text{if } x \geq 0 \\ \lambda\alpha(e^x - 1) & \text{if } x < 0 \end{cases}$	(-∞, ∞)

Note: The search space of activation functions and their output range.

- **L₁-norm and L₂-norm penalty terms**: The LASSO and ridge penalty terms are incorporated into the loss function as regularisation techniques, which effectively mitigate the risk of overfitting. The penalty terms shrink the weights associated with the connections between the neurons within each layer. The key distinction between LASSO and ridge lies in their impact on the weights. In case of LASSO, or L_1-norm regularisation, some of the weights are set exactly to zero by inducing a penalty term based on the absolute value of the weights. This promotes a more sparse layer structure, as the number of active neurons is reduced. Ridge, or L_2-norm regularisation typically only shrinks the size of the weights by a penalty term proportional to the squared magnitude of the weights. L_2-norm regularisation results in a more distributed representation of the input within the layer, to control the overall scale of the weights, making them less susceptible to extreme values. The search space for strengths of the L_1-norm, and L_2-norm penalty terms includes: None, $10^{-1}, 10^{-2}, ..., 10^{-6}$.

The optimal combination of hyperparameter values for the LSTM model is given in Table

Table 2: Hyperparameter search space and optimal values for the LSTM model

Hyperparameter	**Search space**	**Optimal**
Number of LSTM layers	$1, 2, 3, 4, 5$	2
Neurons per LSTM layer	$8, 16, 32, 64$	$32, 16$
Activation function LSTM layers	Sigmoid, tanh, SELU, ReLU	SELU
Activation function output layer	Sigmoid, tanh, SELU, ReLU	SELU
L_1-norm penalty	None, $10^{-1}, 10^{-2}, ..., 10^{-6}$	10^{-5}
L_2-norm penalty	None, $10^{-1}, 10^{-2}, ..., 10^{-6}$	10^{-4}

Note: The search space for each hyperparameter is explored with a grid search over all possible combinations. The hyperparameters are tuned once using the initial training and validation sets, spanning from 1950:01 to 1979:12, and 1980:01 to 1989:12, respectively. The optimal values are determined based on the model's performance on the validation set, using the MSFE as the loss function. The optimal number of neurons per LSTM layer is specified in their respective order. The first LSTM layer contains 32 neurons, while the second layer consists of 16 neurons. Each LSTM layer simultaneously uses multiple activation functions for different purposes. Among these, only the general activation function required tuning and was found to be identical across all layers.

2. The resulting model contains two LSTM layers, with thereafter an output layer. The first LSTM layer contains 32 neurons, while the second LSTM layer incorporates 16 neurons. The output layer helps reduce the dimension of the output to match the one-dimensional target variable, therefore contains 1 neuron. Each LSTM layer simultaneously uses multiple activation functions for different purposes as described in Section 3.2.2. Among these, only the general activation function used within the LSTM layer to decide what information to be outputted, requires fine-tuning. The SELU activation function has been identified as the most effective, yielding optimal performance across both LSTM layers. This function also proved to be the optimal activation function for the output layer. The optimal L_1-norm, and L_2-norm penalty terms are 10^{-5}, and 10^{-4}, respectively, and are incorporated into both LSTM layers.

A distinct grid search over the hyperparameter values for the FNN resulted in a deeper network with three FNN layers, each having 8 neurons. The output layer, consists of 1 single neuron, with the SELU activation function resulting in the best performance. The ReLU activation function was found to be the most effective within the FNN layers, consistently across all layers. The optimal L_1-norm, and L_2-norm penalty terms were determined to be 10^{-5}, and 10^{-4}, respectively. The optimal hyperparameter values for the FNN model are shown in Table 8, Appendix A.

After tuning the hyperparameters, the optimal model configurations are used to forecast the first year in the OOS test set, covering the period from January until December 1990. Subsequently, the training set is updated using an expanding window approach, incorporating an additional year of available data. While maintaining a constant size for the validation set, it is progressively updated to include the most recent 10 years of data prior to the test set.

Due to the computational intensity of retraining the neural network models, these updates do not occur monthly but once every year. While the hyperparameter values remain constant, the models use the updated training and validation set for retraining, to update the weight and bias parameter estimates. The step wise process of updating the model parameters including the forward pass, backward pass, and subsequently the parameter update, is repeated for every observation in the training set. Each iteration trough the entire training set is called an epoch. As the models iterate through additional epochs during retraining to update the parameter estimates each subsequent year, they are susceptible to overfitting. To mitigate the risk of overfitting an early stopping mechanism is employed that ceases the model to iterate through additional epochs after observing an increased loss in the validation set for two consecutive epochs Maier and Dandy (1998). Once the model stops running though additional epochs, new parameter estimates are obtained, and employed to forecast the next year of the OOS test set. Even with early stopping, as the models continue to iterate through additional epochs during subsequent annual retraining, the risk of overfitting persist. Therefore, every five years, the model parameter estimates are reset. Both weight and bias terms are initialised with random values, re-initiating the optimisation process, to form the basis for the forecast of the next five years. This iterative process continues across the OOS forecast horizon, which extends from January 1990 to December 2022.

Before entering the neural network models, all features are standardised to have a mean of zero and a standard deviation of one. This standardisation procedure is performed each time the model undergoes retraining, and the training and validation sets are updated. It ensures all input features are treated equally in the regularisation process, and allows for a relevant range for the random starting parameters in the gradient descent algorithm. This enhances the algorithm's efficiency in exploiting relevant directions (Wüthrich and Merz, 2023; Hastie et al., 2009).

3.4 Combination forecast

The combination forecast is introduced by Rapach et al. (2010), following the controversial findings on return predictability by Welch and Goyal (2008). The researchers demonstrates that, in contrast to traditional univariate and multivariate regression forecasts, the combination forecast model achieves improved OOS accuracy gains relative to the historical average forecast, considerably more consistent over time. By leveraging the (weak) correlation among individual forecasts, a simple average offers reduced volatility, and more reliable tracks movements in the equity premium. The model effectively replaces the coefficient estimates of the multivariate OLS regression model with their respective univariate counterparts, therefore incorporates information from multiple predictive features in a linear way. This approach successfully addresses the problem of multicollinearity, which can lead to imprecise parameter estimates. Moreover, by taking the average, the model offers an effective shrinkage strategy while avoiding overfitting. In this research, the model is employed to evaluate the abilities of the selected input features in forecasting the equity premium. In addition, the model serves as a benchmark to emphasise the capabilities and advancements offered by the more sophisticated, non-linear neural network architectures. The predictive regression model for each individual feature is given in equation (13):

$$r^i_{t+1} = \alpha_i + \beta_i x_{i,t} + \epsilon_{t+1}, \quad \text{for} \quad i = 1, ..., K, \tag{13}$$

where r^i_{t+1} is the equity premium, $x_{i,t}$ denotes one of the K features at time step t, and ϵ_{t+1} is a disturbance term. The equity premium OOS forecast based on each feature is then computed following equation (14):

$$\hat{r}^i_{t+1} = \hat{\alpha}_i + \hat{\beta}_i x_{i,t}, \quad \text{for} \quad i = 1, ..., K. \tag{14}$$

In which $\hat{\alpha}_i$ and $\hat{\beta}_i$ are the OLS regression estimates for their respective counterparts in equation (13), obtained using the in-sample period. A monthly expanding estimation window is used to update the in-sample period and re-estimate the model parameters, in order to forecast the subsequent monthly equity premium. The individual forecasts $\hat{r}^i_{t+1}$, are combined to generate the combination forecast, denoted as $\hat{r}^{cf}_{t+1}$, following equation (15):

$$\hat{r}^{cf}_{t+1} = (1/K)\sum_{i=1}^{K} \hat{r}^i_{t+1}. \tag{15}$$

3.5 Historical average forecast

The final benchmark model is the historical average forecast, which serves as a baseline in the evaluation. The forecast helps to asses the effectiveness of the selected input features, and the added value of the more sophisticated forecasting models. The forecast is constructed following equation (16):

$$\hat{r}^{ha}_{t+1} = (1/t)\sum_{s=1}^{t} r_t, \tag{16}$$

where $\hat{r}^{ha}_{t+1}$ is the equity premium prediction, and r_t the historical excess return at time step t. The OOS forecast is straightforwardly determined by computing the prevailing mean. Hence, the model assumes a relatively constant expected equity premium, and implies that the information in the features, denoted as x_t, is not useful for the predictions. A monthly expanding window is used to predict the OOS equity premium, where all historical excess return observations available up to time step t are used to generate the return forecast for month t+1.

3.6 Statistical accuracy evaluation

The accuracy of the forecasting models is assessed using the MSFE as a performance measure. The measure can be used to compute the OOS R^2, which is a useful statistic to compare statistical accuracy of distinct forecasting models. However, while the MSFE is a useful tool for comparison and model selection, its interpretation can be challenging due to the squared units of measurement. To bring the error metric back into the same units as the original data, the root mean squared forecast error (RMSFE) is computed following equation (17). The RMSFE is simply the square root of the MSFE, offering a more direct reflection of the standard deviation of the prediction errors.

$$\text{RMSFE} = \sqrt{\frac{1}{T}\sum_{s=t+1}^{T}(\hat{r}_{t+1} - r_{t+1})^2}, \tag{17}$$

where $\hat{r}_{t+1}$ represents the return forecast, r_{t+1} the actual observed return at time step t+1, and T corresponds to the total number of OOS forecast observations. To compare the MSFEs for two competing return forecasts the R^2_{oos} statistic is computed using equation (18), in accordance with campbell2008predicting:

$$R^2_{oos} = 1 - \frac{\sum_{s=t+1}^{T}(\hat{r}^m_{t+1} - r_{t+1})^2}{\sum_{s=t+1}^{T}(\hat{r}^b_{t+1} - r_{t+1})^2}. \tag{18}$$

In which $\hat{r}^m_{t+1}$ with $m =$ cf, FNN, LSTM, denotes the return of the model of interest, which is compared to $\hat{r}^b_{t+1}$ with $b =$ ha, cf, FNN, and $m \neq b$, the return of a specific benchmark model. A positive R^2_{oos} signifies that the selected model yields a lower MSFE, demonstrating the relative reduction in the forecast error compared to the benchmark model. In order to evaluate the significance of differences in the forecast accuracy across the competing models, the Clark and West, or adjusted Diebold and Mariano (adj-DM) test, is employed. When comparing forecasts from nested models, the original DM test may yield unreliable results, resulting in less power to detect return predictability. The adj-DM test uses an adjusted MSFE statistic, which produces asymptotically valid results. The test corresponds to evaluating the null hypothesis of equal forecast accuracy, against the one sided alternative hypothesis of improved performance. The adj-DM test defines the loss differential as in equation (19):

$$\text{adj-DM}_{loss} = (r_{t+1} - \hat{r}^{b}_{t+1})^2 - [(r_{t+1} - \hat{r}^{m}_{t+1})^2 - (\hat{r}^{b}_{t+1} - \hat{r}^{m}_{t+1})^2]. \tag{19}$$

The adj-DM test performs a regression of the loss differential on a constant term. By calculating the corresponding t-statistic for the constant and obtaining the p-value using the standard normal distribution, a one-sided test is conducted (Diebold and Mariano, 1995; Clark and West, 2007).

3.7 Measuring economic value

Evaluating the accuracy of forecasting models often involves statistical metrics such as the MSFE, and R^2. Despite their wide application and valuable insights, Leitch and Tanner (1991) argue that profitability serves as a more suitable metric to evaluate stock return forecasts. Their findings show a weak relationship between statistical measures and economic gains, highlighting the distinction between statistical and financial performance. In line with these findings, Ican et al. (2017) observes an inconsistency between MSFE and the potential profits derived from using neural networks. It became evident that achieving a low MSFE does not automatically translate into generating high returns. Put differently, the correlation between statistical improvements and economic gains is not always linear or straightforward. These findings emphasise the importance of using additional measures beyond the conventional MSFE when assessing a model in forecasting the equity premium. Consequently, this study uses Sharpe ratios and a utility based method to asses the economic value of the forecasts. The utility based method measures potential utility gain to a mean-variance investor, following Campbell and Thompson (2008). The utility gain represents the additional utility achieved when using the forecasting model of interest, instead of relying on a benchmark model to make investment decisions. A positive utility gain suggests that the model of interest enhances the investor's investment strategy. In a mean-variance framework, the utility gain can be interpreted as the percentage portfolio management fee that an investor would be willing to pay to have access to the forecast of the model of interest instead of the benchmark model's forecast.

Consider a mean-variance investor who allocates investments between a risky equities portfolio and a risk-free bill. Her optimal portfolio weights are based on the trade-off between expected return and risk, measured as the variance. The investor's objective function is given by equation

(20):

$$\arg\max_{w}(w_{m,t+1}\hat{r}^{m}_{t+1} - 0.5\gamma w^{2}_{m,t+1}\hat{\sigma}^{2}_{t+1}), \tag{20}$$

where $\hat{r}^{m}_{t+1}$ denotes the return forecast after using forecast model $m =$ ha, cf, FNN, or LSTM, $\hat{\sigma}^{2}_{t+1}$ the rolling window estimate for the variance of equity returns[2], and λ the risk aversion coefficient. At the end of period t, the investor determines the risky equity portfolio weights for time period t+1, as the solution to equation (20), following equation (21):

$$\hat{w}_{m,t+1} = (\frac{1}{\lambda})(\frac{\hat{r}^{m}_{t+1}}{\hat{\sigma}^{2}_{t+1}}). \tag{21}$$

In accordance with Campbell and Thompson (2008), the weights are limited to lie between 0.0 and 1.5, ensuring realistic portfolio boundaries by disallowing short sales and limiting leverage to no more than 50%. The share 1 - $\hat{w}_{m,t+1}$ is allocated to the risk-free bill, and the investor's portfolio return at time t+1 is given by equation (22):

$$\hat{R}_{m,t+1} = \hat{w}_{m,t}r_{t+1} + R_{f,t+1}, \tag{22}$$

where $R_{f,t+1}$ is the risk-free return, and $\hat{R}_{m,t+1}$ is the return on the investor's portfolio when utilising the equity return forecast of model m in her trading strategy. The expected average utility to a mean-variance investor can be expressed following equation (23):

$$\overline{U}(\hat{R}_{m}) = \hat{\mu}_{m} - 0.5\lambda\hat{\sigma}^{2}_{m}, \tag{23}$$

where $\hat{\mu}_{m}$ and $\hat{\sigma}^{2}_{m}$, respectively are the mean and variance of the investor's portfolio return over the evaluated forecast horizon. The average expected utility gain is calculated as the difference between the utility achieved when depending on the return forecast from the forecasting model of interest $\overline{U}(\hat{R}_{m})$, compared to that of a benchmark model $\overline{U}(\hat{R}_{b})$, with $b =$ ha, cf, FNN, and $b \neq m$, following equation (24):

$$\hat{U}\text{ gain} = \overline{U}(\hat{R}_{m}) - \overline{U}(\hat{R}_{b}), \tag{24}$$

In a mean-variance framework the utility gain is equal to the excess return an investor requires to be indifferent between to trading strategies. Put differently, it represents the fee an

[2]Following Campbell and Thompson (2008), under the assumption of constant return volatility, the variance $\hat{\sigma}^{2}_{t+1}$ is estimated using the sample variance computed from a five-year rolling window of historical returns.

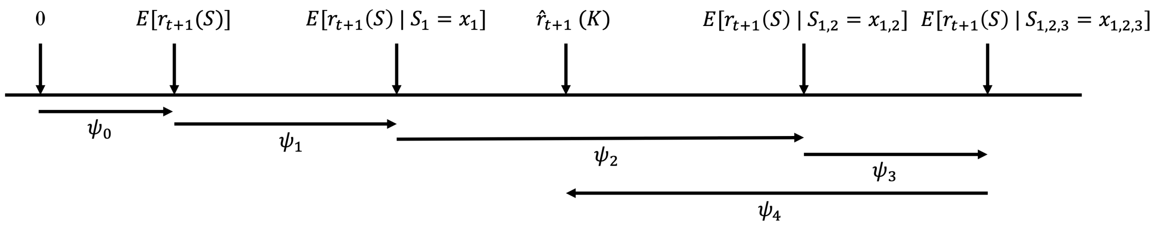
investor is willing to pay to switch from employing one forecasting model to another. Therefore, multiplying the monthly utility gains by 1200 provides a meaningful interpretation as an annual percentage portfolio management fee. The proof for simplifying the utility gain to the fee is given in Appendix C.2. The significance of the portfolio management fee is evaluated using a test of equal economic value proposed by McCracken and Valente (2018). The researchers demonstrate that for two nested forecasting models, the distribution of the portfolio management fee is asymptotically normal and allows for statistical inference. The proposed test corresponds to the null hypothesis of a fee less than or equal to zero, against the alternative hypothesis of a positive portfolio management fee. The p-values are obtained by using standard normal critical values for inference on the studentised value of the estimated portfolio management fee.

3.8 Feature contribution analysis

The rising popularity of powerful, yet highly complex, models presents a nuanced trade-off between a model's accuracy and the interpretability of its outputs. To address this challenge and gain insights into the decision-making processes of a neural network model, the SHAP method is incorporated into the analysis. Proposed by Lundberg and Lee (2017), the SHAP method unifies various existing feature contribution methods, offering a consistent approach to interpret complex model predictions. This enhanced understanding adds transparency and enriches the evaluation, ensuring that the employed neural network models aren't simply 'black boxes' converting features into forecasts.

The SHAP method assigns an importance value, or SHAP value ψ_i, to each feature i. The method originates from cooperative game theory where similar values are used to fairly distribute the total value created by a coalition of players among the players themselves. In machine learning SHAP values are used to assess the contribution of individual features to a prediction. SHAP values quantify the change in the expected model output attributable to each feature when conditioning on that feature. Figure 3 visually depicts how individual SHAP values facilitate the transition from the base value $E[r_{t+1}(S)]$, which is the prediction in the absence of any feature knowledge, to the current model output $\hat{r}_{t+1}(K) = E[r_{t+1}(K)|K]$, which incorporates all predictive features.

To generate SHAP values, every possible subset of features is analysed to determine the marginal impact each feature has on the model's prediction when included in the subset. By averaging these marginal contributions across all possible combinations of features, SHAP values enable a comprehensive and interpretable evaluation of each feature's contribution to the model's predictions. This provides insights into the individual and interactive roles the features have in



0
$E[r_{t+1}(S)]$
$E[r_{t+1}(S) \mid S_1 = x_1]$
$\hat{r}_{t+1}(K)$
$E[r_{t+1}(S) \mid S_{1,2} = x_{1,2}]$
$E[r_{t+1}(S) \mid S_{1,2,3} = x_{1,2,3}]$
ψ_0
ψ_1
ψ_2
ψ_3
ψ_4

Figure 3: SHAP values

Note: SHAP values attribute to each feature the change in the expected model prediction when conditioning on that feature. They illustrate the transition from the base value $E[r_{t+1}(S)]$, which is the prediction in the absence of any feature knowledge, to the current model output $\hat{r}_{t+1}(K) = E[r_{t+1}(K)|K]$, which incorporates all predictive features. In which K represents the set of all predictive features, and S denotes a subset of features. In cases where the input features are not independent, the order in which the features are incorporated into the expectations matters. Consequently, SHAP values are computed by averaging the ψ_i values across all possible sequences.

the model's decision-making process. The SHAP value ψ_i for feature i is computed following equation (25):

$$\psi_i = \sum_{S \subseteq K \setminus \{i\}} \frac{|S|!(|K| - |S| - 1)!}{|K|!} \left(\hat{r}_{t+1}(S \cup \{i\}) - \hat{r}_{t+1}(S)\right), \tag{25}$$

where K represents the set of all predictive features, and S a subset of features excluding feature i, with $|K|$, and $|S|$ denoting the respective cardinalities of these sets. The term $\frac{|S|!(|K|-|S|-1)!}{|K|!}$ serves as a weight, considering that features can appear in different subsets and interact with other features in various ways. The term accounts for the number of ways subset S can be formed, represented as $|S|!$, and the number of sequences in which feature i can be incorporated into the subset, which equals $(|K| - |S| - 1)!$. The relevance of the order in which features are incorporated into the expectations becomes particularly evident when the input features are not independent. The variable $\hat{r}_{t+1}(S)$ is the value of the model's forecast employing feature subset S. Consequently, $(\hat{r}_{t+1}(S \cup \{i\}) - \hat{r}_{t+1}(S))$ illustrates the marginal contribution of feature i to subset S. In essence, the difference in the prediction with and without the feature in the subset. The SHAP value for feature i is determined by averaging its marginal contributions across all possible subsets of features.

Figure 3 illustrates an example of SHAP values attributed to individual features and demonstrates how this results in the desirable additive feature attribution property inherent to the SHAP method. This property implies that the sum of the SHAP values of all individual features, in excess of what would be expected without any feature knowledge, approximates the model's prediction, following equation (26):

$$\hat{r}_{t+1}(K) = \psi_0 + \sum_{i=1}^{K} \psi_i, \tag{26}$$

Through the additive decomposition of a model's output, interaction effects between features can be identified and quantified. This offers valuable insights into the collective feature effects on the model's predictions. Additionally, the additive feature attribution property guarantees that a feature's contribution to the model's predictions can be aggregated across the entire OOS forecast horizon. Consequently, the average individual feature contribution over the forecast period spanning from $t+1$ to T, can be calculated using equation (27):

$$\Psi_i = \sum_{s=t+1}^{T} \psi_i, \quad \text{for} \quad i = 1, ..., K, \tag{27}$$

4 Empirical results

The empirical results section presents and discusses the performance of the forecasting models, focusing on their statistical accuracy and economic value. The section concludes with a comprehensive analysis of the individual feature contributions to the predictions, and explores the aggregate importance of the distinct fundamental and technical feature groups.

4.1 Statistical accuracy of the forecasts

The evaluation starts with testing the performance of the LSTM neural network, FNN, and combination forecast against the historical average forecast. Table 3 presents and compares the forecasting results of all employed models measured over the entire OOS forecast horizon, spanning from January 1990 to December 2022. It outlines statistical performance in terms of the RMSFE, and R^2_{oos} statistic of the forecasts. Additionally, the Sharpe ratio and utility gain reveal economic value in a mean-variance framework, where both the R^2_{oos} statistic and utility gain are reported relative to the historical average forecast.

Table 3: Performance of the forecasting models relative to the historical average forecast

Model	**RMSFE**	**Sharpe ratio**	$\mathbf{R^2_{oos}}$ **(%)**	**Utility gain (%)**
LSTM	4.246	0.235	3.950***	3.315***
FNN	4.265	0.236	3.108***	3.394***
Combination forecast	4.329	0.178	0.166	0.790
Historical average	4.333	0.157	-	-

Note: The RMSFE is calculated over the entire OOS forecast horizon, ranging from 1990:01 to 2022:12 (values are reported $\times 100$). The mean and volatility of the expected portfolio returns obtained from the asset allocations are used to calculate Sharpe ratios. The R^2_{oos} measures the reduction in MSFE for the competing forecast given in the first column relative to the historical average forecast. The average utility gain is given in annualised percent return, indicating the portfolio management fee that an investor, with mean-variance preferences and risk aversion coefficient of five, would be willing to pay to have access to the competing forecasting model relative to the historical average forecast. Where *, **, and ***, indicate significance at the 10%, 5%, and 1% level, based on the p-values corresponding to the adj-DM test to assess significance of the R^2_{oos}, and to assess significance of the utility gain, or portfolio management fee.

The historical average forecast demonstrates the highest RMSFE with a value of 4.333, whereas the LSTM model achieves the lowest RMSFE of 4.246. The resulting LSTM model's R^2oos is 3.95%, indicating the percentage reduction in MSFE compared to the historical average forecast. Even though the R^2oos value seems small, it is common for equity premium forecasting models to have relatively small values due to the limited degree of predictability in stock returns. The adj-DM test concludes that the reduction is significantly greater than zero at the 1% level. Thus, the LSTM model significantly outperforms the historical average based on statistical accuracy. Furthermore, both the FNN and the combination forecast demonstrate enhancements in terms of RMSFE, with respective values of 4.265 and 4.329. Notably, among the two, only the FNN model exhibits a significant R^2oos value of 3.11%. Since the historical average forecast is based on the prevailing mean returns and ignores the predictive features, these findings demonstrate that the selected features carry valuable information for forecasting the equity premium. Consequently, incorporating them into the forecasting models improves the statistical accuracy of equity premium predictions.

Figure 4 displays the cumulative squared forecast error of the historical average forecast minus that of the competing forecasting models. The figure allows for an evaluation of whether the competing forecasts outperform the historical average by comparing the curve's height at the beginning and end points of the relevant period. If the curve is higher (lower) at the end compared to the beginning, it indicates that the competing forecast has a lower (higher) MSFE than the historical average forecast over that period. A consistently superior forecast will exhibit a positive slope throughout the entire forecast horizon. While attaining this ideal is unlikely in practice, the closer the curve aligns with this pattern, the better the forecast performance.

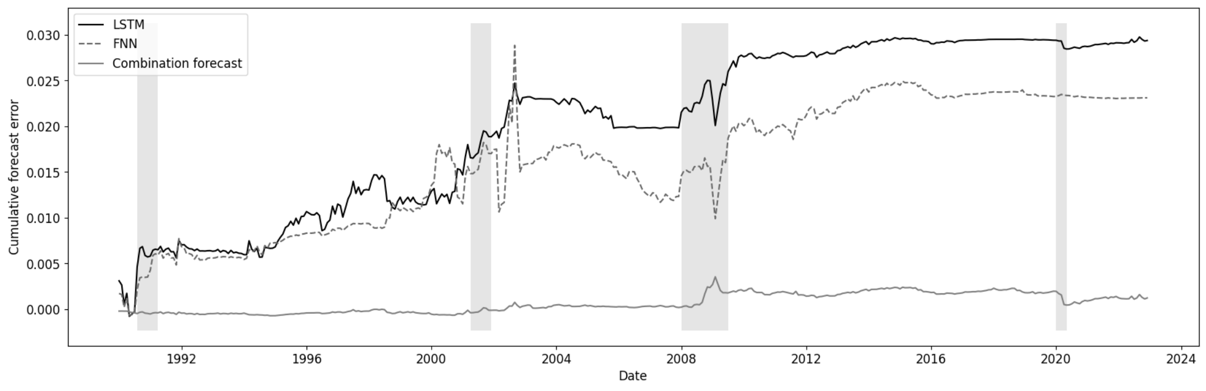


Figure 4: Cumulative difference in squared forecast error relative to the historical average forecast

Note: Cumulative difference in squared forecast error of the LSTM, FNN, and combination forecast relative to the historical average forecast measured over the OOS forecast horizon, spanning from 1990:01 to 2022:12. The legend indicating the colours corresponding to the curves is displayed in the upper left corner. Evaluated based on the entire OOS period, all three competing models demonstrate a lower cumulative squared forecast error than the historical average. The vertical grey bars indicate recession periods as classified by the NBER.

All three models outperform the historical average forecast measured by their MSFE evaluated over the entire OOS forecast horizon, as noticeable in Figure 4 by the increased levels of the curves at the end points of the period. However, the lines do not consistently exhibit a positive slope, indicating that there are months where the historical average forecast demonstrates superior performance. Despite the combination forecast resulting in the smallest improvement in MSFE measured by the model's R^2oos statistic, it most consistently outperforms the historical average, specifically in 56.57% of the months. The LSTM model closely follows, surpassing the historical average in 54.54% of the months, and the FNN beats the model in 53.03% of the months. These findings suggest that the combination forecast, despite the lower R^2oos value, is more consistent and tends to outperform the historical average more frequently. The combination forecast employs an effective shrinkage effect by averaging the forecasts of the univariate regression models. This leads to a multivariate regression model that guards against overfitting, resulting in a stable forecast with a relatively low forecast volatility evident in Figure 7, Appendix B. However, the implied shrinkage effect is too strong and, while occurring more frequently, the improvements in the forecasting errors demonstrated by the combination forecast are relatively smaller compared to those of the LSTM and FNN models.

A closer examination of Figure 4 shows abnormal performance during recession periods, which are indicated by the vertical grey bars. Specifically, the impact of business-cycle fluctuations becomes clear from the elevated levels of the curves observed, when contrasting the starting and ending points of most recession periods. Accordingly, the forecasts that incorporate the predictive features demonstrate enhanced performance during recessions compared to the historical average forecast. The equity premium, and the predictive features exhibit increased volatility during recession, as evident in Figures 7, 8, and 9, in Appendix B, displaying the values of the equity premium and all features. The predictive features provide stronger signals, containing valuable information during these economic downturns, which the competing forecasting models effectively capture and interpret. In contrast, the historical average is based on a constant expected equity premium, ignoring the input features and, by extension, business-cycle fluctuations. Additional evidence supporting these observations is provided in Table 4, where the R^2_{oos} statistic and utility gain are reported separately during expansions and recessions. The disparity in relative OOS forecasting performance is especially noticeable in case of the LSTM and FNN models. During expansions the R^2_{oos} statistics stand at 3.05% and 1.63%, while during recessions, these figures improve to 6.41% and 7.12%, respectively for the LSTM model and FNN.

Remarkably, during the 2020 recession, all three competing forecasting models exhibit diminished performance. Unlike the LSTM and FNN models, the combination forecast fails to compensate for these results during previous recession periods, leading to a higher R^2_{oos} value during expansions. The uniqueness of the 2020 recession, caused by the Covid-19 pandemic, could be a contributing factor to the contrasting performance of the forecasting models. This unique recession might have introduced different and distinct information signals not captured by the predictive features. Notably, the actual equity premium displayed extreme negative, but also extreme positive values during this particular recession, evident in Figure 7, Appendix B. As a result, the average returns are not significantly impacted, partly clarifying the relatively better performance of the historical average forecast.

Table 4: Forecasting performance separately during expansions and recessions

Model	**Expansion**		**Recession**	
	$\mathbf{R^2_{oos}}$ **(%)**	**Utility gain (%)**	$\mathbf{R^2_{oos}}$ **(%)**	**Utility gain (%)**
LSTM	3.046***	2.097*	6.408*	12.989*
FNN	1.630**	2.562*	7.122**	9.834*
Combination forecast	0.215	0.462	0.032	3.570

Note: The R^2_{oos} statistic, and utility gain measure the performance of the competing forecasting model in the first column relative to the historical average forecast. Where *, **, and ***, indicate significance at the 10%, 5%, and 1% level, based on the p-values corresponding to the adj-DM test to assess significance of the R^2_{oos}, and to assess significance of the utility gain, or portfolio management fee. The values are conducted separately during expansions and recessions, according the NBER, with recessions accounting for 11% of the total time period.

Measured over the entire OOS forecast horizon, the LSTM model demonstrates superior statistical performance, surpassing not only the historical average but outperforming all benchmark models. Table 9, Appendix A, shows that the model achieves significant R^2oos statistics of 0.87%, and 3.79%, after comparing the LSTM model's MSFE directly to that of the FNN, and combination forecast, respectively.

In summary, all three forecasts surpass the historical average in terms of their RMSFE. Yet, only the neural network models demonstrate a significant positive R^2_{oos} statistic. The non-linear and high-dimensional processing capabilities of the LSTM and FNN architectures enables these models to capture, interpret and combine the information in the data more effectively relative to the linear combination forecast. When comparing the two neural network models, the LSTM model shows enhanced statistical accuracy and results in a more stable forecast, outperforming the historical average on a more consistent basis over time. The LSTM model's ability to recognise time-dependencies and retain information in its network enhances the robustness of its overall forecast against the noise present in the data. The gating mechanisms provide the model with the ability to use only the information the model considers as important, and ignore the rest. Figure 7, Appendix B presents the OOS forecasts for all models and reveals a smoother curve for the LSTM forecast when compared to that of the FNN. Accordingly, the forecast volatility of the LSTM model measured over the entire forecast horizon is 21% lower than that of the FNN model.

Stock return predictability is closely linked to business-cycle fluctuations, with a greater degree of forecastability observed during recessions. Consequently, model's performance is evaluated separately during recession and expansion periods. Notably, only the neural network models demonstrate enhanced statistical accuracy during recessions. These findings indicate that the additional information available during recession periods is captured more effectively by these non-linear models. Their high-dimensionality enhances their flexibility, and ability to adapt to changing market conditions.

Among both neural network forecasts, Table 4 reveals a relatively higher R^2_{oos} for the FNN model during recession periods. Accordingly, the FNN forecast exhibits a lower RMSFE during economic downturns, while measured over the entire OOS period the LSTM model demonstrates superior statistical accuracy. This difference can be attributed to the fact that the actual equity premium exhibits increased volatility and more extreme values during recessions. The FNN model proves better at forecasting these extreme values, whereas the gating mechanisms in the LSTM model tend to overly smooth its forecast during recessions, resulting in a relatively lower MSFE.

4.2 Economic value of the forecasts

Until now, the analysis has primarily focused on the MSFE metric. However, the model's MSFE might not be the most suitable metric to evaluate equity return predictions. To asses economic value, Table 3 displays the Sharpe ratio and average utility gain to a mean-variance investor with risk aversion coefficient of five. The Sharpe ratio is calculated based on the mean and volatility of the expected portfolio returns obtained from the asset allocations (following equation (22)). The utility gain indicates the portfolio management fee that an investor would be willing to pay to have access to the competing forecasting model in the first column, instead of relying on the historical average. The utility gain, or portfolio management fee, is given in annualised percent return. Over the entire OOS forecast horizon, the historical average forecast yields a Sharpe ratio of 0.157. All three competing forecasting models demonstrate superior economic value, resulting in higher Sharpe ratios and significant utility gains.

The combination forecast leads to a utility gain of 0.79% relative to the historical average forecast. However, Table 9, Appendix A, shows that the model is outperformed by the LSTM forecast, with investors willing to pay an annual portfolio management fee of 2.52% to gain access to the LSTM model's forecasts rather than to use that of the combination forecast in a trading strategy.

The portfolio formed using the LSTM model's forecast in the asset allocations yields a Sharpe ratio of 0.235, which is slightly below the 0.236 achieved by the portfolio based on the FNN model's forecast. The FNN model offers the highest utility gain relative to the historical average at 3.39%, while the LSTM model maintains an annual utility gain of 3.32%. Hence, the utility gain is not linearly correlated with the R^2_{oos} statistic, which is higher for the LSTM model. When comparing the economic value measure of the LSTM forecast directly to that of the FNN, the resulting utility gain, or portfolio management fee, has a negative value of -0.08%, as can be seen in Table 9, Appendix A.

These contradictory findings can be partly attributed to the boundaries imposed on the risky equity portfolio weights during the asset allocations. To ensure realistic portfolios, the risky equity weights are bounded to lie between 0.0 and 1.5. These bounds act as a constraining factor, impacting the actual proportion invested in equities by investors. Particularly, the upper bound significantly affects portfolios constructed using the FNN model's forecast. In absence of the boundaries, a mean-variance investor using the FNN forecast would, on average, fully invest in risky equities. However, with the constraints in place, her average risky equity weights decrease to 0.7. In contrast, a mean-variance investor using the LSTM model's forecast allocates, on average, 0.8 to risky equities without boundaries, which decreases to 0.7 after imposing the weight constraints. Consequently, the weight restrictions have a greater impact in case of the FNN model, resulting in a smoothing effect of the optimal weight allocations. As previously discussed during the assessment of the statistical accuracy, the LSTM model's gating mechanism enable the model to ignore information the model views as noise, resulting in a more robust forecast with a relatively lower forecast volatility. The FNN model seemed better at capturing extreme values in the equity premium. The smoothing effect of weight restrictions on the FNN model's optimal asset allocations diminishes the advantage of the LSTM model in this regard. Hence, somewhat explains the contrasting results in performance measured by economic value compared to statistical accuracy between the LSTM and FNN models.

In conclusion, when focusing on economic gains, a mean-variance investor would prefer to use the FNN model's equity premium forecast, under the restriction of realistic portfolio boundaries. The gates within the LSTM model consider parts of the input information as not useful, that appears of significant value to forecast the equity premium when the aim is to maximise economic value. These results highlight the importance of incorporating economic value-based measures alongside conventional statistical criteria when evaluating OOS equity return predictability. By combining statistical accuracy with economic insights, investors can make more informed decisions, optimise their investment strategies, and achieve better overall performance.

4.3 Feature contribution to the forecasts

The contribution of individual features to the forecasts is evaluated using the SHAP method, where SHAP values are computed individually for each feature and every prediction within the OOS forecast horizon. This provides insight into the impact of each feature on the model's predictions, as the SHAP value represents the contribution of that feature to change in the expected model's prediction when conditioning on that feature. The average aggregate SHAP values serves as a measure of feature importance and contribution over the entire OOS forecast horizon. The absolute value signifies the level of feature importance, while the sign of the value denotes the average direction of the effect that feature has on the predictions. A positive SHAP value signifies that the presence of that feature contributes to an increase in the predicted output value. Conversely, a negative SHAP value implies that including that feature in the model leads to a decrease in the value of the predictions.

Figure 5 illustrates a plot of the SHAP values of the LSTM model for each prediction in the OOS forecast horizon. The 20 most important features are ordered according to their average absolute SHAP value, hence a feature importance ranking. The spread of the values along the x-axis for a given feature indicates the range of impacts that feature has on the model's output. A wide spread means that the feature has a varied impact, while a narrow spread indicates a more consistent impact.

In the LSTM forecast, DY has the highest average absolute SHAP value, and all its individual SHAP values are consistently ≤ 0. This implies that, on average, DY contributes the most to the predictions, and its inclusion typically diminishes the equity premium prediction. Notably, within the five most important features, the model predominantly ranks fundamental features, additionally incorporating DP, DE, and LTY. The spread of values for DE signifies that, in situations where the feature value is low (indicating that stock earnings surpass stock dividends), including the feature tends to have a negative effect on the model's output. Conversely, when DE is high, including the feature has a positive effect on the model's predictions. These insights may be intuitively interpreted as the model linking higher dividends relative to earnings to a form of risk compensation. Investors could be demanding higher dividends as a buffer against perceived risk, thereby driving the equity premium upwards. In contrast, when earnings are high, it might signal stronger corporate health or growth prospects, reducing perceived risk and the required equity premium.

Among technical indicators, VOL(1,12) is distinguished as the most important. This feature

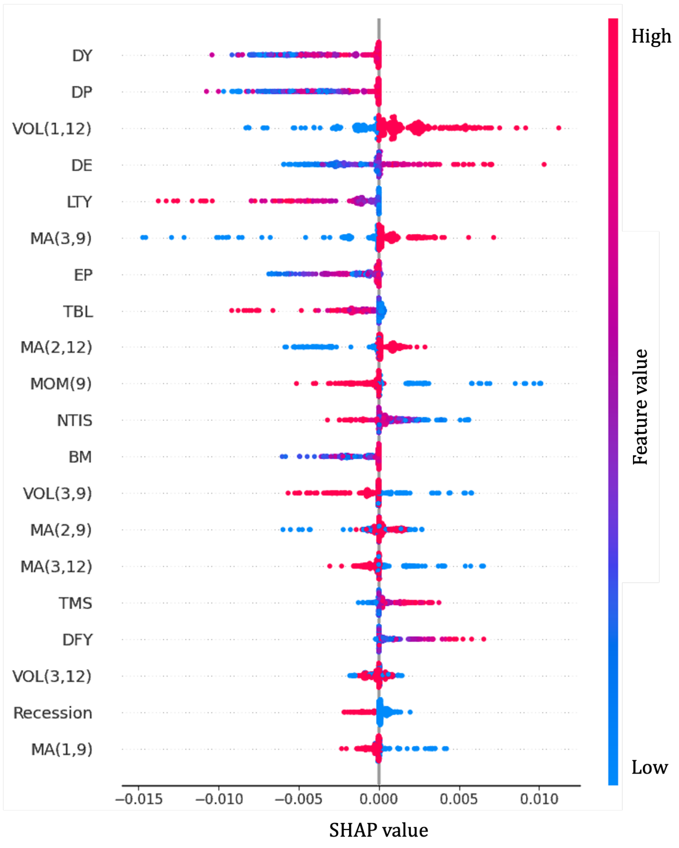


Figure 5: Feature effects for the LSTM model's forecasts

Note: The SHAP values of the 20 most important features for each prediction within the OOS forecast horizon. The features are ordered on the y-axis according to their average absolute SHAP value. Whereas the x-axis represents the individual SHAP values for each prediction. The colour for a given data point is a representative for the value of the feature. Red represents high feature values and blue represents low feature values.

consistently yields a SHAP value ≥ 0 in the case of a buy signal, and a SHAP value ≤ 0 when it indicates a sell signal. Despite the relatively high importance of the VOL(1,12) technical indicator, as illustrated in Figure 10 and Appendix B, its counterparts VOL(3,12), VOL(2,9), VOL(2,12), and VOL(1,9), yield relatively low absolute SHAP values, indicating minor contributions to the forecasts. This observation implies that the additional insights provided by these four features, which are based on the same trading strategy, do not contribute substantially to the information already captured by VOL(1,12), and to a lesser extend, by VOL(3,9).

Interestingly, while VOL(1,12) is associated with positive SHAP values for buy signals, VOL(3,9) demonstrates predominantly negative SHAP values in case of a buy signal. This divergence suggests that the LSTM model captures interactions amongst the features within its network, which can lead to such varied impacts on the equity premium. Similarly, the pronounced values for MA(3,9) and MA(2,12) can be attributed to their unique relevance. In contrast, the SHAP values for MA(3,12) and MA(1,9) diverge and do not contribute substantial additional information, which is evident from their low feature importance ranking.

As illustrated in Figure 5, fundamental features are predominantly ranked higher within the LSTM model's feature importance, suggesting they contribute more to the model's forecasts. To evaluate the varying impacts of fundamental and technical features, the average absolute SHAP values across both feature groups are calculated. In the case of the LSTM model, the average absolute SHAP value for all fundamental features, including the recession dummy, is 0.088 (reported $\times 100$ for comparability), while the corresponding value for technical indicators is 0.061 (reported $\times 100$). Given the slightly higher average absolute SHAP value for fundamental features, their collective contribution is more pronounced, and is thus regarded as more crucial within the predictions made by the LSTM model.

Additionally, to evaluate the overall effect of the distinct feature sets on the predictions, the respective SHAP values are aggregated. Leveraging the additive feature attribution property, as shown in equation (26), the influence of multiple input features can be approximated by summing their individual SHAP values. Notably, this entails summing over the non-absolute values since values in opposite directions can neutralise each other. However, this remains an approximation as the interaction effects among the feature groups when adding or removing them from the expectation is not taken into consideration. The cumulative SHAP value of the fundamental features, including the recession dummy, yields -0.008. This suggests that the incorporation of the collective fundamental features typically results in a decrease in the LSTM model's forecast. Conversely, the technical indicators exhibit an aggregate SHAP value of 0.001, implying their collective inclusion generally increases the output value.

4.3.1 Comparative analysis with the benchmark models

To delve deeper into the inner workings of the LSTM network, the feature effects are contrasted with the feature effects in the benchmark models. Figures 10 and 11, in Appendix B, illustrate the feature importance rankings and feature effects for both benchmark models. When assessing feature importance between the LSTM and FNN models, noticeable is the consistent ranking of DP, VOL(1,12), and DE among the top five most important features. Similarly, SVAR, DFR, and INFL are identified as the three least important features by both models. Thus, a comparable ranking appears present across both forecasting models when analysing the individual feature importance values. Moreover, the estimated aggregate contribution of both features types is relatively similar. Within the FNN model predictions, the estimated aggregate SHAP value is -0.005 for the fundamental features and 0.001 for the technical indicators. The signs and magnitudes of the estimated aggregate SHAP values closely align with those of the LSTM model. However, the average absolute SHAP value within the FNN model is roughly equivalent for the two feature groups, with the fundamental features averaging at 0.123, and the technical indicators at 0.129 (both values reported $\times$100). This suggests that, within the FNN model, technical features contribute slightly more to the predictions, in contrast to the dominant role of the fundamental features in the LSTM model's forecast.

The distribution of the individual SHAP values in Figure 11, Appendix B, reveals that the FNN model exhibits relatively more extreme and less consistent feature effects (note the discrepancy in values along the x-axis when compared to the feature effects within the LSTM model). These observations indicate less uniform effects on the predictions when including the individual features in the predictions. For instance, the features DP and EP exhibit instances of positive SHAP values in the FNN model's forecasts, whereas they consistently display SHAP values ≤ 0 within the LSTM model's forecasts. These observations suggest that the gating and memory mechanisms within the LSTM architecture enable the model to recognise feature patterns over time. Consequently, the model retains only information it views as important, filters out the noise and ignores outliers, aligning with the observed more stable forecast of the LSTM model. As a result, the feature effects within the model seem more consistent and interpretable. However, the FNN model offers enhanced economic value to an investor. This implies that some of the feature values considered as outliers by the LSTM model appears to be of significant value for forecasting the equity premium.

The results from the combination forecast highlight the importance of the DP and VOL(1,12) features in forecasting equity returns, with both features ranked among the top five most important across all three models. In contrast, features such as INFL and DFR are consistently valued as less important. The average absolute SHAP value for the fundamental features in this model is 0.335, compared to 0.228 for the technical indicators (both values reported $\times$100). This signifies that the fundamental features have a marginally higher contribution to the model's predictions. The approximated aggregate SHAP value stands at -0.026 for the fundamental features and 0.004 for the technical indicators within the combination forecast model. Given the consistency of these effects across all forecasts, this study concludes that the integration of collective fundamental features generally results in a negative impact on the model's output, whereas the inclusion of the combined technical features yields a minor positive influence on the output.

Interestingly, Figure 11, Appendix B, demonstrates that the combination forecast consistently assigns positive SHAP values to the technical indicators in case of buy signals, and negative SHAP values to sell signals. In contrast, within both neural network models, as explained for VOL(1,12) and VOL(3,9), these signs may differ. The combination forecast models the features in a linear and independent manner and subsequently combines the results by taking the average of all predictions. This way the model disregards any potential interaction effects amongst the features. In contrast, the layered structure within the neural network architectures allows the models to capture, not just non-linear relations of the features with respect to the equity premium, but also complex relationships and patterns among the features. This capability yields results that might not be immediately intuitive, but can offer substantial value to investors.

In conclusion, the memory mechanism inherent to the LSTM network enables the model to retain information from previous time steps, allowing the model to recognise feature patterns over time. Consequently, the network exhibits enhanced robustness against outliers and presents more consistent feature effects compared to the FNN. However, this resilience to outliers diminishes the LSTM model's ability to model extreme values in the equity premium, in which the FNN model excels. Moreover, the ability of both neural networks to capture non-linear relationships and interaction effects among the features, yields valuable insights for investors. This is particularly noteworthy regarding technical indicators, where a buy signal is not always positively related with the equity premium.

Lastly, the ranking of individual feature importance is relatively similar across all three models. DY and DP are the most important fundamental features, and including the features in the models, on average, lowers the value of a prediction. Among all technical indicators, VOL(1,12) contributes the most to the models' forecasts, resulting in an increased value of the predictions when the feature indicates a buy signal. Evaluating the aggregate contributions of the fundamental and technical features reveals that these feature sets impose effects in opposite directions.

5 Robustness check

This section discusses two robustness checks intended to validate and reinforce the conclusions drawn regarding the capabilities of the LSTM model in forecasting the equity premium. A first check is performed by retraining both neural network models, this time using utility as the optimisation criterion with the aim of enhancing the economic value of the forecasts. Subsequently, a second check is carried out to assess the sensitivity of the LSTM model to its initialisation and parameter estimates.

5.1 Reinforce forecasting performance to maximise economic value

The parameter estimates for the LSTM and FNN models are derived by minimising the MSFE loss function. Accordingly, the models are optimised with the aim of attaining optimal statistical accuracy. The empirical findings reveal that the forecasts generated by the LSTM model achieve a lower MSFE, indicating superior statistical accuracy. Nonetheless, the forecasts from the FNN model yield greater utility to a mean-variance investor, thus offer enhanced economic value.

To reinforce these conclusions, both neural network models are re-optimised to maximise the economic value of the forecasts, as opposed to the statistical accuracy. This implies using the utility function, as given in (23), as loss function in the optimisation process to obtain parameter estimates for the models' weight and bias terms. The same SGD method as for the original models is employed to minimise the negative utility function. However, this algorithm requires the loss function to be differentiable in order to update the parameter estimates. Therefore, the constraints on the equity portfolio weights within the utility function, previously bounded between 0.0 and 1.5, are removed.

Following the re-optimisation process, where the utility function is used as loss function to obtain optimal parameter estimates, the resulting models are used to forecast the OOS period. Table 5 presents the forecasting performance of both models. To ensure comparability with the empirical results in Section 4, the R^2_{oos} statistic and utility gain are reported relative to the historical average forecast.

Table 5: Reinforce forecasting performance to maximise economic value

Model	**RMSFE**	**Sharpe ratio**	$\mathbf{R^2_{oos}}$ **(%)**	**Utility gain (%)**
LSTM	4.795	0.084	-22.485***	-3.672
FNN	5.161	0.159	-41.898***	-0.301

Note: Forecasting results of the LSTM model and FNN being optimised with the aim of maximising realised utility to an investor. The RMSFE is calculated over the entire OOS forecast horizon, ranging from 1990:01 to 2022:12 (values are reported $\times$100). The mean and volatility of the expected portfolio returns obtained from the asset allocations are used to calculate Sharpe ratios. The R^2_{oos} measures the reduction in MSFE for the forecast given in the first column relative to the historical average forecast. The average utility gain is given in annualised percent return, indicating the portfolio management fee that an investor, with mean-variance preferences and risk aversion coefficient of five, would be willing to pay to have access to the competing forecasting model relative to the historical average forecast. Where *, **, and ***, indicate significance at the 10%, 5%, and 1% level, based on the p-values corresponding to the adj-DM test to assess significance of the R^2_{oos}, and to assess significance of the utility gain, or portfolio management fee.

The findings show a decrease in forecasting performance for both models after optimising the models' weight and bias parameter estimates using utility as a loss function. This reduction in performance is likely attributable to the models retaining the same hyperparameter values as the original model configurations. These hyperparameter configurations are selected based on optimal model performance as assessed by the MSFE loss function. Hence, with the aim of maximising the statistical accuracy of the models.

Nevertheless, despite the observed decline in both the statistical accuracy and economic value of the models, the relative results are consistent with the primary conclusions drawn from the forecasts of the original models. Specifically, the FNN forecast achieves a higher Sharpe ratio and provides a greater utility gain to a mean-variance investor, making it the most financially beneficial model. Conversely, the LSTM model achieves a lower RMSFE and a relatively higher R^2_{oos} statistic, indicating superior statistical accuracy. Thus, these results confirm that, even when the models are optimised with the aim of maximising economic value, the FNN forecast continues to surpass the LSTM model in performance, offering more value to an investor within a mean-variance framework.

5.2 Robustness to model parameter estimates

An LSTM model is characterised by two types of parameters: hyperparameters and model parameters. As outlined in Section 3.3, the hyperparameters are tuned once and remain constant throughout the entire forecast horizon. In contrast to the model parameters, namely the weight and bias terms, which undergo annual updates. Since the neural network models use an expanding estimation window, the in-sample period is annually updated, resulting in the yearly updates of the model's parameter estimates. Moreover, to mitigate the risk of overfitting and to adapt the models to changing time periods, the weight and bias terms are systematically reset after every five years of consecutive forecasting. Following this reset, newly initialised model parameter estimates are derived and leveraged to forecast the equity premium in the subsequent five years.

This procedure of re-initialising the weight and bias terms for every five-year forecasting horizon is a computationally intensive, and time-consuming process. In addition, it may reduce the model's robustness across different data-sets or time periods. To evaluate the model's performance without the five-yearly reset and re-initialisation of parameter estimates, Table 6 presents the forecasting results for the same LSTM network, with the difference being that the initial weight and bias terms are employed to generate forecasts for the entire OOS period. This approach implies that the initial parameter estimates are leveraged to generate a forecast for the equity premium in the first year of the test period. In the subsequent years, a fixed number of epochs is consistently used for retraining to derive parameter updates, and to generate a forecast for the equity premium in the following year of the OOS test period. Early stopping is implemented to mitigate the risk of overfitting.

The results demonstrate that the LSTM model struggles to forecast the equity premium when

Table 6: Performance of the LSTM model without re-initialising its parameter estimates

Model	**RMSFE**	**Sharpe ratio**	$\mathbf{R^2_{oos}}$ **(%)**	**Utility gain (%)**
LSTM (without re-initialising)	4.401	0.132	-3.196***	-1.818

Note: The RMSFE is calculated over the entire OOS forecast horizon, ranging from 1990:01 to 2022:12 (values are reported $\times 100$). The mean and volatility of the expected portfolio returns obtained from the asset allocations are used to calculate Sharpe ratios. The R^2_{oos} measures the reduction in MSFE for the forecast given in the first column relative to the historical average forecast. The average utility gain is given in annualised percent return, indicating the portfolio management fee that an investor, with mean-variance preferences and risk aversion coefficient of five, would be willing to pay to have access to the competing forecasting model relative to the historical average forecast. Where *, **, and ***, indicate significance at the 10%, 5%, and 1% level, based on the p-values corresponding to the adj-DM test to assess significance of the R^2_{oos}, and to assess significance of the utility gain, or portfolio management fee.

its parameters are not reset and re-initialised once every five years. The model's RMSFE of 4.401 is significantly higher compared to the historical average forecast, substantiated by the corresponding R^2_{oos} value of -3.20%. In addition, the utility gain is negative and substantiated by both statistical as economical results, the entire OOS forecast horizon of 33 years appears to be too long, and the model is outperformed by the historical average.

Figure 6 illustrates that there are instances where the LSTM model without the re-initialisation of its parameters, surpasses the historical average forecast in performance. However, these instances are limited, and the model generally underperforms when evaluated over the entire OOS test set. It is notable that throughout the latter half of the OOS forecast horizon, where the original LSTM model's forecast maintains relative stable, the model without parameter re-initialisation seems to perform comparatively well. Conversely, in periods characterised by improved performance of the original LSTM model, the model without parameter re-initialisation tends to perform significantly worse. During these specific intervals, such as the period prior to 1996, the original LSTM model exhibits heightened volatility, evident from Figure 7a, Appendix B. In these more volatile periods, there appears to be a greater degree of predictability in the equity premium for the LSTM model, resulting in the increased difference in cumulative forecast error relative to the historical average forecast. And the re-initialisation of the model's weight and bias terms significantly enhances its performance.

The findings emphasise the sensitivity of the LSTM model to its weight and bias parameters and the updates thereof. This is particularly evident during periods where the original LSTM model's forecast shows increased volatility, and on average, increased performance. In these instances, the parameter estimates of the model become increasingly important, and their re-initialisation considerably enhances the forecasting performance of the LSTM model.

The model is not able to generate accurate equity premium predictions for an extended time period, while leveraging the same initial parameter estimates. Hence, integrating the LSTM

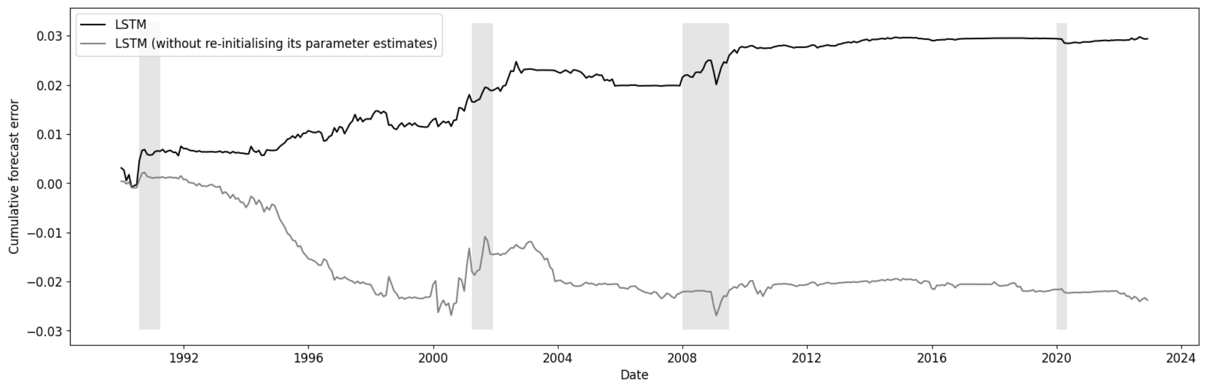


Figure 6: Performance of the original LSTM model versus the LSTM model without re-initialising the model's parameter estimates

Note: Cumulative difference in squared forecast error of the two LSTM models relative to the historical average forecast measured over the OOS forecast horizon. The black line represents the original LSTM model, while the grey line illustrates the LSTM model without re-initialising its parameter estimates for every five-year forecast horizon. The vertical grey bars indicate recession periods as classified by the NBER.

forecast into a trading strategy carries a degree of risks, as the model's forecasting results are highly sensitive to precise parameter estimates which are less robust and less stable over a relatively long forecast horizon.

6 Conclusion

Extending the literature on stock return predictability, this paper introduces the LSTM model to predict the U.S. equity premium. The LSTM network architecture incorporates an internal memory mechanism, enhancing the model's capability to recognise time dependencies within time-series data. While the main objective is to improve OOS forecasting performance, this study also presents an in-depth analysis of the complex inner workings of the LSTM network, providing valuable insights to researchers and investors using financial forecasting models.

The main conclusion and answer to the research question of this study is, that the LSTM neural network model can be employed to forecast the equity premium, yielding significant improvements in forecasting performance compared to the linear benchmark models. However, while the LSTM model's forecast demonstrates enhanced statistical accuracy, the FNN's forecast offers higher economic value.

The memory mechanism of the LSTM model controls the flow of information through the network, it decides what information to retain, and what to ignore. This enables the model to filter out noise and only use information the model considers as important. The analysis on feature effects, particularly the distribution thereof, reveals that the LSTM network's ability to recognise feature patterns over time, yields more consistent and uniform feature effects on the predictions than within the FNN model. As a result, the LSTM model generates a relatively more stable forecast, with a lower forecast volatility, compared to the FNN model. However, the enhanced economic value of the FNN model suggests that the LSTM model considers certain extreme feature values as outliers or noise, although these appear to incorporate valuable information for predicting the equity premium.

In addition, the analysis on feature contribution to the forecasts shows that the individual feature importance ranking is relatively similar across all models. Integrating the collective set of fundamental features in the models generally results in a lower value of the equity premium predictions, whereas this effect is in the opposite direction for the collective technical features. Furthermore, the high-dimensional structure of the neural network architectures allows the models to capture and interpret interactions among the features, yielding results that might not be immediately intuitive, but can offer valuable insights.

However, this research also has its limitations. The study relies on the neural network architectures as obtained through the described hyperparameter tuning process. The choice of hyperparameters in the LSTM model, although tuned, may not represent the optimal configuration to achieve the highest economic value. Future research could explore a wider range of hyperparameter configurations for both neural network models, specifically tuned to optimise economic value as opposed to statistical accuracy. Additionally, varying sizes for the training and validation sets, potentially a shorter validation set, such that the training sample is closer to the test set, could be explored. By building upon the findings, and addressing the limitations of this study, future researchers can contribute to the evolving landscape of financial forecasting, possibly finding models that balance statistical accuracy with substantial economic value.

Forecasting equity returns holds considerable interest to both researchers and investors. Given the inherent level of uncertainty and randomness in the behaviour of equity returns, a limited degree of improvements in forecasting performance is to be expected. However, it is important to realise that little goes long way, and even minimal enhancements can yield substantial value. This paper demonstrates that the canonical problem of measuring the equity premium continues to be an exciting and dynamic area of research.

References

Brock, W., Lakonishok, J., and LeBaron, B. (1992). Simple technical trading rules and the stochastic properties of stock returns. *The Journal of finance*, 47(5):1731–1764.

Campbell, J. Y. and Thompson, S. B. (2008). Predicting excess stock returns out of sample: Can anything beat the historical average? *The Review of Financial Studies*, 21(4):1509–1531.

Clark, T. E. and West, K. D. (2007). Approximately normal tests for equal predictive accuracy in nested models. *Journal of econometrics*, 138(1):291–311.

Cowles, A. (1933). Can stock market forecasters forecast? *Econometrica: Journal of the Econometric Society*, pages 309–324.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Diebold, F. and Mariano, R. (1995). Comparing predictive accuracy. *Journal of Business Economic Statistics*, 13(3):253–63.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273.

Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Ican, O., Celik, T. B., et al. (2017). Stock market prediction performance of neural networks: A literature review. *International Journal of Economics and Finance*, 9(11):100–108.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Leitch, G. and Tanner, J. E. (1991). Economic forecast evaluation: profits versus the conventional error measures. *The American Economic Review*, pages 580–590.

Lettau, M. and Ludvigson, S. (2001). Consumption, aggregate wealth, and expected stock returns. *the Journal of Finance*, 56(3):815–849.

Leung, M. T., Daouk, H., and Chen, A.-S. (2000). Forecasting stock indices: a comparison of classification and level estimation models. *International Journal of forecasting*, 16(2):173–190.

Linardatos, P., Papastefanopoulos, V., and Kotsiantis, S. (2020). Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Maier, H. R. and Dandy, G. C. (1998). Understanding the behaviour and optimising the performance of back-propagation neural networks: an empirical study. *Environmental Modelling & Software*, 13(2):179–191.

McCracken, M. W. and Valente, G. (2018). Asymptotic inference for performance fees and the predictability of asset returns. *Journal of Business & Economic Statistics*, 36(3):426–437.

Neely, C. J., Rapach, D. E., Tu, J., and Zhou, G. (2014). Forecasting the equity risk premium: the role of technical indicators. *Management science*, 60(7):1772–1791.

Rapach, D., Strauss, J. K., and Zhou, G. (2010). Out-of-sample equity premium prediction: Combination forecasts and links to the real economy. *The Review of Financial Studies*, 23(2):821–862.

Rapach, D. and Zhou, G. (2013). Forecasting stock returns. In *Handbook of economic forecasting*, volume 2, pages 328–383. Elsevier.

Rapach, D. and Zhou, G. (2020). Time-series and cross-sectional stock return forecasting: New machine learning methods. *Machine learning for asset management: New developments and financial applications*, pages 1–33.

Taddy, M. (2018). The technological elements of artificial intelligence. In *The economics of artificial intelligence: An agenda*, pages 61–87. University of Chicago Press.

Welch, I. and Goyal, A. (2008). A comprehensive look at the empirical performance of equity premium prediction. *The Review of Financial Studies*, 21(4):1455–1508.

Wüthrich, M. V. and Merz, M. (2023). *Statistical foundations of actuarial learning and its applications*. Springer Nature.

Yu, Y., Si, X., Hu, C., and Zhang, J. (2019). A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270.

A Appendix: Tables

Table 7: Summary statistics

Variable	**Mean**	**Standard deviation**	**Minimum**	**Maximum**	**Autocorrelation**
Equity premium (%)	0.57	4.32	-24.76	14.73	0.06
DP	-3.55	0.43	-4.52	-2.60	0.99
EP	-2.81	0.43	-4.84	-1.90	0.99
DY	-3.55	0.43	-4.53	-2.59	0.99
DE	-0.74	0.29	-1.24	1.38	0.99
SVAR	0.21	0.01	0.00	0.07	0.40
BM	0.50	0.25	0.12	1.21	0.99
NTIS	0.01	0.02	-0.06	0.05	0.98
TBL (annual %)	4.03	3.09	0.01	16.30	0.99
LTY (annual %)	5.68	2.86	0.62	14.82	0.99
LTR (%)	0.48	2.78	-11.24	15.23	0.07
TMS (annual %)	1.64	1.36	-3.65	4.55	0.96
DFY (%)	0.96	0.43	0.32	3.38	0.97
DFR (%)	0.04	1.44	-9.76	7.37	-0.05
INFL (%)	0.29	0.37	-1.92	1.81	0.55

Note: The summary statistics for the log equity premium and the 14 fundamental variables. The total sample period spans from 1950:01 to 2022:12, consisting of 876 observations.

Table 8: Hyperparameter search space and optimal values for the FNN model

Hyperparameter	**Search space**	**Optimal**
Number of FNN layers	$1, 2, 3, 4, 5$	3
Neurons FNN layer	$8, 16, 32, 64$	$8, 8, 8$
Activation function FNN layers	Sigmoid, tanh, SELU, ReLU	ReLU
Activation function output layer	Sigmoid, tanh, SELU, ReLU	SELU
L_1-norm penalty	None, $10^{-1}, 10^{-2}, ..., 10^{-6}$	10^{-5}
L_2-norm penalty	None, $10^{-1}, 10^{-2}, ..., 10^{-6}$	10^{-4}

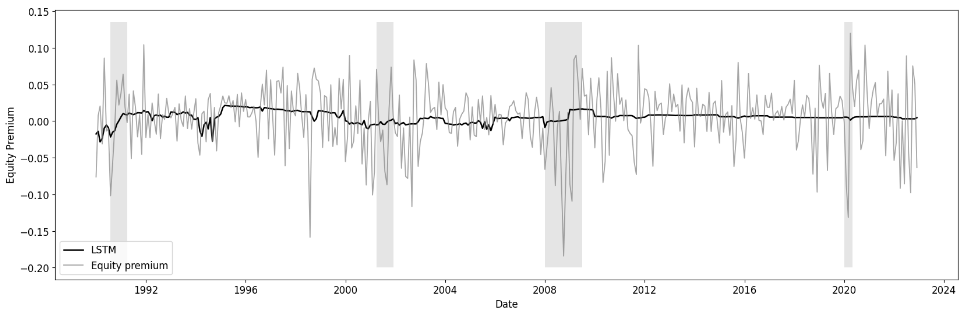
Note: The search space for each hyperparameter is explored with a grid search over all possible combinations. The hyperparameters are tuned once using the initial training and validation sets, spanning from 1950:01 to 1979:12, and 1980:01 to 1989:12, respectively. The optimal values are determined based on the model's performance on the validation set, using the MSFE as the loss function. The number of neurons per FNN layer is specified in their respective order, with all three FNN layers containing 8 neurons. The optimal activation function is found to be consistent across all FNN layer, and equal to the ReLU activation function.

Table 9: Forecasting performance of the LSTM model relative to the FNN and combination forecast

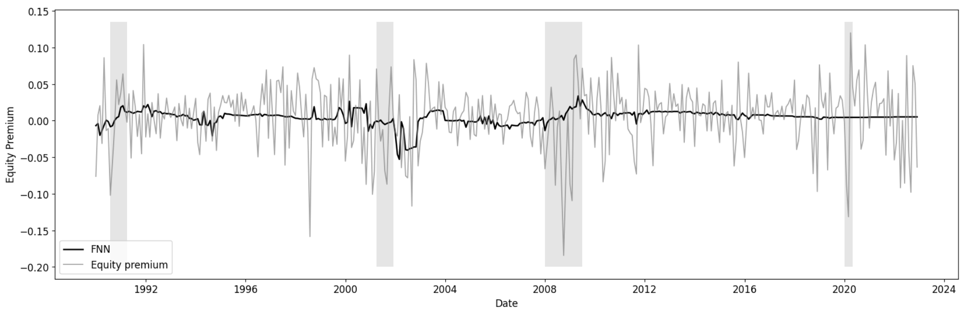
Model	$\mathbf{R^2_{oos}}$ **(%)**	**Utility gain (%)**
Combination forecast	3.791***	2.517***
FNN	0.870*	-0.081

Note: The R^2_{oos} measures the reduction in MSFE for the LSTM model relative to the benchmark model given in the first column. The average utility gain is given in annualised percent return, indicating the portfolio management fee that an investor, with mean-variance preferences and risk aversion coefficient of five, would be willing to pay to have access to the LSTM model instead of the benchmark models. Where *, **, and ***, indicate significance at the 10%, 5%, and 1% level, based on the p-values corresponding to the adj-DM test to assess significance of the R^2_{oos}, and to assess significance of the utility gain, or portfolio management fee.

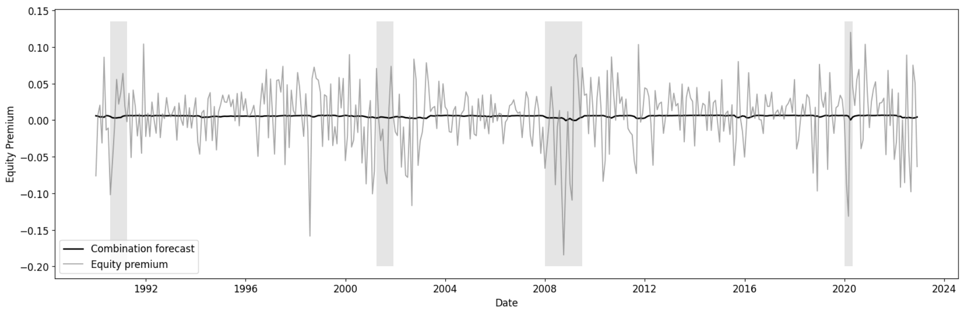
B Appendix: Figures



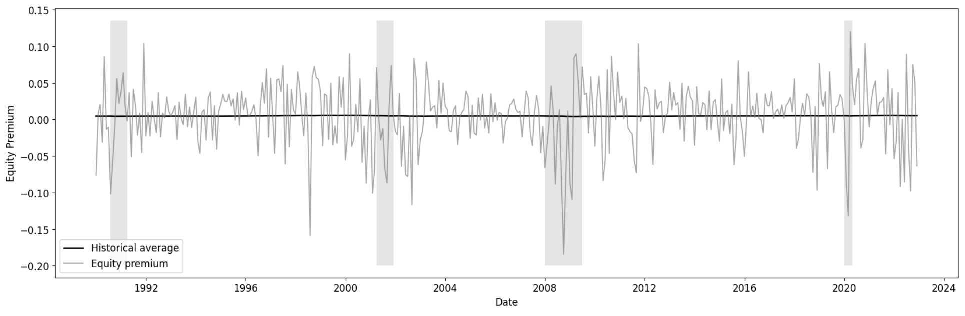
(a) LSTM forecast



(b) FNN forecast



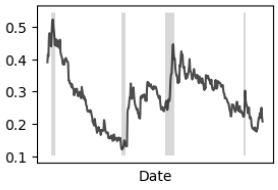
(c) Combination forecast



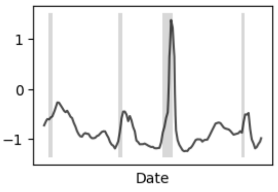
(d) Historical average forecast

Figure 7: Equity premium forecasts

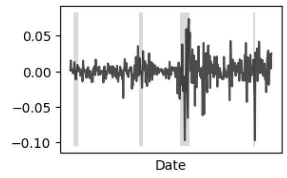
Note: Forecast of the LSTM model and the three benchmark models through the entire OOS forecast horizon, spanning from 1990:01 to 2022:12. The vertical grey bars indicate recession periods as classified by the NBER.



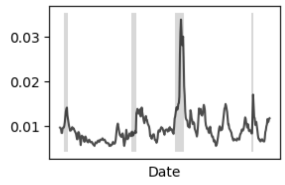
(a) BM



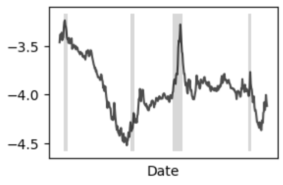
(b) DE



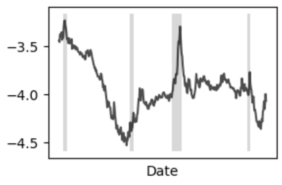
(c) DFR



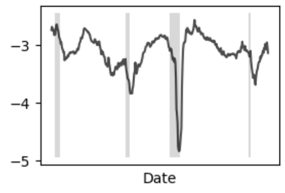
(d) DFY



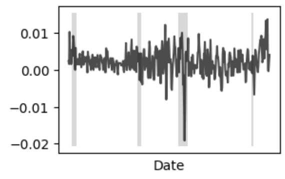
(e) DP



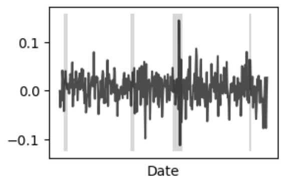
(f) DY



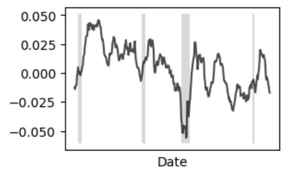
(g) EP



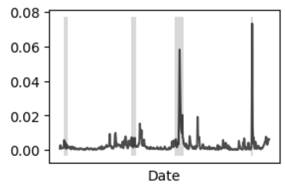
(h) INFL



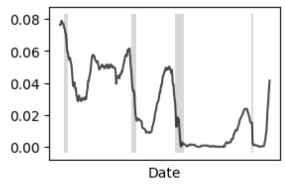
(i) LTR



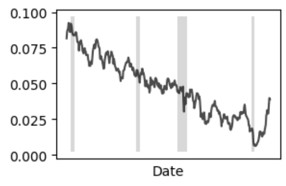
(j) NTIS



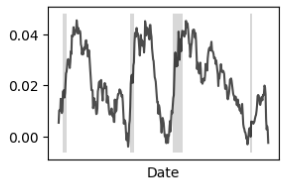
(k) SVAR



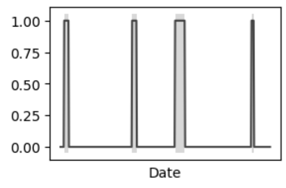
(l) TBL



(m) TLY



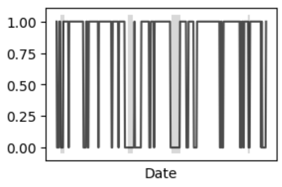
(n) TMS



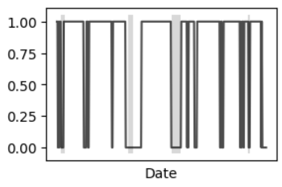
(o) Recession dummy

Figure 8: Fundamental input features

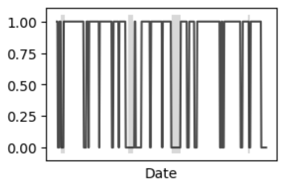
Note: Values of the fundamental input features throughout the entire OOS forecast horizon, spanning from 1990:01 to 2022:12. The vertical grey bars indicate recession periods as classified by the NBER.



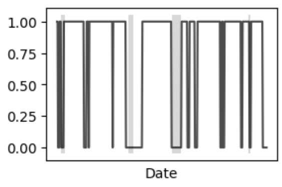
(a) MA(1,9)



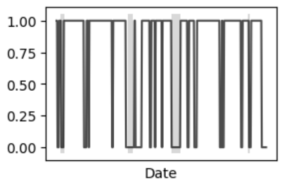
(b) MA(1,12)



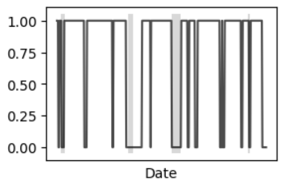
(c) MA(2,9)



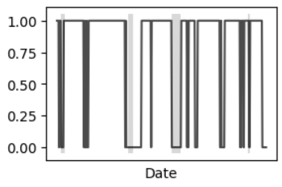
(d) MA(2,12)



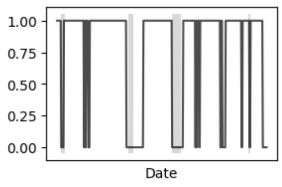
(e) MA(3,9)



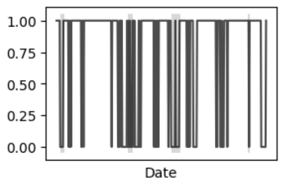
(f) MA(3,12)



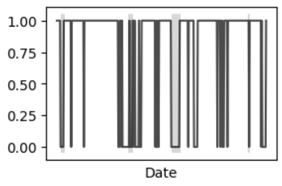
(g) MOM(9)



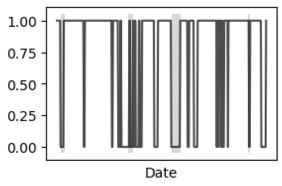
(h) MOM(12)



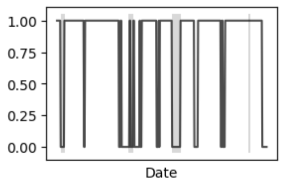
(i) VOL(1,9)



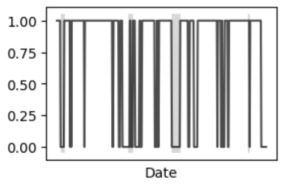
(j) VOL(1,12)



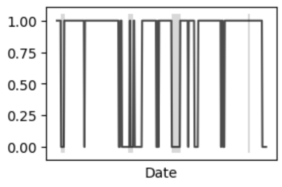
(k) VOL(2,9)



(l) VOL(2,12)



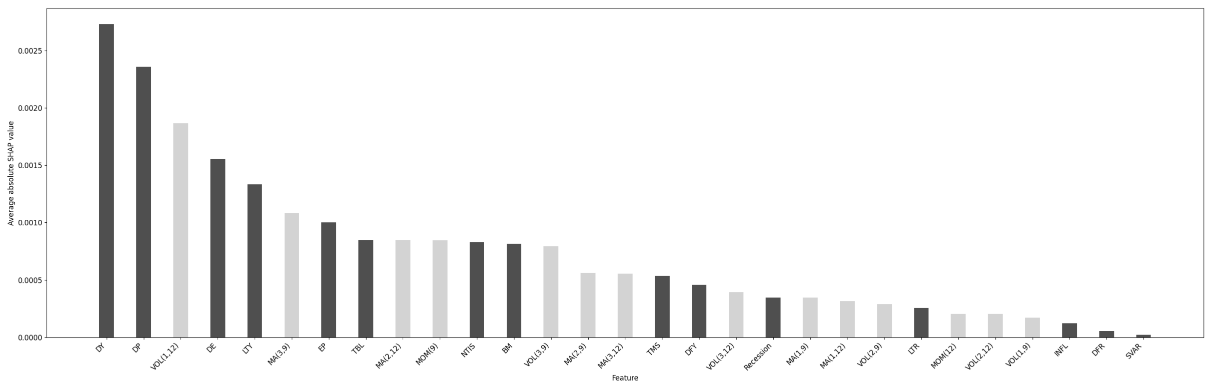
(m) VOL(3,9)



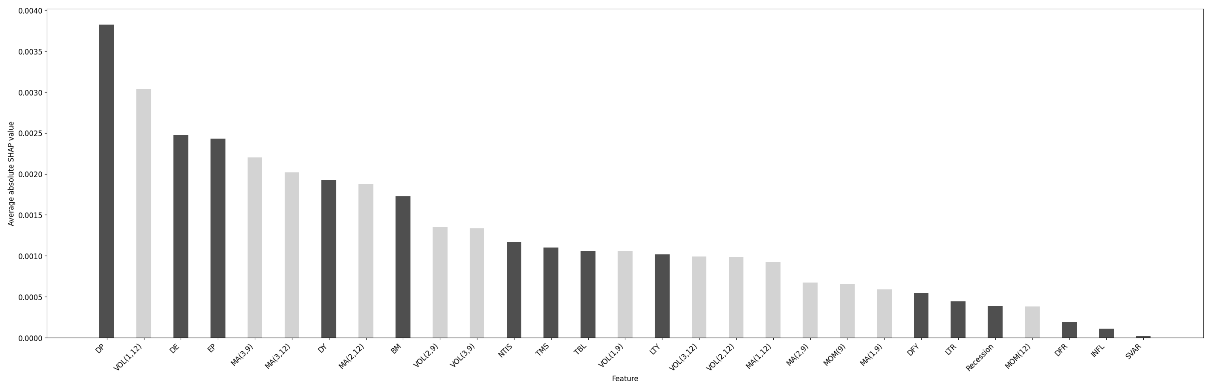
(n) VOL(3,12)

Figure 9: Technical input features

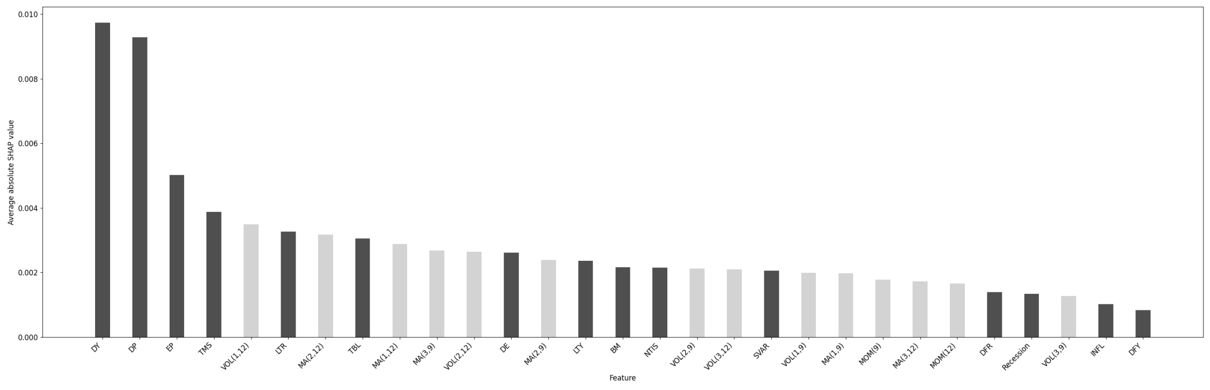
Note: Values of the technical input features throughout the entire OOS forecast horizon, spanning from 1990:01 to 2022:12. The vertical grey bars indicate recession periods as classified by the NBER.



(a) LSTM forecast feature importance



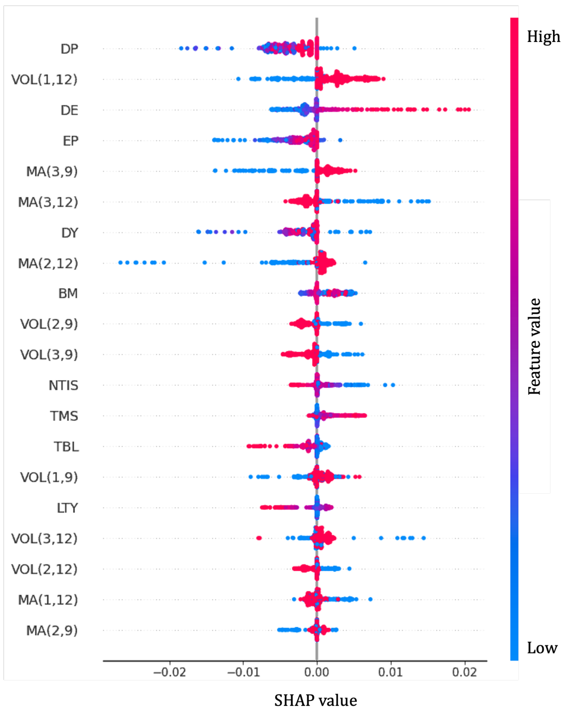
(b) FNN forecast feature importance



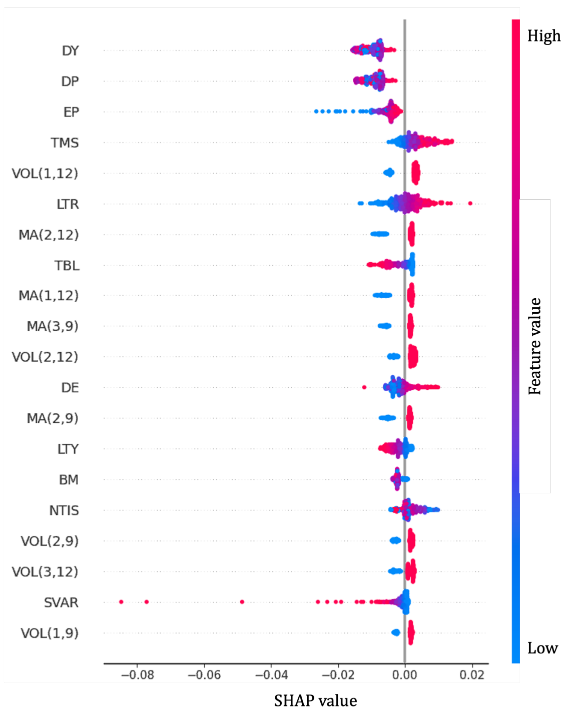
(c) Combination forecast feature importance

Figure 10: Feature importance

Note: Ranking of the average absolute SHAP values for all individual features. The feature importance values are computed for each prediction, and the average values are measured over the OOS forecast horizon. The dark grey bars indicate the fundamental features, and the light grey bars indicate the technical features.



(a) FNN forecast



(b) Combination forecast

Figure 11: Feature effects for the benchmark model's forecasts

Note: The SHAP values of the 20 most important features for each prediction within the OOS forecast horizon. The features are ordered on the y-axis according to their average absolute SHAP value. Whereas the x-axis represents the individual SHAP values for each prediction. The colour for a given data point is a representative for the value of the feature. Red represents high feature values and blue represents low feature values.

C Appendix: Additional details and equations

C.1 Trading strategies

The first strategy revolves around the moving average (MA) rule, creating trading signals based on the change in price trends. The MA price index is denoted in equation (28):

$$\text{MA}_{j,t} = (1/j)\sum_{i=0}^{j-1} P_{t-i} \quad \text{for} \quad j = s, l, \tag{28}$$

where P_t represents the S&P 500 index or its price level at time t, and j = s, l, corresponding to the length in months of the short or long MA, respectively. A total of six monthly MA signals, denoted as MA(s,l), are generated by comparing two MAs with s = 1, 2, 3, and l = 9, 12 months, following equation (29):

$$\text{MA(s,l)}_t = \begin{cases} 1 \text{ if } \text{MA}_{s,t} \geq \text{MA}_{l,t} \\ 0 \text{ if } \text{MA}_{s,t} < \text{MA}_{l,t} \end{cases}. \tag{29}$$

The rationale behind this strategy is that the short $\text{MA}_{s,t}$ is more responsive to recent price movements compared to the long $\text{MA}_{l,t}$. For instance, when prices start trending upwards, the short $\text{MA}_{s,t}$ tends to rise faster than the long $\text{MA}_{l,t}$, eventually exceeding it. On the other hand, during periods of declining prices, the short $\text{MA}_{s,t}$ generally remains below the long $\text{MA}_{l,t}$. As a result, a higher short $\text{MA}_{s,t}$ initiates a buy signal, while a lower short $\text{MA}_{s,t}$ indicates a sell signal.

The second trading strategy is based on momentum. Momentum refers to a positive (negative) price trend in the recent past that is likely to continue this positive (negative) trend in the near future. The strategy is based on the belief that assets that gained value or experienced positive returns will continue to do so in the short term. Therefore, when the current price exceeds the price of m months ago, this generates a buy signal. Conversely, if the current price is below the price of m months ago this generates a sell signal. Two momentum signals, MOM(m), are generated based on equation (30), with m = 9 and 12, representing the length in months,

$$\text{MOM(m)}_t = \begin{cases} 1 \text{ if } P_t \geq P_{t-m} \\ 0 \text{ if } P_t < P_{t-m} \end{cases}. \tag{30}$$

The third strategy incorporates traded volume alongside prices to identify overall market trends. This strategy builds upon the OBV, as defined in equation (31):

$$\mathrm{OBV}_t = \sum_{t}^{k=1} VOL_k D_k, \tag{31}$$

where VOL_k represents the traded volume during period k, and D_k a binary variable that takes a value of 1 if $P_k - P_{k-1} \geq 0$, and -1 if $P_k - P_{k-1} < 0$. The MAs based on OBV are generated following equation (32):

$$\mathrm{MA}_{j,t}^{OBV} = (1/j) \sum_{i=0}^{j-1} OBV_{t-i} \quad \text{for} \quad j = s, l, \tag{32}$$

in the same manner as for the first trading strategy, where j = s, or l, correspond to the length in months of the short or long MA, respectively. Lastly, the MA^{OBV} signals are obtained, according to equation (33):

$$\mathrm{VOL(s,l)}_t = \begin{cases} 1 \text{ if } \mathrm{MA}_{s,t}^{OBV} \geq \mathrm{MA}_{l,t}^{OBV} \\ 0 \text{ if } \mathrm{MA}_{s,t}^{OBV} < \mathrm{MA}_{l,t}^{OBV} \end{cases}. \tag{33}$$

The motivation behind the final strategy is that a relatively high trading volume in recent months, combined with a recent price increase, implies a strong positive market trend. As a result, the short MA exceeds the long MA, and a buy signal is generated. Conversely, in periods of declining prices D_k, thus OBV, have negative values, resulting in a lower short MA compared to the long MA, and a sell signal is generated. Therefore, the created variables incorporate the relationship between trading volume and price movements. Six monthly volume signals are generated, denoted as VOL(s,l), with s = 1, 2, 3, and l = 9, 12 months.

C.2 Portfolio management fee in a mean-variance framework

To asses economic value in a mean-variance framework, the difference in utility to an investor when employing two distinct forecasts in her trading strategy, can be seen as a fee the investor would be willing to pay to have access to the alternative forecasting model to make investment decisions. This portfolio management fee can be interpreted as the additional portfolio return an investor requires to be indifferent between using both forecasting models, and is defined as the value for $\hat{\zeta}$ that satisfies equation (34):

$$\overline{U}(\hat{R}_m - \hat{\zeta}) - \overline{U}(\hat{R}_b) = 0. \tag{34}$$

In a mean-variance framework, substitution of the expected utility to an investor when employing models m and b in the asset allocations, results in equation (35):

$$\left(\hat{\mu}_m - \hat{\zeta} - 0.5\lambda\hat{\sigma}_m^2\right) - \left(\hat{\mu}_b - 0.5\lambda\hat{\sigma}_b^2\right) = 0 \tag{35}$$

After rearranging the terms, the portfolio management fee can be defined as the utility gain, defined in equation (36):

$$\begin{aligned}\hat{\mu}_m - \hat{\zeta} - 0.5\lambda\hat{\sigma}_m^2 - \hat{\mu}_b + 0.5\lambda\hat{\sigma}_b^2 &= 0\\ \hat{\zeta} &= \left(\hat{\mu}_m - 0.5\lambda\hat{\sigma}_m^2\right) - \left(\hat{\mu}_b - 0.5\lambda\hat{\sigma}_b^2\right)\\ \hat{\zeta} &= \hat{U} \text{ gain} = \overline{U}(\hat{R}_m) - \overline{U}(\hat{R}_b)\end{aligned} \tag{36}$$