Deep-learning models for forecasting financial risk premia and their interpretations

Andrew W. Lo & Manish Singh

To cite this article: Andrew W. Lo & Manish Singh (2023) Deep-learning models for forecasting financial risk premia and their interpretations, Quantitative Finance, 23:6, 917-929, DOI: 10.1080/14697688.2023.2203844

To link to this article: https://doi.org/10.1080/14697688.2023.2203844

Quantitative Finance, 2023
Vol. 23, No. 6, 917–929, https://doi.org/10.1080/14697688.2023.2203844

Deep-learning models for forecasting financial risk premia and their interpretations

ANDREW W. LO[†‡§*] and MANISH SINGH[§¶]

†Sloan School of Management, MIT, Cambridge, MA, USA
‡Laboratory for Financial Engineering, MIT Sloan School of Management, Cambridge, MA, USA
§Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA, USA
¶Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, USA

(*Received 6 March 2021; accepted 31 March 2023; published online 12 May 2023*)

The measurement of financial risk premia, the amount that a risky asset will outperform a risk-free one, is an important problem in asset pricing. The noisiness and non-stationarity of asset returns makes the estimation of risk premia using machine learning (ML) techniques challenging. In this work, we develop ML models that solve the problems associated with risk premia forecasting by separating risk premia prediction into two independent tasks, a time series model and a cross-sectional model, and using neural networks with skip connections to enable their deep neural network training. These models are tested robustly with different metrics, and we observe that our models outperform several existing standard ML models. A known issue with ML models is their 'black box' nature, i.e. their opaqueness to interpretability. We interpret these deep neural networks using local approximation-based techniques that provide explanations for our model's predictions.

Keywords: Machine learning; Risk-premia prediction; Explainable AI; Deep models; Return prediction; Random forest; Fintech

JEL Classification: G10, G11, G12, G17, C21, C22, C23

1. Introduction

Two significant problems have historically demanded solutions in empirical asset pricing: the first, explaining the variance in the cross-section of asset returns, and the second, explaining the asset's risk premia. Fama and MacBeth (1973) gave early evidence for the relationship between returns and the characteristics of assets via cross-sectional regression, while Fama and French (1993, 2015) described the famous multi-factor models that explain the cross-section of returns, now widely used in risk management.

The other problem of asset pricing, the measurement of an asset's risk premia, the amount by which the returns of a risky asset outperform the risk-free asset, is primarily a predictive task. Given the complexity of financial markets, a linear relationship is often not enough to capture the full scope of interactions between asset factors and excess returns (risk premia). In this work, we focus on the use of machine learning (ML) methods that can model nonlinear relationships for risk premia forecasts using the cross-section and time series of returns of stocks.

ML techniques are currently being adopted in many fields, and researchers in finance have likewise adopted ML methods for modeling financial relationships. In this article, ML methods are applied to nonlinear asset pricing models. Due to the high dimensionality and multi-collinearity of the data, linear models by themselves often fail to generalize in this area. Deep neural networks can therefore be of particular use for this problem.

Multiple examples exist in the literature which use ML-based methods for asset pricing. The work of Gu *et al.* (2018) uses several ML models to predict asset risk premia and compares their performance. Gu *et al.* (2020) use two different networks, one to learn the risk loading, β, and the other to learn factors, using linear factor type modeling. Chen *et al.* (2019) formulate the problem of asset pricing as an adversarial optimization to learn the stochastic discount factor. Given the time-series nature of stock market data, Feng *et al.* (2018) use LSTM (Long Short-Term Memory) recurrent neural networks to forecast returns.

However, the existing wide variety of ML models has typically been restricted to neural networks 3 to 5 layers deep.

The views and opinions expressed in this article are those of the authors only, and do not necessarily represent the views and opinions of any institution or agency, any of their affiliates or employees, or any of the individuals acknowledged above.
*Corresponding author. Email: alo-admin@mit.edu

Gu *et al.* (2018) observe that the performance of a network becomes saturated with increasing depth, a problem tackled in this work. This work also decouples risk premia prediction into two independent tasks. In later sections, we will observe that this decoupling leads to better models.

Another important aspect of our analysis is in the interpretability of deep models. A significant reason finance has been slow in adopting ML-based methods is their lack of interpretability. Neural networks are often a black box in which interactions between the input–output are intractable. An investor may be reluctant to invest based on a black-box model's output when the factors driving the output are unknown. Much work has been done to interpret deep neural network predictions in the domain of computer vision using a gradient-based approach, including saliency maps in Simonyan *et al.* (2014), DeconvNets (deconvolution networks) in Zeiler and Fergus (2014), guided backpropagation in Springenberg *et al.* (2014), and integrated gradients in Sundararajan *et al.* (2017). These methods, however, are most appropriate for image-related datasets and convolutional neural network models. Nakagawa *et al.* (2018) have explored the use of layer-wise relevance propagation to explain a neural network's prediction of stock returns, but this analysis is restricted to small datasets.

To solve the issue of interpretability in our work, we look at the relative feature importance in our models. Methods such as Local Interpretable Model-agnostic Explanation (LIME) (Ribeiro *et al.* 2016) and SHapley Additive exPlanation (SHAP) (Lundberg and Lee 2017) use local approximation-based methods to explain network predictions. In this work, we use LIME to find the interpretable explanation for every prediction of our model (a similar analysis can be performed using SHAP). These explanations can be used to find a model feature's local importance in the neighborhood of a data point, as well as its importance globally, aggregated over all data points. Obtaining the feature contribution and the relationship between factors and risk premia gives greater insight into the importance of each factor, and is a step in the direction of solving the problem of deep model interpretability.

Overall, we make the following contributions to the risk premia problem:

- We model the risk premia forecast into two independent prediction tasks, separating the time series and the cross-sectional components. This leads to better performance, additionally solving the non-stationarity problem inherent to stock returns while predicting the cross-sectional spread in returns.
- We modify the architecture of deep neural networks using skip connections, leading to the training of deeper models with better performance.
- We test models robustly by evaluating the prediction power for high- and low- capitalization stocks, and across different time intervals. We also create ML-based portfolios and evaluate our models.
- We explain our black-box ML models, finding features that contribute to their predictions and to the input–output relationship.

2. Risk premia estimation

In our notation, the asset risk premia for an asset i at time t is given by $r_{t,i}$. It can be modeled as:

$$r_{t,i} = \hat{r}_{t,i} + \epsilon_{t,i},\ \hat{r}_{t,i} = f(z_{t-1,i})$$

where $r_{t,i}$ is the observed excess return, $\hat{r}_{t,i}$ is the predicted excess return, $\epsilon_{t,i}$ is the prediction error, f is the model to be learned and $z_{t-1,i}$ are input features.

The aim of this ML model is to minimize the error, $\epsilon_{t,i}$, while learning the parameters of model f. It is a regression task, for which we use the mean squared error. This is given by:

$$Loss(L) = \frac{1}{NT}\sum_N \sum_T (r_{i,t} - \hat{r}_{i,t})^2$$

where T is the number of time periods, and N is the number of stocks. We discuss the different models used to minimize the $Loss(L)$ in the next subsection.

2.1. Loss decomposition

One contribution of this work is to separate risk premia prediction into two independent tasks, which we examine in greater detail here. Monthly returns can be decomposed into two components:

$$r_{i,t} = r^{prime}_{i,t} + \bar{r}_t, \quad \hat{r}_{i,t} = \hat{r}^{prime}_{i,t} + \bar{\hat{r}}_t \tag{1}$$

where $\bar{r}_t$ is the average value of monthly returns of all stocks and $r^{prime}_{i,t}$ is the deviation from the mean value, taken from the definition $(\sum_N r_{i,t})/N = \bar{r}_t$; while $\hat{r}^{prime}_{i,t}$ is the predicted deviation of the mean value, and $\bar{\hat{r}}_t$ is the predicted average monthly return, taken from the definition $(\sum_N \hat{r}_{i,t})/N = \bar{\hat{r}}_t$.

Using the above equations and taking the summation over stocks i on both sides,

$$\sum_N r_{i,t} = \sum_N r^{prime}_{i,t} + \sum_N \bar{r}_t; \quad \sum_N \hat{r}_{i,t} = \sum_N \hat{r}^{prime}_{i,t} + \sum_N \bar{\hat{r}}_t \tag{2}$$

$$\sum_N r_{i,t} = \sum_N r^{prime}_{i,t} + N\bar{r}_t; \quad \sum_N \hat{r}_{i,t} = \sum_N \hat{r}^{prime}_{i,t} + N\bar{\hat{r}}_t \tag{3}$$

$$\sum_N r^{prime}_{i,t} = 0; \sum_N \hat{r}^{prime}_{i,t} = 0 \tag{4}$$

$$\sum_N r^{prime}_{i,t} - \sum_N \hat{r}^{prime}_{i,t} = 0 \tag{5}$$

We can expand the overall loss in the following way:

$$\begin{aligned} L &= \frac{1}{NT}\sum_N \sum_T (r_{i,t} - \hat{r}_{i,t})^2 \\ &= \frac{1}{NT}\sum_N \sum_T (r^{prime}_{i,t} + \bar{r}_t - \hat{r}^{prime}_{i,t} - \bar{\hat{r}}_t)^2 \\ &= \frac{1}{NT}\sum_N \sum_T \left[(r^{prime}_{i,t} - \hat{r}^{prime}_{i,t}) + (\bar{r}_t - \bar{\hat{r}}_t)\right]^2 \end{aligned}$$

$$= \frac{1}{NT}\sum_N\sum_T\Big[(r_{i,t}^{prime} - \hat{r}_{i,t}^{prime})^2 + (\bar{r}_t - \hat{\bar{r}}_t)^2$$
$$+ 2(r_{i,t}^{prime} - \hat{r}_{i,t}^{prime})(\bar{r}_t - \hat{\bar{r}}_t)\Big]$$
$$= \frac{1}{NT}\sum_N\sum_T(r_{i,t}^{prime} - \hat{r}_{i,t}^{prime})^2 + \frac{1}{NT}\sum_N\sum_T(\bar{r}_t - \hat{\bar{r}}_t)^2$$
$$+ \frac{2}{NT}\sum_T(\bar{r}_t - \hat{\bar{r}}_t)\sum_N(r_{i,t}^{prime} - \hat{r}_{i,t}^{prime})$$
$$= \frac{1}{NT}\sum_N\sum_T(r_{i,t}^{prime} - \hat{r}_{i,t}^{prime})^2 + \frac{1}{T}\sum_T(\bar{r}_t - \hat{\bar{r}}_t)^2$$

Here, $\frac{1}{NT}\sum_N\sum_T(r_{i,t}^{prime} - \hat{r}_{i,t}^{prime})^2$ is the cross-sectional (CS) loss, and $\frac{1}{T}\sum_T(\bar{r}_t - \hat{\bar{r}}_t)^2$ is the time series (TS) loss. The third term in the decomposition, $\frac{2}{NT}\sum_T(\bar{r}_t - \hat{\bar{r}}_t)\sum_N(r_{i,t}^{prime} - \hat{r}_{i,t}^{prime})$, is zero by Equation (5).

Since the total loss can be decomposed into two parts that are independent of each other (TS and CS), this allows to use two separate models for the prediction of the risk premia. Using two separate models adds two important benefits to our predictive framework.

First, nonlinear ML models are well known for reaching a local optimum rather than the global optimum during optimization. When a single model f is trained, the best parameters minimizing the total loss function in the regression may not be optimal for both TS and CS loss components individually. Since these losses are independent, we can say there exists a separate set of parameters that are individually optimized for each part, hence the need to have two separate models. We therefore train two separate models; one is a TS model that predicts average monthly returns, and the other is a CS network, which captures the variation in the cross-section by predicting deviations from the mean value.

Second, it is also well known that stock market returns are non-stationary. However, an ML model is typically trained with the assumption that the out-of-sample data distribution will be the same as the in-sample data distribution. Due to the non-stationarity of stock market data, however, this assumption is violated. Removing the TS component from returns gives the remaining CS component a zero mean.† This makes it simpler to train deep networks for the CS model.

In our empirical analysis, we will demonstrate that the performance in predicting total returns improves due to the loss decomposition into TS and CS components.

2.2. *Deep neural networks using skip connections*

Another contribution of this work is to modify the architecture of the neural networks to permit the training of deeper neural networks.

It is widely observed that additional layers in neural networks lead to an increase in network capacity. However, this is true only up to a certain depth. After a certain number of layers, the performance of the network begins to degrade (He *et al.* 2016). This phenomenon is not due to overfitting, as would be the case in linear regression, because it is observed that the training performance also degrades.

It has been speculated that vanishing or exploding gradients are the reason for this degradation in performance. However, even after the addition of normalization layers that remove exploding and vanishing gradients, the degradation problem still exists. This problem illuminates the inability of existing deep learning optimizers to learn very deep models.

This degradation problem, however, can be solved by models that use skip connections. Inspired by ResNet (He *et al.* 2016) and DenseNet (Huang *et al.* 2017), we develop a model that adds a skip connection from every layer to the penultimate layer, as shown in Figure 1. Later in our results, we show that with this architecture, it is possible to train neural networks up to 10 layers without degradation, in fact achieving superior performance.

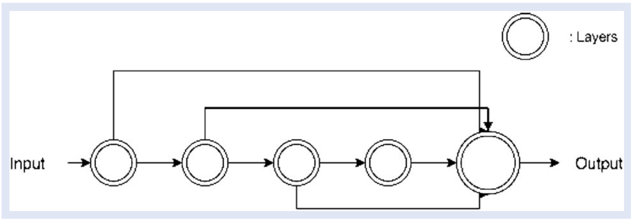

Figure 1. Illustration of a neural network with skip connections. The outputs from each layer are fed directly to the last layer, as shown in the figure. This modification allows the training of a deeper model with better performance.

3. Empirical analysis

3.1. *Data*

We obtained data for the monthly equity returns of the firms listed in the NYSE, NASDAQ, and AMEX exchanges from CRSP.‡ The sample period ranges from March 1957 to December 2016. This dataset consists of approximately 30,000 unique stocks, and contains returns for approximately 6,200 stocks every month. The data were divided into a training set consisting of 12 years of validation data, and testing data from 1987 to 2016. Using the expanding window method for training models, we recursively fitted the network for every testing year. For every testing year y_t, validation data were taken from years $[y_t - 12, y_t - 1]$, and training data were taken from $[1957, y_t - 13]$. We used the Treasury bill rate as a proxy for the risk-free rate from Welch and Goyal (2008). These design choices were inspired by Gu *et al.* (2018).

We used 94 firm-specific features in our model. Out of these, 61 are updated annually, 13 are updated quarterly,

† Demeaning the stock market data will not make the returns stationary, as there will still be finite variance in the out-of-sample returns. Only the mean will be zero.

‡ These values were calculated or derived from data from the Center for Research in Security Prices (CRSP) at the University of Chicago Booth School of Business.

and 20 are updated monthly. We also added 74 variables, corresponding to the first two digits of their Standard Industrial Classification (SIC) codes. These features are extracted from the CRSP and Compustat datasets. We obtained the processed version of this dataset from Gu *et al.* (2018).† All these features are preprocessed by ranking stock characteristics cross-sectionally month by month, and normalizing ranks to the interval $[-1, 1]$ following Kelly *et al.* (2017). This removes the effect of outliers from the training of the model. The macroeconomic data were obtained from Welch and Goyal (2008) and the Fred-MD (McCracken and Ng 2016) database.

3.2. *Time series model*

The TS network is trained to predict the average monthly returns. The cross-sectional average of monthly returns does not necessarily depend on individual stocks, since it is a single value each month. It is trained using macroeconomic variables as its input features, since the addition of extra features may lead to overfitting. Given the smaller amount of data, since the number of data points is equivalent to the number of months in the training data, ridge regression is used. In order to avoid overfitting, variable selection is done prior to fitting the ridge regression model. More detail about these parameters and models is included in Appendix A.1.

3.3. *Cross-sectional model*

The CS network captures the cross-sectional variance of returns, and takes 94 firm-specific features and 8 macroeconomic features as its input. The Kronecker product is used to examine the interactions between firm and macroeconomic features. The total input dimensions for this CS network are thus 94 * (8 + 1) + 74 SIC variables, or 920 in total. We use several different ML models to learn the cross-sectional distribution of the returns, including ridge regression, random forests, neural networks with hidden layers (NNx, where x is the number of hidden layers) and neural networks with skip connections (NNs). For all the neural network models, an ensemble of 10 networks are trained, due to the randomness of the initialization and optimization procedure. The out-of-sample R^2 is used to determine the goodness of fit for each model, calculated as follows:

$$R^2_{OOS_{CS}} = \frac{\sum_N \sum_T (r^{prime}_{i,t} - \hat{r}^{prime}_{i,t})^2}{\sum_N \sum_T (r^{prime}_{i,t})^2}$$

where $r^{prime}_{i,t}$ is the 'ground truth' deviation of the returns from the mean value, and $\hat{r}^{prime}_{i,t}$ is the deviation from mean value predicted by the model. More detail about these parameters and models is included in Appendix A.2.

† https://dachxiu.chicagobooth.edu/download/datashare.zip.

3.4. *Total returns model*

The total returns model is trained to predict the total excess returns (that is, the sum of the time series and cross-sectional returns) of the stocks in the dataset. This model serves as the baseline model for comparison to the independent component models. For the total returns model, the predicted returns were demeaned every month to obtain the cross-sectional spread, and the $R^2_{OOS_{CS}}$ was calculated.

3.5. *Model performance*

The performance of the CS models is shown in Table 1. This table also contains the $R^2_{OOS_{CS}}$ for the total returns (baseline) model. All the values of R^2 presented in this paper are given in percentage values. We include the out-of-sample R^2 obtained from our testing data, along with the training and validation R^2.

Our CS models have a much better performance than models trained on the overall returns. We also observe that performance of NN CS models becomes saturated at 7 layers, then decreases at 10 layers. At the same time, however, neural networks with skip connections can be trained up to 10 layers with the performance. The performance of networks with skip connections ultimately saturates at 15 layers, perhaps due to the exploding size of the last hidden layer. Overall, the neural network models perform much better than ridge regression and random forests. It should be noted that the deep model NN15s CS had the highest training R^2 over all CS models, which implies that such a deep model is possibly suffering from overfitting.

We also report the R^2 of high- and low- capitalization stocks for the out-of-sample and testing dataset. For every month, the thousand highest capitalization stocks (*top*1000) and the thousand lowest capitalization stocks (*bottom*1000) are filtered, and their R^2 is calculated. This is done to ensure that the model is not artificially limited to learning the inefficiencies present in low-capitalization stocks. We observe that the models do well for both groups of stocks.

The predictions made by the CS networks are added to the TS predictions to obtain the overall forecast return, $\hat{r}_{i,t}$. We also evaluate the overall predictive out-of-sample R^2 of the model, given by:

$$R^2_{OOS} = \frac{\sum_N \sum_T (r_{i,t} - \hat{r}_{i,t})^2}{\sum_N \sum_T (r_{i,t})^2}$$

where $r_{i,t}$ is the 'ground truth' return and $\hat{r}_{i,t}$ is the predicted value, given by $\hat{r}^{prime}_{i,t} + \bar{\hat{r}}_t$.

The overall R^2_{OOS} is presented in Table 2. We select a subset of our CS models to combine for the overall total return. The best TS model and the best CS model (NN10s) combine to give the best-performing model. We thus observe that loss decomposition and the use of a neural network with skip connections lead to a threefold increase in performance compared to the best performing model trained on total returns (the baseline model, NN3). We are also able to compare our model predictions to those found in Gu *et al.* (2018) because of its use of a similar dataset; we find that there is a 2.5x increase in R^2_{OOS}, as the best-performing model of Gu *et al.* (2018)

Table 1. Cross-sectional $R^2_{OOS_{CS}}$ of tested ML models. The highest values of R^2 are in bold. The first four models are trained on total returns, then the $R^2_{OOS_{CS}}$ was calculated by demeaning the monthly returns. The $R^2_{OOS_{CS}}$ of cross-sectional models are greater than those of comparable models trained on total returns (e.g. NN3 CS $>$ NN3). The performance of cross-sectional neural network models saturates at 10 layers (NN10 CS), while models trained with skip connections can be trained up to 10 layers (NN10s CS) without any degradation in performance. The R^2 values for the training and validation sets are not available for the non-cross-sectional models because they were not trained to predict the cross-sectional returns.

Model	$R^2_{train_{CS}}$	$R^2_{valid_{CS}}$	$R^2_{OOS_{CS}}$	Top 1000	Bottom 1000
Ridge	–	–	0.16	0.01	0.22
RF	–	–	0.07	0.10	0.09
NN1	–	–	0.19	0.12	0.28
NN3	–	–	0.28	0.13	0.37
Ridge CS	0.51	0.24	0.15	−0.14	0.23
RF CS	0.84	0.23	0.25	0.07	0.43
NN1 CS	0.93	0.52	0.26	0.12	0.36
NN3 CS	1.10	0.60	0.34	0.29	0.52
NN5 CS	1.04	0.58	0.34	0.25	0.50
NN7 CS	1.07	0.59	0.38	0.15	**0.52**
NN10 CS	0.55	0.34	0.24	0.21	0.32
NN5s CS	1.13	0.62	0.34	**0.36**	0.45
NN7s CS	1.11	0.63	0.37	0.27	0.47
NN10s CS	1.00	0.39	**0.40**	0.27	**0.52**
NN15s CS	1.16	0.62	0.38	0.16	0.50

Table 2. Overall R^2_{OOS}. The upper part of the table shows the performance of models trained on total returns. The lower part of the table shows the performance obtained by combining the time series (TS) & different cross-sectional (CS) models. The R^2_{OOS} of the best performing CS + TS model is three times better than the R^2_{OOS} of the best performing ML model trained on total returns. The training and validation R^2 is not available for CS+TS models because these models were not trained to predict the total return. The highest values of R^2 are in bold.

Model	$R^2_{OOS_{train}}$	$R^2_{OOS_{valid}}$	$R^2_{OOS_{test}}$	*Top* 1000_{test}	*Bottom* 1000_{test}
Ridge	0.45	0.23	0.31	0.00	0.41
RF	1.55	0.23	0.34	0.51	0.31
NN1	0.77	0.41	0.36	0.49	0.48
NN3	0.64	0.46	0.33	0.50	0.45
Models CS + TS	$R^2_{OOS_{train}}$	$R^2_{OOS_{valid}}$	$R^2_{OOS_{test}}$	*Top* 1000_{test}	*Bottom* 1000_{test}
Ridge	–	–	0.77	0.08	0.75
RF	–	–	0.99	**0.95**	1.07
NN1	–	–	0.85	0.46	0.91
NN5	–	–	0.97	0.81	1.08
NN7	–	–	0.99	0.80	1.08
NN5s	–	–	0.93	0.66	1.04
NN7s	–	–	1.00	0.88	1.04
NN10s	–	–	**1.01**	0.77	**1.11**

achieves an R^2_{OOS} of 0.40. A fair comparison to the other existing models in the literature is not possible due to the unavailability of similar datasets or implementation.

3.6. *Performance analysis*

In this section, we analyze the yearly and monthly performance of our best-performing model, TS + NN10s CS.

Figure 2 shows the yearly performance of this model over time. At the beginning, the model performs well. However, there are several years for which the R^2 value becomes negative (2000, 2005, 2006, 2007, 2011, and 2014). The poor performance of the model for some of these years may be due to higher volatility, which caused the market to behave in unusual ways, e.g. the dot-com bubble in 2000, the Great Recession of 2007, and the period of high market volatility during 2011.

Figure 3 shows the monthly performance of the model aggregated over our sample years. September is the month with the visibly worst performance. The exact reason for this behavior is not known, but it may be related to the well-known empirical 'September Effect,' in which the Standard and Poor's 500 has had an average decline of 1% since 1928 in September.† Similarly, Figure 3 shows the model's performance across different industry divisions, based on the underlying firm's SIC classification. As can be seen, firms classified in the mining industry perform visibly more poorly in our model compared to others.

† https://www.investopedia.com/ask/answers/06/septworstmonth.asp.

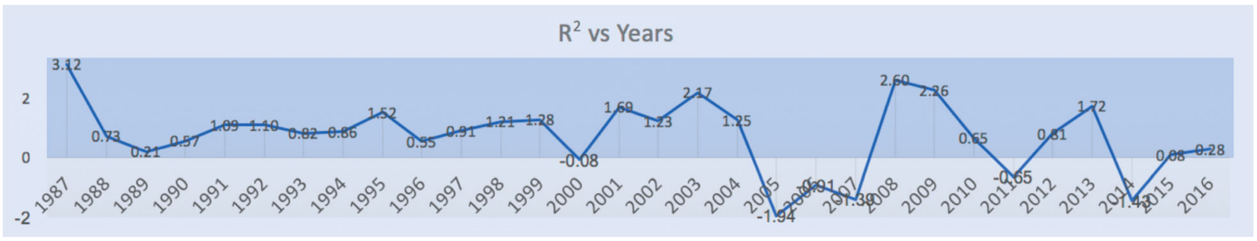

Figure 2. The yearly performance of our best-performing model, TS + NN10s CS. It consistently performs well apart from some exceptional years with negative R^2 (2000, 2005, 2006, 2007, 2011, and 2014). This poor performance may be due to higher volatility.

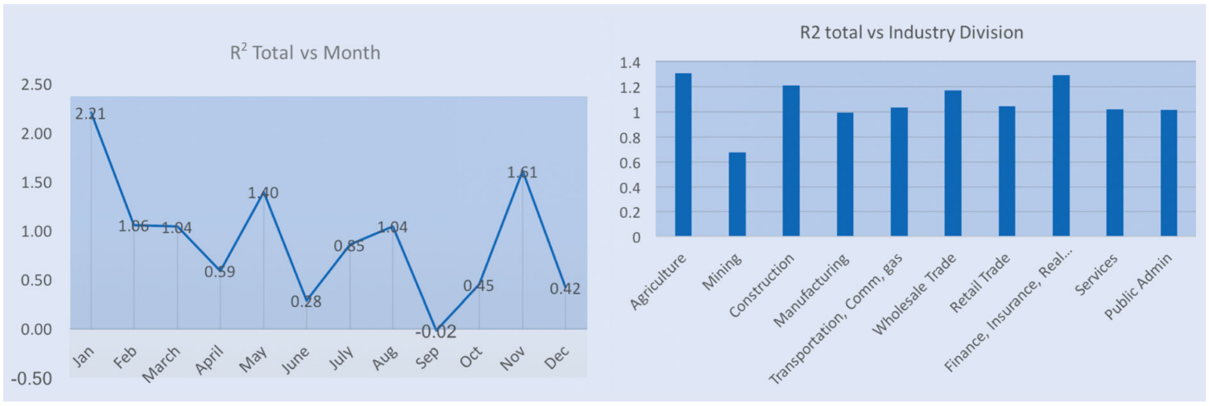

Figure 3. Left: Monthly performance of the best-performing model aggregated over years in the testing sample. September is the month with the worst performance. Right: Industry-wise performance of the model. Industry groups are categorized based on their SIC classification. The best-performing model performs slightly worse for companies in the mining industry group compared to those in other industry groups.

3.7. Machine learning portfolios

Risk premia forecasting can also be used for portfolio creation by investors. We generate test portfolios using different ML models to compare their effectiveness. At the start of every month, we obtain the model's predictions for the stocks in our dataset. These predictions are then sorted into deciles, and portfolios are created using a simple long–short heuristic. Stocks belonging to the top decile are bought, and those from the bottom decile are shorted.

Two types of portfolios, equally weighted and value-weighted (i.e. weighted by market capitalization), are constructed. We select the best-performing models in these categories for our comparison. For every portfolio, we evaluate a number of different metrics, as shown in Table 3. These metrics include the yearly Sharpe ratio, the value of the maximum drawdown, and the percentage turnover. For the task of portfolio creation, only the cross-sectional spread is required for decile sorting. We also calculate the log cumulative returns for the equally weighted and value-weighted portfolios, as presented in Figure 4.

It can be observed in Table 3 that equal-weighted portfolios have a larger Sharpe ratio and smaller drawdowns than value-weighted portfolios. Similarly, the value of log cumulative returns is smaller for value-weighted portfolios than for equal-weighted ones. This is due to the mean squared error, which is an equal-weighted loss function used in cross-sectional models for optimization. It necessarily does not give a greater importance to high-capitalization stocks, leading to this behavior. For equal-weighted portfolios, the performance

Table 3. Selected metrics for equal-weighted (upper table) and value-weighted (lower table) portfolios constructed using ML model prediction. The Sharpe ratio of the ten-layer neural network with skip connections (NN10s CS) has the maximum value. The performance of equal-weighted portfolios is consistently better than that of the value-weighted portfolios. Also, it can be observed that the turnover is high for all ML models. This is as expected as the ML models are not optimized for lower turnover. Bolded values indicate the best-performing model according to that metric.

Model	Sharpe Ratio	Max. Drawdown (%)	Turnover (%)
NN3	2.24	**5.56**	108.57
Ridge	1.37	23.87	128.06
RF	0.85	14.71	99.38
Ridge CS	1.36	22.59	131.34
RF CS	1.44	6.13	116.15
NN5 CS	2.27	6.46	108.83
NN5s CS	2.26	6.62	113.14
NN7 CS	2.21	7.62	111.66
NN7s CS	2.16	5.97	112.21
NN10 CS	1.94	5.85	116.03
NN10s CS	**2.43**	5.99	113.50
NN3	0.98	26.60	148.47
Ridge	0.50	12.47	134.40
RF	0.19	21.36	110.50
Ridge CS	0.50	15.44	139.09
RF CS	0.45	13.44	154.52
NN5 CS	1.13	**8.24**	145.80
NN5s CS	1.05	19.74	142.09
NN7 CS	1.08	19.13	143.41
NN7s CS	1.00	24.36	141.66
NN10 CS	0.93	9.57	147.36
NN10s CS	**1.16**	20.49	145.89

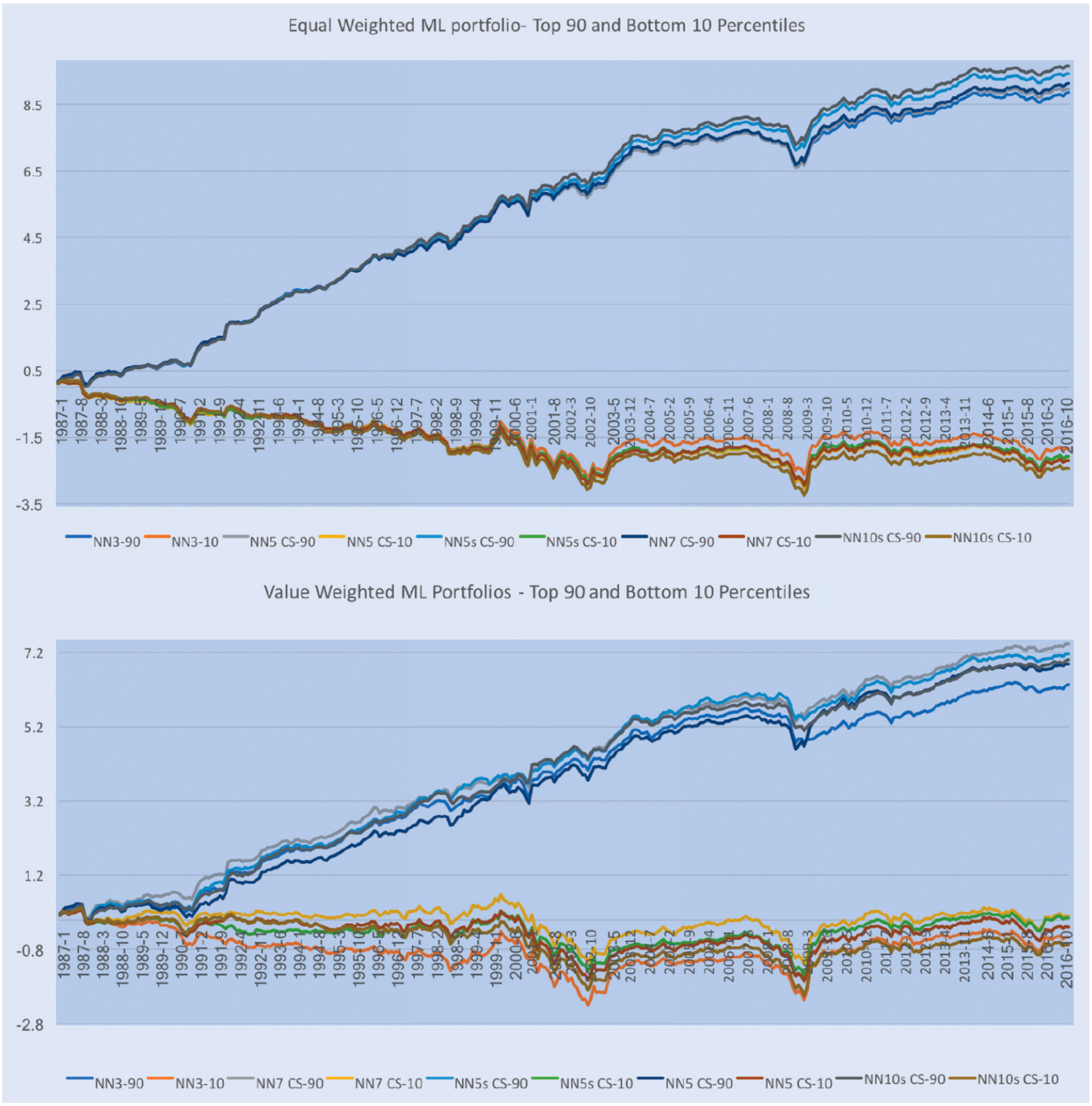

Figure 4. The monthly log cumulative returns of portfolios formed from the top and bottom decile stocks in our sample. We present a selection of representative models for each category of portfolio. The spread in the returns of these long–short portfolios is greater for equal-weighted portfolios than for value-weighted portfolios.

metrics are ranked in order of the model's R^2. The equal-weighted portfolio constructed from the **NN10s CS** model has the highest Sharpe ratio.

3.8. *Forecasting known portfolios*

Once trained, these ML models can also be used for forecasting the returns of known portfolios. A return forecast can be obtained as:

$$\hat{r}^p_{t+1} = \sum w_i * \hat{r}_{i,t+1}$$

where $\hat{r}^p_{t+1}$ is the forecast portfolio return, w_i are the portfolio weights, and $\hat{r}_{i,t+1}$ is the predicted return.

The known portfolios used in our analysis are double-sorted (3×2) book-to-market, momentum, investment, and size, constructed using a similar methodology to that used in the Fama–French data library.† The R^2_{pf} for different portfolios are given in Table 4, and are calculated as follows:

$$R^2_{pf} = \frac{\sum_T (r^p_t - \hat{r}^p_t)^2}{\sum_T (r^p_t - r^p_m)^2}$$

where r^p_t is the portfolio return at time t, $\hat{r}^p_t$ is the predicted value of the return, and r^p_m is the mean historic return.

Portfolio returns are more predictable than those of individual stocks due to the averaging of the noise present in stock returns. Hence, as expected, we observe a higher R^2 in our portfolio forecasts. From these results, the random forest model and the neural network with 5 hidden layers and skip connections perform best on these portfolios. In our earlier

† https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html.

Table 4. Forecast results of known portfolios by (R^2_{pf}). The highest values of R^2 are in bold. CMA is Conservative Minus Aggressive, UMD is Up Minus Down, HML is High Minus Low, and SMB is Small Minus Big. The performance of the models in the construction of the above portfolios is included in Table A1. More detail about portfolio construction is available at the Fama–French data library found at https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html.

Pf	NN3	Ridge CS	RF CS	NN7 CS	NN5s CS	NN10s CS
SMB	0.88	– 0.38	0.75	1.33	**2.20**	1.81
HML	1.03	2.04	1.38	2.00	**2.33**	1.70
CMA	1.16	– 0.99	– 0.13	1.96	**3.13**	2.78
UMD	0.03	2.45	**5.09**	– 0.81	– 1.38	– 1.36

results, we observe that RF and NN5s achieved higher R^2 values for high capitalization stocks (i.e. the top 1000). Since our portfolios are constructed using a value-weighted average, this may be the reason for the better performance of the RF and NN5s models.

4. Explainability

ML models are functionally black boxes, that is, they are often opaque in operation beyond their inputs and outputs. In this section, we try to find an interpretable relationship between the inputs and features that drives the output of these models.

We apply a method called the Local Interpretable Model-Agnostic Explanation (LIME) (Ribeiro *et al.* 2016) to explain the predictions of a black-box model. This is a model-agnostic technique that can approximate the decision boundary near a data point. The assumed linearity of the approximate decision boundary near this point makes it interpretable.

LIME explains the behavior of the model around the instance that is being analyzed or predicted. To determine which features contribute to the prediction of an input x, it perturbs input x and calculates the prediction for each perturbed input. Once the perturbed dataset is ready, a weighted

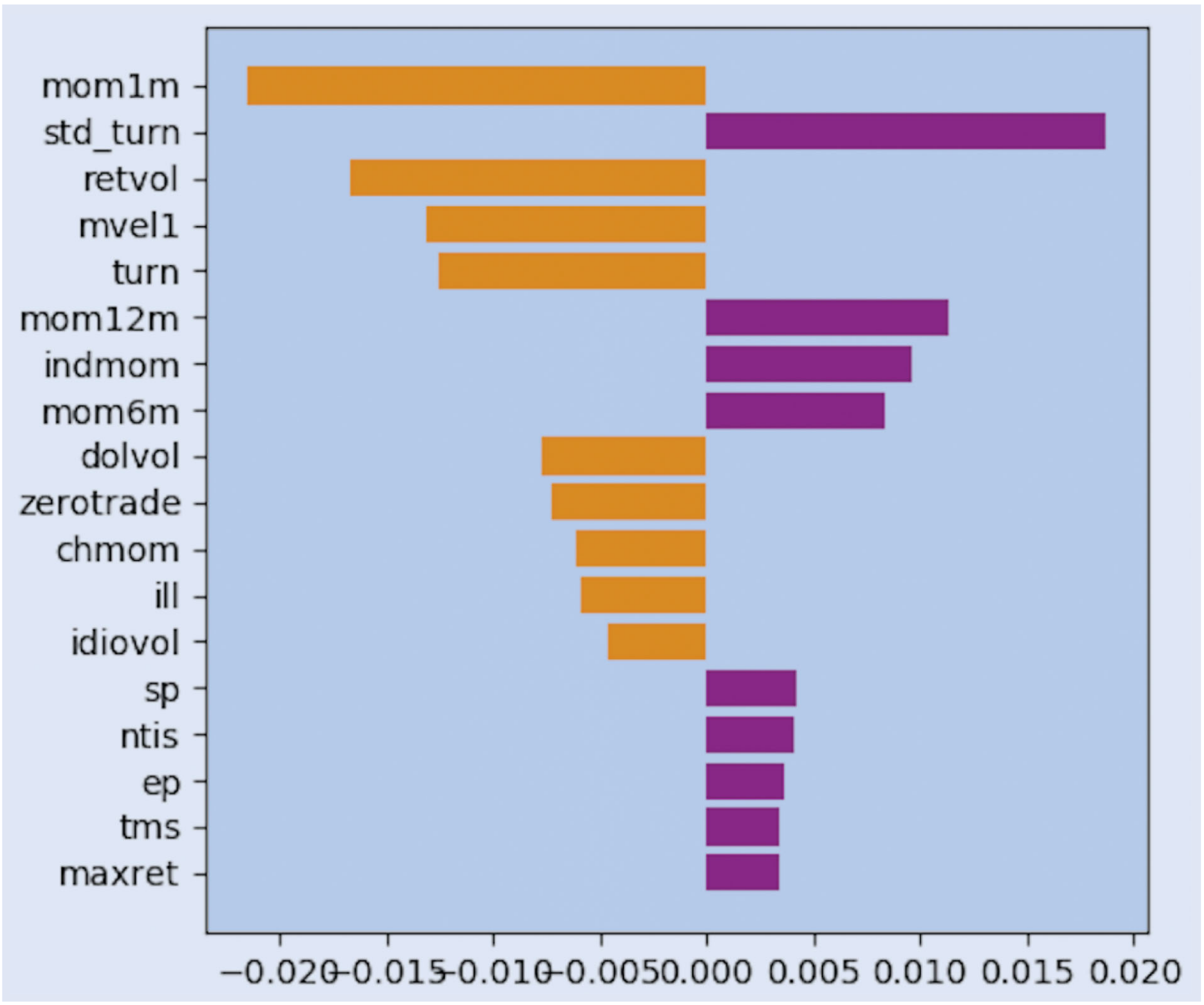

Figure 5. LIME feature importance for a single data point. Only the leading features are plotted. For this particular example, short-term reversion (*mom1m*), volatility of liquidity (*std_turn*), and returns volatility (*retvol*) are the three factors with the most contribution to the model. The *mom1m* factor contributes negatively to the risk premia. A brief description of these features is available in Appendix 3.

ridge regression model is trained in which its weights are decided by the distance of the perturbed data point to the original data point. The subsequent regression coefficients give an interpretable contribution of each feature in the prediction output for a given input x. The above procedure can be performed for any black-box model, making the LIME method model-agnostic.

Ridge regression models can easily be analyzed due to their linear nature. In comparison, random forests and deep neural networks suffer from their black box opacity. In this section, we analyze our best-performing model, the deep 10-layer neural network with skip connections, (**NN10s CS**), but a similar analysis can be performed for any other black-box model.

4.1. *Local feature importance*

For every test point, the importance or contribution of a feature can be calculated using LIME. An example of a sample data point's leading features and their importance is shown in Figure 5. For this particular example, short-term reversion ($mom1m$) is the highest contributing feature, and it is negatively correlated to the predicted cross-sectional returns. Only a few features out of all the features used in the training network contributed to the output, and their relative importance is shown. The feature contribution for all data points in the sample can be obtained in a similar way.

4.2. *Global feature importance*

Once the feature importance for each data point is obtained, it can be aggregated to obtain the global feature importance. We take the average of the absolute value of each feature to obtain the global feature importance. Figure 6 shows the global feature importance for the top 15 features out of 94 firm-specific and 8 macroeconomic features. Again, only a very few features contribute to the total output. Short-term reversion ($mom1m$), size ($mvel1$), and the volatility of returns ($retvol$) are the top three most important features.

4.3. *Importance histogram*

While the global feature analysis gives us the overall feature importance, it does not tell us anything about the relationship between the input features and the output variables. To determine this relationship, a histogram of the LIME coefficients (i.e. the feature importance) is plotted for all the data points. Histograms for some features are shown in Figure 7. The presence of histogram values on both sides of the y-axis (e.g. for the maximum return) implies that a particular factor conditionally affects returns positively or negatively. It demonstrates the nonlinear nature of deep models, which cannot be fully learned by linear models. The histogram of factors that are unimportant to the model are concentrated around 0, e.g. leverage. Most of the relationships learned by the model between factors and risk premia are consistent with the literature. We analyze some important features below:

- **mom1m:** Short-term reversion ($mom1m$) is known to be correlated negatively to returns (Cakici and Topyan 2014). The histogram of $mom1m$ shows large negative coefficients.
- **mom12m:** Momentum over 12 months ($mom12m$) is known to be correlated positively to returns

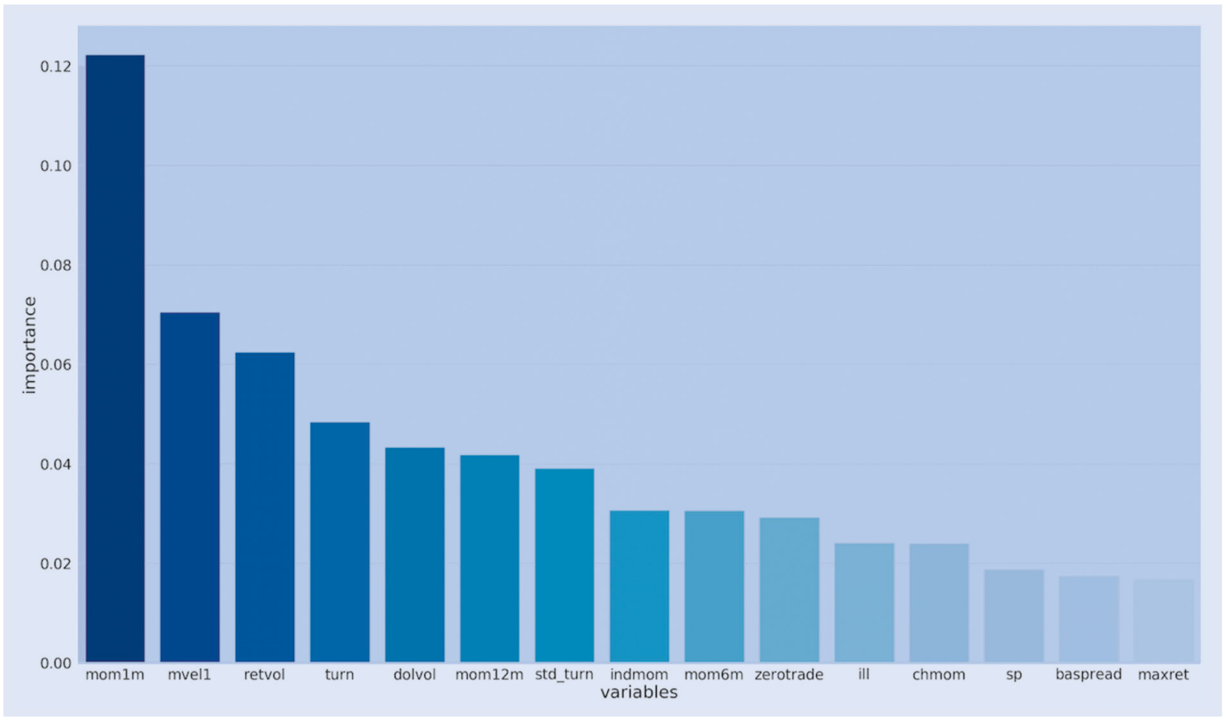

Figure 6. LIME feature importance aggregated over all data points. Short-term reversion, size, and return volatility are the three most important features. The top 15 features are plotted in the bar plot. Variables from left to right: *mom*1*m*, *mvel*1, *retvol*, *turn*, *dolvol*, *mom*12*m*, *std_turn*, *indmom*, *mom*6*m*, *zerotrade*, *ill*, *chmom*, *sp*, *baspread*, *maxret*. A description of features is available in Appendix 3.

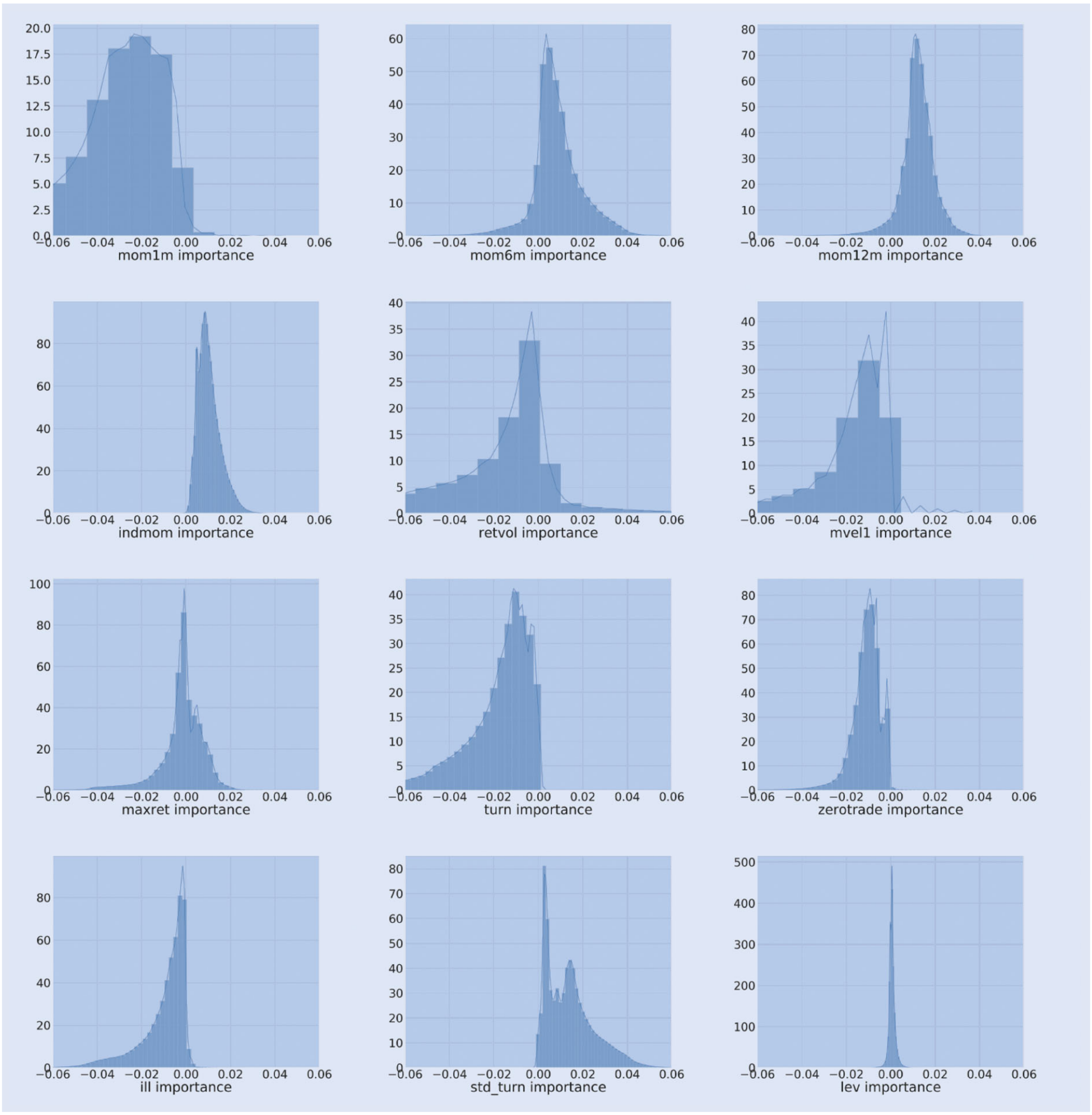

Figure 7. Histogram of LIME coefficients for selected features. It can be observed that for more important factors, histograms have longer tails, for example, *mom*1*m*, *size*, and *turnover*. The majority of the behavior of different features can be observed in return performance, e.g. the negative coefficients of the *mom*1*m* histogram imply that short-term reversion (*mom*1*m*) is negatively correlated to the output.

(Figelman 2007). The histogram of *mom*12*m* shows large positive coefficients.

- **indmom:** If a particular industry is doing well, companies belonging to that industry group are likely to be doing well. The histogram of industry momentum shows large positive coefficients.
- **mvel1 (size):** Small-cap stocks are known to outperform large-cap stocks, as evident from the Fama–French factor model (Fama and French 1993).
- **retvol:** Return volatility is known to be both positively and negatively related (Harvey and Lange 2015). The histogram obtained for *retvol* has values on both sides of the y-axis, indicating it can be both positively and negatively related to returns.
- **turn:** High turnover stocks are known to have smaller risk premia (Hu 1997). The histogram shows a negative relationship between turnover and returns.
- **std_turn:** The volatility of liquidity should be negatively related to the risk premia (Pereira and Zhang 2010). The histogram obtained from the model, however, shows a positive relationship. For this factor, the relationship learned by the ML model is not consistent with the existing literature. When the explanation gives this much importance to *std_turn*, the investor should be made aware of it and then decide whether to actually trust the model.

While we used LIME to analyze the deep 10-layer neural network with skip connections, (**NN10s CS**), any black-box model can be analyzed using LIME. Such an explanation of a neural network prediction gives important information about the essential variables that affect risk premia. It also gives confidence to any investor using the model regarding its fundamental validity.

5. Conclusion

Our results show that macroeconomic and firm-specific features can be used to forecast risk premia, using ML models to uncover these relationships. In this work, we trained a selection of ML models and observed that, in general, deep neural networks outperform linear ML models. We explored some of the problems associated with deep neural networks including performance degradation with increasing numbers of layers and lack of interpretability. Inspired by the deep learning literature, we modified the architecture of the network, leading to the training of deeper networks with better performance. We also modeled the risk premia using two different models, solving the non-stationary problem of the stock return data, and allowing us to decouple the learning process into two independent components, a time series component and a cross-sectional one, leading to a significant improvement in performance. The trained models were evaluated on a yearly basis, and on a dataset separated into high-capitalization versus low-capitalization stocks. We also constructed potential investment portfolios using the model's predictions and evaluated them by several different metrics.

We also attempted to illuminate the black box nature of these deep learning networks using preexisting methods in the literature to explain the model predictions. Investors can use this method of explanatory analysis to gain confidence in a model's predictions before making any investment decisions.

Overall, our approach captures the interaction between factors and risk premia. It can be extended to predict returns at multiple horizons and offers a robust framework that can be applied in various applications for portfolio construction, performance attribution, and risk management, and to better understand the factors related to risk premia.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

Cakici, N. and Topyan, K., Short-term reversal. In *Risk and Return in Asian Emerging Markets: A Practitioner's Guide*, pp. 91–103, 2014 (Palgrave Macmillan US: New York).

Chen, L., Pelger, M. and Zhu, J., Deep learning in asset pricing. Available at SSRN 3350138, 2019.

Fama, E.F. and French, K.R., Common risk factors in the returns on stocks and bonds. *J. Financ. Econ.*, 1993, **33**, 3–56.

Fama, E.F. and French, K.R., A five-factor asset pricing model. *J. Financ. Econ.*, 2015, **116**, 1–22.

Fama, E.F. and MacBeth, J.D., Risk, return, and equilibrium: Empirical tests. *J. Polit. Econ.*, 1973, **81**, 607–636.

Feng, G., He, J. and Polson, N.G., Deep learning for predicting asset returns. arXiv preprint arXiv:1804.09314, 2018.

Figelman, I., Stock return momentum and reversal. *J. Portf. Manag.*, 2007, **34**, 51–67.

Gu, S., Kelly, B. and Xiu, D., Empirical asset pricing via machine learning. Working Paper 25398, National Bureau of Economic Research, 2018.

Gu, S., Kelly, B. and Xiu, D., Autoencoder asset pricing models. *J. Econom.*, 2021, **222**, 429–450.

Harvey, A. and Lange, R.J., Modeling the interactions between volatility and returns. Cambridge Working Papers in Economics 1518, Faculty of Economics, University of Cambridge, 2015.

He, K., Zhang, X., Ren, S. and Sun, J., Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision*, pp. 630–645, 2016.

Hu, S.y., Trading turnover and expected stock returns: The trading frequency hypothesis and evidence from the Tokyo Stock Exchange. Available at SSRN 15133, 1997.

Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q., Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

Kelly, B., Pruitt, S. and Su, Y., Some characteristics are risk exposures, and the rest are irrelevant. Unpublished Manuscript, University of Chicago, 2017,

Lundberg, S.M. and Lee, S.I., A unified approach to interpreting model predictions. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

McCracken, M.W. and Ng, S., FRED-MD: A monthly database for macroeconomic research. *J. Bus. Econ. Stat.*, 2016, **34**, 574–589.

Nakagawa, K., Uchida, T. and Aoshima, T., Deep factor model. In *Proceedings of the ECML PKDD 2018 Workshops*, pp. 37–50, 2018.

Pereira, J.P. and Zhang, H.H., Stock returns and the volatility of liquidity. *J. Financ. Quant. Anal.*, 2010, **45**, 1077–1110.

Ribeiro, M.T., Singh, S. and Guestrin, C., 'Why should i trust you?': Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13–17, 2016*, pp. 1135–1144, 2016.

Simonyan, K., Vedaldi, A. and Zisserman, A., Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proceedings of the Workshop at International Conference on Learning Representations*, 2014.

Springenberg, J.T., Dosovitskiy, A., Brox, T. and Riedmiller, M., Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806, 2014.

Sundararajan, M., Taly, A. and Yan, Q., Axiomatic attribution for deep networks. In *Proceedings of the International Conference on Machine Learning*, pp. 3319–3328, 2017.

Welch, I. and Goyal, A., A comprehensive look at the empirical performance of equity premium prediction. *Rev. Financ. Stud.*, 2008, **21**, 1455–1508.

Zeiler, M.D. and Fergus, R., Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision*, pp. 818–833, 2014.

A. Appendix 1. Model Details

A.1. Time series model

The time series model is a ridge regression model. In it, feature selection is first performed using mutual information[†] between the individual features and output variables. In the feature selection procedure, the top 1% of features are selected, and a ridge regression model is trained using the selected features. It is trained to predict the monthly average value of returns, $\bar{r}_t$.

[†] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html.

A.2. *Cross-sectional model*

All the neural network models are implemented using keras-tensorflow library in Python. The ridge regression and random forest models are implemented in Matlab. The random seed was fixed to maintain the reproducibility of experiments. The grid search was performed over parameters like L1 norm penalty, learning rate. The Adam optimizer was used for the training model. Early stopping was used during the training. All other details related to parameters required for reproducibility of experiments can be obtained from the authors upon publication. The different kinds of models used in our analysis are listed below:

- **Ridge**: This is a ridge regression model. It predicts total returns $r_{t,i}$.
- **RF**: This is a random forest model. It predicts total returns $r_{t,i}$.
- **NN1**: This is a neural network with one hidden layer. The hidden layer dimension is 32. It is trained to predict total returns $r_{t,i}$.
- **NN3**: This is a neural network with 3 hidden layers. The hidden layer dimensions are 32,16 & 8. It predicts total returns $r_{t,i}$.
- **Ridge CS**: This is a ridge regression model. It predicts $r_{i,t}^{prime}$, the deviation from the mean value (cross-sectional returns).
- **RF CS**: This is a random forest model. It predicts $r_{i,t}^{prime}$, the deviation from the mean value (cross-sectional returns)
- **NN1 CS**: This is a neural network with one hidden layer. The hidden layer dimension is 32. It is trained to predict $r_{i,t}^{prime}$, the deviation from the mean value (cross-sectional returns).
- **NN3 CS**: This is a neural network with 3 hidden layers. The hidden layer dimensions are 32,16 & 8. It is trained to predict $r_{i,t}^{prime}$, the deviation from the mean value (cross-sectional returns).
- **NN5 CS**: This is a neural network with 5 hidden layers. The hidden layer dimensions are 32, 16, 8, 4 & 2. It is trained to predict $r_{i,t}^{prime}$,the deviation from the mean value (cross-sectional returns).
- **NN7 CS**: This is a neural network with 7 hidden layers. The hidden layer dimensions are 32, 32, 16, 16, 8, 4 & 2. It is trained to predict $r_{i,t}^{prime}$, the deviation from the mean value (cross-sectional returns).
- **NN10 CS**: This is a neural network with 10 hidden layers. The hidden layer dimensions are 32, 32, 16, 16, 8, 8, 4, 4, 2 & 2. It is trained to predict $r_{i,t}^{prime}$, the deviation from the mean value (cross-sectional returns).
- **NN5+s CS**: This is a neural network with five hidden layers with skip connections. The hidden layer dimensions are 32, 16, 8, 4 & 2. The features from all hidden layers are concatenated with the last hidden layer (skip connections) to make a vector of size 62 that is used as input for the final (output) layer. It is trained to predict $r_{i,t}^{prime}$, the deviation from the mean value (cross-sectional returns).
- **NN7+s CS**: This is a neural network with five hidden layers with skip connections. The hidden layer dimensions are 32, 32, 16, 16, 8, 4 & 2. The features from all hidden layers are concatenated with the last hidden layer (skip connections) to make a vector of size 110 that is used as input for the final (output) layer. It is trained to predict $r_{i,t}^{prime}$, the deviation from the mean value (cross-sectional returns).
- **NN10+s CS**: This is a neural network with 10 hidden layers with skip connections. The hidden layer dimensions are 32, 32, 16, 16, 8, 8, 4, 4, 2 & 2. The features from all hidden layers are concatenated with the last hidden layer (skip connections) to make a vector of size 124 that is used as input for the final (output) layer. It is trained to predict $r_{i,t}^{prime}$, the deviation from the mean value (cross-sectional returns).
- **NN15+s CS**: This is a neural network with 15 hidden layers with skip connections. The hidden layer dimensions are 32, 32, 32, 16, 16, 16, 8, 8, 8, 4, 4, 4, 2, 2 & 2. The features from all hidden layers are concatenated with the last hidden layer (skip connections) to make a vector of size 186 that is used as input for the final (output) layer. It is trained to predict $r_{i,t}^{prime}$, the deviation from the mean value (cross-sectional returns).

Appendix 2. Forecasting Known Portfolios

In this section, we provide the results of forecasting known double-sorted portfolios, including those sub-portfolios used in the construction of the double-sorted portfolios. The R_{pf}^2 values are given in Table A1. The best values for each portfolio are presented in bold.

Appendix 3. Feature Names

A brief description of selected important input features used in our analysis is given below:

- *mom*1*m* : 1-month momentum
- *mvel*1: Size
- *retvol*: Return volatility
- *turn*: Share turnover
- *dolvol* : Dollar trading volume
- *mom*12*m*: 12-month momentum
- *std_turn*: Volatility of liquidity (share turnover)
- *indmom* : Industry momentum
- *mom6m* : 6-month momentum
- *zerotrade*: Zero trading days
- *ill*: Illiquidity
- *chmom*: Change in 6-month momentum
- *SP*: Sales to price
- *baspread*: Bid–ask spread
- *maxret*: Maximum daily return
- *idiovol*: Idiosyncratic return volatility
- *ntis*: Net equity expansion
- *ep*: Earnings to price

Appendix 4. Computational Overhead

In this section, we compare the computational time and number of parameters used in each model. The computational times were computed for all the models on the same machine (10 core CPU). The run-times were computed for the prediction of a single data point, averaged over 3000 samples over 50 different runs. The results are included in Table A2. We can observe that the addition of extra layers clearly leads to a higher run-time. There is no difference in the number of parameters and run-time for the cross-sectional and the total returns models because they use exactly the same underlying model. The small microsecond run-time difference is statistically insignificant.

Table A1. Results (R^2_{pf}) of forecasting known double-sorted portfolios. The naming convention of these portfolios is as follows: in the abbreviation XY, X refers to the first variable sorted (**book-to-market, momentum, investment**) which can be Small (S), Middle (M), or Big (B), and Y refers to the second variable sorted (**size**) which can be Small (S) or Big (B). CMA indicates a Conservative Minus Aggressive portfolio, UMD Up Minus Down, HML High Minus Low, and SMB Small Minus Big. More details about portfolio construction are available in the Fama–French data library found at https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html. The best model values for each portfolio type are highlighted in bold.

Pf	NN3	Ridge CS	RF CS	NN7 CS	NN5s CS	NN10s CS
SMB	0.88	– 0.38	0.75	1.33	**2.20**	1.81
HML	1.03	2.04	1.38	2.00	**2.33**	1.70
BB	1.25	3.32	**4.38**	3.94	3.56	3.72
BS	0.57	**4.90**	4.52	4.28	4.08	4.30
MS	0.51	2.26	**2.89**	2.74	2.08	2.49
MB	0.24	– 1.63	**1.52**	0.81	– 0.54	0.60
SS	0.49	– 1.37	**1.76**	1.06	– 0.29	0.84
SB	– 0.09	– 1.96	**1.19**	0.48	– 0.88	0.27
CMA	1.16	– 0.99	– 0.13	1.96	**3.13**	2.78
BB	0.23	– 4.19	**1.18**	0.74	0.01	0.66
BS	0.05	2.25	**3.70**	3.09	2.38	2.67
MS	0.24	2.37	**3.02**	2.66	1.92	2.42
MB	**0.19**	– 3.65	– 0.41	– 0.68	– 1.44	– 0.66
SS	**3.57**	– 0.14	2.99	2.73	2.00	2.75
SB	**0.88**	– 2.94	0.28	0.02	– 0.74	0.03
UMD	0.03	2.45	**5.09**	– 0.81	– 1.38	– 1.36
BB	**0.44**	– 2.37	0.07	– 0.02	– 0.38	0.05
BS	5.77	**10.39**	9.52	9.98	9.33	9.35
MS	3.44	9.25	**9.39**	9.40	8.73	8.93
MB	– 0.48	– 3.33	**1.41**	0.63	– 0.02	0.46
SS	– 0.45	– 3.31	**1.43**	0.65	0.00	0.49
SB	– 0.48	– 3.34	**1.41**	0.63	– 0.02	0.46

Table A2. Deep model run-time in microseconds and number of parameters.

Model	Time (μS)	#Parameters
NN1	53	29,633
NN3	74	30,369
NN1 CS	53	29,633
NN3 CS	74	30,369
NN5 CS	75	30,433
NN7 CS	77	31,953
NN10 CS	80	32,107
NN5s CS	72	30,445
NN7s CS	79	32,061
NN10s CS	84	32,229
NN15s CS	101	33,965