Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics and Operations Research

Reimagining Asset Pricing: A Fresh Perspective using an Instance Ranking Formulation

Hub Knops (583402tk)

Supervisor:	Terri van der Zwan
Second assessor:	Alberto Quaini
Date final version:	2nd July 2023

Abstract

In this research, I compare the performance of instance ranking formulations with a more traditional return prediction model in asset pricing. I use stock data of American companies to train and test four different neural network model types: return regression, ranking regression, binary classification and decile classification. The return regression is a standard return prediction model. The ranking regression, binary classification and decile classification models allow for multiple levels of ranking aggregation to predict the relative ranking of future stock returns. A statistical analysis using the mean squared error based on the error in the ranking of stock returns and an adjusted Diebold-Mariano test leads to the conclusion that the instance ranking formulations are able to outperform the return prediction model. However, empirically the 1-layer return regression model outperforms any other model. This implies that the essence of asset pricing is not in the type of model but in the choice of covariates used for prediction.

1 Introduction

Asset pricing is a fundamental topic in finance that has received a lot of attention both in academic research and in practice. Accurately predicting asset returns can lead to huge economic gains for investors. The traditional approach to asset pricing involves developing prespecified theoretical models that capture underlying economic principles to predict future returns. However, after decades of academic research in this field, lots of different characteristics have been proposed as significant predictors of asset returns. Therefore, choosing the most impactful characteristics to use in a model is a difficult task. The rapid advances in machine learning techniques propose a solution to this problem. By using a highly flexible model structure, the interactions between different variables do not have to be specified before training. This paper explores the use of machine learning in asset pricing, including the various types of models and techniques that have been developed in this field. More specifically, I use neural networks to compare a traditional asset pricing model to multiple instance ranking formulations.

A big issue in asset pricing is the fact that returns show a low signal-to-noise ratio. Research has shown that aggregating data can lead to an improvement of this ratio. Therefore, I investigate the performance of neural network models using a ranking approach. Instead of predicting the realized stock returns, I construct models to predict the relative rankings of stock returns. I use three levels of aggregation: continuous ranking, decile ranking and binary ranking. A continuous ranking uses a scale from 1 to n, where n is the number of observations in a certain time period, thereby getting rid of the relative distance between stock returns. Decile ranking and binary ranking use 10 and 2 ordered classes respectively, indicating the quantile in which a stock return is located based on its ranking. In order to accommodate the different dependent variables, the continuous ranking model employs a regression structure, whereas the decile and binary ranking models employ a classification structure. I analyze the performance of the different model structures using neural networks containing 1-5 densely connected layers.

All models have different output types. Therefore, the models are analyzed using a model-specific measure and a standardized ranking measure. The model-specific measures include an out-of-sample R^2, precision and recall. To construct a standardized ranking, each model uses a scoring system. The predictions of each model are transformed into a single value, which can be used to rank the stock returns. This ranking is compared to the ranking of the realized returns using the mean squared error. To test for statistically significant differences between models, I use an adjusted Diebold-Mariano test. On top of this, I provide an empirical analysis of the models by comparing the out-of-sample realized returns of portfolios constructed using the predicted rankings.

To train and test the models, I use monthly data from companies in the AMEX, NYSE and Nasdaq. The dataset ranges from 1977 to 2021 and consists of monthly returns, 92 stock specific characteristics and the first two digits of the Standard Industrial Classification code. These characteristics are updated either monthly, quarterly or annually. On top of this, 8 macro-economic predictors proposed by Welch and Goyal (2007) are used. In order to make sure that the training data is up-to-date, every model is retrained after each year of data.

Although the instance ranking formulations show better performance using the statistical measures, empirically they do meet expectations. The simplest model, the 1-layer return regression neural network, shows the best empirical performance, while it is one of the worst in terms of statistical performance. This implies that the choice of variables is the most important part of asset pricing.

This article proceeds as follows. Section 2 presents a literature review containing a discussion of previous research concerning machine learning models in asset pricing as well as the deviations I make from the literature. I discuss the data that are used in Section 3. The methods and evaluation techniques are presented in Section 4. A complete overview of the results is given in Section 5 and finally I conclude and propose further research in Section 6.

2 Literature Review

Asset pricing has seen a great deal of attention over the past century. Traditional models include the capital asset pricing model (CAPM) proposed by Sharpe (1964) and the three-factor and five-factor models by Fama and French (1993). However, research by Bai, Philippon and Savov (2016) shows that price efficiency has increased significantly over the past 50 years, suggesting that the traditional models are not good enough anymore. This is also confirmed by Fama and French (2004) since they provide compelling evidence that the performance of the CAPM model is rather poor. Recent innovations in terms of computing power have led to the rise of more complex models. Gu, Kelly and Xiu (2020) provide a comparative analysis of different types of econometric models, ranging from traditional 3-factor linear models to complex non-linear models like regression trees and feed-forward neural networks. They find that the non-linear models perform a lot better and attribute this to the flexible setup and the room for complex interactions between variables. These findings are also confirmed by Chen, Pelger and Zhu (2021). In order to provide a good comparison to the paper by Gu et al. (2020), I reproduce their main results by constructing models based on the 5 different architectures of their feed-forward neural network.

A big issue in asset pricing is that stock returns tend to have an extremely low signal-to-noise ratio. Regression-based methods focus on predicting the numerical values of stock returns. Gu et al. (2020) provide evidence that this method is capable of outperforming the market. However, due to the low signal-to-noise ratio, the method does show poor predictive performance (Simin, 2008). Alberg and Lipton (2017) propose to solve this issue by predicting company fundamentals rather than stock returns since the signal-to-noise ratio is higher when forecasting fundamentals than in returns. Rasekhschaffe and Jones (2019) approach this by dividing the asset returns into categories since forecasting discrete variables limits the effect of outliers. By ranking stocks in outperformers and underperformers for each date in the training set instead of using the actual returns, the deteriorating effect of noise is reduced (Rasekhschaffe & Jones, 2019). Takeuchi (2013) finds that using a deep learning model to predict a binary classification (outperform vs underperform) shows statistically significant higher returns than the overall market portfolio. Another benefit of using categories is the fact that the probabilities can be used to calculate a measure of confidence in a prediction. This can be used in portfolio construction since high confidence in a low return might be more attractive than low confidence in a high return. To compare the performance of the return-based methods and ranking-based methods, I analyze the performance of the return model proposed by Gu et al. (2020) as well as other models using various amounts of aggregation in terms of ranking.

Werner (2020) introduces the problem of correctly ranking all instances as the hard ranking problem. They define a ranking rule as a function $r(X_i, X_j, s) = 2I_{(s(X_i) \geq s(X_j))} - 1$, where s is a scoring function and I is the indicator function. The ranking function returns 1 if the score of X_i is greater than or equal to the score of X_j and -1 vice versa. Therefore, a ranking can be constructed using scores based on the output of a model. Werner (2020) also distinguishes three types of instance ranking problems. If the dependent variable is binary-valued, the problem is referred to as a binary instance ranking problem. If the dependent variable has d classes, the problem turns into a d-partite instance ranking problem, and for continuously-valued responses, the problem is called a continuous instance ranking problem. In order to compare the different problem specifications, I implement a version of all three types of instance ranking problems.

3 Data

The dataset used in this paper is an updated version of the data used by Gu et al. (2020) ranging from 1977 to 2021. It is a private dataset owned by Van der Zwan (2023). It consists of monthly stock returns as well as 92 stock-specific characteristics, based on those of Green, Hand and Zhang (2017), and the first two digits of the Standard Industrial Classification (SIC) code of each company. For a detailed description of the stock-specific variables, I would like to refer to Green et al. (2017) or Gu et al. (2020). I also construct 8 macro-economic predictors using variables proposed by Welch and Goyal (2007) to allow for interactions between stock-specific and macro-economic variables.

3.1 Data transformation

In order to account for outliers and the magnitude of variables, I use a ranking system to transform the dataset. Each month, each characteristic is ranked cross-sectionally and mapped to the [-1, 1] interval. Afterward, I use the Kronecker product to allow for interactions between variables. This is defined by

$$z_{i,t} = x_t \otimes c_{i,t}, \tag{1}$$

where x_t denotes the vector of macro-economic predictors, including a constant, at time t and $c_{i,t}$ is the vector stock-specific characteristics of stock i at time t. Next to the interactions, I also construct 67 industry dummies based on the first two digits of the SIC code. This results in $92 \times (8 + 1) + 68 = 895$ stock-specific covariates used for prediction.

3.2 Model validation

Each model is constructed using a training dataset and a validation dataset. The parameters of the model are obtained using the training set. Afterward, the loss is calculated using the validation set to allow for a pseudo-out-of-sample analysis, which can be used to reduce the risk of overfitting. The specific implementation concerning the use of the validation set is explained in Section 4.3. For model evaluation, I compare 1 year of monthly out-of-sample predictions to the realized values. The datasets are constructed using an expanding moving window. They are initialized as 18 years of training, 12 years of validation and 1 year of out-of-sample testing. Each year, the training set is expanded by 1 year and the validation and test set are shifted forward 1 year. This results in 15 iterations and therefore 15 years of out-of-sample testing ranging from the start of 2007 to the end of 2021.

4 Methodology

I analyze the predictability of monthly stock returns using neural networks. Specifically, I compare the performance of models that predict future returns to models that predict the relative ranking of future stock returns. The analysis contains four neural networks of which two are regression types and two are classification types.

4.1 Neural networks

A neural network is a mathematical model which consists of neurons (also called nodes) that are linked together. The most simple form is the feed-forward neural network (FFN). This is a model that consists of multiple layers of neurons that only contain forward connections, as illustrated in Figure 1.

Neurons represent a value which is called the activation. The first layer of nodes denotes the vector of inputs. The activation of each node is based on a single characteristic or feature of the input data. The number of nodes in the first layer is determined by the dimension of the input data.

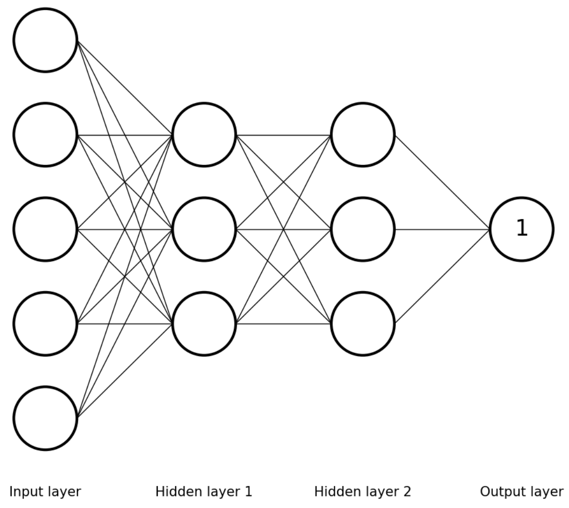
The input layer is fully connected to the first hidden layer. The activation of each node in the first hidden layer is determined by a weighted sum of the activations of the input layer and a specified activation function. Mathematically, this is represented as

$$x_k^{(l)} = f(\boldsymbol{\theta}_k^{(l)\intercal}\boldsymbol{x}^{(l-1)} + b_k^{(l)}), \tag{2}$$

where $x_k^{(l)}$ denotes the activation of node k in layer l, f represents the activation function, $\boldsymbol{\theta}_k^{(l)}$ is the vector of weights, $\boldsymbol{x}^{(l-1)} = (x_1^{(l-1)}, x_2^{(l-1)}, ...)$ is the vector of activations in layer $l-1$ and $b_k^{(l)}$ denotes the added bias. The first hidden layer is fully connected to the second hidden layer and so forth until the output layer is reached. The activation of the node(s) in the last layer represents the output of the model.

The choice of activation function can be very influential on the overall performance of the model. Lots of different factors determine the optimal choice. First of all, the intended application of the model is important. The range of values that the activation function can represent is important for the desired output. As an example, a simple image recognition model could have a desired output that consists of a probability indicating whether a certain image contains a cat. Since the output is a probability, its range needs to be $[0, 1]$. Therefore, in most probabilistic models the output layer will contain the sigmoid activation function. The activation function also determines the degree of non-linearity in the model. The most simple activation function is linear, denoted by $f(x) = x$, but there are also more complex functions that allow for non-linear interactions.

Figure 1 Structure of FFN



Note: Structure of a feed-forward neural network containing 5, 3, 3 and 1 nodes in the input layer, first hidden layer, second hidden layer and output layer respectively.

4.1.1 Training the model

Using an FFN to compute an output based on a certain input is not difficult. However, determining the optimal weights and biases which result in the best prediction is a complex problem. The algorithm I use is called Stochastic Gradient Descent (SGD).

Before starting the training, all weights and biases in the network are set to a random value. These values are optimized in an iterative manner. SGD assumes that there is a training dataset containing (x, y) pairs, where x is the input or explanatory variable and y is the output. This dataset is divided into batches, which are subsets of the entire dataset. For each iteration, a random batch of the dataset is selected. The input values of this batch are used to perform a forward pass through the network, resulting in a prediction $\hat{y}$. Using a loss function, we can calculate the error between the predicted $\hat{y}$ and the realized y. In order to obtain a better prediction we adjust the weights slightly to reduce the error. The direction of this adjustment is determined using backpropagation. This involves iteratively calculating the gradients of the loss function with respect to the parameters in the model. The gradients in the output layer of the model are simple to obtain since the parameters directly influence the loss function. Using the gradients of the output layer, the gradients of the subsequent layers are iteratively calculated. At layer l, the gradients are calculated by taking the weighted sum of the gradients in layer $l+1$ and applying the chain rule. The parameters of the models are updated by taking a small step in the direction opposite to the gradients, such that the loss function is minimized.

The step size is also called the learning rate of the model. A learning rate that is too high makes the algorithm overshoot the optimal parameters and a learning rate that is too low results in a very slow convergence. To make the optimization more efficient, I use the Adam optimizer. This optimizer employs learning rate shrinkage. The learning rate starts at an initial value and in each iteration, it is lowered. Therefore, the algorithm is more efficient at the beginning of optimization and slows down to avoid overshooting the optimal solution. Based on the performance during a few test runs using a learning rate of 0.1, 0.01 and 0.001, I decide to use an initial learning rate of 0.01.

By iteratively selecting new batches and performing forward and backward passes the weights and biases in the model are optimized. The iteration is stopped after a certain criterion is met. This might be a threshold in terms of loss, after a prespecified amount of iterations or any other suitable measure.

Due to the randomness in the initialization of the weights and biases and the choice of a batch in each iteration, each time you train a model the outcome is different. To reduce the impact of this randomness on the predictions I make use of ensemble learning. This means that each model is trained 5 times and the final prediction is the average of the prediction of the separate models.

4.2 Model structures

In principle, the idea of predicting asset returns is to use all the available data in period t to predict the returns of period $t+1$. Gu et al. (2020) have shown that the overall market can be outperformed by using this method. However, since asset returns show a very low signal-to-noise ratio, trying to predict returns is very much prone to overfitting on noisy data. On top of this, accurately predicting the actual return of a single stock is not a feasible goal. A general ranking of stocks indicating the likeliness to outperform the market seems much more useful. Therefore, to try to improve on the method used by Gu et al. (2020), I construct models that predict rankings of stocks, both in a regression structure and a classification structure. Each model type is analyzed using five different architectures, following the pyramid structure proposed by Masters (1993). The first architecture consists of a single hidden layer with 32 nodes, the second architecture consists of two hidden layers with 32 and 16 nodes respectively. This continues up to the fifth architecture which contains five layers with 32, 16, 8, 4 and 2 nodes respectively. All models are trained using a batch size of 10.000 observations and for a maximum of 100 epochs.

4.2.1 Return regression

In order to compare the performance of the ranking models to the traditional regression models, the first model I employ is similar to the neural network analysis in Gu et al. (2020). The model is a regression neural network used to predict stock returns. Figure 2a shows an illustration of the output layer of this network. A major difference compared to the model by Gu et al. (2020) is that I make use of the linear activation function instead of the rectified linear unit. Research by Crooijmans, Dekker, Van Steenis and Knops (2023) shows that the linear activation function performs better in asset pricing. The loss function used to train this model is the mean squared prediction error defined by

$$MSPE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2, \tag{3}$$

where N is the number of observations, y_i is the realized return of stock i and $\hat{y_i}$ is the predicted return of stock i.

4.2.2 Ranking regression

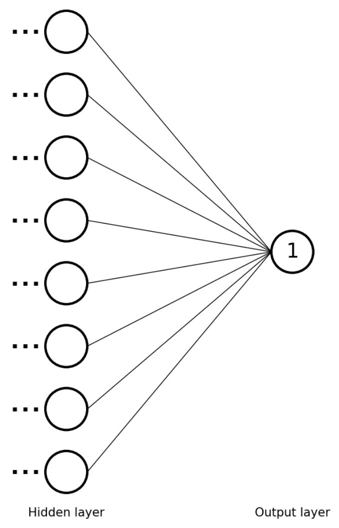
The second regression model is very similar to the first regression model. It only differs in terms of the dependent variable. Instead of predicting stock returns, it is trained to predict the ranking of stock returns, thereby turning the problem into an instance ranking problem. By taking the monthly stock returns and ranking them from 1 to n, where n indicates the number of observations in the month, I obtain the dependent variable. This results in the loss function specified by

$$MSPE(\boldsymbol{r}, \hat{\boldsymbol{r}}) = \frac{1}{N}\sum_{i=1}^{N}(r_i - \hat{r_i})^2, \tag{4}$$

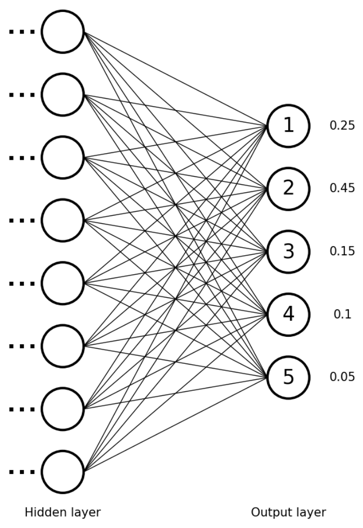
where N is the number of observations, r_i is the monthly ranking of the realized return of stock i and $\hat{r_i}$ is the predicted ranking of the return of stock i.

The output of this model is the predicted rank of a certain stock return. However, this output is not immediately usable for portfolio construction. Therefore I take the ranking of the monthly predicted rankings which results in the monthly output ranging from 1 to n, where n indicates the number of stocks in the predicted month. The highest ranking stock has the highest likelihood of performing well according to this model.

Figure 2 Illustration of the output of the FFN models



(a) Regression structure



(b) Classification structure

Note: Output structures of the FFN model. Figure 2a indicates the structure of a single numerical output denoting the asset return of period $t+1$. Figure 2b shows the structure where a classifier is implemented to indicate the relative ranking of returns in period $t+1$. The numbers behind the output layer indicate the probabilities of each class.

4.2.3 Binary ranking classification

The first classification model uses a binary dependent variable, which consists of 2 classes: outperform and underperform. By ranking the monthly stock returns and splitting them in the middle, each stock is classified as either outperforming or underperforming. I then implement one-hot encoding, which turns the classification of each stock return into a vector containing 2 values indicating the class it belongs to. As an example, a stock return in the upper 50% is turned into the vector $(0, 1)$, whereas a stock return in the lower 50% turns into $(1, 0)$. This allows for a classification structure using a logistic approach instead of a regression structure. In order to accommodate for this type of dependent variable, the output layer of the neural network contains 2 nodes. Similar to Figure 2b, the output nodes represent a probability of a stock return to be in a certain class. To accommodate for the probabilistic output, the final layer in the binary classification model uses the sigmoid activation function defined by

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{5}$$

The key feature of this activation function is the range being $[0, 1]$. This yields $\lim_{x\to\infty} f(x) = 1$ and $\lim_{x\to-\infty} f(x) = 0$. Therefore, high values of x result in a probability close to 1, whereas low values lead to a probability close to 0. The loss function used to train this model is the categorical cross-entropy (CE) which is specified as

$$CE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{K} (y_i \cdot \log(\hat{y}_i)), \tag{6}$$

where K is the number of classes, y_i is the i'th element of the true one-hot encoded vector and $\hat{y}_i$ is the i'th element of the predicted vector. By minimizing the CE loss function, the probabilities in the output layer are shifted toward the correct class.

The output of the model can be interpreted as two probabilities indicating the likelihood of the stock outperforming or underperforming the market. I analyze the output in two different ways. The first way is to classify the output using the highest of the two probabilities. A vector containing $[0.55, 0.45]$ would be classified as 'underperform'. However, this way of classifying does not provide much clarity for portfolio construction, since there are still many stocks in the same predicted class with no distinction between them. Therefore, I also use a scoring system to rank the stocks. Stocks are ranked using the $score(\boldsymbol{y}) = y_1 - y_0$, which is positive if the likelihood to outperform is higher than the likelihood to underperform and negative vice versa.

4.2.4 Decile ranking classification

The decile ranking classification model is similar to the binary ranking classification model since it uses the same structure of one-hot encoded vectors. However, as opposed to the binary ranking model, the vector now contains 10 elements, indicating 10% groups instead of 50% groups. Therefore, the structure changes slightly, since the output layer contains 10 nodes instead of 2. A limitation of the sigmoid activation function is that it provides isolated probabilities. This can result in an output that does not sum to 1 and therefore is not a probability distribution. When dealing with a binary output, this is not much of an issue, since there is no sense of relative distance between classes. However, if the output has more than 2 ordered dimensions, the relative distance between classes is important, and therefore the output should be normalized. In order to ensure that the output sums to 1, I make use of the softmax activation function, specified by

$$f(\boldsymbol{x}) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}. \tag{7}$$

Instead of using a single value in the output layer, the softmax function uses the entire vector of values as input. Therefore, it can divide by the sum of all values and thereby turn the output into a probability distribution.

A limitation of the loss function of the binary classification model is that there is no ordering. This leads to problems when there are more than 2 groups. If a realized stock return is in the upper 10%, the regular CE loss function punishes a prediction of the lowest 10% exactly the same as a prediction of the 80-90% group. To account for the relative distance between the classes, I use an ordinal version of the cross-entropy loss proposed by Hart (2022). The function is specified as

$$OCE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \left(\frac{|\arg\max_i(\boldsymbol{y}) - \arg\max_i(\hat{\boldsymbol{y}})|}{K-1} + 1 \right)^2 CE(\boldsymbol{y}, \hat{\boldsymbol{y}}), \tag{8}$$

where K denotes the number of classes, y is the true one-hot encoded vector, $\hat{y}$ is the predicted vector of probabilities, $\arg\max_i(\boldsymbol{x})$ indicates the index of the highest element in vector x and CE is the categorical cross entropy specified by equation 6. By adding the extra term, an incorrect prediction is punished based on the distance toward the correct class.

As for the binary classification model, the output is processed in two ways. The first way is by classifying the return using the highest probability in the predicted vector. The second way uses a scoring system, however, this system is more complex than the system of the binary classification model. I make use of the negative empirical skewness defined by

$$score(\boldsymbol{y}) = -\frac{\sum_{i=1}^{N}(y_i - \bar{y})^3}{(N-1)\sigma^3}, \tag{9}$$

where y_i is the i'th element of the predicted vector of probabilities, $\bar{y}$ denotes the mean of the elements in vector $\boldsymbol{y}$, N is the number of elements in the vector and σ is the standard deviation. A positive skewness indicates that the tail on the right of the mean is more pronounced and more mass of the distribution is on the left. Therefore, I add the minus sign, such that a stock is rated as positive when the distribution is skewed to the right and as negative when the distribution is skewed to the left.

4.3 Regularization techniques

A major issue when using neural networks is the risk of overfitting. Neural networks are very powerful models, which can be used to approximate any function. However, most datasets contain varying sizes of unpredictable errors. Therefore, perfectly modeling a function based on the training dataset results in a terrible performance in the test set. In order to overcome this issue, regularization techniques are applied.

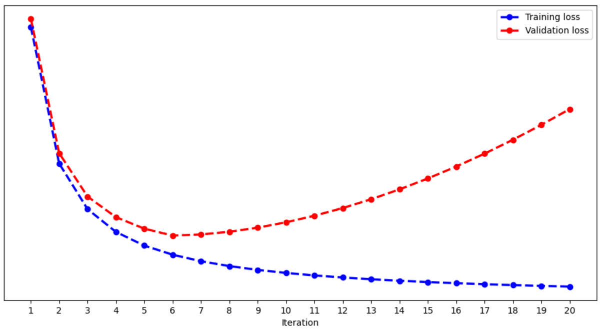
The first regularization method I implement is L1 penalization. Neural networks find optimal solutions by minimizing a loss function. However, doing so can lead to very complex models which account for every single variable in a very specific manner. L1 penalization is a method which penalizes the weight terms in the neural network by appending the loss function with a penalty term. This penalty is represented by

$$L1 = \lambda \cdot |w|, \tag{10}$$

where λ denotes the penalization factor and w indicates the weight vector. Using the absolute value, this penalty term encourages sparse solutions, which results in less complex models.

Another method of regularization is called early stopping. As mentioned in Section 4.1.1, neural networks are trained in an iterative manner. Each iteration, small changes are made to reduce the prediction error within the training set. The only issue is that the optimization of the parameters and the calculation of the loss is done using the same dataset. This can lead to overfitting on specific characteristics in the training set. The essential idea behind early stopping is to monitor the loss on a validation dataset. After each epoch, the model's prediction error on the validation set is calculated and whenever the validation loss starts rising, training is halted. Figure 3 shows an illustration of the progression of the training loss as well the validation loss. In practice, the progression of the loss is not as smooth. Therefore I apply a patience of 5 epochs. This means that each time a new minimum is found a counter is reset. Whenever the minimum is not breached for 5 consecutive epochs, the training is stopped.

Figure 3 Training versus validation loss



Note: Illustration of the progression of training loss and validation loss

The last regularization technique I apply is batch normalization. It works by normalizing each batch using its mean and standard deviation. Therefore, it changes the absolute scale of the characteristics used for prediction, while the relative scale is preserved. In regression models, absolute scale is important. Therefore, I only apply batch normalization to the classification models.

4.4 Performance measures

All of the models discussed in the previous sections have a different type of output and thus a different interpretation. This yields issues when trying to conduct a fair statistical comparison of their respective predictive performances. To try to accommodate a comparison, all models, except for the ranking regression, are analyzed using model-specific and ranking measures. The first of these measures is used to compare the different architectures within a specific output structure. The second measure is a ranking-based measure, which is used to compare across models. The output of each model is transformed into a single value such that a monthly ranking can be constructed. Since the output of the ranking regression is already a ranking, only the second measure is applied.

4.4.1 Return regression

The output of the return regression model consists of predictions of future returns. The predictions can be compared to realized returns, therefore, I make use of the out-of-sample R^2, which can be calculated as

$$R^2_{oos} = 1 - \frac{\sum_i y_i - \hat{y}_i}{\sum_i y_i^2}, \tag{11}$$

where $\hat{y}_i$ denotes the predicted return of stock i and y_i denotes the actual return of stock i. This R^2 measure deviates from the regular R^2, since the divisor is not demeaned. Therefore, the predictions are compared to a constant prediction of zero instead of the mean.

4.4.2 Classification models

Both the binary classification and the decile classification model returns a vector of probabilities indicating the likelihood that a stock return belongs to a certain quantile. In order to analyze the accuracy of the classification, I make use of the precision and recall defined by

$$precision = \frac{TP}{TP + FP} \tag{12}$$

and

$$recall = \frac{TP}{TP + FN}, \tag{13}$$

where TP indicates the number of true positives, FP indicates the number of false positives and FN indicates the number of false negatives. The two different criteria are used to differentiate between two types of accuracy. Precision is used to find the number of correct predictions divided by the total number of predictions in a certain class. On the other hand, the divisor of the recall measure is defined by the number of actual observations in a certain class. By using both the precision and recall, I can more accurately determine the weaknesses of the models.

4.4.3 Comparison between models

I use the scores based on the outputs of each model, as discussed in Section 4.2, to construct a monthly ranking. To compare the predicted ranking with the realized ranking, I employ a standardized mean squared error. Since the number of observations does not stay the same across all months in the dataset, I map the monthly rankings to the $[0, 1]$ interval thereby reducing the bias which would otherwise be introduced. The mean squared ranking error is defined by

$$MSRE = \frac{1}{n} \sum_{i=1}^{n} (r_i - \hat{r_i})^2, \tag{14}$$

where n is the number of observations, r_i denotes the scaled realized monthly ranking of stock i and $\hat{r_i}$ is the scaled prediction of the monthly ranking of stock i.

4.4.4 Diebold-Mariano test

To test for a statistically significant difference in predictions I use an adapted Diebold-Mariano test. The test-statistic is defined as $DM = \bar{d}/\hat{\sigma}_{\bar{d}}$, where

$$d_t = \frac{1}{n_t} \sum_{i}^{n_t} \left((e_{i,t}^{(1)})^2 - (e_{i,t}^{(2)})^2 \right), \tag{15}$$

$e_{i,t}^{(1)}$ and $e_{i,t}^{(2)}$ denote the prediction errors for stock i in month t for the two different models and n_t is the number of observations in period t. d_t is calculated for every period t in the testing sample. $\bar{d}$ and $\hat{\sigma}_{\bar{d}}$ are the mean and the Newey-West standard error of d_t over the testing sample. This statistic uses a cross-sectional average prediction error instead of the individual prediction errors of each stock. This is done to satisfy the regularity conditions for asymptotic normality, which results in more accurate p-values.

4.5 Variable importance

Another interest when predicting stock returns is the impact of individual predictors on the overall performance of the model. Although causal analysis is very difficult to conduct using neural networks, variable importance is measurable. Normally, when using a model to predict, all variables are given as input to the model. However, by setting the values of a specific variable to 0 and using that as input, I can analyze the effect on the loss function and therefore observe the impact of a single variable on the model's predictive performance.

My specific implementation is similar to the analysis performed by Gu et al. (2020). For each year of testing, I use the trained models to predict future output and calculate the increase of the loss function due to setting all values of a single predictor to 0. I average the increase in loss across the years, resulting in a single importance measure for each predictor, which I use to rank the variables.

5 Results

In this section, I provide a comprehensive overview of the findings based on the neural network models described in Section 4. The results are organized into subsections, each focusing on a specific model structure. I provide an overview of the return regression, binary classification and decile classification models, followed by a general comparison including a statistical analysis, an empirical analysis and the variable importance. The ranking regression only appears in the comparison, since there is no model-specific measure.

5.1 Return regression

Table 1 shows the comparison in terms of out-of-sample predictive R^2 of the five neural network models. Based on the first line of Table 1, it is clear that the 3-layer feed-forward neural network outperforms the others since it has the highest out-of-sample R^2 at 0.43. This indicates that a deeper network does not necessarily lead to greater out-of-sample performance. On top of this, too shallow or too deep learning increases the variance of the performance of separate models.

Table 2 shows the performance of the separate models used for the ensemble learning. The range of the predictive R^2 of the 3-layer model is around 0.2, whereas the other models show a larger range. The 1-layer model has a range of 1.22 and it even contains negative values. Therefore, the 3-layer models performs better and is more consistent than the other models.

Although this conclusion can be justified for the entire set of stocks, when we assess different subgroups, it does not hold. By ranking the stocks using the market value at the start of each month, I assess the performance on high market cap stocks and low market cap stocks. Lines 2 and 3 of Table 1 show the out-of-sample predictive R^2 of the highest and lowest 500 stocks for each month in terms of market value. The performance of the model when predicting low-cap stocks follows a similar pattern to the performance using the entire sample, since it peaks at a 3-layer neural network. However, the predictive performance on high-cap stocks shows that deeper networks perform better. This gives reason to believe that predicting returns of high-cap stocks involves more complex interactions than predicting returns of low-cap stocks.

Table 1 Monthly out-of-sample R^2 return FFN

	FFN1	FFN2	FFN3	FFN4	FFN5
All	0.18	0.25	0.43	0.32	0.29
Top 500	0.07	0.41	0.64	0.77	0.84
Bottom 500	0.24	0.25	0.40	0.27	0.20

Note: Table containing the monthly out-of-sample R^2 in percentages of the return neural networks, as described in Section 4.4.1. Bottom 500 and top 500 indicate the out-of-sample R^2 of the 500 lowest and highest value companies of each month.

Table 2 Out-of-sample R^2 separate return FFN models

	FFN1	FFN2	FFN3	FFN4	FFN5
1	-0.93	0.19	0.35	0.02	0.26
2	-0.51	0.29	0.41	0.14	0.44
3	-1.07	0.16	0.25	0.42	0.26
4	-0.29	0.04	0.21	0.08	-0.16
5	0.15	0.14	0.40	0.33	0.06
Range	1.22	0.25	0.20	0.40	0.60

Note: Table containing the out-of-sample R^2 in percentages of the separate return neural networks that combine into the ensemble learning of Table 1.

The results in Table 1 show that, while using the entire sample, the 3-layer network has the best performance. In order to statistically analyze this notion, I evaluate the Diebold-Mariano test statistics, shown in Table 3. At a 10% significance level, the 3-layer network statistically outperforms the 2-layer and 5-layer networks. Although the rest of the models show no statistically significant results, it is probable that the 3-layer network is optimal.

On top of the analysis of monthly returns, I also analyze the aggregated yearly returns. For each year in the testing set, I use twelve consecutive (predicted or realized) returns from January to December to compute the yearly returns as $r_i^y = (\prod_{j=1}^{12}(1 + r_{i,j}^y)) - 1$, where r_i^y is the yearly return of stock i in year y and $r_{i,j}^y$ is the monthly return of stock i in month j of year y. Table 4

Table 3 Diebold-Mariano statistics return FFN

	FFN1	FFN2	FFN3	FFN4
FFN2	-0.24			
FFN3	-0.96	-2.45(**)		
FFN4	-0.57	-1.18	1.03	
FFN5	-0.42	-1.06	1.78(*)	0.55

Note: Table reporting the Diebold-Mariano test statistics of the return neural networks, as described in Section 4.4.4. A positive value indicates that the model in the column outperforms the model in the row. (**) and (*) indicate significance at 5% and 10% levels respectively

shows the out-of-sample predictive R^2 on a yearly basis. The patterns are similar to the monthly returns, where the 3-layer network performs best on the entire dataset and the low-cap stocks. Deeper learning performs better when predicting the returns of high-cap stocks.

Table 4 Yearly out-of-sample predictive R^2 return FFN

	FFN1	FFN2	FFN3	FFN4	FFN5
All	1.79	1.32	2.34	1.74	1.67
Top 500	0.42	3.44	4.79	6.30	6.29
Bottom 500	2.51	1.50	2.38	1.64	1.48

Note: Table containing the yearly out-of-sample R^2 in percentages of the return neural networks. Bottom 500 and top 500 indicate the out-of-sample R^2 of the 500 lowest and highest value companies at the start of each year.

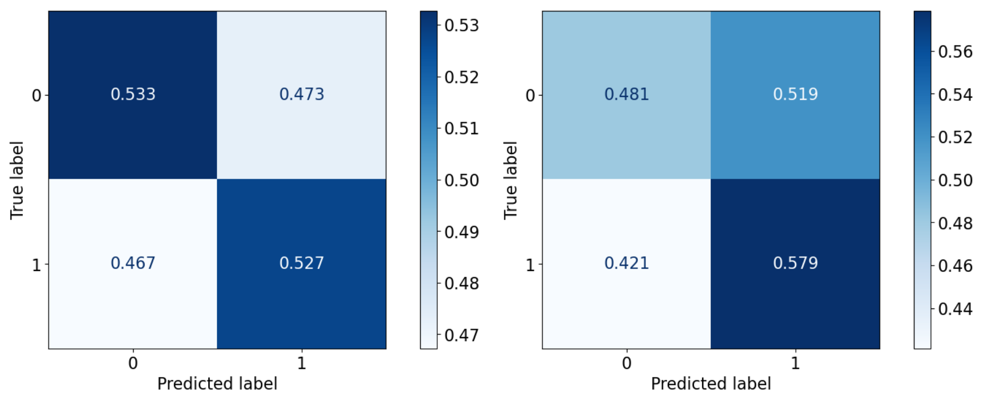
5.2 Binary classification

I analyze the binary classification models using the precision and recall, as specified in Section 4.4.2. The number of layers in the model does not have much influence on the predictive performance, therefore I discuss the 3-layer binary classification model. Conclusions obtained from the 3-layer model can be generalized to the 1, 2, 4 and 5-layer models. To illustrate the generalization, confusion matrices of the other models are shown in the Appendix (Figure A.1).

Figure 4 shows a graphical representation of the precision and recall of the 3-layer binary classification model. The most important feature in these figures is the diagonal from the upper left corner to the bottom right corner. This indicates the percentage of predictions that are in the correct class. In the precision matrix, the percentage indicates the number of correct predictions out of the total predictions of a certain class. On the other hand, the recall matrix indicates the percentage of correct predictions out of the total number of observations in a certain class. Based on Figure 4a, around 53% of the predicted classifications is correct. This is similar to results obtained by Takeuchi (2013).

Figure 4b shows that the model prefers predicting high returns since the percentage of 'outperform' predictions is higher than the percentage of 'underperform' predictions regardless of the correct class. 51.9% of the times that the correct label is 'underperform', the model predicts 'outperform'.

Figure 4 Confusion matrices 3-layer binary classification model



(a) Precision **(b)** Recall

Note: Confusion matrices of the binary classification model. The statistics in Figure 4a and Figure 4b indicate the precision and recall of the predictions made by the model, as specified in Section 4.4.2.

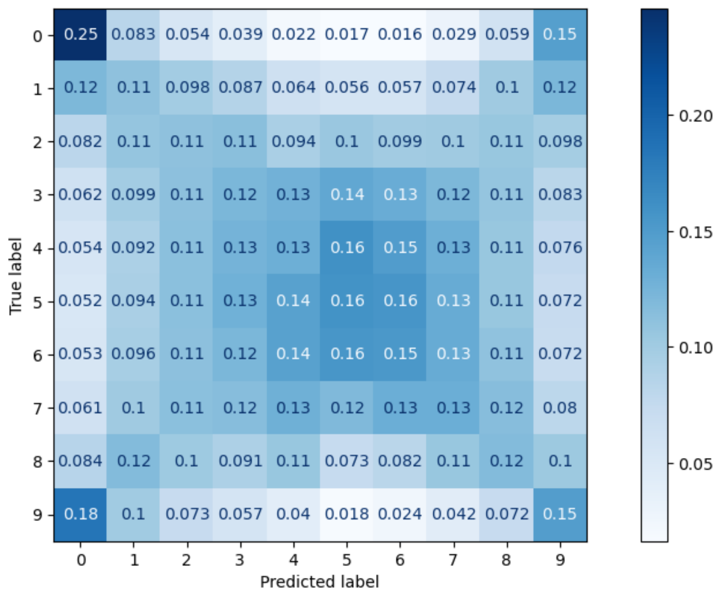
5.3 Decile classification

As with the binary classification, the confusion matrices are very similar across the different amounts of layers in the model. Figures 5 and 6 show the precision and recall of the 3-layer ranking classification model, the rest of the confusion matrices can be found in the appendix (Figure A.2). The overall accuracy of the models is around 14.5% for all numbers of layers. This suggests that they perform better than a model using random predictions since its accuracy is expected to be around 10%.

A clearly striking result is the fact that the darker colors in the precision matrix (Figure 5) show a cross pattern. On the extreme sides of the predictions, the distributions of the correct labels are heavily tailed, whereas the distributions are more centered in the middle of predictions. This pattern indicates that the model is able to distinguish between high-volatility stocks and low-volatility stocks. However, it lacks the power to determine whether the stock moves up or down.

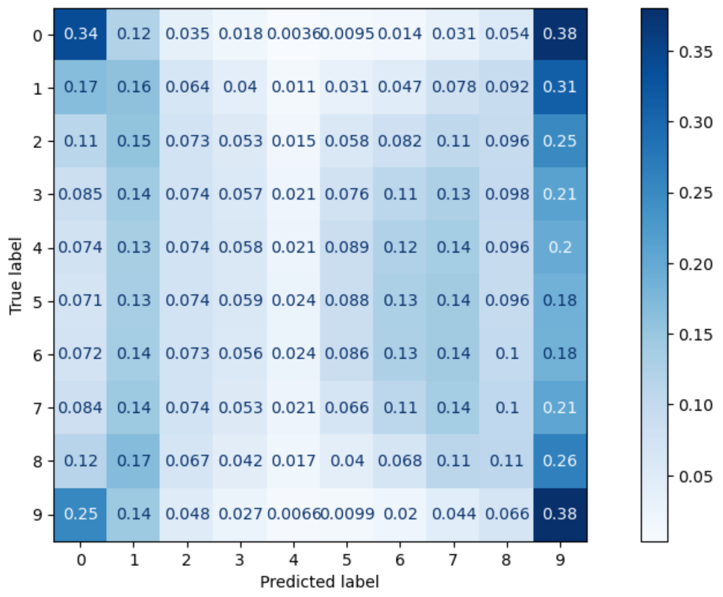
Another interesting observation is the high prevalence of predictions in the highest class. Figure 6 shows that the number of predictions of class 9 is disproportionately high since the number of 9-predictions compared to the total number of observations in a certain class is the highest for every observed class. A similar yet less pronounced case can be made for the lowest classes, indicating the overrepresentation of extreme predictions. This issue is likely caused by the specification of the loss function. The ordinal cross-entropy, as defined in equation 8, looks at the predicted probability of the correct class and punishes based on the distance between the predicted class and the correct class. The average distance between a predicted class and a middle class is much lower than the average distance between a predicted class and an extreme class. Therefore, in order to minimize the loss, high priority is given to extreme classes. Although the addition of distance to the loss function makes sense, an improved approach might be to add an extra parameter to the loss function to regulate the punishment of the distance.

Figure 5 Precision 3-layer decile classification model



Note: Confusion matrix of the decile classification model. The statistics indicate the precision of the model, as specified in Section 4.4.2

Figure 6 Recall 3-layer decile classification model



Note: Confusion matrix of the decile classification model. The statistics indicate the recall of the model, as specified in Section 4.4.2

5.4 Comparison

To compare the performance of the different models, I make use of the mean squared ranking error (MSRE), as specified in Section 4.4.3. A lower MSRE indicates less error in terms of predicted ranking. Table 5 shows that the three instance ranking formulations have a lower MSRE than the return regression. This is as expected since they are specifically designed to predict rankings. Similar to the conclusions from the model-specific measures, there seems to be no major difference in performance across the different numbers of layers within the ranking formulations, since the MSRE shows only a very slight range of values.

An interesting notion is the fact that a Monte-Carlo simulation, that randomly ranks the stock returns, shows an MSRE of around 0.167. This suggests that the return regression is unable to perform better than a randomized model in terms of ranking.

Table 5 Mean Squared Ranking Error

MSRE	1 layer	2 layer	3 layer	4 layer	5 layer	Range
Return	**0.1664**	0.1686	0.1679	0.1689	0.1691	0.027
Ranking	0.1573	0.1572	0.1570	0.1571	**0.1570**	0.003
Bin. Clas.	**0.1545**	0.1546	0.1548	0.1548	0.1547	0.003
Dec. Clas.	0.1545	**0.1543**	0.1544	0.1544	0.1546	0.003

Note: Table containing the mean squared errors based on the ranking as described in Section 4.4.3.

Using the ranking error, I calculate the Diebold-Mariano (DM) test statistics. Table 6 shows the DM-statistics for the 3-layer variants of the different model structures. A full table containing all the DM-statistics can be found in the appendix (Table A.1). In line with the MSRE, the binary classification and decile classification models significantly outperform the other models.

Table 6 Diebold-Mariano statistics ranking error

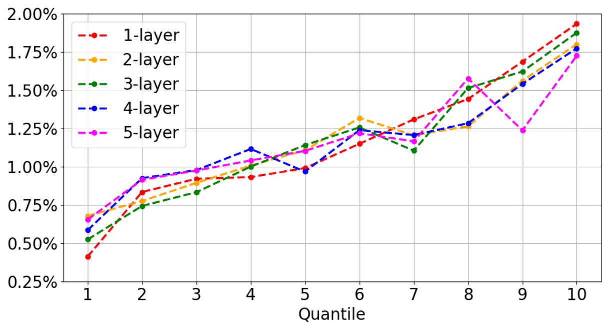
	Return 3	Ranking 3	Bin. Clas. 3
Ranking 3	-6.51(**)		
Bin. Clas. 3	-6.60(**)	-3.05(**)	
Dec. Clas. 3	-5.73(**)	-2.32(**)	-0.30

Note: Table containing the Diebold-Mariano test statistics of the 3-layer neural networks using the return regression, ranking regression, binary classification and decile classification structures. A positive value indicates that the model in the column outperforms the model in the row. (**) indicates significance at a 5% level.

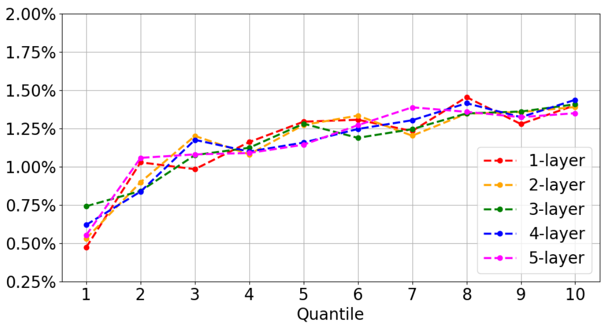
5.4.1 Empirical performance

The empirical performance of the models shows a major contrast to the MSRE. Figure 7 shows the monthly average realized returns of equal-weighted portfolios constructed using 10% quantiles based on the predicted rankings of each model. The return regression (Figure 7a) shows a clear upward pattern, indicating a correlation between the predicted ranking and the realized returns. The ranking regression (Figure 7b) shows a similar but less pronounced pattern. On the other hand, the classification models (Figures 7c and 7d) show a nearly flat, sometimes even downward, line. This indicates that the empirical performance of the classification models is underwhelming. Where the highest quantile of the 1-layer return regression model has an average return of nearly 2%, the returns of the highest quantile of the decile classification models are barely higher than 1%.

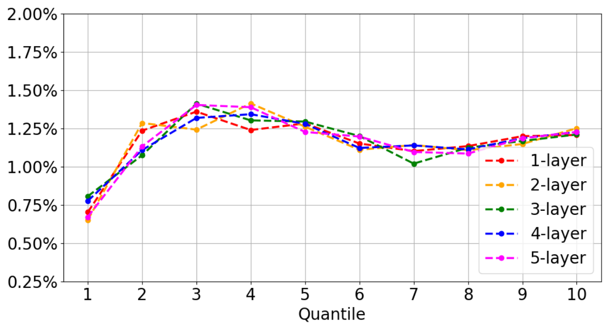
Figure 7 Empirical performance



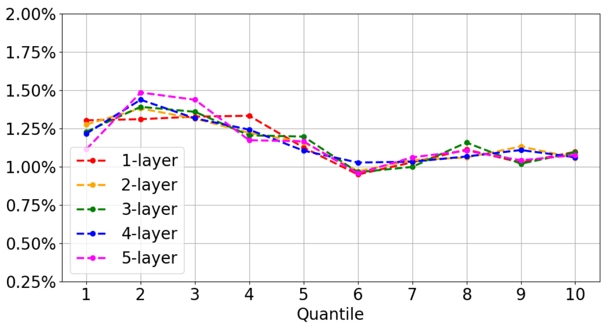
(a) Return regression



(b) Ranking regression



(c) Binary classification



(d) Decile classification

Note: Average realized monthly returns of 10% quantiles based on the prediction-sorted rankings of stocks within each model.

5.4.2 Variable importance

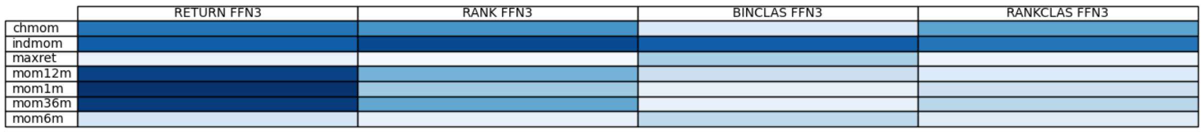
The variation in the importance of variables between the different model types is limited. As an illustration, Figure A.4 in the appendix shows the variable importance across the 1-5 layer return regressions. However, across the model types, there are major differences in the importance of variables. Therefore, I analyze the 3-layer variants of each model type.

I specify four groups containing a selection of the variables: momentum, risk, liquidity and fundamentals. The table containing all variables can be found in the appendix (Figure A.3). To visualize the results, each figure contains a table with colors. The darker the color, the higher the importance of the variable.

The first group consists of momentum variables. This includes recent price trends in the stock returns and industry trends. Figure 8 shows the importance of the momentum variables. Interestingly, the importance of momentum variables is very high for the return regression. They are a little less important in the ranking regression, and the classification models seem to be almost uninfluenced by momentum. Given that the return regression showed the best empirical performance, this suggests that momentum variables are highly important for predicting future returns.

Figure 9 shows the importance of the market beta (beta), squared beta (betasq), volatility

Figure 8 Momentum variables

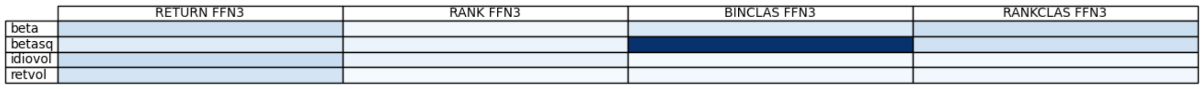


Note: Importance of the momentum variables.

of returns (retvol) and the volatility of idiosyncratic returns (idiovol). In contrast to the results obtained by Gu et al. (2020), risk measures seem to play no major role in asset pricing. This is likely due to the difference in the dataset used. The data used for testing in the paper by Gu et al. (2020) ranges from 1987 to 2016, whereas the testing set in this paper ranges from 2007 to 2021. This suggests that risk measures have become less important over time.

Another interesting observation is that the decile classification model does not make use of the risk measures. Based on Figure 5, the model performs well in predicting volatility. However, it does so without using the most prominent variables associated with volatility.

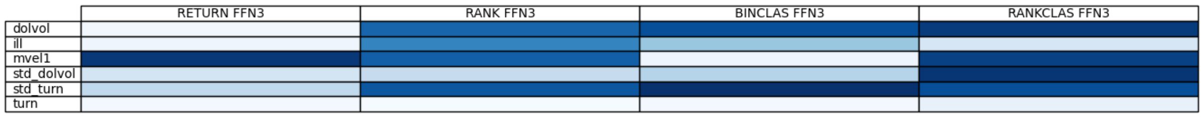
Figure 9 Risk variables



Note: Importance of the risk variables.

The third group focuses on the liquidity of the stocks. This group includes variables like trading volume (dolvol), the market size of the company (mvel1), share turnover(turn) and illiquidity (ill, ratio of stock return to dollar volume), as depicted in Figure 10. Except for the market size of the company, liquidity variables are almost unused in the return regression model, whereas they see great use in the other models. This suggests that market activity is useful for predicting a ranking of the stock returns, but not as much for predicting actual returns.

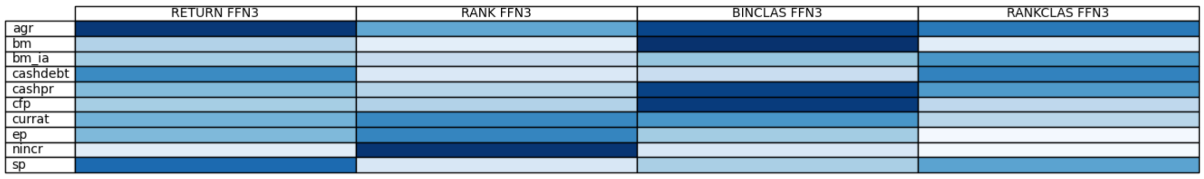
Figure 10 Liquidity variables



Note: Importance of the liquidity variables.

The last group consists of fundamental variables. These are variables that are not specifically linked to the stock market but provide a more general indication of the health of a company. This group includes, among others, the sales-to-price ratio (sp), book-to-market ratio (bm), cashflow-to-debt ratio (cashdebt), cash productivity (cashpr) and current ratio (currat). Based on Figure 11, all models use the fundamental variables for prediction.

Figure 11 Fundamental variables



Note: Importance of the fundamental variables.

6 Conclusion

This study compares the performance of different models for asset pricing, specifically instance ranking formulations and a traditional return prediction model. American stock data is used to train and test four feed-forward neural network model types: return regression, ranking regression, binary classification, and decile classification. The return regression model predicts standard stock returns, whereas the ranking regression, binary classification, and decile classification models predict three levels of ranking aggregation: continuous ranking, bipartite ranking and 10-partite ranking. To analyze the models individually, model-specific measures are employed, while ranking-based measures are used to compare the different model types. I determine the effect of complexity by analyzing the performance of each model type using 1-5 densely connected layers.

Through statistical analysis of the return regression models, it is observed that a 3-layer model performs the best, this implies that deeper models do not necessarily yield better results. The classification models demonstrate a better-than-random performance.

A statistical analysis, using a mean squared error to analyze the ranking errors in stock returns and an adjusted Diebold-Mariano test, shows that the instance ranking formulations outperform the return prediction model. However, empirically, the 1-layer return regression model outperforms all other models. This implies that the essence of asset pricing lies not in the model type itself, but in the selection of covariates used for prediction. The most crucial variable in this context is the momentum variable, which reflects the recent price trends of a stock.

For future research, I would recommend the use of forecast combinations. The instance ranking formulations show a clear statistical advantage over traditional models. Therefore, it is reasonable to assume that there is a way to utilize these models in practice. Another improvement would be to perform a more in-depth analysis of high- and low-market-value stocks. Based on my results, predicting high-market-value stock returns involves more complex interactions, therefore a deeper model would be better.

References

Alberg, J. & Lipton, Z. (2017, 11). Improving factor-based quantitative investing by forecasting company fundamentals.

Bai, J., Philippon, T. & Savov, A. (2016). Have financial markets become more informative? *Journal of Financial Economics*, *122*(3), 625-654.

Chen, L., Pelger, M. & Zhu, J. (2021). Deep learning in asset pricing.

Crooijmans, L., Dekker, M., Van Steenis, T. & Knops, H. (2023, April). Activation Functions in Neural Networks for Asset Pricing: A Comparative Analysis and Portfolio Optimization.

Fama, E. F. & French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, *33*(1), 3-56.

Fama, E. F. & French, K. R. (2004, September). The capital asset pricing model: Theory and evidence. *Journal of Economic Perspectives*, *18*(3), 25-46.

Green, J., Hand, J. R. M. & Zhang, X. F. (2017, 03). The Characteristics that Provide Independent Information about Average U.S. Monthly Stock Returns. *The Review of Financial Studies*, *30*(12), 4389-4436.

Gu, S., Kelly, B. & Xiu, D. (2020, 02). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, *33*(5), 2223-2273.

Hart, J. (2022). *Keras ordinal categorical crossentropy.* GitHub. Retrieved from `https://github.com/JHart96/keras_ordinal_categorical_crossentropy`

Masters, T. (1993). *Practical neural network recipes in c++.* USA: Academic Press Professional, Inc.

Rasekhschaffe, K. C. & Jones, R. C. (2019). Machine learning for stock selection. *Financial Analysts Journal*, *75*(3), 70-88.

Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, *19*(3), 425-442.

Simin, T. (2008). The poor predictive performance of asset pricing models. *The Journal of Financial and Quantitative Analysis*, *43*(2), 355–380.

Takeuchi, L. (2013). Applying deep learning to enhance momentum trading strategies in stocks.

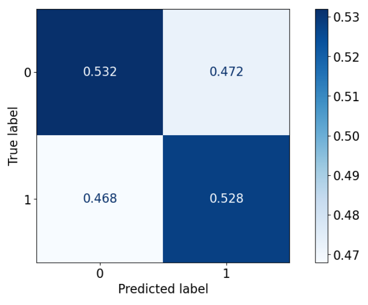
Van der Zwan, T. (2023). *Dataset containing updated data used in the paper "empirical asset pricing via machine learning" by shihao gu, bryan kelly and dacheng xiu".* Unpublished dataset.

Welch, I. & Goyal, A. (2007, 03). A Comprehensive Look at The Empirical Performance of Equity Premium Prediction. *The Review of Financial Studies*, *21*(4), 1455-1508.

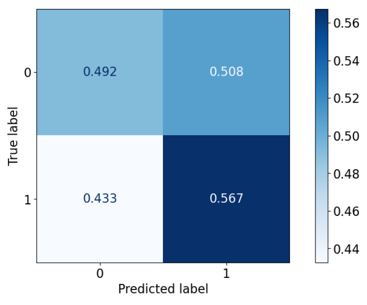
Werner, T. (2020). A review on ranking problems in statistical learning. *Machine Learning*, *111*, 415-463.

A Appendix

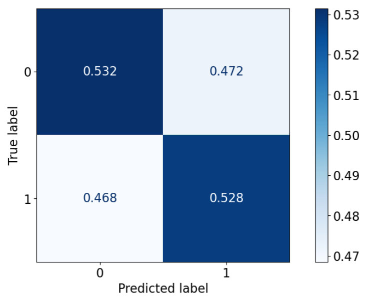
Figure A.1 Confusion matrices 1, 2, 4 and 5-layer binary classification models



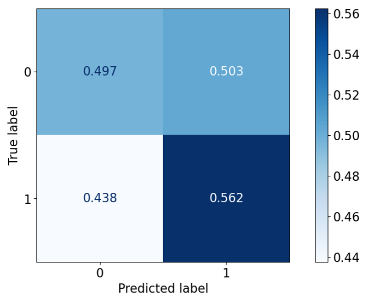
(a) Precision 1-layer



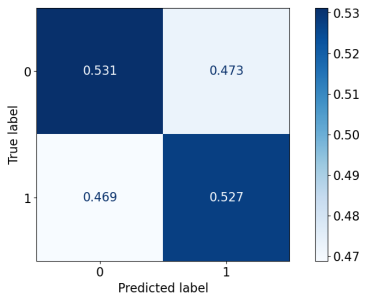
(b) Recall 1-layer



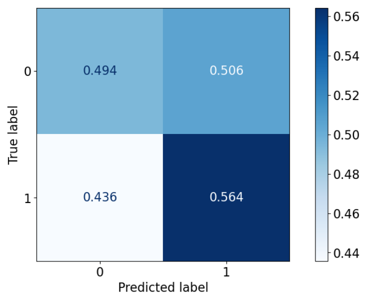
(c) Precision 2-layer



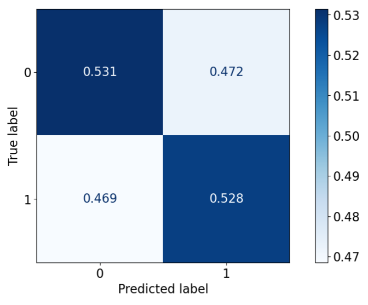
(d) Recall 2-layer



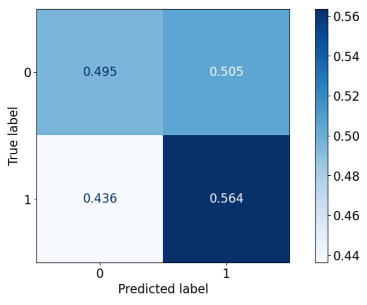
(e) Precision 4-layer



(f) Recall 4-layer



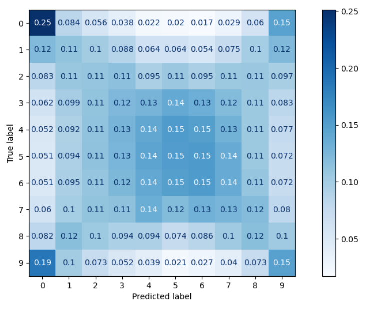
(g) Precision 5-layer



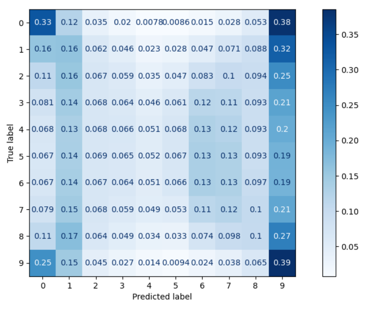
(h) Recall 5-layer

Note: Confusion matrices of the binary classification models. The statistics indicate the precision and recall of the predictions made by the model, as specified in Section 4.4.2.

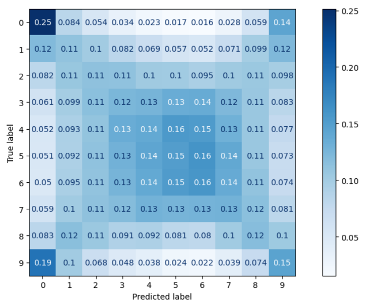
Figure A.2 Confusion matrices 1, 2, 4 and 5-layer decile classification models



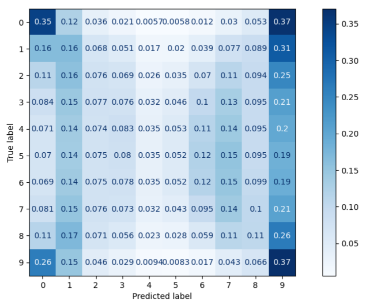
(a) Precision 1-layer



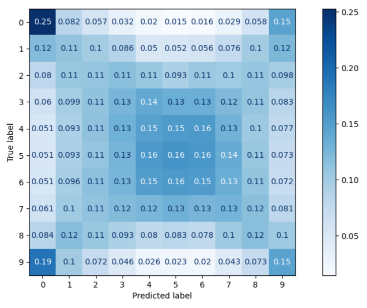
(b) Recall 1-layer



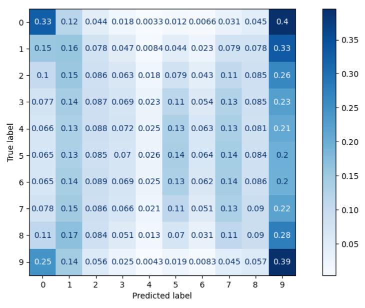
(c) Precision 2-layer



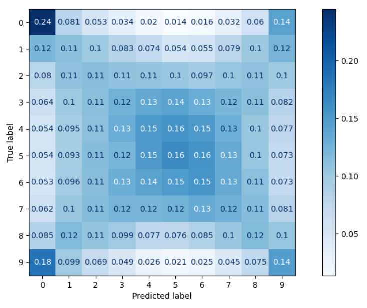
(d) Recall 2-layer



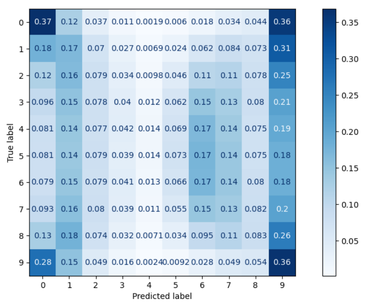
(e) Precision 4-layer



(f) Recall 4-layer



(g) Precision 5-layer



(h) Recall 5-layer

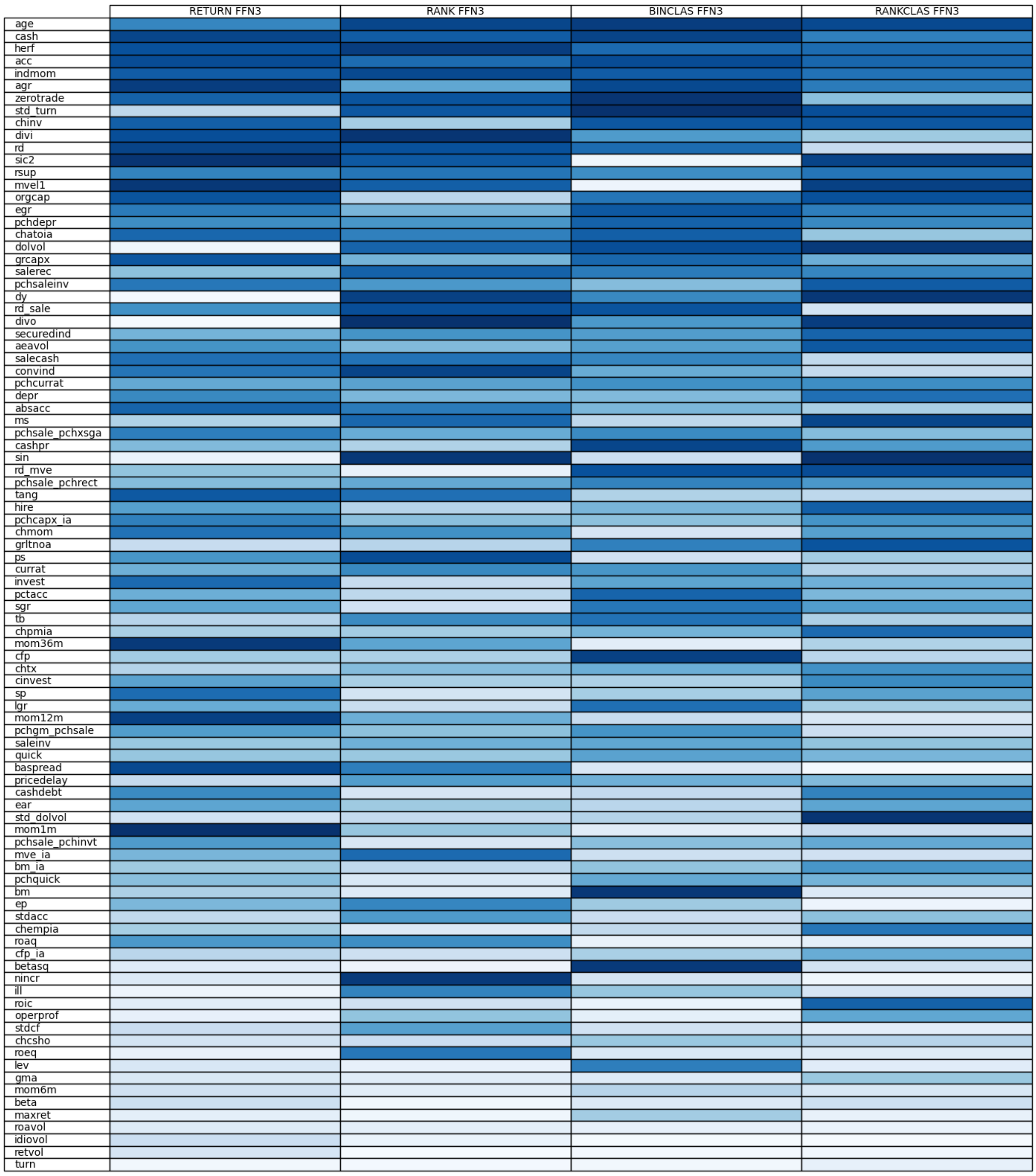
Note: Confusion matrices of the decile classification models. The statistics indicate the precision and recall of the predictions made by the model, as specified in Section 4.4.2.

Table A.1 Diebold-Mariano test statistics

	RE1	RE2	RE3	RE4	RE5	RA1	RA2	RA3	RA4	RA5	BC1	BC2	BC3	BC4	BC5	DC1	DC2	DC3	DC4
RET2	2.36																		
RET3	1.70	-2.00																	
RET4	2.52	1.17	2.45																
RET5	3.27	1.24	2.83	0.31															
RANK1	-5.97	-6.77	-6.48	-6.85	-6.95														
RANK2	-5.93	-6.69	-6.42	-6.77	-6.85	-0.58													
RANK3	-6.06	-6.78	-6.51	-6.86	-6.98	-1.18	-0.63												
RANK4	-5.73	-6.64	-6.33	-6.72	-6.78	-1.06	-0.34	0.16											
RANK5	-5.93	-6.70	-6.43	-6.77	-6.89	-1.41	-0.59	-0.06	-0.32										
BINCLAS1	-6.20	-7.05	-6.66	-6.99	-7.10	-3.60	-3.47	-3.33	-3.48	-3.27									
BINCLAS2	-6.16	-6.99	-6.61	-6.93	-7.06	-3.59	-3.44	-3.32	-3.45	-3.25	0.21								
BINCLAS3	-6.13	-6.97	-6.60	-6.89	-7.03	-3.39	-3.21	-3.05	-3.26	-3.07	1.37	1.20							
BINCLAS4	-6.23	-7.10	-6.71	-7.02	-7.15	-3.42	-3.16	-3.02	-3.28	-3.06	1.16	1.05	0.12						
BINCLAS5	-6.22	-7.07	-6.68	-6.99	-7.11	-3.50	-3.25	-3.11	-3.38	-3.15	1.02	0.73	-0.29	-0.55					
DECCLAS1	-5.50	-6.13	-5.73	-6.20	-6.21	-2.52	-2.45	-2.29	-2.28	-2.20	-0.04	-0.07	-0.25	-0.27	-0.20				
DECCLAS2	-5.53	-6.15	-5.75	-6.22	-6.22	-2.61	-2.53	-2.38	-2.37	-2.29	-0.18	-0.22	-0.38	-0.41	-0.34	-2.12			
DECCLAS3	-5.49	-6.13	-5.73	-6.20	-6.20	-2.55	-2.47	-2.32	-2.31	-2.22	-0.09	-0.13	-0.30	-0.33	-0.26	-0.80	1.40		
DECCLAS4	-5.48	-6.11	-5.71	-6.18	-6.18	-2.53	-2.45	-2.30	-2.28	-2.20	-0.09	-0.12	-0.29	-0.31	-0.25	-0.58	1.29	0.09	
DECCLAS5	-5.42	-6.04	-5.65	-6.13	-6.13	-2.37	-2.29	-2.14	-2.12	-2.04	0.09	0.06	-0.11	-0.14	-0.07	1.12	2.54	1.68	2.20

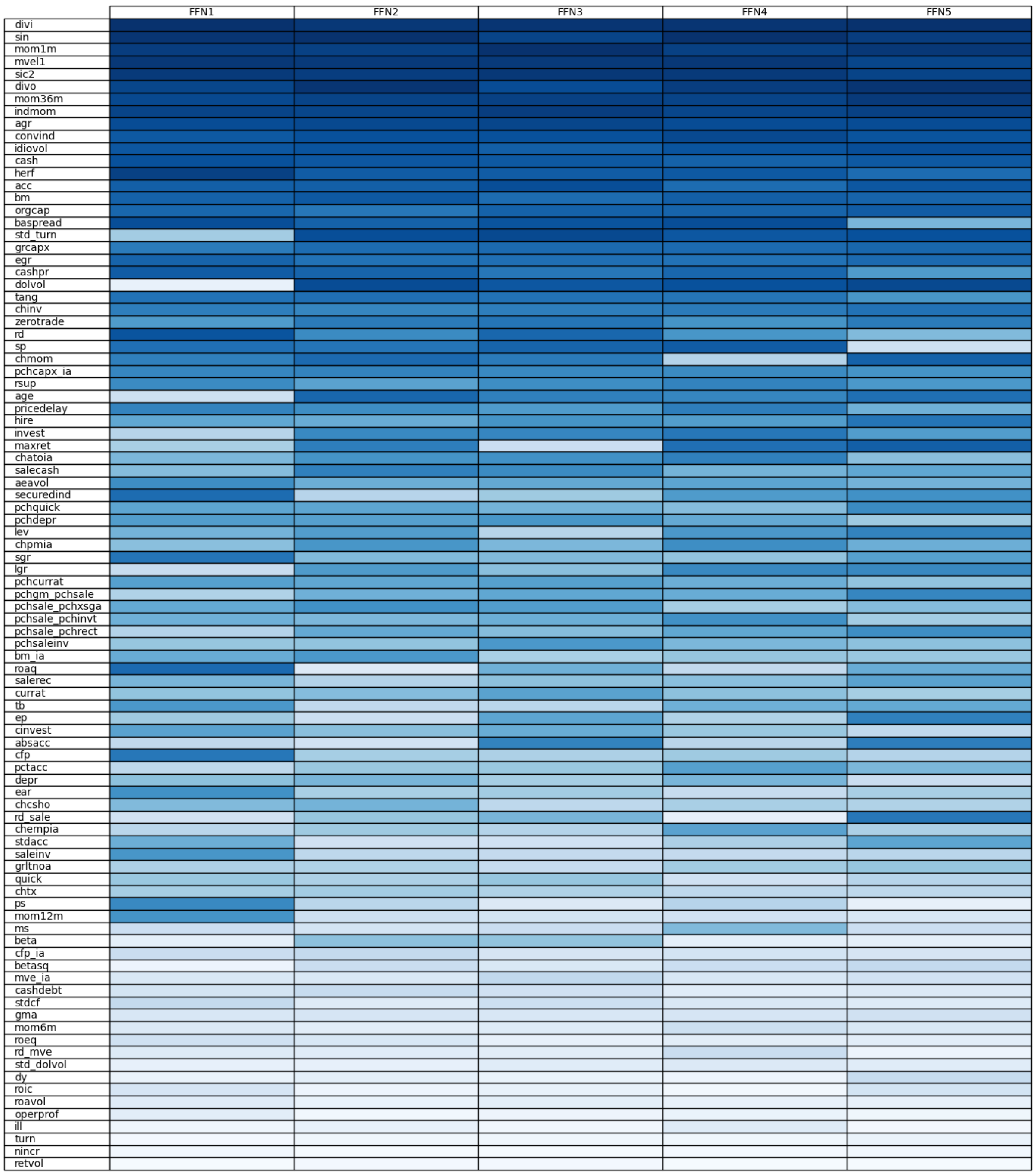
Note: Table containing the Diebold-Mariano test statistics for all architectures and all model structures.

Figure A.3 Variable importance all variables



Note: Importance of all variables in the dataset across the 3-layer variants of the four different model types.

Figure A.4 Variable importance return regression



Note: Variable importance of the 1-5 layer return regression models.