Department of Business and Management
Bachelor degree in Management and Computer Science

Machine Learning in Asset Pricing

Relatore:
Prof. Michela Altieri

Candidate:
Davide Manniello

Academic Year 2020/2021

Table of Contents

Acknowledgements

I would like to express my sincere thanks to my supervisor, Dr. Michela Altieri, who guided and encouraged me to be professional and do the right thing to achieve the best possible result. Without her invaluable help, the goal of this project would not have been achieved.

I would like to thank my parents from the bottom of my heart. Thank you for always supporting me and allowing me to complete my studies. A special thanks goes to my mother without whom I probably wouldn't be here.

A special thanks goes to my girlfriend who has always supported me especially in difficult moments. And finally I would like to thank my great friend Leonardo Scalzi who contributed to the realization of the graphic part of this work.

List of Figures

List of Tables

Introduction

In this paper I perform a comparative analysis of four Machine Learning methods across the US market in the field of 'Empirical Asset Pricing'. "Asset Pricing theory studies what factors determine asset prices and rates of return. Thus, the main objective of Asset Pricing is to understand the behavior of risk premiums" (J. David Cummins, 1990). But the risk premium is difficult to measure: often the problem is that there are variations in returns due to unpredictable events that hide the real excess return. Therefore, the aim is not to predict the exact value of the risk premium, but to see which covariates influence it most. In addition to the analysis of the covariates, I perform a comparative analysis between Machine Learning methods.

The asset pricing literature is still dominated by traditional approaches that simply run cross-section regressions on stock returns (Lewellen, 2015). This type of approach has many limitations that can be overcome by performing analyses based on Machine Learning methods. Many researches have shown how using Deep learning in the field of Empirical Asset Pricing can result in very good performance in terms of R^2 (Feng et al , 2018).

Gu et al (2020) is one of the most recent research regarding the use of deep learning methods in the field of asset pricing and shows that using neural networks would lead to much better results. But this research also shows how investors can benefit from using these methods. Gu et al. (2020) shows how an investor who bases his strategy on the prediction of the $S\&P500$ using deep learning methods manages to achieve an average Sharpe Ratio of 0.77 which is much higher than that achieved by an investor who adapts a Buy and Hold strategy.

The main purpose of my work is to understand which method perform better in predicting stock returns, and which feature in the dataset affect the most returns predictability. All the methods use the same set of features, chosen based on the results of previous research (Gu et al, 2020) and (Kelly et al, 2019).

The set of Machine Learning methods was constructed based on the findings in the paper 'The Review of Financial Studies' (Gu et al,2020). I used this research as a benchmark for my analysis as it is one of the most recent research in this field and also provides a set of models that can achieve good results.

For my work, I start with a linear regression optimized by Gradient Descent as the first model, in order to compare with non-linear models. The non-linear models I use are Ensamble methods like the Random Forest and Boosted Regression Tree, and Neural Networks. This choice is not casual as Gu et al. (2020) has shown that these methods achieve excellent results in terms of R^2. My analysis shows that the best performing methods are Random Forest and Neural Networks. Regarding the implementation of Neural Networks I decided to use two different types of architectures: a Long-Short term memory architecture and a Feed Forward Network. FNN is one of the most widely used neural network types in the Asset Pricing literature, this structure is able to achieve good results in terms of R^2.

The results of my analysis show that among the stock-level indicators the momentum indicator is the most influential, while for the macroeconomic variables the indicators: 'housing', 'Volume' and 'Vix' are the most influential ones. The results obtained are in line with those obtained by Gu et al. (2020) which shows that one of the most influential features are the momentum indicators and in my analysis they reach a correlation of 40% with the dependent variable, in most of the methods used.

Because Messer (2017) and Gu et al. (2020) rely on the use of a FNN (Feed-Forward network) , I also compared. Therefore, in my analysis I tried to compare two different types of neural networks : LSTM (Long-Short term memory) and FNN. However, the results obtained show that the most efficient type of neural network is the FNN, which achieves an R^2 of 74% compared to 57% achieved by the LSTM architecture. This last topic can be an interesting insight that future research should explore.

The rest of the thesis proceeds as follows. Chapter 1 reports the literature review in the field of empical asset pricing and then some of the main capital market theories. Chapter 2 presents the theory and the methods used in my analysis. In Chapter 3 I analyze the results of the analysis, then conclusions follow. Appendix A shows the results achieved by alternative algorithms, Appendix B displays the Knime workflows used for the creation of the dataset, and Appendix C shows the results achieved in the feature importance analysis.

Chapter 1

Theory and Methods

1.1 Literature Review

Despite all the advances made in Machine Learning, traditional methods in recent years are still dominating the Empirical Asset Pricing literature and only recently there have been some advances in the use of these techniques in the field of empirical asset pricing. The literature on traditional methods of forecasting stock returns can be divided into two parts. The first one focuses on the analysis of differences in expected returns between stocks as a function of stock level features Fama and French (2008) and Lewellen (2015), among many others. This type of literature is characterized by a very simple approach, limited to running cross-sectional regressions on realized stock returns. The second part analyzes the time series of stock returns, one of the first to examine this was Welch and Goyal (2008), followed by Koijen and Nieuwerburgh (2011). This type of literature uses time-series regressions on the returns of some portfolios but taking into account a low number of macroeconomic variables such as Rapach and Zhou (2013). One important thing to note is that the regressions and portfolios used cannot handle a large number of variables, so they do not include all the possible variables that the literature has identified over several decades. Traditional methods have some limitations that Machine Learning methods can easily overcome, which I will discuss later. Despite that, there is still a small amount of work analyzing the cross- section and time series of stock returns using Machine Learning techniques.

Harvey and Liu (2016), for example used the Bootstrap method in order to analyze the most influential features. The results of this research were that the 'dominant' feature are the 'market factor' and the other relevant factors are those related to profitability. But beyond the results obtained, the importance of this work lies in the way they addressed the problem of multiple testing, one of the most common problems when working with 'Big Data'.

Giglio and Xiu (2017) used a Principal Component Analysis (PCA), a dimension reduction technique, to prove that the risk premium can be estimated using a linear model. The work of Kelly et al (2019) also used a dimension reduction method, IPCA, which allowed for more accurate results than the work of Giglio and Xiu (2017). The results of Kelly et al (2019) showed that out of all the variables identified from the previous literature only seven of them are statistically significant: market beta, size, earnings-to-price, book-to-market, assets-to-market, short-term reversal, and momentum.

One of the first works to use tree based models is that of Moritz and Zimmermann (2016) obviously applied to the cross-section of stock returns. Tree models are one of the best performing Machine Learning techniques in this field as Gu et al (2018) demonstrated in later studies.
Moritz and Zimmermann (2016) compared the results from their research with those obtained from previous studies, based on linear functions, and showed that these left out a lot of relevant information. Therefore, they conclude their work by stating that tree-based methods achieved much more accurate results than those in the classical literature.

Starting in 2007, a number of works have been developed using neural networks to analyze the cross-section of stock returns. Cao et al (2011) is one of the first works to venture into this field. This study performed a comparison analysis between: a single-factor model, the three-factor model of Fama and French, and an ANN. Obviously, the best results were achieved by the ANN model. The ANNs are a particular type of Neural Networks and a very important rule when using the ANNs is that the performance of the network is optimal when the sample of data is big, while on smaller samples there is a great probability of overfitting.

Messmer (2017) was one of the first papers to use a Feed-forward Neural Network (FNN). FNNs are a type of neural network in which the signal only flows in one direction, and the author acknowledges the limitations of this method by arguing that FFNs are not the best method for predicting the US cross-section of stock returns. The author concluded his paper by identifying the most significant variables: short-term reversal and twelve-month momentum.

Feng et al (2018) built a dynamic deep learning model to predict asset returns. This type of Neural Network is much more complex than the previously analyzed work also because it achieved particularly relevant results at R^2 level. But one of the main problems with using deep learning techniques is that the model loses interpretability while gaining accuracy.

Gu et al. (2020) is one of the most recent researches in the application of machine learning techniques in the field of Empirical Asset Pricing. In the first part of their work they implement a comparative analysis of Machine Learning methods, taking into account all the methods highlighted in previous works. Their research shows that the best results are obtained by ensamble methods, such as Random Forest and Boosted Regression tree, and Deep Learning. In the second part of their analysis they estimate the most relevant ”predictive signals”. My work takes this research as a benchmark for both the methods used and the features selected.

1.2 Capital Market theories

1.2.1 Markowitz’s theory

Much of Capital market theory is based on the assumption that investors hold diversified portfo lios. The first comprehensive theory of diversification was developed by Harry Markowitz (1952) and (1959). Markowitz’s theory of portfolio selection is defined as a normative theory, which means that it describes a standard rule of behaviour that investors should follow when constructing a portfolio. Markowitz’s diversification theory only considers the mean and variance of asset returns, it is often called Mean-Variance Diversification or Mean-Variance Portfolio theory (J.David.Cummins, 1990). The theory relies on several assumptions:

1. All investors are risk-adverse, they accept more risk if and only if they are compensated with an higher expected return.
2. Investors take into account only the expected return and variance of asset returns.
3. All markets are perfectly efficient, (frictionless markets).
4. Investors seek to maximize the expected return of total wealth.

Suppose our portfolio has N assets and the expected return on each asset is defined as R_i, where $i = 1,...,N$. Then the portfolio return, R_p, will be the weighted average of the various expected returns included in our portfolio :

$$R_p = \sum_{i=1}^{N} w_i R_i \tag{1.1}$$

The portfolio variance,σ_p^2, instead is calculated with the following formula :

$$\sigma_p^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j \sigma_i \sigma_j \rho_{ij} \tag{1.2}$$

where ρ_{ij} is the correlation coefficient between the returns on assets i and j. So one of the main messages of MPT is that assets in a portfolio cannot be selected simply by looking at their characteristics, but also by taking into account the correlation that exists between the various securities. The Equation 1.2 shows how the risk of a portfolio depends on : the asset weights, the standard deviations of the asset returns and especially on the correlation of the asset returns. The objective of this theory is to find a set of efficient portfolios. All portfolios involve risk, σ, and return, $E(r)$, so the efficient frontier will be formed by the combinations of risk and return that minimize σ and maximize $E(r)$, (Top-Left).

The word risk could be substituted with the expression " dispersion of outcomes around the expected value", so riskier simply means more dispersion of outcomes around the expected value. The variance and the standard deviation measure dispersion of outcomes around the expected value.

This theory is the basis of the Capital asset pricing model, CAPM, which describes the relationship that should exist between asset returns and risk assuming that investors construct and select portfolios according to Mean-Variance Portfolio theory.

1.2.2 CAPM

The Capital Asset Pricing Model (CAPM) was the first major asset pricing model, developed independently by William Sharpe, John Lintner, and Jan Mossin during the 1960s. CAPM is a one-factor model and it is based on the following assumptions :

1. Investors consider only efficient portfolios (Markowitz's theory).
2. A riskless asset exists, and investors can borrow and lend unlimited amounts at the risk free rate.
3. There are no transactions costs, taxes, or restrictions on short sales, frictionless markets.
4. Securities are assumed to be infinitely divisible.
5. Investors can always borrow and lend at the same risk-free rate.
6. Investors have the same expectations about future returns

This model requires the parameter "beta" to describe the association between systematic risk and stock rates of return. Beta measures the degree of co-movement between asset returns and market portfolio returns:

$$\beta_i = \frac{\sigma_{im}}{\sigma_m^2} \tag{1.3}$$

σ_{im} is the covariance between stock returns and market returns while σ_m^2 is the variance of the returns on the market. In other words, beta quantifies the systematic risk of an asset, the amount of risk that cannot be diversified away.

The CAPM establishes that the expected rate of return on any risky asset satisfies the following equation :

$$E(R_i) = R_f + \beta_i E(R_m - R_f) \tag{1.4}$$

where $E(R_i)$ is the expected return on the asset i, R_f is the risk free rate and R_m is the return on the market portafolio. The term $(R_m - R_f)$ is called market risk premium and it represents the return over the risk-free rate required by investors to hold the market portfolio. The Equation (1.4) can be rewritten as :

$$E(R_i) - R_f = \beta_i E(R_m - R_f) \tag{1.5}$$

The interpretation of equation (1.5) is really simple, it merely says that the risk premium on an individual asset equals its beta times the market risk-premium. The empirical applications of the formula is based on the security market line (SML). ”This line is obtained by regressing its holding-period returns on the returns for the ”market” (S&P500) over some historical period” (Viviana Fernandez, 2006). The equation for the SML is:

$$R_i = \alpha_i + \beta_i(R_m - R_f) + \varepsilon_i \tag{1.6}$$

where α and β are the coefficients to be estimated and ε is a random error. Equation (1.6) introduces α, which is one of five popular technical investment risk ratios. The others are Beta, Standard Deviation, R-squared, and the Sharpe ratio. The α typically refers to a percentage measuring how the portfolio performed compared to the referenced benchmark index.

1.2.3 Fama French

The cross-section of average returns on U.S. common stocks shows little relation to either the market β of the Sharpe (1964) Lintner (1965) Asset Pricing model (Fama and French, 1993). Taking all critique towards the CAPM into account, Fama and French develop their famous three-factor model. This model augments the market return factor with size, based on market capitalization, and value, based on book-to-market value:

$$R_i = r_f + \beta_m(r_m) + \beta_s(r_s) + \beta_{bm}(r_{bm}) \tag{1.7}$$

where r_m is the return on the market index minus the risk free rate, r_s is the return on small-firm stocks less the return on large-firm stocks and r_{bm} is the return on high book-to-market-ratio stocks less return on low book-to-market ratio stocks. Many criticisms have been addressed to this model, Ferson and Harvey (1999) demonstrated that the three-factor model fails to explain expected return, but surely it is one of the most used model in the literature.

Fama and French (2015) find a way to improve their three-factor model, adding two additional factors:

$$R_i = r_f + \beta_m(r_m) + \beta_s(r_s) + \beta_{bm}(r_{bm}) + \beta_{rmw}(r_{rmw}) + \beta_{cma}(r_{cma}) \tag{1.8}$$

where the first three variables are already explained in equation (1.8), r_{rmw} is the return spread of most profitable firms minus least profitable and r_{cma} is the return spread of firms that invest conservatively minus aggressively. The decision to add these two variables is due in part to the 'Dividend Discount Model' but also to the 'Miller and Modigliani (1961) theory'. They come up with three statements about expected stock returns :

1. A higher book-to-market equity ratio is generally associated with a higher expected return.
2. higher expected earnings imply a higher expected return.
3. higher expected growth in book equity implies a lower expected return.

Chapter 2

Methodology

2.1 Liner Regression and Gradient Descent

Typically, a linear model makes predictions through a weighted average of its features, including a bias term :

$$y_p = (\theta_0 + \theta_1 x_1 + ... + \theta_n x_n) \tag{2.1}$$

Generally, the equation (2.1) is expressed through a Vectorized form :

$$y_p = h_\theta(x) = \theta x \tag{2.2}$$

In the equation (2.2), θ contains the bias term θ_0 and all the features weights $\theta_1 ... \theta_n$, then we have the hypothesis function $h_\theta(x)$ and the feature vector x.
Training a Linear Regression means finding the value of θ that minimizes the MSE :

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(\theta^T x_i - y_i) \tag{2.3}$$

One of the standard approaches in order to find the value of θ that minimizes the cost function (MSE) is to use a Normal Equation :

$$\theta_{min} = (X^T X)^{-1} X^T y \tag{2.4}$$

Without going into too much detail let's say right away that I'm going to discard this method. This choice is due to the computational complexity of inverting the matrix $X^T X$. The best way to understand the complexity of such a matrix is by using the Big O notation :

$$\mathcal{O}(n^2) \tag{2.5}$$

In the equation (2.5) 'n' represents the number of features, if this value doubles then the computation time will increase by four times. This kind of approach is very slow when the number of features increases. Therefore, I discard this method a priori.

An other method that can minimize the loss function is that one of the Gradient Descent. Unlike the Normal equation, Gradient Descent iteratively adjusts its parameters until the loss function is minimized. In the case of a Linear Regression the MSE cost function is a convex function, Figure 2.1. Therefore, being a convex function with no continuity problems, there is only one global minimum and no local minimum. These considerations are very important because after a significant number of steps, which depend on the learning rate, the algorithm will reach points that are very close to the absolute minimum.

There are various types of GD, the most popular are : Batch Gradient Descent and Stochastic Gradient Descent. The main difference between these two types is that Batch GD uses the entire training set to calculate the gradient at each step, whereas Stochastic GD takes a random instance from the training set and calculates gradients based on that single instance only. For my purposes I will use Stochastic GD since it allows to work easily on large datasets, the only drawback of this method is that it will not have the same accuracy as the GD Batch.

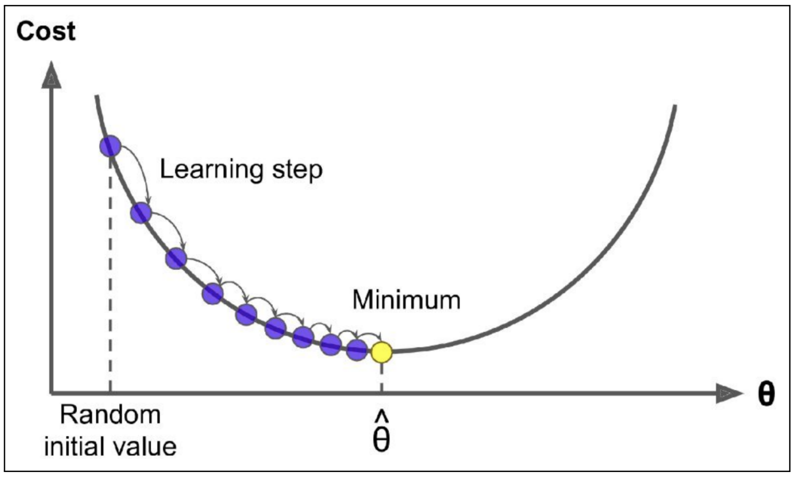


Figure 2.1: MSE cost function

The figure shows how a convex function is minimized. The parameters are randomly initialized and iteratively adjusted in order to minimise the function. The learning step is proportional to the slope of the function, so the more we approach the minimum, the smaller the step becomes.

2.2 Random Forest

Random Forest is a supervised learning algorithm, and is classified as an Ensemble method, which means building a model on top of many others. A random forest is a predictor consisting of a collection of randomized base regression trees $r_n(x,\theta_m,D_n), m \geq 1$, where $\theta_1,\theta_2,..$ are the output of a randomized variable. As I said Random Forest is an Ensamble method so all these regression trees are combined in order to generate a regression estimate :

$$\bar{r}_n(X,D_n) = E_\theta[r_n(X,\theta,D_n)] \tag{2.6}$$

where D_n is our training sample and E_θ is the expectation with respect to the random parameter. This random parameter θ is used to determine the cuts performed when building trees and by definition we know that each tree has exactly $2^{\log_2 k_n}$ terminal nodes. Thus, if X has uniform distribution there will be on average about n/k_n observations per terminal node, and this is one of reasons why Random Forest has high performance in asset pricing problems.

As I previously stated the general principle on Random Forest is to aggregate a collection of random individual decision trees. "Since individual trees are drawn randomly , the constructed forest benefits from a more extensive exploration of the space of all possible tree predictors, which, in practice, results in better predictive performance" (M.R.Segal, 2004). The hyperparameters in random forest are used to improve the predictive accuracy of the model :

1. *n_estimators* hyperparameter, denotes the number of trees that the algorithm constructs before taking the average predictions. Generally, the larger the number of trees associated with it, the better the performance, but it also slows down the computation time.
2. *max_features* hyperparameter, indicates the maximum number of features that the random forest considers to divide a node.
3. *max_bootstrap* , if set up to 'False' then the entire dataset is used to construct the trees, otherwise the Boostrap sample is used.

Like any other method, Random Forest has both advantages and disadvantages. One of its main advantages is definitely its versatility, as it can be used for both regression and classification problems (M.R.Segal, 2004). Furthermore, Random Forest can easily handle high dimensionality data, as the algorithm is able to discard the least significant features. For this reason, random forest is considered as a dimensionality reduction method. Another advantage of this method is that in the case of missing values it has a relatively effective way of replacing them and at the same time maintaining good prediction accuracy. However, when a very high number of trees is used, the algorithm becomes extremely slow.

2.3 Boosted Regression Tree

The Boosted Regression Tree, BRT model, uses regression trees to solve regression problems. The BRT method relies on two powerful procedures: regression trees and boosting. ”Regression trees are decision-tree based models formed by dividing the predictor space into a number of mutually exclusive regions” (Elith et al, 2008). For regression problems, boosting is very similar to the ‘Gradient Descent’ so we can consider it as a numerical optimization technique for minimizing a loss function by adding, at each step, a new tree that best minimize the loss function. The hyperparameters in BRT are used to improve the predictive accuracy of the model :

1. *Learning_rate* hyperparameter : shrinks the contribution of each tree. If we decrease the learning rate then we increase the number of trees required and in general a lower learning rate is preferable. I estimate the optimal number of trees and learning rate using CV.
2. *Tree_complexity* hyperparameter : determines the number of nodes in each tree. This hyperparameter regulates the number of trees and is generally related to the learning rate.As the complexity of the tree increases, the learning rate must be decreased.

To find the optimal setting I used Cross Validation, which provides a means to test the model on small portions of the dataset, but still using all the data to fit the model (J. Elith, 2008). I will now briefly describe how this method works:

1. Randomly divide the dataset into n subset (usually n=10, but it can assume any value)
2. For each of the n training set there is one omitted subset that is used to test data.

3. The model training will start with the selected number of trees, then develop 'n' BRT models on each training set and test the data on the omitted set
4. The previous step is repeated by changing the hyper-parameters. Remember that at each step the average performance and standard errors are recorded.
5. Select the best combination to to find out what the best parameters are.

2.4 Neural Networks

The idea behind Neural Networks is to mimic the human brain, or more precisely how it works. The Perceptron is one of the simplest structures in Neural Network field. It is based on an artificial neuron called TLU whose inputs and outputs can only be numbers. The TLU calculates a weighted sum of its input features:

$$z = (w_1 \cdot x_1 + w_2 \cdot x_2 + ... + w_n \cdot x_n) = X^T \cdot W \tag{2.7}$$

Then it applies a step function to (2.7) and outputs the result.

$$h_w(u) = step(z) \tag{2.8}$$

The most common step function used in (2.8) are : the heaveside and Sign functions.
The simplest type of Neural Network is called the 'Single Layer Perceptron' which consists of a single output layer, as explained in Figure 2.2. Consequently, the inputs are inserted directly into the outputs through a series of weights. The network simply calculates a linear combination between the weights and the inputs. This type of Neural Network , being very simple, can be used only for binary linear classification. When the results of the linear combination exceed a threshold, the output will be the positive class, otherwise the result will be the negative one.

In Figure 2.2 the Neural Network consists of a single layer of TLUs, with each TLU (each neuron) connected to all input nodes, the layer is fully connected or a Dense layer. The output of a fully connected layer or Dense Layer is :

$$h_{w,b}(x) = \phi(XW + b) \tag{2.9}$$

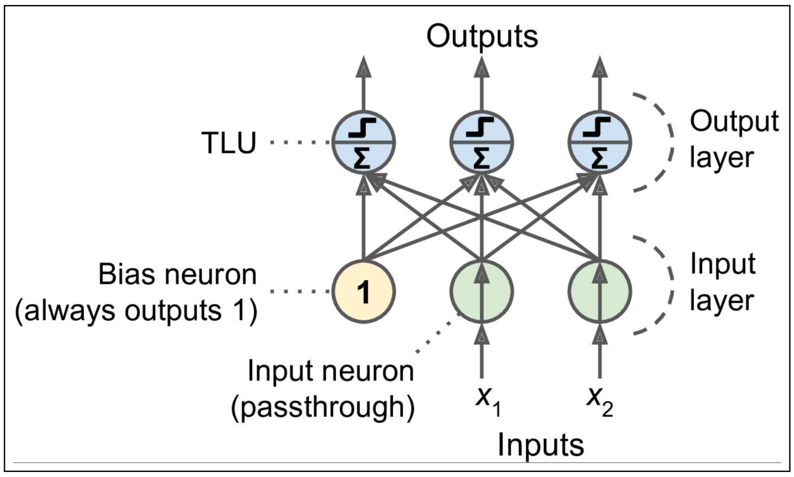


Figure 2.2: Single-Layer Perceptron

The figure shows a Single-Layer Perceptron network, which is made up of an output node layer; the inputs are feed directly to the outputs through a set of weights.

where X is the matrix of input features, W the connection weights, b the bias vector and ϕ the activation function.

2.4.1 MLP and Backpropagation Algorithm

A more complex algorithm is the Multi-Layer Perceptron (MLP), as reported in Figure 2.3. This is composed by an input layer, one or more TLU layers, called Hidden layers, and a last TLU layer called Output layer. Each layer is fully connected to the next layer and all layers have a bias neuron except the Output layer. In this type of Neural Network the signal flows in only one direction so it is classified as a Feedforward Neural Network (FNN).

To understand how an MLP is trained I need to introduce the Backpropagation algorithm, which was initiated by David E. Ruineihart (1985). This procedure handles one mini-batch at a time so the algorithm will go through the training set several times, each pass is called an epoch. The Backpropagation algorithm follows five steps:

1. Each mini-batch is passed to the input layer, which sends it to the first hidden layer
2. 'Forward Pass' : the algorithm calculates the output of all the neurons, the result is passed to the next layer, its output is calculated and passed to the next layer and so on, until the output of the last layer is obtained.
3. After the Forward pass, the algorithm uses a loss function to calculate the error of the neural

network.

4. Then, using the Chain Rule, the algorithm computes how much each output connection contributes to the error.

5. The algorithm performs a Gradient Descent step to adjust all the connection weights.

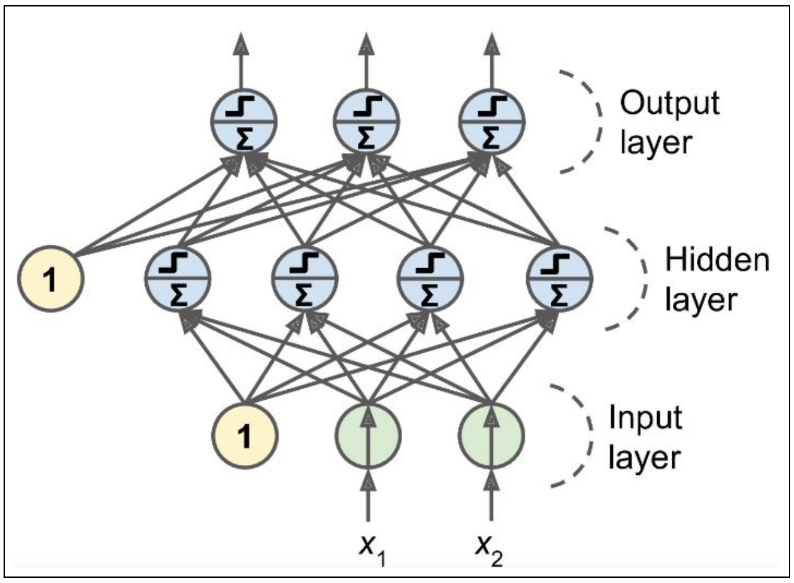


Figure 2.3: Multi-Layer Perceptron

The figure shows a Multi-Layer Perceptron network, which is made up of an input layer, several TLU layers and a final layer, the output layer. Each layer apart from the output layer incorporates a bias neuron.

2.4.2 Recurrent Neural Networks

One of the main differences between an RNN and an MLP is that only the first one is able to work on sequences of arbitrary length, so it has a much more flexible structure. Another very important difference between an MLP and an RNN is the way the signal circulates in the network. The MLP is classified as a Feed-Forward Network (FNN), so the signal only goes in one direction, whereas the RNN also has connections that point backwards, bidirectional.
The simplest possible RNN is composed by one neuron receiving inputs, producing an output and sending that output back to itself. In other words at each time step t every neuron receives both the input vector x_t and the output vector from the previous time step y_{t-1}. Each Recurrent neuron has two set of weights : one for the inputs, W_x, and the other one for the outputs of the previous step, W_y. The output vector of the whole recurrent layer can be described by the following equation:

$$Y_t = \phi(X_t W_x + Y_{t-1} W_y + b) \tag{2.10}$$

This type of network has a sort of short-term memory, capable of remembering only short term information.

A deep RNN, reported in Figure 2.4, is composed of multiple layers of Recurrent neurons. One of the problems of this type of structure is that the non-saturating activation functions can lead the network to be very unstable during training. To avoid this problem it is necessary to replace the Output layer with a Dense layer. It will allow the network to be slightly faster but most importantly it will give us the possibility to use any activation function.

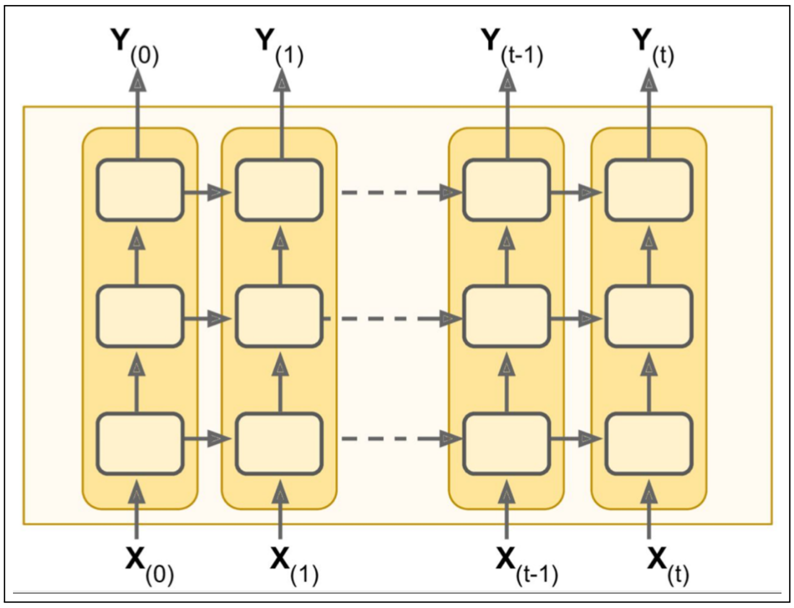


Figure 2.4: Deep RNN

The figure shows a Deep RNN, the network is composed by multiple layers of Recurrent neurons and it generates an output for each time step.

2.4.3 LSTM

As I stated earlier SimpleRNN cells have a short term memory which means that some information is lost at each time step. This type of problem is solved by long-term memory cells. In the LSTM (Long-Short term memory) the memory is divided into two vectors: h_t, the short-term memory, and c_t, the long-term memory.

Figure 2.5 shows us how an LSTM works : The long term memory c_{t-1} first goes through a forget gate, dropping some memories, and then adding new memory through an addition operation. So at each time step some memories are wasted and some are added. Before the output gate a copy of the long term state is passed through a tanh function, and this produces the short term memory, h_t , which is equal to the cell's output.

The short term memory h_{t-1} is fed together with the input vector x_t to four Fully connected layers:

1. The main layer is the one that outputs g_t, and it has the role of analyzing the current inputs x_t and the short term memory h_{t-1}
2. The forget gate f_t, controls which parts of the long term memory should be erased.
3. The input gate i_t, controls which parts of g_t should be added to the long term state.
4. The output gate o_t, controls the process to output h_t and y_t

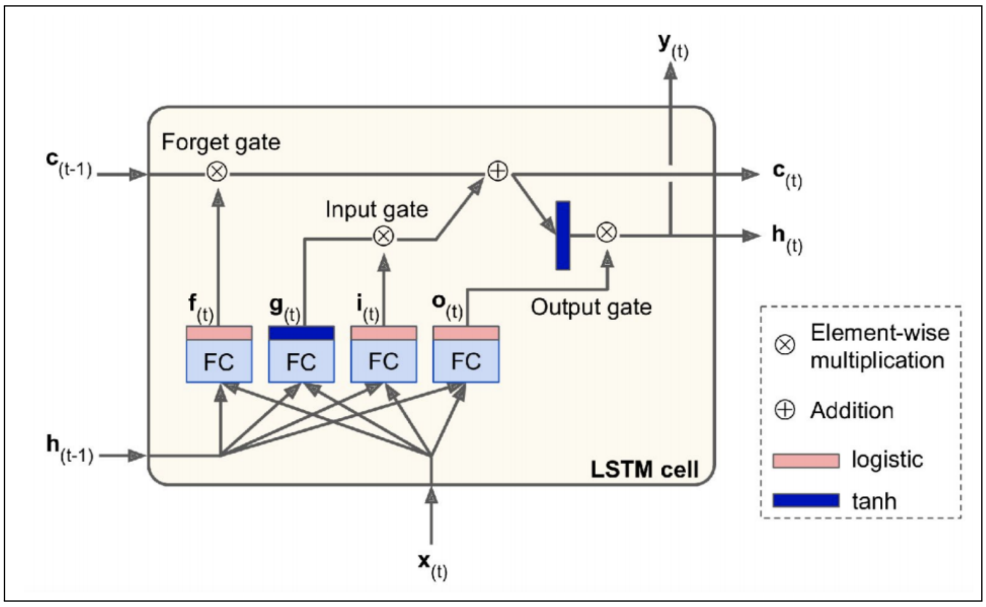


Figure 2.5: LSTM Architecture

The figure shows the architecture of an LSTM neural network, in particular the path that is taken by short-term h_{t-1} and long-term memory c_{t-1} .

Chapter 3
Empirical Study

3.1 Data and Samples

I selected monthly total individual equity returns from Bloomberg for firms listed in the NASDAQ, Dow Jones and also for the largest firms in terms of capitalization listed in the NYSE. The sample begins in May 1991 and ends in February 2021, totaling 30 years. In addition, I collected stock-level predictive characteristics and the most relevant macroeconomic variables in the US economy.

Let’s start by analyzing the most important macroeconomic variables of the dataset: non-farm payrolls (nfp) is the most important employment indicator in the US economy and it measures the monthly change in the number of employees in the US, excluding seasonal workers. This indicator can be considered particularly relevant because personal consumption make up more than the 69% of US GDP. As a consequence, the economy tends to shrink when more people lose their jobs. Other important indicators are: the CPI (cpi) to track the level of inflation, PMI (pmi) for Business Confidence and Housing (hs). This last indicator was selected not for the influence that the real estate sector has on the economy, but because a good level in Housing indicator means that consumers are confident enough to assume a 30-year mortgage.

Two commodities were also included in the dataset: Crude Oil (oil) and Gold (gold), I decided to choose these two because they are both the most imported and exported commodities in US economy. The Fama/French 3 Research Factors were also added to the dataset[1]. I also obtain the Treasury-bill rate to proxy for the risk-free rate from which I calculate individual excess returns.

[1]The data were collected from the web-site : Fama/French Data Library

In addition to the macroeconomic variables there are also some stock-level variables. One of the most important indicators is undoubtedly the 'Relative Strength Index' (mom). The RSI is generally used in technical analysis to measure the magnitude of recent price movements to assess overbought or oversold conditions in the price of a stock. The value of this indicator is between 0 and 100, (over 70 is overbought, under 30 oversold) and it is calculated using a particular formula consisting of two parts. The first part of the equation is described by the following formula :

$$RSI_1 = 100 - \frac{100}{1 + \frac{ag}{al}} \tag{3.1}$$

Where 'ag' and 'al' in the calculation is the average percentage of gain or loss during a look-back period, 14 days in my case. Once the look-back period (14 days) is passed we can move on to calculate the RSI using the second part of the equation :

$$RSI_2 = 100 - \frac{100}{1 + \frac{ag \cdot 13 + ag_0}{-(al \cdot 13 + al_0)}} \tag{3.2}$$

Another important indicator is the Relative Share Price Momentum, which measures the percentage change over the last 6 months in the one-month moving average of the share price against a reference index. Then I added other standard indicators such as : Volume (vol), Volatility(vlt) and EPS (eps).

3.2 Summary Statistics

Table 3.1 shows the mean and standard deviation values for the various features. In addition, the dataset has been divided into three groups - Nasdaq, Dow Jones, NYSE - to see how the mean and standard deviation values for the various features change depending on which group they belong to. Looking at the table we can notice that the three groups of stocks do not have great disparity for the mean and standard deviation values. Although for some features such as volume (vol) we can find a higher mean value in the NYSE stocks than the other two groups. This means that the NYSE stocks have a generally higher change in volume than the other stocks in the dataset. While the highest value for the standard deviation is reached by Nasdaq stocks, there is a greater dispersion of outcomes around the expected value.

A similar analysis can be made for the feature volatility (vlt), as the standard deviation of Blue Chips stocks is significantly higher than that achieved by Nasdaq and NYSE stocks. On the other hand, there are no major differences between the three groups in terms of the average value of the latter feature. Unlike all the other features for ’Non farm Payrolls’(nfp) and Inflation (cpi) I did not consider their change so they have a larger value than all the other features.

Table 3.1: Summary statistics

The table reports the mean and standard deviation for all the variables used in the analysis. The table reports summary statistics for monthly individual total stock returns from Bloomberg for companies listed on the NASDAQ, Dow Jones, and NYSE during the period May 1991 through February 2021. The table was divided according to the group to which the actions belonged in order to observe how the mean and standard deviation values of the various characteristics changed according to the group to which they belonged.

Features	Nasdaq		Blue Chips		NYSE	
	Mean	std. dev	Mean	std. dev	Mean	std. dev
nfp	74.50	1.585	91.92	1.221	85.33	1.326
pmi	53.01	4.644	52.7249	4.6449	52.8311	4.6808
mkt*	0.0081	0.0445	0.0074	0.0437	0.0078	0.0441
smb*	0.0019	0.0308	0.0017	0.0314	0.0018	0.0308
hml*	−0.006	0.0316	0.0009	0.0312	0.0002	0.0311
gold*	0.0075	0.1196	0.0078	0.1154	0.0069	0.1176
gold_v^*	0.0516	0.328	0.0553	0.3398	0.0526	0.337
oil*	0.0088	0.119	0.0088	0.1189	0.1104	0.8838
oil_v^*	0.0209	0.1794	0.0233	0.1883	0.1812	0.1396
hs*	0.0039	0.0787	0.0039	0.0787	0.0042	0.0808
cpi*	0.0208	0.0112	0.0223	0.0114	0.0215	0.0115
bdiy*	0.0311	0.2809	0.0244	0.2451	0.0269	0.2593
dxy*	0.0003	0.0219	0.0002	0.0228	0.0003	0.0225
vix*	0.0263	0.2372	0.0222	0.2195	0.0232	0.2261
vol*	0.0527	0.532	0.0434	0.329	0.0937	0.327
mom*	0.0042	0.0823	0.0054	0.0823	0.0043	0.0911
vlt*	0.0009	0.0479	0.0434	0.329	0.003	0.0617

3.3 Models Performance

3.3.1 Results

Figure 3.1 shows the comparison of the various algorithms implemented in Python used in terms of their out-of-sample R^2. I compared four methods in total, the first one is a simple Linear Regression that uses a Gradient Descent to optimize its parameters (Linear GD), then we have Random Forest (RF), Boosted Regression Tree (GBRT) and finally a Neural Network composed of three layers (NN1). Analyzing Figure 3.1, we can notice that three bars are associated with each method: The green bar and the yellow one show the R^2 obtained within subsamples that include only the top 100 securities (yellow bar) or the bottom 100 (green bar) by Earning per share (eps) while the blue bar indicates the R^2 obtained using the whole dataset.

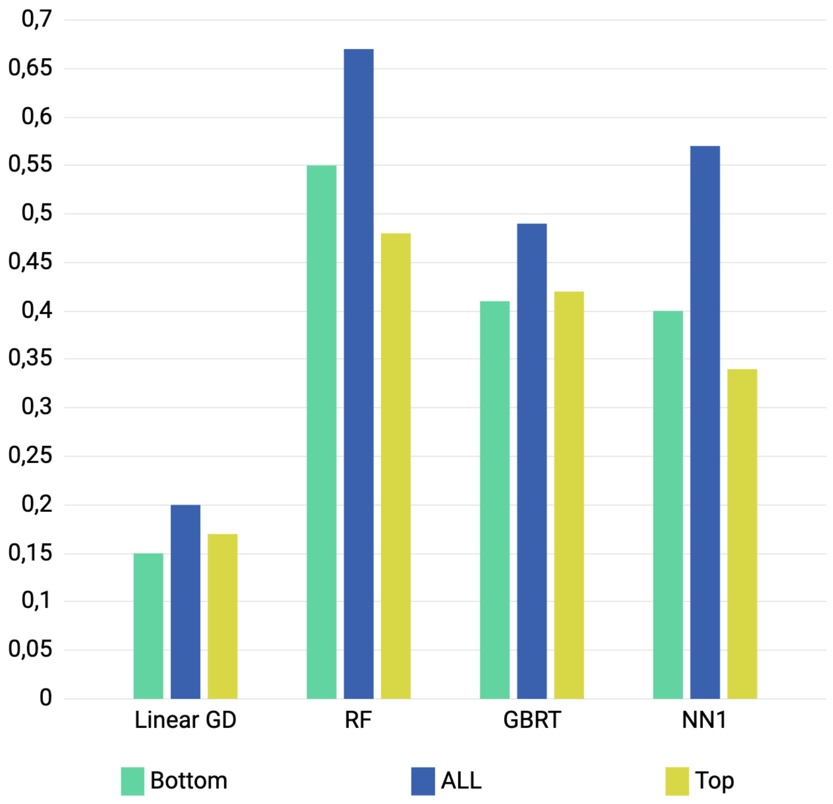


Figure 3.1: Empirical Results

The figure shows the comparison of the various algorithms in terms of their out-of-sample R^2. Three bars are associated with each method: the green and yellow bars show the R^2 obtained in subsamples that include only the first 100 securities (yellow bar) or the last 100 (green bar) per Earning per share (eps) while the blue bar shows the R^2 obtained using the entire dataset

If we analyze the results achieved using the whole dataset, we can see that RF is the best performing method, with an R^2 of 0.67%. The results are not in line with those obtained by Gu et al. (2020) , which finds that the use of neural networks provides the best results in terms of prediction. I can explain these differences with two reasons: I use a LSTM as a neutral network model, which is certainly a robust and efficient structure, as it still reaches an R^2 of 0.57%, but the results cannot be compared with those achieved by research such as that of Gu et al. (2020) who instead of an LSTM use a Feed-Forward network. The second reason is the number of observations: in order to achieve better results, at least twice observations would be needed.

Regarding GBRT, it performs much worse in terms of R^2 than the other Ensamble method (RF). But despite this, GBRT is the only method that manages to have a similar R^2 even when subsamples are used. So it is certainly a method that does not lose efficiency when the sample is reduced, but in this case it does not reach satisfactory R^2 levels. The last method I am going to analyse is Linear Regression (Linear GD) which is not very suitable for this type of analysis and in fact it reaches 0.21% R^2.

3.3.2 Advanced Model Results

The Deep Learning models used by Rapidminer software are very similar to those used by Gu et al. (2020). In fact, it uses a deep feed-forward neural network that is trained with stochastic gradient descent with the back-propagation algorithm. Using this typology of neural network there is an improvement ,in terms of R^2, with respect to the LSTM architecture. In fact the FNN achieves a R^2 of 0.70% compared to the previous 0.57% for the LSTM structure (NN1). While for Random Forest (RF) the results achieved by RapidMiner are practically identical to those of Python. Therefore the best performing method is no longer Random Forest but Neural Networks.

The performance of the Boosted Regression Tree (GBRT) and Linear GD remains almost unchanged. From these results we can conclude that there is not one single model which is optimal in every condition. Tbest Ensamble method is certainly Random Forest (RF), which achieves performances in some cases even higher than those of Neural Networks. But we can consider RF as the best method only in the case in which the neural networks used are LSTM or simpler structures. In the case in which more complex structures are used, NN1 reaches levels of R^2 that RF cannot reach.

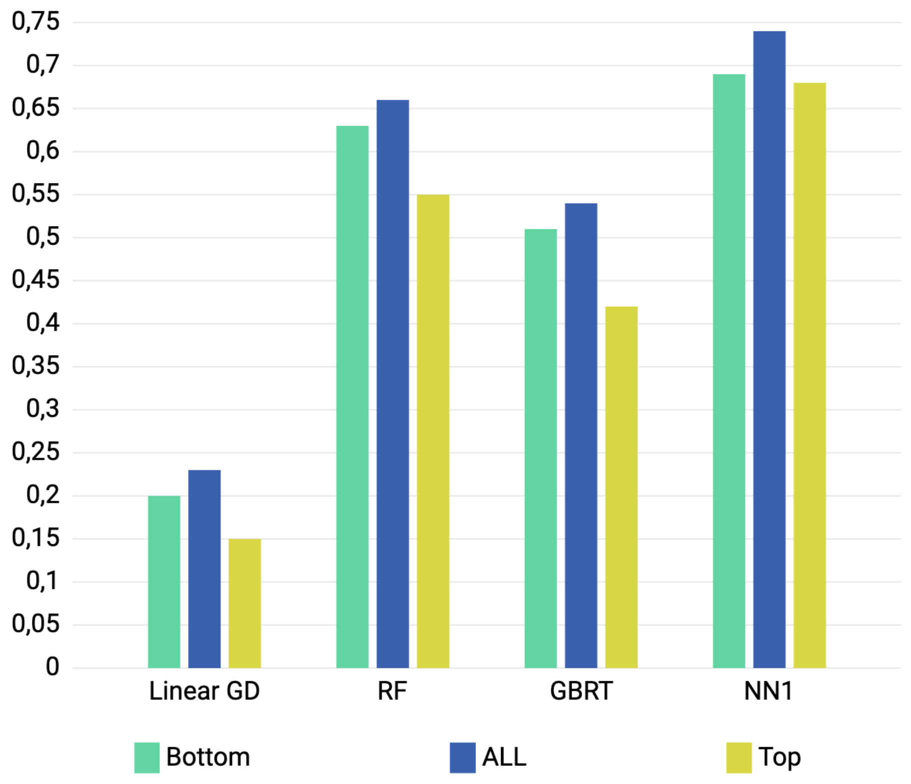


Figure 3.2: Advanced Model Results

The figure shows the comparison of the various algorithms in terms of their out-of-sample R^2 achieved using more advanced methods (see the legend to Figure 3.1)

3.4 Features Importance

The purpose of my work is not to interpret the various machine learning techniques, but to identify the features that have the most influence on the dependent variable. For each model taken into analysis I calculated the reduction in R^2 by setting all predictors to zero. The first result of this procedure is shown in Figure 3.3 which shows the top ten features for each method. We can immediately see that practically all methods give particular importance to the momentum indicator (mom) and (gv30_mom). This result is in line with Gu et al (2020). As far as the other features are concerned, this image does not give us a clear idea of which are the most relevant. We can therefore move on to the analysis of Figure 3.4.

Figure 3.4 shows the general feature rankings. This result was obtained by first ranking the importance of the features for each method (ordering them with a rank attribute), then once normalised I sum up their rank. Looking at the figure it is clear that the most important indicator, as mentioned above, is the momentum (mom) which reaches a correlation of 0.39 with the boosted regression tree method, while with random forest it reaches a very low correlation, a value of 0.04. Random forest seems to be the only method that does not give importance to this indicator because with

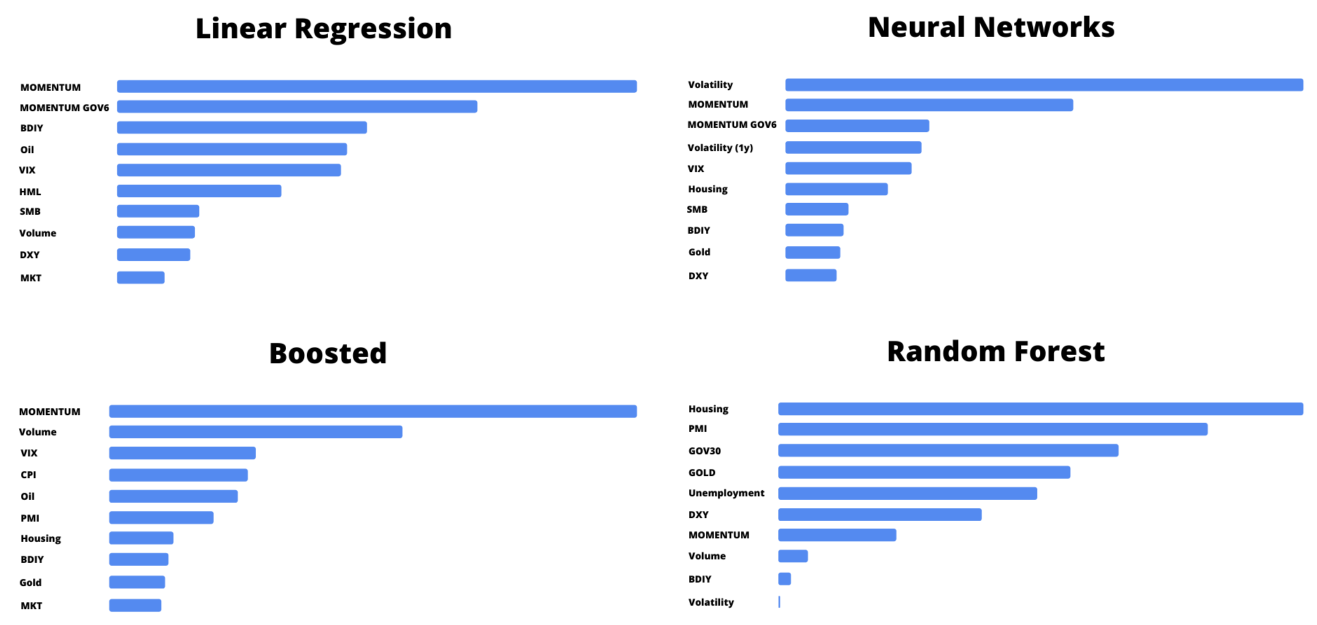


Figure 3.3: Top 10 features for each method

The figure shows the variable importance for the top 10 most influential variables using liner regression, boosted regression tree (Boosted), random forest, deep learning model (Neural Networks)

linear regression it reaches a correlation of 0.29 while with neural networks it reaches a value of 0.15. The second most important indicator is housing (hs) which achieves significant results only with random forest and neural networks where it achieves a correlation of 0.35 and 0.11 respectively. Then there is the volume indicator (vol) which reaches good results in terms of correlation with the methods Boosted regression tree and neural networks with values of 0.15 and 0.23 respectively. The last method to reach significant results in at least two methods is another momentum indicator (gov30_mom) which reaches a correlation of 0.16 with both linear regression and neural networks methods. Regarding the other features, they do not reach very significant results, with some exceptions as in the case of the pmi and gold indicators which reaches a correlation of 0.20 with the random forest method. Regarding the other features, they do not reach very significant results, with some exceptions as in the case of the pmi and gold indicators which reach a correlation of 0.20 and 0.14 respectively with the random forest method.

I conclude by saying that the analysis shows that the best stock-level characteristics are: momentum indicators and the change in the volume of a stock. On the other hand for the macroeconomic characteristics the analysis is not very clear on the housing indicator so the vix index can be considered as the most relevant macroeconomic variable.

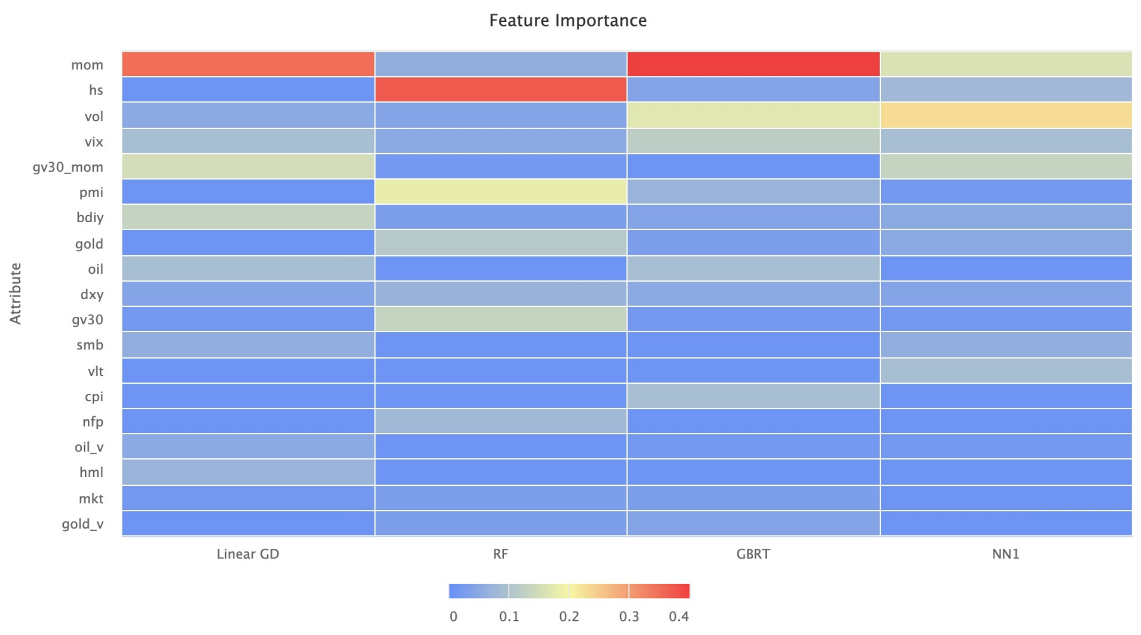


Figure 3.4: Top 10 features for each method

The figure shows the general feature rankings. Features are sorted summing their ranks over the models. Each column represents a model while the various colors within each column indicate the most influential variables (red) to the least influential variables (yellow).

3.5 Summary of Results

Table 3.2 shows the results achieved by the various methods used in my analysis. The columns of the table in fact show the values reached in the train and in the test for the mean square error, the root mean square error and the R^2. As I stated in previous sections, the model that performs best at R^2 level is the FNN. The results achieved at R^2 level by almost all methods, except for linear regression, are very similar both on the test and on train. This means that the models generalize well since the generalization gap is generally low compared to the R^2 levels achieved.

Looking at the results achieved for MSE by linear regression, it is clear that the error is particularly significant compared to those achieved by non-linear methods. This shows how linear regression is unsuitable for solving this type of problem. The other methods instead manage to reduce the error, in particular neural networks can reach very low levels of MSE in both train and test. This shows that the R^2 level reached by these structures is not due to overfitting.

Regarding Ensamble methods, Random Forest performs much better than Boosted Regression Tree in terms of both R^2 and MSE. Both methods however reach similar values both in the train and in the test set.

Table 3.2: Summary results

The table reports the results achieved by all the methods used in the analysis. The table reports the summary results for monthly individual total stock returns from Bloomberg for companies listed on the NASDAQ, Dow Jones, and NYSE during the period May 1991 through February 2021. Each row of the table is associated with a method while the columns show the values reached in the train and in the test for the mean square error (MSE), the root mean square error (RMSE) and the R^2. The best results at the R^2 level, both in the train and in the test, are achieved by the Feed-Forward network, NN1 (FNN), and by the random forest, RF. While for the MSE the results caught up from the two neural nets, NN1 (FNN) and NN1 (LSTM), are the best ones.

Method	MSE		RMSE		R^2	
	Train	Test	Train	Test	Train	Test
Linear GD	1.45	7.89	1.20	2.81	0.40	0.20
RF	0.0456	0.0876	0.2135	0.2959	0.71	0.67
GBRT	0.0756	0.167	0.275	0.408	0.59	0.48
NN1 (LSTM)	0.0146	0.0196	0.382	0.140	0.66	0.57
NN1 (FNN)	0.002 38	0.005 45	0.0487	0.0738	0.74	0.70

.

Conclusion

This thesis studies the predictive ability of some Machine Learning and Neural Network methods on the US market in the field of ”Empirical Asset Pricing”. My analysis is aimed at finding two types of results: the first one is to identify the best performing methods in terms of R^2 while the second one is to understand which features in the dataset affect the dependent variable the most.

Regarding the best performing methods, the best results were achieved by the Feed-Forward neural network. The results obtained are similar to those of Gu et al. (2020), in which FNN achieves similar results at R^2 level. Another type of neural network, the LSTM, was also used in the analysis, but it fails to achieve similar results to FNN in terms of R^2. As far as Ensamble methods are concerned, the best results are achieved by Random Forest which reaches much higher levels of R^2 than the Boosted Regression tree. Also Gu et al. (2020) shows how Random Forest is one of the best methods in the field of Empirical Asset Pricing.

As I mentioned one of the purposes of my work is to understand which features of the dataset most influences the dependent variable. The findings of my analysis indicate that among the stock level features the momentum indicator is the most influential one while for the macroeconomic variables the indicators: ’housing’, ’Volume’ and ’Vix’ are the most influential ones. The results obtained are in line with those obtained by Gu et al. (2020) which demonstrate that the most influential characteristics are the momentum indicators and in my analysis they reach a correlation of 40% with the dependent variable.

The field of empirical asset pricing is not yet fully exploiting the potential of machine learning techniques and in particular of neural networks. Therefore , the development of these techniques can be an interesting topic of future research.

Bibliography

[1] J. David Cummins. Asset Pricing Models and Insurance RateMakings, 125–166, ASTIN bulletin, 1990.

[2] Gu Shihao, Bryan Kelly, and Dacheng Xiu. Empirical Asset Pricing via Machine Learning, 2223–2273, The review of financial studies, 2020.

[3] Fama, E. F., and K. R. French. Dissecting Anomalies, 1653–78., TJournal of Finance, 2008.

[4] Lewellen. The Cross-section of Expected Stock Returns, 1–44, Critical Finance Review, 2015.

[5] Welch, I., and A. Goyal. A comprehensive look at the empirical performance of equity premium prediction, 1455–508, Review of Financial Studies, 2008.

[6] Koijen, R., and S. V. Nieuwerburgh. Predictability of Returns and Cash Flows, 467–91, Annual Review of Financial Economics, 2011.

[7] Harvey, C. R., Y. Liu, and H. Zhu. And the cross-section of expected returns, 5–68, Review of Financial Studies, 2016.

[8] Giglio, S. W., and D. Xiu. Asset Pricing with Omitted Factors, Tech. Report, 2016.

[9] Bryan T.Kelly,Seth Pruitt,Yinan Su. Characteristics are covariances: A unified model of risk and return, 501-524, Journal of Finance and Economics, 2019.

[10] Moritz, B., and T. Zimmermann. Tree-based conditional portfolio sorts: The relation between past and future stock returns, Working Paper, Ludwig Maximilian University of Munich, 2016.

[11] Cao,Q., Parry,M. E. Leggio,K. B. The three-factor model and artificial neural networks: predicting stock price movement in China,25–44, Annals of Operations Research, 2011.

[12] Messmer, M. Deep learning and the cross-section of expected returns., Working paper, 2017.

[13] Feng, G., He, J., Polson, N. G Deep learning for predicting asset returns, arXiv:1804.09314, 2018.

[14] Fama, E. F., and K. R. French. Common risk factors in the returns on stocks and bonds, :3–56, Journal of Financial Economics, 1993.

[15] Fama, E. F., and K. R. French. A five-factor asset pricing model, 1–22,Journal of Financial Economics, 2008.

[16] Fernandez Viviana. The CAPM and value at risk at different time-scales, 203–219, International Review of Financial Analysis, 2006.

[17] David E. Ruineihart, Geoffrey E. Hint.., and Ronald J. Williams. LEARNING INTERNAL REPRESENTATIONS BY ERROR PROPAGATION, INSTITUTE FOR COGNITIVE SCIENCE, 1985.

[18] Segal, M. R. Machine Learning Benchmarks and Random Forest Regression, UCSF, 2004.

[19] G. Biau, L. Devroye, and G. Lugosi. Consistency of random forests and other averaging classifiers, Journal of Machine Learning Research, 2008.

[20] J. Elith, J. R. Leathwick, and T. Hastie. A working guide to boosted regression trees, 802–813, Journal of Animal Ecology, 2008.

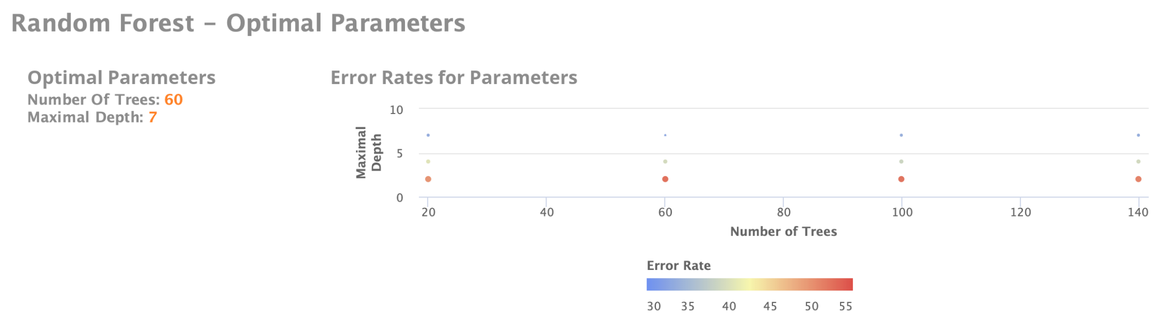
[21] DAVID E. RAPACH, JACK K. STRAUSS, GUOFU ZHOU International Stock Return Predictability: What Is the Role of the United States?, The Journal of Finance, 2013.

[22] Wayne E. Ferson, Campbell R. Harvey Economic, Financial, and Fundamental Global Risk in and Out of the Emu, NBER Working Paper No. w6967, 1999.

Appendix A

RapidMiner

A.1 Random Forest



Optimal Parameters
Number Of Trees: 60
Maximal Depth: 7

Figure A.1: Random Forest Model specifics

Figure A.2: Random Forest Model specifics

The figure shows the optimization of random forest hyperparameters performed by the RapidMiner software.

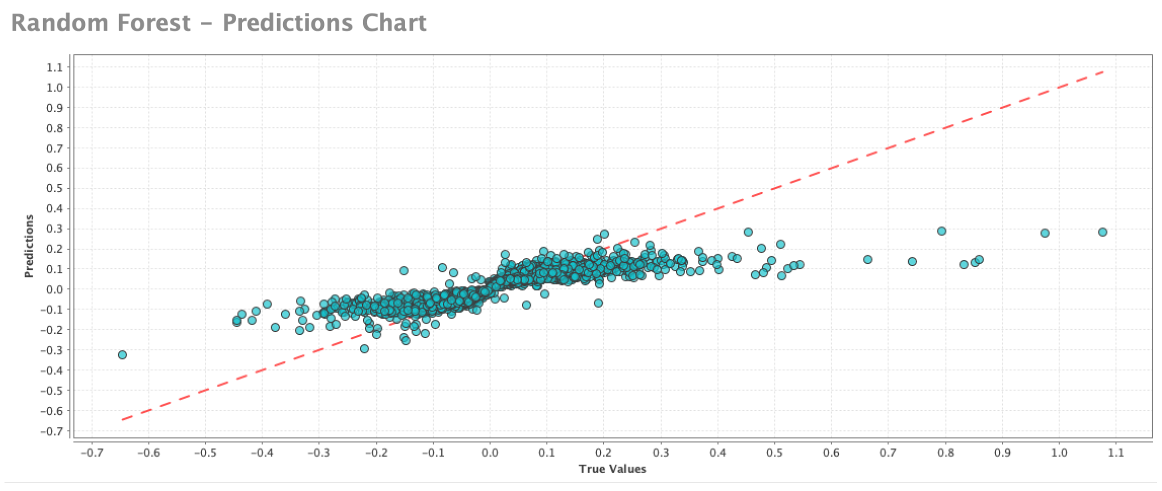
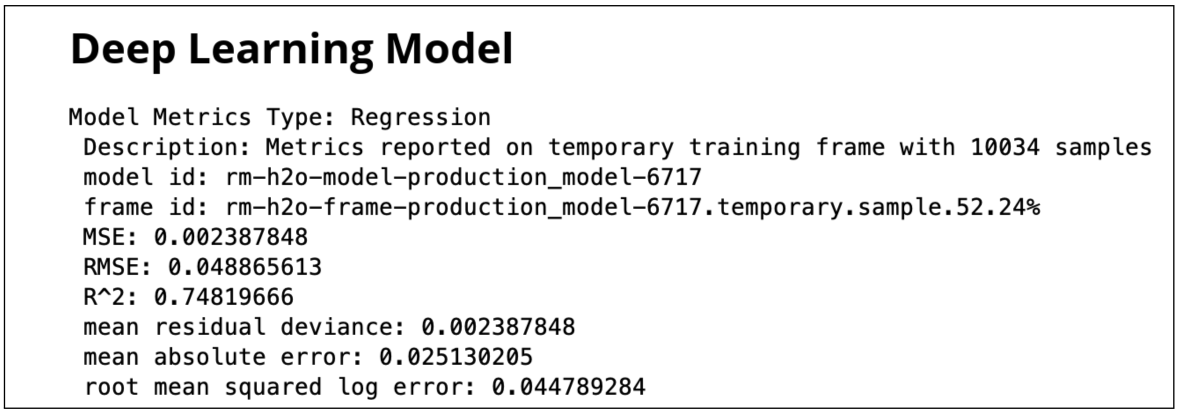


Figure A.3: Random Forest - Prediction Charts

The figure shows the prediction accuracy achieved in the test set by Random forest. An optimal model would follow the dashed red line.

A.2 Deep Learning



Deep Learning Model

```
Model Metrics Type: Regression
 Description: Metrics reported on temporary training frame with 10034 samples
 model id: rm-h2o-model-production_model-6717
 frame id: rm-h2o-frame-production_model-6717.temporary.sample.52.24%
 MSE: 0.002387848
 RMSE: 0.048865613
 R^2: 0.74819666
 mean residual deviance: 0.002387848
 mean absolute error: 0.025130205
 root mean squared log error: 0.044789284
```

Figure A.4: Deep Learning Model specifics

The figure shows the results achieved by the Feed-Forward neural network, using the RapidMiner software. It shows the values reached for the mean square error, the root mean square error and R^2

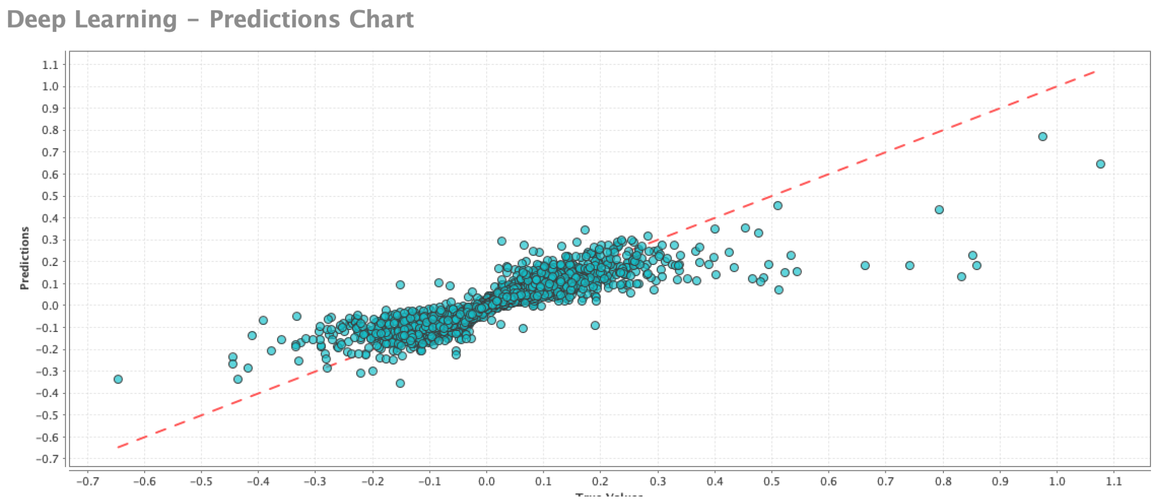
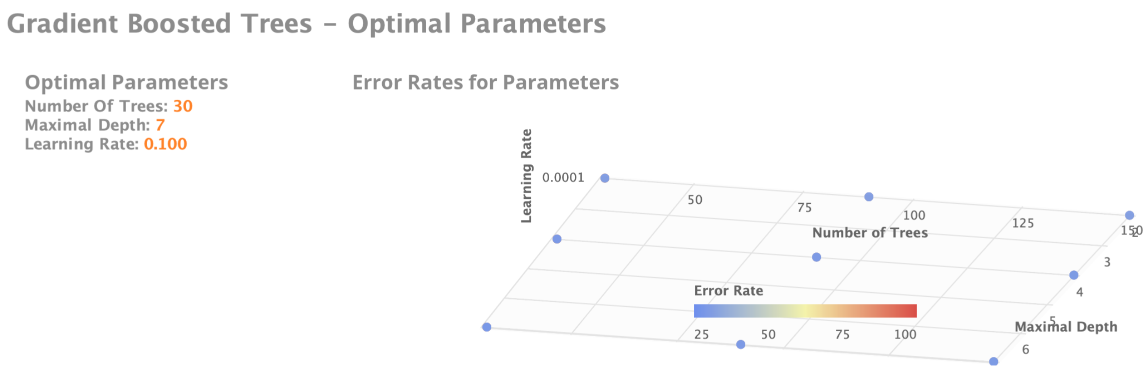


Figure A.5: Deep learning - Prediction Charts

The figure shows the prediction accuracy achieved in the test set by the Feed-Forward neural network. An optimal model would followthe dashed red line.

A.3 Boosted Regression Tree


Gradient Boosted Trees – Optimal Parameters

Optimal Parameters
Number Of Trees: 30
Maximal Depth: 7
Learning Rate: 0.100

Figure A.6: Boosted Regression Tree Model specifics

The figure shows the optimization of the boosted regression tree model hyperparameters performed by the Rapid-Miner software.

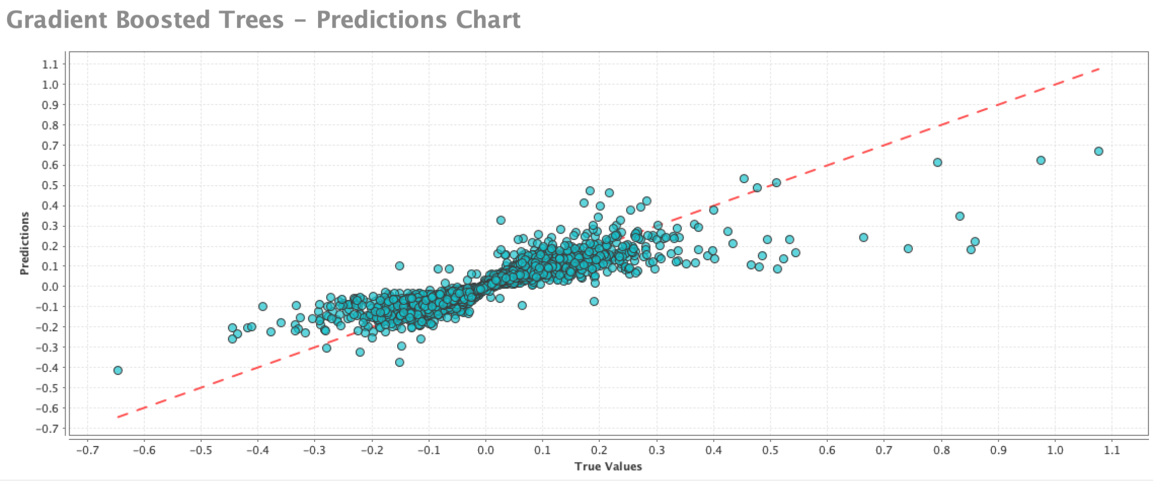


Figure A.7: Boosted Regression Tree - Prediction Charts

The figure shows the prediction accuracy achieved in the test set by the boosted regression tree model. An optimal model would follow the dashed red line.

Appendix B

Knime

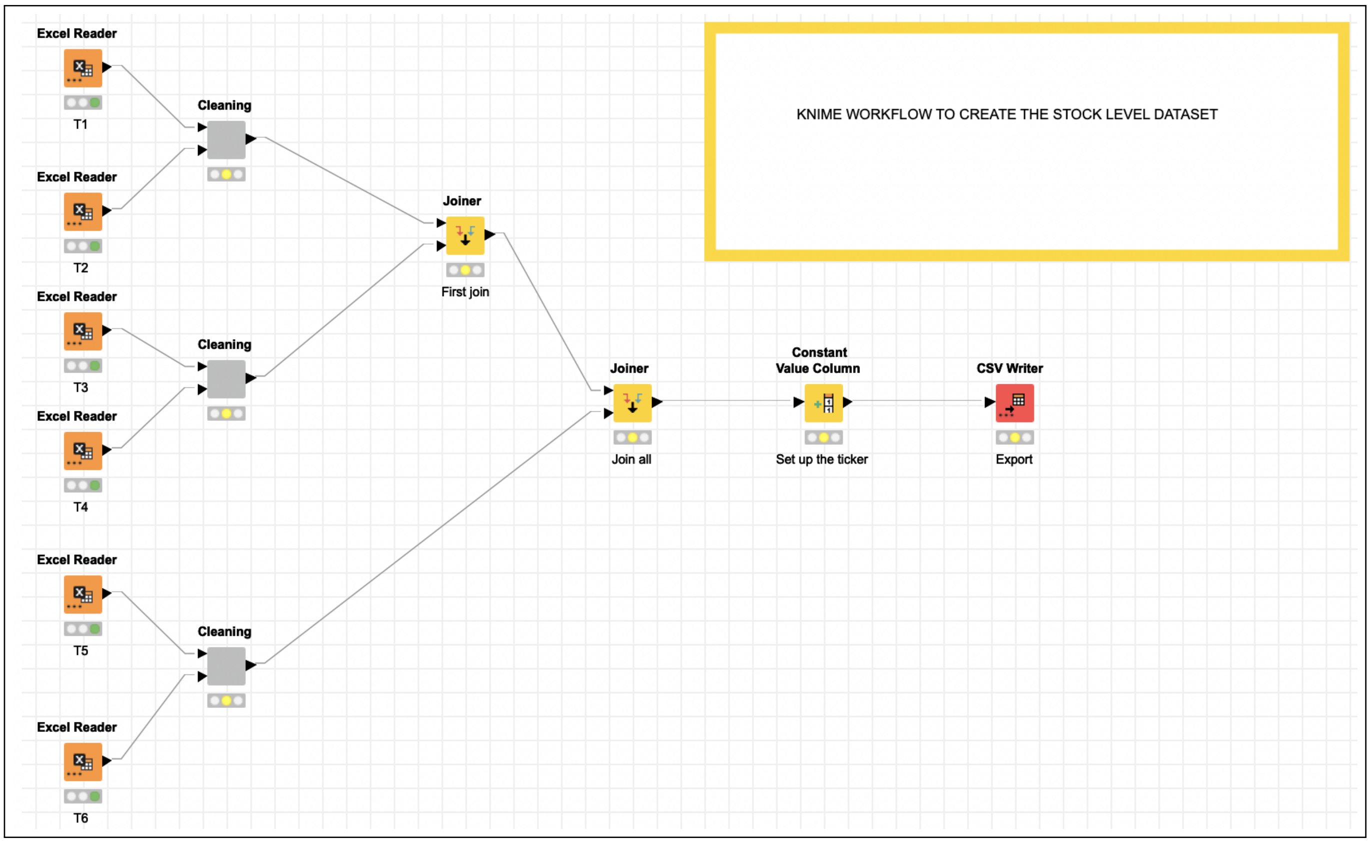
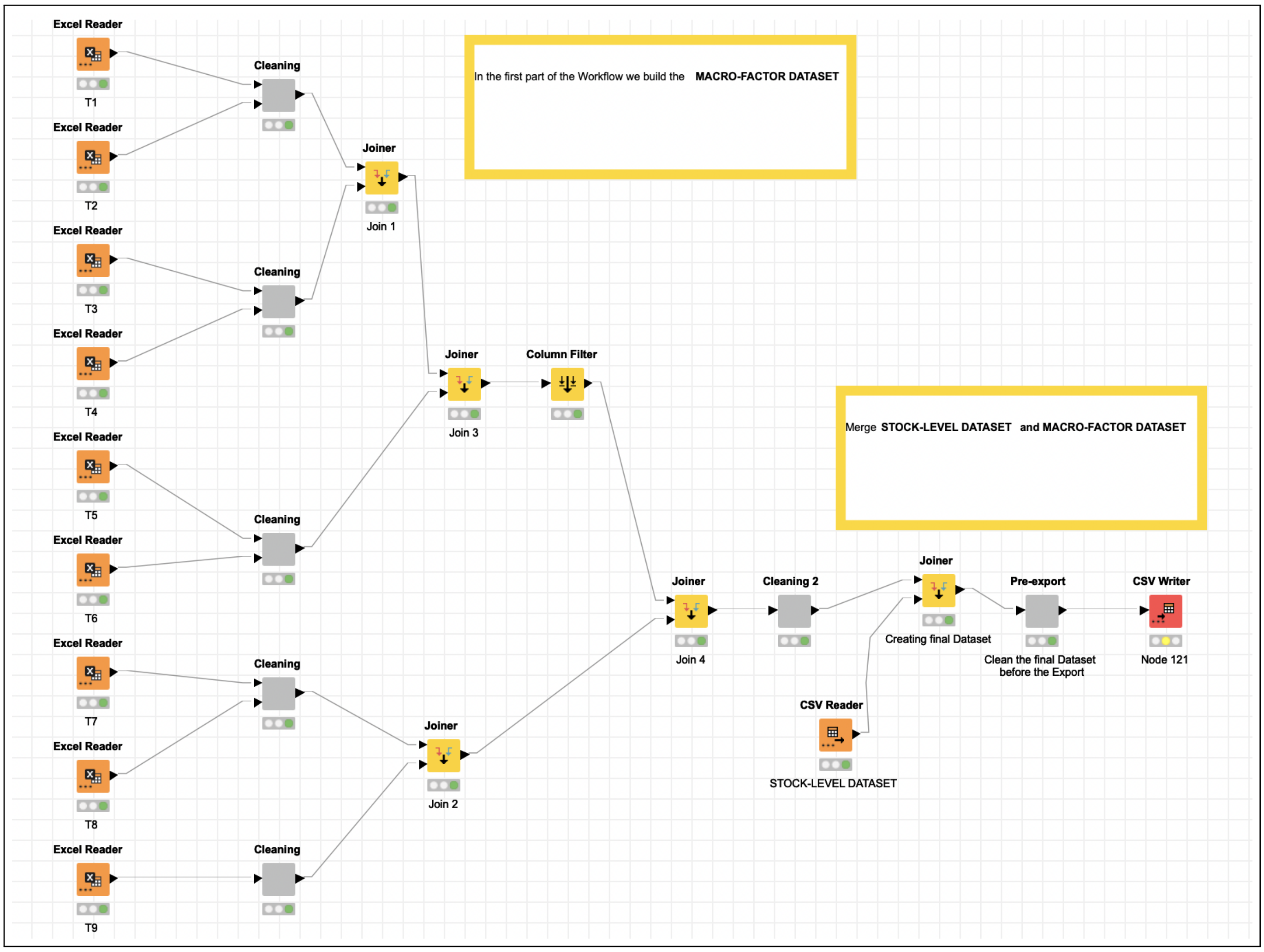


Figure B.1: Building Dataset - Stock Level

The figure shows the Knime workflow to create the stock-level dataset. The data is first imported via the 'excel reader' node, then it is processed by a metanode to clean the data. After the Data Cleaning part the join of the various datasets is made.



In the first part of the Workflow we build the **MACRO-FACTOR DATASET**

Merge **STOCK-LEVEL DATASET** **and MACRO-FACTOR DATASET**

Figure B.2: Building Dataset - Macro and Total

The figure shows the Knime workflow to create the macro-level dataset, which is then merged with the stock-level dataset to obtain the final dataset.

Appendix C

Results Feature Importance

Table C.1: Results feature importance

The table reports the correlation with the dependent variable of the features included in the dataset for each method used. The table reports the feature importance results for monthly individual total stock returns from Bloomberg for companies listed on the NASDAQ, Dow Jones, and NYSE during the period May 1991 through February 2021.

Feature	Linear regression	Random Forest	Boosted regression tree	Neural network
mom	0.29	0.04	0.39	0.15
hs	0.01	0.35	0.02	0.11
vol	0.03	0.03	0.15	0.23
vix	0.10	0.03	0.14	0.11
gv30_mom	0.16	0.01	0.01	0.16
pmi	0.01	0.20	0.04	0.01
bdiy	0.16	0.02	0.02	0.06
gold	0.01	0.14	0.02	0.05
oil	0.12	0.01	0.13	0.01
dxy	0.02	0.08	0.03	0.03
gv30	0.01	0.18	0.01	0.01
smb	0.05	0.01	0.01	0.04
vlt	0.01	0.01	0.01	0.12
cpi	0.01	0.01	0.13	0.01
nfp	0.01	0.12	0.01	0.01
oil_v	0.03	0.01	0.01	0.01
hml	0.04	0.01	0.01	0.01
mkt	0.01	0.02	0.02	0.01
gold_v	0.01	0.02	0.02	0.01