technische universität
dortmund

Case Studies

Winter 2023

Project II
Forecasting the equity premium: Approach using machine learning

Ly Le Thi
December 14, 2023

in collaboration with Alexander Langnau, Sudhir Pratap Singh Rathore, and Jonathan Albert Karras

Lecturers: Prof. Matei Demetrescu
Dr. Paul Navas Alban

Contents

1 Introduction

Excess returns are defined as the return obtained from the return on stock (or portfolio of stocks) above the risk-free rate. The prediction of excess stock returns has been a central theme in financial economics. Accurate forecasts of excess stock returns are important for investors, policymakers, and financial analysts as they impact investment decisions and portfolio management strategies. In this paper, we go deeper into stock excess return forecasting by looking at different machine learning forecasting models.

The motivation for this research stems from Goyal and Welch's project (Welch and Goyal, 2008), where they contend that the historical average excess stock return outperforms the combination of other predictor variables. In this study, we explore the predictive power of various models relative to the historical data of series return compared to the general model which combines the historical data and all other covariates as predictors.

To achieve this goal, three machine learning methods (Regression Tree and Random Forest and Feed-forward Neural Network) are employed to create forecasts of excess stock returns. These forecasts will be generated using lagged predictor variables created from series excess returns alone (we called it a historical model) and using a combination of lagged predictors and all covariates (we called it a general model). For the historical model, we take the lag up to ten days of the series excess returns and train the model on the training set using three machine learning methods then use the trained model for prediction on the testing set. For the general model, we combine the historical data of serries excess returns with all covariates in the dataset in a model and also apply the procedure for the historical model, train the model on the training set, and make predictions on the testing set. The quality of these forecasts is assessed using the mean squared forecast error (MSFE) as an evaluation metric, providing a comprehensive measure of forecast accuracy. The results show that, in general, Neural Networks outperformed the other two models, exhibiting the smallest MSFE in both the general and historical models. For the Regression Tree, the general model, incorporating all covariates and lagged predictors, demonstrated superior performance compared to the historical model. In contrast, Random Forest exhibited an intriguing pattern, with the historical model outperforming the general model. For the feature importance results, we observed that the lagged of series excess returns contributes mostly to predictions of both historical and general models, other than that, inflation, dividend and earning 12 months, and net equity expansion also have a large contribution to the general model performance.

In addition to this introduction, the report comprises four additional sections. The second section of this report provides a description of the dataset and the preprocessing data before coming to the model structure. The following section describes the different machine learning methods and software used in this report, such as Regression Tree, Random Forest, Feed-forward Neural Network, Mean Squared Forecast Error, and Feature Importance calculation. The fourth section of this report states all the results that are obtained using all three machine learning methods mentioned in the third section of the report and discusses the limitations of the findings. Finally, the fifth section summarizes all the results.

2 Data

2.1 Data Description

2.1.1 All data

The dataset used in this project is taken from Welch Goyal's project which is provided on the website $https://sites.google.com/view/agoyal145$. The original multivariate dataset contains a total of fourteen variables in monthly frequency with 1824 data points starting from 1871-01 and ending in 2022-12. For the purpose of this research, we drop three variables from the original dataset. Firstly, the two series $CRSP\_SPvw$ and $CRSP\_SPvwx$ will be removed since they are stock returns series computed by Welch Goyal's project Welch and Goyal (2008) and will not be used as predictors for our target variable, excess returns series. We also removed the series Cross-Sectional Premium since more than half the data of this series is missing value and the period of data is from 1947 to 2003, which can not cover the intent of this research, till 2022. The final dataset which is used for this research then contains fourteens variables and all variables are plotted in Figure 1.

From the plot, we can see that the compounded returns on the S&P 500 index, Dividend 12, Earning 12, have an increasing trend since they are the cumulative sum of the dividends paid on the S&P 500 index (or earnings from the S&P 500 index) over the measures period. The time series Treasure-Bills (the short-term U.S. government bonds), Corporate Bond Returns (including AAA and BAA) which indicates the performance of corporate debt markets, Long-Term Yield (the yield on long-term government bonds), and Risk-free rate (the interest rate derived from Treasury-bill rates) all have increasing trends from 1950 and reach the peak in the early 1980s then decrease until the end of the

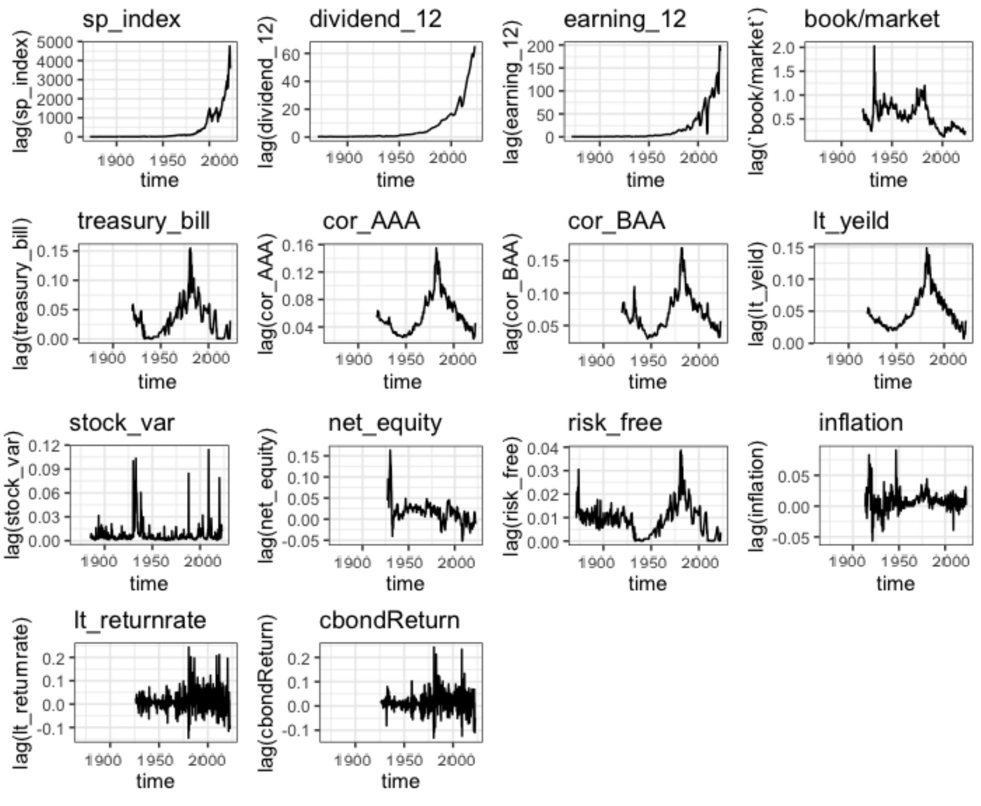

Figure 1: Plot of all fourteen variables over time.

research period (2022). The other variables such as Net Equity Expansion (measures the net issuance of equity by NYSE-listed stocks over 12 months), Inflation (the Consumer Price Index of all urban consumers), Stock Variance (the sum of squared daily returns on the S&P 500 index)... have large volatility in the research period.

2.1.2 Excess Return

Excess return is the surplus return on the investment above the risk-free rate, providing insights into the additional return earned for taking on additional risk. By subtracting the risk-free rate from the asset's return, it's possible to measure the additional return generated by the asset in comparison to a virtually risk-free investment.

Based on the definition of stock excess returns, the time series excess returns are generated by first calculating the percentage change between consecutive values in the S&P 500 index data, effectively determining the daily returns of the index with the help of *Delt* function in package *quantmod* in R. The intuitively of the *Delt* package is taking the percentage change of stock price to have the stock returns $R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$ with P_t is the S&P 500 stock price at time t.

Following this, the function subtracts the risk-free rate from the calculated returns to derive the excess returns $ER_t = R_t - RF_t$ with ER_t as the excess return at time t, R_t as the stock return which has been calculated above and RF_t is the risk-free rate at time t. This is a common approach in finance known as the "excess return formula". The series excess returns is plotted in Figure 2

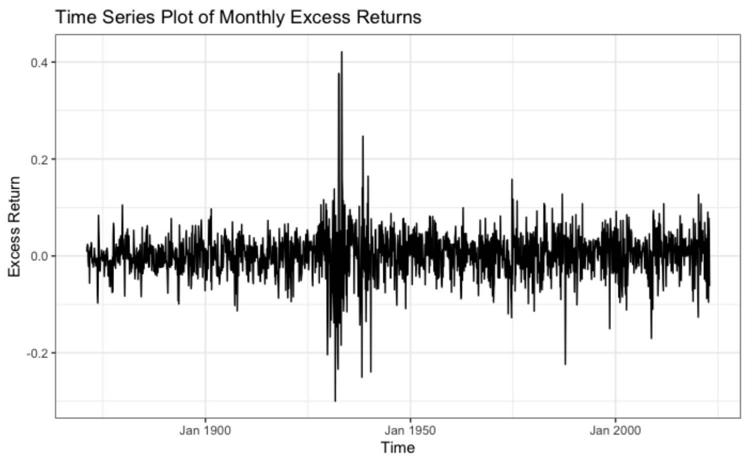

Figure 2: Monthly excess returns time series.

The time series excess returns show a constant or stationary trend, just vary around the mean that close to zero and do not show any upward or downward trend along the research period. This can be because the excess returns are generated by taking first differences in short-term future prices that will remove the trend of the S&P 500 and then subtract the risk-free rate, the whole process makes the variables stationary. In table 1 the minimum observation, the 25% quantile, the median, the mean, variance, and the 75% quantile, and the maximum observation for the excess returns time series for different frequencies (monthly, quarterly, and annually time series data).

Table 1: Summary statistics of excess returns series.

	Min	25% Quantile	Median	Mean	Variance	75% Quantile	Max
Excess return	-0.30	-0.02	0.003	0.002	0.002	0.03	0.42

Analyzing the series excess returns series, the data reveals a balanced trend with a mean value equal to -0.3, and close to zero variance (0.002) indicating overall stability. The range extends from a minimum of -0.02 to a maximum of 0.42, suggesting potential fluctuations in first differences of future S&P 500 prices.

2.2 Data Prepossessing

2.2.1 Creating historical data

For the entire dataset, we generate one-month lags for all covariates, which will become instruments for subsequent calculations, model fitting, and forecasting. This involves creating a prediction $\hat{y}_{t+1} = \hat{f}(y_t, \ldots, y_{t-n+1}, X)$, where $\hat{f}(y_t)$ represents the estimated function.

In the context of time series excess returns, historical data is constructed by incorporating one to ten-month lags, utilizing these historical lags as predictors for subsequent historical modeling. After removing all missing values and aligning with the primary dataset encompassing all covariates, the range of the research period is set from January 1927 to December 2022 and consists of 1152 data points. The historical dataset incorporates a target variable, time series excess returns, along with ten predictors representing lags of the excess returns series. The resulting historical dataset then be combined with 12 covariates (excluding the time series S&P 500 and time series risk-free rate which were used in excess returns calculation) creating the general dataset that includes one target variable, time series excess returns, and a total of 22 predictors.

2.2.2 Splitting time series

After having the final dataset which consists of 1152 data points, from 1927-01 to 2022-12, we then split the time series into training and testing sets following the timeline order. The whole dataset then will be split with 80% proportion of the data allocated to the training set (from 1927-01 to 2003-10) and the 20% left allocated to the testing set (from 2003-10 to 2022-12). Subsequently, two distinct datasets are created: comprising the first portion of the data up to the calculated index, and containing the remaining data from the dataset. This split enables the model to be trained on one subset of the data and evaluated on another, facilitating the assessment of its performance on unseen examples. Adjusting the parameter allows for flexibility in the allocation of data for training and testing purposes.

3 Methods

In this section, we provide an overview and description of the various methods used in this project, as well as the software utilized for their implementation.

3.1 Introduction to Machine Learning and Tree-based Algorithms

Machine learning is a part of artificial intelligence that focuses on creating smart algorithms. Its core concept revolves around solving problems by constructing models that effectively represent specific datasets by developing and studying the statistical algorithms that enable computers to learn from data (Verdhan, 2020). Supervised learning is widely recognized as the foremost application of machine learning. It involves acquiring a function to map given input data (X) to specific output values (Y) and subsequently utilizing this function to determine the corresponding Y for new input data. One common technique in this approach involves applying statistical algorithms with historical data to understand and capture the patterns (Verdhan, 2020). This procedure is known as algorithm training. The historical data, or training data, comprises both input and output variables, constituting a collection of training examples that the algorithm learns from.

Tree-based algorithms, such as decision trees, random forest, gradient boosting, stand out as among the most effective and widely employed methods in supervised learning. These algorithms enhance predictive models by offering impressive accuracy, stability, and ease of interpretation. In contrast to linear models, they excel in capturing non-linear relationships. Their versatility allows them to adeptly address various problems, be it classification or regression.

3.2 Regression Tree

A regression tree is a specific type of decision tree used in the context of regression analysis, where the goal is to predict a continuous numerical output. Decision trees, in general, are a popular class of algorithms in machine learning that can be applied to both classification and regression tasks. A decision tree is a flowchart-like structure where each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (in the case of classification) or a numerical value (in the case of regression). In contrast to traditional classification trees, where the objective is to partition the feature space into disjoint sets corresponding to distinct classes, in a regression tree, the goal is to predict a continuous numeric target value (Y) based on a set of predictor variables ($X_1, \ldots, X_p$) (Loh, 2011).

In the decision tree algorithm, the nodes are formed through a specific process that involves selecting the best feature at each node based on certain criteria (e.g., mean squared error for regression tree). The algorithm then splits the data into subsets based on the chosen feature, creating branches and forming child nodes. This process is repeated recursively until a stopping criterion is met, resulting in the complete tree structure. Algorithm 1 below outlines the pseudocode for constructing a regression tree through an exhaustive search:

1. Start at the root node.

2. For each variable X, identify the set S that minimizes the sum of node impurities [1] in the two child nodes. Opt for the split ($X^* \in S^*$) that results in the minimum overall impurity across X and S.

3. If a predetermined stopping criterion is met, terminate the process. Otherwise, repeat step 2 for each child node consecutively (Loh, 2011).

Figure 3 below describes the structure of a regression tree, as described by the accompanying pseudocode Algorithm 1. A regression tree is a hierarchical structure that begins at the root node representing the entire dataset. The process involves iteratively splitting nodes based on variable values to create homogeneous subsets, leading to decision nodes and, eventually, terminal nodes known as leaves. The crucial criterion for the splitting process is to minimize the sum of impurities in child nodes, ensuring the effectiveness of the divisions. In the decision tree also comes the concept of pruning, depicted as the removal of sub-nodes from decision nodes, counterbalances splitting, emphasizing a balance between complexity and precision. In essence, a regression tree is a predictive model that organizes data into a structured tree, facilitating clear decision paths and allowing for effective modeling of relationships within the dataset. This leads to the creation of piecewise constant models. While these models offer simplicity in interpretation, their predictive accuracy often falls short compared to models with greater smoothness.

3.3 Random Forest

Random Forest is a machine learning algorithm rooted in tree-based methods that harnesses the collective strength of multiple decision trees for decision-making. Unlike a single decision tree where nodes are formed through a specific process, in a Random Forest, the nodes of individual trees are formed through a randomized process, as each tree is constructed using a subset of randomly chosen features. This randomness is a key charac-

[1] Impurity is a measure of how mixed the classes or values are within a node. For regression trees, impurity represents the variability of the target variable. The goal of tree algorithms is to reduce impurity by splitting nodes based on certain criteria, ultimately creating more homogeneous subsets. The common impurity measure is the mean squared error for regression tree.

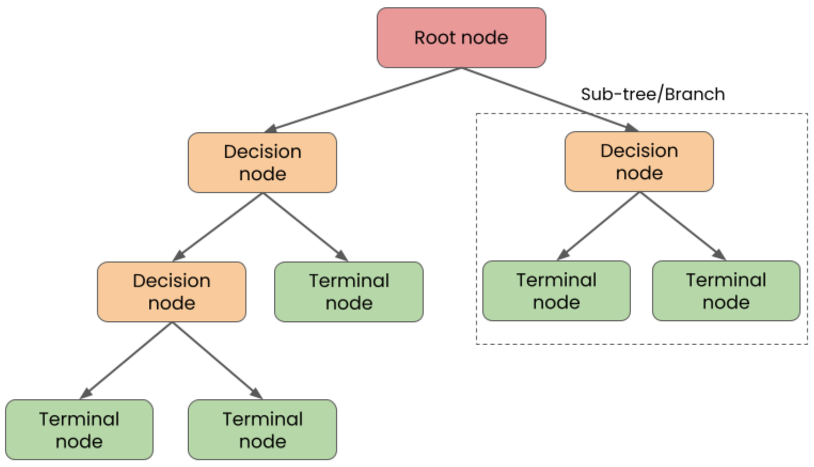

Figure 3: Architecture of Regression Tree ((Chauhan, 2022))

teristic of Random Forest, contributing to its diversity and ability to mitigate overfitting. Each node in these trees operates on a random subset of features to compute its output. Subsequently, the Random Forest consolidates the outputs from individual decision trees to produce the final result.

The technique of bootstrapping is integral to Random Forest, involving the random selection of items from the training dataset - a method characterized by its stochastic nature. This approach amalgamates diverse decisions derived from various trees and culminates in a final decision determined by majority voting. The overarching process of amalgamating outputs from multiple individual models, often referred to as weak learners, is known as Ensemble Learning.

Random Forest operates as Figure 4 and can be described as follows:

1) To cultivate each tree, a random sample is drawn, with replacement, from the training set containing N cases. This sampled subset becomes the training set for the tree.

2) Assuming there are M input variables, a specific value m (where $m < M$) is designated. At each node, m variables are randomly chosen from the M, and the node is split using the best split among these m variables. This value of m remains constant throughout the growth of the forest.

3) Each tree is allowed to grow to its maximum extent, without any pruning applied. The final predictions for new data are made by aggregating the predictions of all the trees.

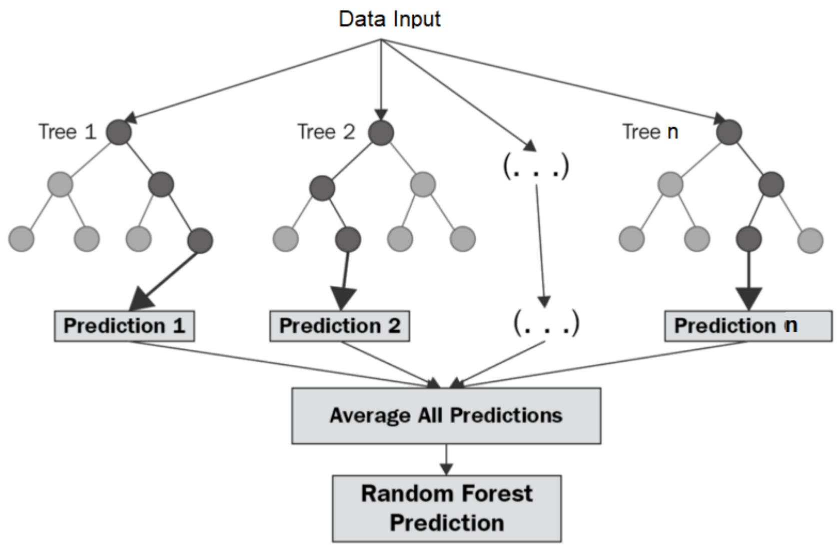

Figure 4: Architecture of Random Forest ((Raj, 2020))

3.4 Feed Forward Neural Networks

Neural Networks are a type of function that establishes connections between inputs and outputs. A Feed Forward Neural Network, a subtype of Artificial Neural Network, features nodes interconnected circularly. This model processes input data exclusively in one direction, with information flowing forward and not in reverse. Unlike traditional machine learning methods, neural networks possess an inductive bias, enabling them to capture both linear and nonlinear relationships effectively. In Artificial Neural Network, a weight is assigned to each input. First, the inputs are multiplied by their weights, and then a bias is applied to the outcome. This is called the weighted sum, then this weighted sum passes through a non-linear activation function (as described in Figure 5 (b)). The network comprises layers, including input, output, and hidden layers, with the number of neurons determined by the dataset and task as in Figure 5 (a). Trial and error are employed to ascertain the optimal configuration of hidden layers and neurons. Connectivity involves linking all inputs from the preceding layer to neurons in the subsequent layer, forming a complex network.

A Feed-Forward Neural Network, in its simplest form, mirrors a single-layer perceptron, wherein input values undergo multiplication by weights, summation, and threshold comparison for output (Figure 5 (b)). The network refines its weights through training, employing methods like gradient descent for single-layer perceptrons and back-propagation for multi-layered perceptrons. This dynamic adaptation ensures continuous improvement in the network's ability to generate precise output values.

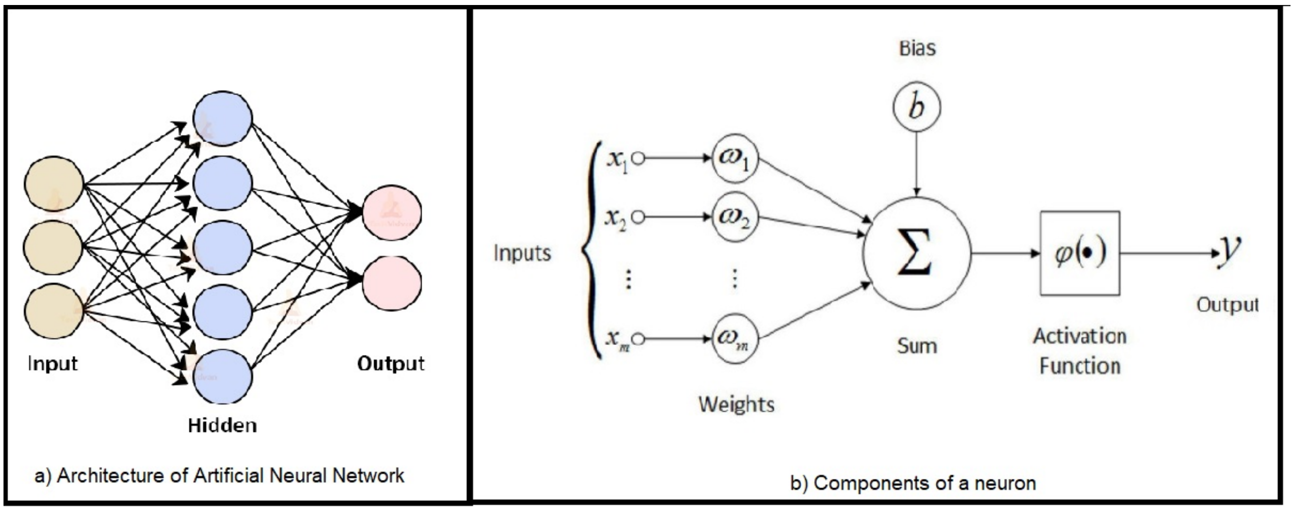

Figure 5: Architecture of Artificial Neural Network ((Sharma, 2023))

3.5 Feature Importance

Machine learning models exhibit the ability to capture complex patterns and relationships within datasets, enhancing computational performance but they lack clarity, particularly in complex models such as deep neural networks or ensemble models with sophisticated algorithms. To understand the behavior of these complex predictive models, the importance of features is assessed by applying permutation-based variable importance. Features are classified as ”important” if their shuffling increases model error, signifying reliance, and ”unimportant” if shuffling leaves error unchanged, indicating disregard by the model.

To achieve this assessment, the *model_parts*() function in DALEX package in R is employed. The DALEX (mo**D**el **A**gnostic **L**anguage for **E**xploration and e**X**planation) package is specifically designed for model-agnostic exploration, explanation, and visualization of machine learning models. The model-agnostic procedure in DALEX is very useful for calculating variable importance. We can calculate the validity of the variables by considering only one permutation, but repeat it several times and average the results. By default, in the *model_parts*() function, we have 10 times of permutation. This approach is based on the permutations approach introduced initially for Random Forest (Breiman, 2001) and then extended to other models by (Aaron Fisher and Dominici, 2018)

The permutation feature importance procedure can be described as follows:

1) Initialization

- Input: Trained model $\hat{f}$, feature matrix X, target vector y, and an error measure $L(y, \hat{f})$.

- Estimate the original model error $e_{orig} = L(y, \hat{f}(X))$, typically using a metric like root mean squared error.

2) Permutation Iteration: For each feature j in the range $1, \ldots, p$, where p is the number of features:

- Generate a permuted feature matrix X_{perm} by shuffling the values of feature j in the data X. This disrupts the association between feature j and the true outcome y.

- Estimate the error $e_{perm} = L(y, \hat{f}(X_{perm}))$ based on the predictions of the permuted data.

- Calculate the permutation feature importance as the difference $FI_j = e_{perm} - e_{orig}$.

3) Feature Ranking: Sort the features by their permutation feature importance values in descending order, identifying the most influential features.

This algorithm provides a systematic approach to assess the impact of each feature on the model's performance by perturbing individual features and measuring the resulting change in prediction accuracy. It is a valuable tool for understanding the relative importance of different features in a predictive model. In essence, feature importance serves as a key tool in striking a balance between the complexity of advanced models and the imperative need for interpretability.

In neural networks, calculating feature importance is not as straightforward as in some other machine learning models like random forests. Neural networks operate as complex, interconnected systems, and the contribution of each input to the output is distributed across the network's layers and weights. To get some insights into feature importance in a neural network we start looking at the weights associated with the input features and extract the weights from the trained model and analyze them.

3.6 Mean Square Forecast Error

To evaluate the accuracy of forecasts in this project, we employed Mean Square Forecast Error (MSFE). MSFE is used to measure the accuracy of forecasts in time series analysis. It's computed by taking the average of the squared differences between forecasted values and actual values over a specified period as formula

$$MSFE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2} \tag{5}$$

.

MSFE is the key metric for evaluating the performance of forecasting methods or models and helps in comparing different forecasting techniques. A high MSFE implies that the fitted values are further away from the actual data, the lower MSFE indicates the better forecasting accuracy.

3.7 Statistical Software

The programming language R (R Core Team, 2021) is used for analyzing and processing the dataset along with the package dplyr (Wickham et al., 2022). The *ggplot*2 package (Wickham, 2016) is also used in this project for data visualization. For the calculation of the excess return series, we employed the *Delt* function in package *quantmod* (Jeffrey A. Ryan, 2023). Other than that, the three most important packages used to fit three models in this research are package *rpart* (Terry M. Therneau and Foundation, 2023) for fitting Regression Tree, package *randomForest* (Breiman and Cutler, 2022)for fitting Random Forest, and package *keras* (Chollet, 2022) for fitting Neural Network. To access the feature importance of each model performance we employed package *DALEX* (law Biecek, 2022)

4 Results

This section is dedicated to analyzing the data using the methods defined in the preceding section. It is divided into the following subsections:

4.1 Hyperparameter

In machine learning, hyperparameters are values that control the learning process and influence the values of model parameters learned by a learning algorithm. They are the external configuration settings that must be specified before the training process and play a crucial role in optimizing the overall performance (MSFEs) of the model. In this project, we explore various combinations and evaluate the sets of hyperparameters that result in the smallest MSFEs, optimizing our models for better predictive accuracy.

For the regression tree, the complexity parameter (cp) is a tuning parameter that controls the trade-off between the complexity of the tree and its goodness of fit to the training data. A larger cp leads to simpler trees, helping to prevent overfitting. We set the range of cp from 0.005 to 0.02 with steps is 0.001 and we get the results for the best model that leads to the smallest MSFEs is $cp = 0.01$ for the historical model and 0.015

for the general model (Table 2).

For the random forest, the hyperparameters are the number of decision trees (*ntree*) in the ensemble and the number of randomly chosen variables to consider for splitting (*mtry*) in a decision tree within the random forest is considering for the model. Increasing the number of trees and the number of splits generally improves model performance (MSFEs) but also increases computational complexity. We set the range for the number of trees from 50 to 300 with steps is 10 and the split in each tree from 1 to 10, the results show that $ntree = 209$ and $mtry = 3$ for both historical and the general model (Table 2).

For the neural network model, we will consider the architecture of the neural network (how many *layer* in the network should we consider), including the number of hidden layers and the number of neurons in each layer, then the (*units*) which determines the number of neurons in a specific layer of a neural network and the learning rates control the step size during the optimization process. We also consider the ($layer_dropout$) as a regularization technique to prevent overfitting with the dropout rate control the fraction of randomly selected neurons that are ignored during training at each update. The results will be presented in Table 2.

After optimizing the hyperparameter we will use the best hyperparameter for further steps like fitting the model with the best set of hyperparameters and using the model for prediction.

Table 2: Hyperparameters for all models

	Historical model	General model
Regression Tree	cp=0.01	cp=0.015
Random Forest	ntree=209, mtry=3	ntree=209, mtry=3
Neural Networks	layer_dense=2, units=96 (for the 1st layer), learning rate=0.2, 1 layer_dropout	layer_dense=4, units=400 (for the 1st layer), learning rate=0.3, 2 layer_dropout

4.2 Model Performance

To evaluate the model performance, MSFEs are then computed for each model on the testing set to quantify and compare the accuracy of the model predictions, the results are presented in Table 3. In general, Neural Networks exhibit the best performance in terms of MSFE, with the smallest MSFE recorded as 0.0437 in general model and even better than historical model with 0.0441. This suggests that the Neural Networks model, with its ability to capture complex relationships, performs well in making accurate predictions on the testing set, and might benefit from additional features, resulting in improved performance in the general model. Along with Neural Network, for Regression Tree the general model, which includes all covariates and lagged predictors of series excess returns, outperforms the historical model. This suggests that incorporating additional features beyond the lagged series of excess returns contributes to improved predictive accuracy for these models. However, Random Forest exhibits an inverse pattern. With the smallest MSFE in historical type of model with just 0.0439, performs much better than the general model, which has an MSFE of 0.04536. The behavior of Random Forest suggests that its performance might be more sensitive to the inclusion of additional features, and in this case, a simpler model performs better.

Table 3: Mean square forecast error for all models

	Regression Tree	Random Forest	Neural Networks
Historical model	0.0456	0.0439	0.0441
General model	0.04538	0.04536	0.0437

4.3 Feature Importance

The feature importance is computed and then ranked in descending order using *model_parts* function in *DALEX* package, as illustrated in Figure 6.

The plot describes permutation-based variable importance and is specifically designed to assess the impact of permuting (shuffling) each feature on the model's performance. The x-axis represents the Root Mean Squared Error (RMSE) loss after permutations. RMSE is used to measure the accuracy of a regression model. The term "loss" here refers to how much the model's performance (measured by RMSE) degrades after permuting the values of a specific feature. The length of each bar corresponds to the extent of RMSE degradation, revealing the importance of each feature for accurate predictions. Boxplots accompanying each bar show the distribution of RMSE losses across random permutations, providing insights into the variability in the feature's impact. Each bar starts at the RMSE value for the model trained on the original, unpermuted data. This is the baseline performance of the model without any feature shuffling. A longer bar indicates a larger negative impact on model performance, emphasizing the permuted feature is crucial for model accuracy and robustness.

For the feature importance results, we observed that the lagged of series excess returns contributes mostly to predictions of both historical and general models, other than that,

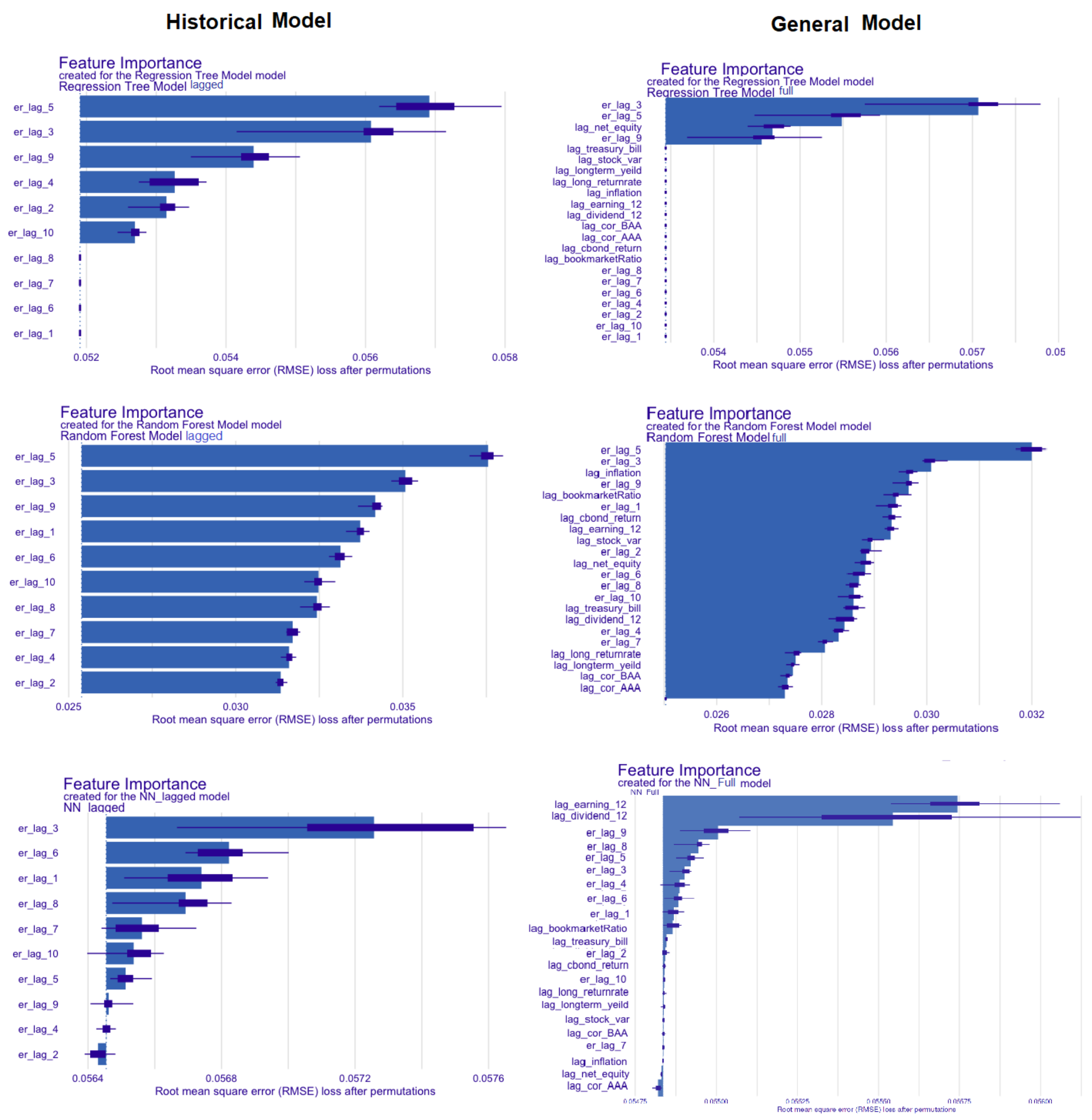

Figure 6: Feature importance for all models

inflation, dividend and earning 12 months, and net equity expansion also have a large contribution to the general model performance. Notably, the regression tree model relies heavily on particular variables to make decisions and assigns high importance to specific features (such as lag 5, 3,9,4,2,10 orderly of series excess returns in historical model and lag 3,5, net equity expansion and lag 9 in general model respectively, and all other lags are unimportance for the model), while Random Forest, known for its randomized feature selection, exhibits a more distributed importance across variables, it gives weight for all its predictors in the model. Some most important ones are lag 5,3 and 9 in the historical model and lag 5,8 and inflation in the general model, but in general, there is not much difference in the ranking and none predictors have zero contribution. This characteristic explains the superior generalization ability of Random Forest, outperforming the

Regression Tree in the historical model. However, the general model with a larger set of predictors, requires more computing time to select random subsets during training and this additional "burden" in feature selection can slightly decrease the forecasting power of the model, contributing to a higher MSFE compared to the simpler Regression Tree in the general model (see Table 3). For Neural Networks, feature importance is challenging to quantify and the network might distribute importance across multiple layers and neurons, but its pattern resembles that of the Regression Tree, assigning higher weights to specific variables such as lag 3 and 6 of the historical model and for general model dividends and earnings 12 months, lag 9 and 8 receive higher weights than other variables.

4.4 Forecast Visualization

Figure 7 illustrates the predicted values generated by three distinct models—Regression Tree, Random Forests, and Neural Networks—applied to both historical and general model scenarios against the actual test data.

Notably, the predicted values of the Regression Tree for both models exhibit minimal fluctuation during the testing period. Particularly in the general model, fluctuations occur primarily on the negative side, and the model tends to align with the mean value of the time series of excess returns. Under specific complexity parameters (*cp*), the Regression Tree produces consistent results throughout the testing period with a small MSFE. This constancy is a characteristic of the Regression Tree model, especially when assigning substantial importance to specific variables and employing categorical splits.

In contrast, the Random Forest model yields more dynamic predictions around the mean during the testing period, providing relatively precise forecasts. The predicted results closely follow the fluctuations in the time series of excess returns for both the general and historical models. On the other hand, the Neural Networks model demonstrates varying predictions around the mean of the excess returns series in the historical model, exhibiting a similar pattern of fluctuation as the Regression Tree when considering the mean of the excess returns series. However, it introduces slight fluctuations at the peaks of the actual excess returns series.

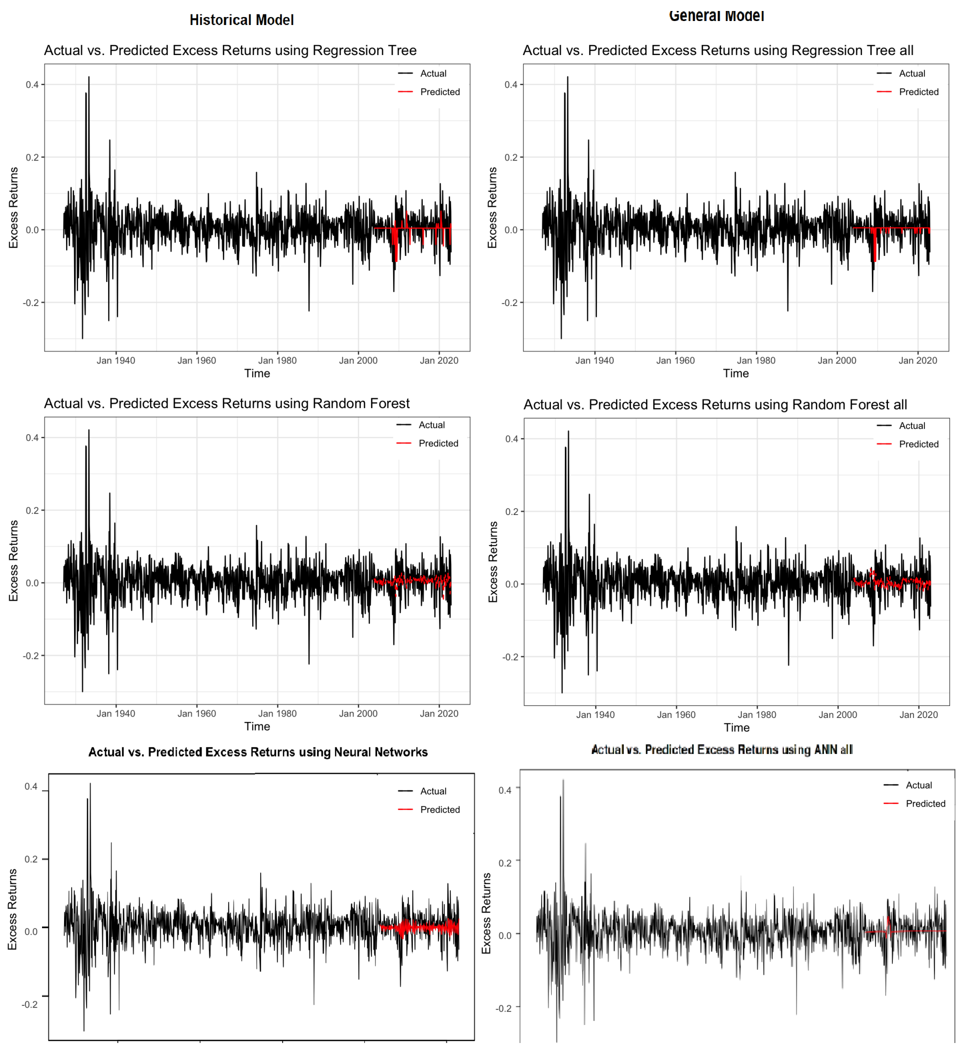

Figure 7: Forecast visualization for all models

5 Conclusion

In this study, we employed three distinct machine learning models—Regression Tree, Random Forest, and Feed-forward Neural Network—to generate forecasts of excess stock returns. These forecasts were created using two approaches: a historical model based on lagged predictor variables from excess returns alone and a general model incorporating both lagged predictors and additional covariates. Evaluating the models' performance using the MSFEs, we observed that Neural Networks outperformed, exhibiting the smallest MSFE in both the general and historical models. This underscores the model's capability to capture complex relationships and make accurate predictions. Furthermore, for the Regression Tree, the general model, incorporating all covariates and lagged predictors, demonstrated superior performance compared to the historical model, emphasizing the value of additional features. In contrast, Random Forest exhibited an intriguing pattern, with the historical model outperforming the general model. This suggests that the model's performance might be more sensitive to the inclusion of additional features, and in this case, a simpler model performed better. Feature importance analysis revealed distinctive characteristics of each model. Regression Tree heavily relies on specific variables for decision-making, while Random Forest exhibits a more distributed importance across variables. For Neural Networks, quantifying feature importance is challenging, but the pattern resembles that of Regression Tree. Notably, the visual representation of Regression Tree predictions showed minimal fluctuation during the testing period, especially in the general model, aligning closely with the mean value of excess returns. In contrast, Random Forest provided dynamic predictions around the mean, closely following the fluctuations in excess returns. The Neural Networks model demonstrated varying predictions around the mean, introducing slight fluctuations at the peaks of actual excess returns series. These findings contribute insights into the strengths and nuances of each model in forecasting excess stock returns.

Bibliography

Cynthia Rudin Aaron Fisher and Francesca Dominici. *All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously.*, 2018. URL `http://arxiv.org/abs/1801.01489`.

Leo Breiman. *Random Forests*, 2001.

Leo Breiman and Adele Cutler. *randomForest: Breiman and Cutler's Random Forests for Classification and Regression*, 2022. URL `https://cran.r-project.org/web/packages/randomForest/index.html`.

Nagesh Singh Chauhan. *Decision Tree Algorithm, Explained*, 2022.

François Chollet. *Getting Started with Keras*, 2022. URL `https://cran.r-project.org/web/packages/keras/vignettes/`.

Joshua M. Ulrich Jeffrey A. Ryan. *quantmod: Quantitative Financial Modelling Framework*, 2023. URL `https://cran.r-project.org/web/packages/quantmod/index.html`.

Przemys law Biecek. *DALEX: Explainers for Complex Predictive Models in R*, 2022. URL `https://jmlr.org/papers/v19/18-416.html`.

Wei-Yin Loh. *Classification and regression trees*, 2011.

R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2021. URL `https://www.R-project.org/`.

Ashwin Raj. *A Quick Guide to Random Forest Regression*, 2020.

Pranshu Sharma. *Basic Introduction to Feed-Forward Network in Deep Learning*, 2023.

Elizabeth J. Atkinson Terry M. Therneau and Mayo Foundation. *An Introduction to Recursive Partitioning Using the RPART Routines.* R Foundation for Statistical Computing, 2023. URL `https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf`.

Vaibhav Verdhan. *Supervised learning with Python.* Okänd, Irland, 2020.

Ivo Welch and Amit Goyal. *A Comprehensive Look at The Empirical Performance of Equity Premium Prediction*, 2008.

Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL `https://ggplot2.tidyverse.org`.

Hadley Wickham, Romain François, Lionel Henry, and Kirill Müller. *dplyr: A Grammar of Data Manipulation*, 2022. URL `https://CRAN.R-project.org/package=dplyr`. R package version 1.0.9.