Commodity Futures Pricing Via Machine Learning: An Empirical Approach

Master Thesis I:

Regression-Based Approach to Futures Pricing via Machine Learning

Submitted to Università Commerciale Luigi Bocconi (Milan, Italy) in June 2021

Defended at Università Commerciale Luigi Bocconi (Milan, Italy) on July 20, 2021

Master Thesis II:

Classification-Based Approach to Futures Pricing via Machine Learning

Submitted to Stockholm School of Economics (Stockholm, Sweden) in July 2021

Anton Nartov

M.Sc. in Finance Double Degree

Stockholm School of Economics /

Università Commerciale Luigi Bocconi

July 2021

Abstract

The goal of this thesis is to use established methodologies in the field of machine learning in finance to extend the list of current applications to commodity futures, reviewing and refining the established empirical approaches to return forecasting and hyperparameter optimization. We thus investigate the out of sample predictive accuracy of tree-based machine learning (ML) techniques and neural networks applied to monthly commodity futures returns, relying on conventional regression and classification accuracy metrics. We find that a large selection of machine learning techniques cannot consistently outperform the benchmark AR(1) model when applied to monthly data, and that there is no specific ML method suitable to all the analyzed commodity series at once. While we also find potential portfolio-level investor gains from using ML techniques, the robustness of these gains is questionable. Finally, we suggest an updated approach to hyperparameter optimization and find that different commodity series have to be modelled separately, which is suggested by large differences in the optimal architectures estimated by our Grid Search and Bayesian Optimization tuner algorithms.

Keywords: Machine learning, asset pricing, neural networks, decision trees

Supervisors: Dr. Tobias Sichert (SSE) and Prof. Massimo Guidolin (Bocconi University)

Table of Contents

MASTER THESIS I

1. Introduction

"Machine Learning Revolution" is a term that is commonly applied in scientific publications and media in order to describe a dramatic increase in the use of algorithms in different scientific applications, ranging from mathematics and physics to social sciences. Finance did not become an exception, and different methodologies have been developed for such purposes as risk management, macroeconomic analysis, and, of course, active investing (Dixon and Halperin, 2019). A significant part of the existing literature finds increased predictive power and potential portfolio gains from the use of machine learning (henceforth, ML) techniques, which is fostering further research in this field.

The purpose of this thesis is thus to test the out of sample forecasting power of a wide selection of ML methods applied in the context of commodity futures returns. We rely on the methodology developed by Gu et al. (2020) who report high out of sample investor gains from using ML techniques, applying this methodology to the case of commodity futures. We also rely on previous academic research to review the approach to hyperparameter optimization suggested by Gu et al. (2020), adding a stronger focus on neural networks. The study by Guidolin and Pedio (2020), in which the authors use hidden Markov chain models and stepwise variable selection algorithms to forecast returns of commodity futures, will be our main guide in adapting the methodology by Gu et al. (2020). Namely, this research will determine our choice of variables and provide a benchmark for assessing the out of sample performance of our models.

Even though we aim to adapt the ML approach to return forecasting suggested by Gu et al. (2020) to a different asset class, the contribution of the thesis to the existing body of knowledge is not limited to this. One of our key contributions is a more careful and consistent approach to model specification and hyperparameter tuning: we rely on such optimization algorithms as Grid Search and Bayesian Optimization to tune the hyperparameters of our models. Hyperparameter optimization will be explained in more detail in the Methodology section. Moreover, we extend the list of models proposed by Gu et al. (2020) by adding Long Short-Term Memory (LSTM) neural networks, explaining why this class of models carries

the potential to achieve a higher out of sample forecast accuracy compared with more conventional, feedforward neural networks.

We will focus on two broad classes of ML models within this thesis: tree-based methods and neural networks. Tree-based methods include random forests and boosted decision tree algorithms, and neural networks include feedforward and recurrent neural networks. Each model will be tested on different sets of prediction variables, discussed in more detail in the Dimensionality Reduction subsection of the methodology chapter. We will estimate three tree-based methods on three sets of variables, and two types of neural networks on four sets of variables, which gives us a total of 17 models the out of sample forecasting accuracy of which will be compared against the AR(1) benchmark.

Guidolin and Pedio (2020) show that hidden Markov chain models and stepwise variable selection algorithms cannot consistently outperform the AR(1) benchmark, so a closer analysis of this conclusion would be beneficial. Namely, it is unclear whether the reasons why the models fail to show consistent outperformance are driven by model selection, the structural instability of the data generating process, or a combination of both. We will thus focus on the methodology in Guidolin and Pedio (2020) and their suggested set of predictors, aiming to explore whether tree-based methods and neural networks can show a better out of sample forecasting performance.

Hyperparameter optimization is another contribution of the current thesis: to the best of our knowledge, hyperparameter tuning algorithms have not been analyzed and used to optimize neural networks aiming to forecast the returns on commodity futures. For example, the papers by Struck and Cheng (2020) have used ML methods to predict commodity futures returns; however while they use tree-based methods and neural networks on portfolios of commodities, they state that default model parameters are used, which completely neglects the hyperparameter tuning stage. While it is commonly established that hyperparameters are a crucial element of applying ML algorithms (see, for example, Pereyra et al., 2017), this aspect of the methodology often has not received adequate scrutiny in financial applications. In turn, Gu et al. (2020) approach hyperparameter tuning in a more academic and consistent way. While they use exhaustive grid searches for tree-based methods, their neural network

hyperparameter tuning stage is limited to five architectures, which can be justified by the massive size of the dataset used in their study.

In order to choose the hyperparameter values within the five architectures, Gu et al. (2020) rely on more conventional approaches to hyperparameter optimization, such as the pyramid rule for neurons proposed by Masters (1993). The rule states that a rough approximation of the optimal number of neurons in the hidden layer can be found by taking the square root of the product of the number of neurons in the input layer and the output layer. However, despite its efficiency in empirical applications, the pyramid rule only serves as a first approximation and does not suggest an undisputedly optimal solution. Given the common opinion within ML that hyperparameter optimization is "more art than science", our goal is not to provide exhaustive evidence that optimization algorithms can outperform established academic approaches. We are rather trying to empirically test whether hyperparameter tuning algorithms can suggest a better model specification compared with more conventional rules and notions. This way, an undisputable advantage of our approach is that we test a significantly larger number of architectures: while Gu et al. (2020) tested five different architectures of feedforward neural networks, our approach to dimensionality reduction and the use of tuning algorithms will enable us to find optimal models from over 1,000 different specifications.

In this thesis, we work with fourteen monthly commodity futures return series: Light Crude Oil, Corn, Soybeans, Wheat, Coffee, Cocoa, Sugar, Cotton No.2, Gold, Silver, Platinum, Frozen Orange Juice, Lumber, and Live Cattle. Descriptive statistics of these series can be found in Table 1. The sample spans the period January 31, 1989 - June 30, 2020, which gives us a total of 377 observations per series. While one could argue that "data-hungry" ML algorithms perform best on a significantly larger amount of data, an important advantage of shorter series is the enormous reduction in computational costs. When working with smaller datasets, modern hyperparameter optimization algorithms allow for a significantly more careful choice of hyperparameters that provide a better balance between over- and underfitting models. Indeed, while ML techniques do perform better on large data sets, the advantage of working with a relatively short series of monthly returns is that we can afford a larger number of simulations and model specifications. This will allow us to minimize the

probability that potential underperformance of our models is driven by parameter misspecification, making the limited sample size and the complexity of dependencies within the data the only factors that can affect the performance of our models relative to the benchmark.

Consistently with Gu et al. (2020), we close our empirical work with a portfolio exercise in which we explore whether the ML algorithms are capable of outperforming the benchmark in terms of realized portfolio performances. In this exercise, the models that displayed the lowest out of sample forecast errors will be used to forecast the returns of commodity futures. The forecasts will be ranked from highest to lowest, and the investor will take a long position in the top four commodities while taking a short position in the bottom four commodities. However, an important restriction is that due to the positive correlation between commodity futures (see the evidence in Table 2), there is no guarantee that there will be four positive and four negative forecasts at the same time. To account for this issue, we set a restriction that will only allow the algorithm to assume a long position in a commodity futures contract if the forecasted return is positive, and vice versa.

The research questions of this thesis are thus the following:

1. How do tree-based methods and neural networks perform relative to the benchmark AR(1) model in the context of forecasting returns of commodity futures?
2. Given that Gu et al. (2020) show that their methodology leads to larger portfolio performance gains, how will it perform when a different asset class is analyzed?

We find that the set of analyzed ML techniques cannot consistently outperform the AR(1) benchmark, and that the complexity of the dependencies between commodity futures and forecast variables forces the forecast from most models to converge to their long-run mean, suggesting that the analyzed data frequency is not suitable for neural networks and tree-based methods. Our findings are consistent with Guidolin and Pedio (2020); however, we believe that our approach to hyperparameter optimization can be used in further research on the topic of forecasting returns of commodity futures. While we also find potential portfolio performance gains from applying ML methods, we cannot vouch for the robustness

of this forecast, which makes real-world applications of our methods for commodity futures return prediction infeasible.

The rest of the thesis is structured as follows. In Section 2, we survey the literature on return predictability and previous attempts to apply ML models in different areas of finance. In Section 3, we describe the methodology, dimensionality reduction techniques, the key principles behind the tree-based methods and neural networks used in this thesis, and our approach to hyperparameter optimization, showing how our research deviates from and contributes to the existing body of literature. In Section 4, we describe the set of commodity futures whose returns we aim to forecast, and the set of aggregate and commodity-specific predictors. In Section 5, we present the empirical results of our study, analyzing both the out of sample forecast accuracy metrics and the performance of our models at a portfolio level. Finally, Section 6 concludes and provides suggestions for further research.

2. Literature Review

This section is dedicated to a review of past applications of ML models within finance and a brief discussion of predictability of returns. We first motivate our choice of data frequency and sample window, reviewing the studies on efficiency of commodity markets and market efficiency overall. Next, we briefly discuss the history and development of applications of ML models for investment management and return forecasting.

2.1 Return Predictability

Any empirical research on the use of quantitative techniques to predict asset returns has to start with a consideration of the Efficient Market Hypothesis and a discussion of return predictability overall. Indeed, one's opinion on market efficiency is always implied in their research design choices, and one's decision to apply a certain model and a certain frequency of the data implies one's attitude towards the notion of return predictability.

The earlier pieces of research on efficiency of commodity markets date back to Kaminsky and Kumar (1990) and Kellard et al. (1999). These studies emphasize a large degree of

ambiguity regarding the evidence of market efficiency in commodity markets: multiple studies come up with contradicting conclusions, and these conclusions largely depend on the choice of methodology and sample window. More recent studies have found evidence of inefficiencies in certain commodity markets, such as wheat, soybeans, and maize (Algieri and Kalkuhl, 2019). The absence of unambiguous evidence of commodity market efficiency implies that there can be sources of relevant information which can forecast returns of commodity futures with reasonable accuracy and potential investor gains, so a large number of studies emerged to tackle this issue.

Next, we refer to the debate on the efficiency of equity markets to motivate our choice of data frequency. As stated in Shiller (2013), if day to day returns were indeed predictable, making money day trading would be an easy way to get rich; however, there is no evidence of that in the real world. Shiller argues that investors have to remain patient and aware not only of short-term market inefficiencies, but also of their own psychological biases that often drive irrationality and, as a result, financial losses. In his 2013 Nobel Prize lecture, Shiller emphasizes that in terms of R squared there is better return predictability observed in longer time series and longer horizons, and investors should rely on long-term strategies as opposed to speculation in order to generate consistently high positive returns.

In an attempt to follow Shiller's proposition and to keep our analysis consistent with recognized academic research, we focus on fourteen monthly series of commodity futures returns, spanning the period from January 1989 to June 2020. We thus believe that the focus on monthly data will allow us to minimize the short-term noise component, allowing the models to capture established long-term dependencies between the returns of futures contracts and explanatory variables, or, in the worst case, the lack thereof.

2.2 Machine Learning in Finance

As argued by Dixon and Halperin (2019), machine learning techniques have been actively used in financial services for more than 40 years; however, the rise of these methods in investment management is relatively recent. While such established quantitative hedge funds as D.E. Shaw, AQR, and Two Sigma have successfully implemented ML and deep

learning in investment management, Dixon and Halperin (2019) emphasize that healthy skepticism must be present, since multiple companies failed from their over-reliance on ML. Indeed, the complicated mathematical foundation behind these models and the ease which modern programming languages and packages enable users to run these models with promote the "black-box" approach to ML, which leads to irrational decisions driven by a lack of understanding of the underlying models. This emphasizes how careful analysis, testing, and reliance on established methodologies are vital for a successful implementation of machine learning-driven investment decisions over the long term, which creates a vast research field for the modern academia.

Some of the earliest academic applications of ML in investment management date back to, for example, Swanson and White (1995) and Trippi and Desieno (1992); however, as ML started to gain more popularity, improved methodologies and regularization techniques started to emerge. Besides, a higher availability of data allowed researchers to extend the field of application from the conventional case of equities to other asset classes. For example, Bali et al. (2020) provide an extensive study on corporate bond pricing, Ye and Zhang (2019) and Ni (2019) analyze various classes of conventional derivatives, and Cramer et al. (2017) and Akyildirim et al. (2020) research weather derivatives and cryptocurrencies respectively. Despite the fact that multiple authors developed methodologies for the application of ML within investment management and return forecasting, multiple contributions are still required. Taking into account fundamental differences between asset classes, it is unreasonable to expect that methodologies used to forecast, for example, equity returns will perform equally well unless asset-specific adjustments are implemented.

As we mentioned previously, one of the goals of this thesis is to introduce these adjustments to an established methodology suggested by Gu et al. (2020). This study provides a very comprehensive introduction to the use of ML methods in the case of stock returns, and many conclusions from it will be the basis for the current thesis. For example, the authors find that after considering a wide set of model specifications, simpler models with fewer parameters tend to outperform their more complicated, "deeper" counterparts. Indeed, the authors argue that conventional financial data stands far from what ML algorithms are most "comfortable" with, despite the large number of factors and observations available. This way,

smaller data sets imply the need for simpler model specifications to find the balance between sufficiently high prediction accuracy and avoidance of overfitting the training data. This finding is of particular relevance in our case, given that our chosen data frequency implies a small number of observations, especially by ML and deep learning standards.

3. Methodology

The purpose of this section is to introduce the benchmark AR(1) model, out of sample forecast accuracy metrics, and the ML methods used in this thesis. We also dedicate a paragraph to describing the issue of the "curse of dimensionality", why it is highly relevant in our application, and how this issue will be solved using established academic approaches. The section ends with a detailed description of our approach to hyperparameter tuning and how it deviates from previously published academic research: we describe the tuner algorithms used to find optimal sets of hyperparameters and outline regularization techniques used to prevent the neural networks from overfitting the training data.

3. 1 Preliminary Conditions and Model Specifications

Prior to introducing the models used in this thesis and discussing their optimization process, several preliminary conditions have to be established. The first important consideration is the choice of the benchmark model and accuracy metrics: consistently with Guidolin & Pedio (2020), we use AR(1) as the benchmark. In order to make the comparison as realistic as possible, the coefficients of the benchmark AR(1) model are re-estimated after each data point: this way, all information available at time t is used for forecasting the returns at time $t + 1$. Given that AR-type models converge to their ergodic means when the forecast period is too large, this approach would allow us to get the most accurate assessment of the benchmark AR(1) in the context of forecasting commodity futures returns.

Next, consistently with Guidolin and Pedio (2020), forecast accuracy will be assessed via two out of sample metrics: Root Mean Squared Forecast Error (RMSFE):

$$RMSFE^j(M) = \sqrt{\frac{\sum_{t=1}^{n}(\hat{R}^j_{t+1|t}(M) - R^j_{t+1})^2}{n}} \quad (1)$$

and Mean Absolute Forecast Error (MAFE):

$$MAFE^j(M) = \sqrt{\frac{\sum_{t=1}^{n}|\hat{R}^j_{t+1|t}(M) - R^j_{t+1}|}{n}} \quad (2)$$

where n is the number of observations in the test subsample, $\hat{R}^j_{t+1|t}(M)$ is the forecasted return of commodity futures j at time $t+1$, and R^j_{t+1} is the realized return. Guidolin and Pedio (2020) find that different models are considered optimal when these accuracy metrics are compared, so the use of two metrics will provide an additional robustness check.

We also divide the models in four categories based on the number and the nature of predictor variables (also known as "features" in the ML jargon). The first class is the so-called "kitchen sink" models where all the features are used without any prior dimensionality reduction; the second class of models will use the first five principal components, the third class of models will use the first ten principal components, and the fourth class will use the variables selected by the Decision Tree Regressor. In the next section we outline why dimensionality reduction is of extreme importance given the structure of our data and explain the choice of the sets of predictor variables.

3.2 Dimensionality Reduction

The Curse of Dimensionality, a term coined by Richard E. Bellman to describe issues that arise in the analysis of multidimensional spaces and data, is of extreme relevance and importance in any ML application. In the context of data analysis, Willmott (2019) describes the curse of dimensionality using an example where a researcher is trying to explain N data points using M independent variables. The author assumes for simplicity that the numerical data for each independent variable is binary, which gives us a total of 2^M combinations. Willmott thus argues that unless N is larger than 2^M, the researcher is facing the risk to classify each realization of the dependent variable as "unique", making the classification exercise meaningless.

This way, as it might appear logical that adding more independent variables will lead to a higher aggregate explanatory power, the curse of dimensionality makes this decision a double-edged sword. While it is apparent that the dataset has to be "large enough", exactly how large should it be and how many independent variables can be considered at the same time? Unfortunately, this largely depends on the choice of models and variables, and no rules are set in stone; this way, researchers have to rely on multiple model specifications and established methodologies from published academic research to find the optimal ways to minimize the impact of the curse of dimensionality. The issue is of extreme relevance in the current thesis as well, given that the sample size is only limited to 377 observations; thus, several approaches to dimensionality reduction will be applied.

Consistently with Gu et al. (2020), we will start with Principal Component Analysis (PCA) to reduce the number of predictor variables. Principal Component Analysis (PCA) uses the variance-covariance matrix of predictor variables to structure these variables into factors and rank these factors by the share of explained aggregate in-sample variance. This way, a small selection of principal components would allow us to reduce the total number of variables while still retaining most of their explanatory power. Indeed, we find that the first five principal components explain over 50% of variance for each commodity series, while the value for the first ten principal components is over 70% (Table 3).

In spite of a sound theoretical foundation, a significant drawback of PCA discussed in Gu et al. (2020) is the fact that principal components are constructed without taking the dependent variable into consideration. Indeed, while the principal components do capture the degree of variation of predictor variables, this degree of variation could potentially be less relevant for forecasting purposes than, for example, a small number of comparatively static variables which the dependent variable has a high fundamental dependence on. Thus, to take this issue into account and conduct an additional robustness check, we will be relying on Decision Tree Regressors.

The methodology of using decision trees for feature selection is discussed in Kira and Rendell (1992): the main idea is to run a simple decision tree regressor using the full set of independent variables and to look at the variables in the top "leaves" of the decision tree.

The variables that lead to some of the first splits of the decision tree are expected to have the highest forecasting power: this way, decision trees tackle the drawback of principal components, allowing us to select the key variables based on their forecasting ability. With decision trees, we will limit the number of key predictor variables to five: this will allow us to train models without overloading them with too many variables, which is the desired outcome given the structure of the analyzed dataset. Besides, the predictor variables selected by the Decision Tree Regressor will only be tested on neural networks, since this class of models is significantly more parametrized than decision trees, and such a significant reduction in the number of variables would arguably benefit neural networks more.

This way, we distinguish between four classes of models based on the set of predictor variables: full set of variables (also known as "kitchen sink" models), 5 of the first principal components, 10 of the first principal components, and 5 predictor variables suggested by decision tree regressors. The set of ML models used in the current thesis is mainly based on the study by Gu et al (2020), and these models can be broadly divided into two main categories: tree-based methods and neural networks. We now proceed with a description and explanation of the founding principles behind each of these classes of models.

3.3 Tree-Based Methods

Decision trees and tree-based models in general have proven to be highly effective in many prediction tasks, which led to a spike in their popularity within economics and finance (Athey and Imbens, 2019). In the case of commodity futures, Struck and Cheng (2020) find that tree-based methods significantly outperform the more conventional linear models, regardless of the horizon. Ease of interpretation and visualization, combined with a relatively fast training process and availability of boosting algorithms make decision trees a widely used tool that is capable of handling complex classification, regression, and forecasting problems. Breiman et al. (1984) outline the algorithms that are used to construct decision trees in modern programming languages and statistical software; however, such statistical packages as Keras and Tensorflow within the Python programming language make decision trees significantly more intuitive to build and estimate.

As discussed in Hull (2020), a single decision tree represents a flowchart-like structure, consisting of three main parts: the root, internal nodes, and leaves. Decision trees split the values of the dependent variables based on the optimal thresholds found in the values of independent variables, and the first split takes place in the root. More formally, as outlined in Coqueret and Guida (2020) and Athey and Imbens (2019), given a sample of size I, the regression tree will minimize the total variation of y_i inside the subclusters. The process consists of two steps: first, the model will find the best splitting point for each independent variable, and then pick the variable that leads to the optimal split. The optimal split in regression problems is defined as the one which minimizes the variability (or dispersion) inside each subcluster.

$$V_I^{(k)}\left(c^{(k)}\right) = \sum_{x_i^{(k)} < c^{(k)}} \left(y_i - m_I^{k,-}(c^k)\right)^2 + \sum_{x_i^{(k)} - c^{(k)}} \left(y_i - m_I^{k,+}\left(c^{(k)}\right)\right)^2 \tag{3}$$

Where

$$m_I^{k,-}\left(c^{(k)}\right) = \frac{1}{\#\{i, x_i^{(k)} < c^{(k)}\}} \sum_{\{x_i^{(k)} < c^{(k)}\}} y_i \tag{4}$$

$$m_I^{k,+}\left(c^{(k)}\right) = \frac{1}{\#\{i, x_i^{(k)} > c^{(k)}\}} \sum_{x_i^{(k)} > c^{(k)}} y_i \tag{5}$$

The optimal splits thus satisfy the following condition:

$$k^* = \underset{k}{\operatorname{argmin}}\, V_I^{(k)}(c^{k,*}) \tag{6}$$

However, despite the advantages, one of the key issues of decision trees discussed in Hull (2020) and Gu et al. (2020) is that they tend to quickly overfit the training data, unless sufficiently strict regularization is imposed. Of course, there are multiple ways to regularize decision trees: the researcher could limit the minimum number of observations per leaf, maximum "depth" of a decision tree, the minimum number of observations required for a split, etc. However, decision trees are still infamous for overfitting despite these regularization options, and this is part of the reason why improved and modified versions of tree-based models have been gaining popularity in the recent years. For this reason, we will

not be looking at simple decision trees within this thesis, but rather start the analysis with their more advanced extensions; namely, we will be looking at Random Forests, Adaptive Boosting, and Extreme Gradient Boosting.

3.3.1 Random Forests

Out of the three tree-based methods used in this thesis, random forests are the easiest to understand and interpret. Random forests are an ensemble method whose main idea is to train multiple decision trees (hence the name "forest") and to take the average of their forecasts. While the "perfect" decision tree is different for each problem and is impossible to estimate unequivocally, intuitively it is reasonable to train multiple trees and use their average forecast as the final output. Apart from Gu et al. (2020), random forests were proven successful in forecasting asset returns, for example, in Sadorsky (2021), where the author demonstrates how random forests outperform the standard logit model in forecasting the direction of price movements of Clean Energy stock prices. Besides, another advantage of random forests is that they can be trained on different subsamples: predictor variables for each tree are selected randomly, which allows us to avoid inconclusive or overfitted results due to the curse of dimensionality and to reduce the risk of model misspecification at the same time. This way, the curse of dimensionality becomes close to irrelevant in the case of random forests, since the number of randomly chosen predictor variables used to estimate each tree in the forest, and the number of trees for that matter, are tunable parameters that will be chosen to optimize the out of sample forecast accuracy.

3.3.2 Adaptive Boosting (AdaBoost)

Adaptive Boosting or AdaBoost is the first boosting decision tree-based algorithm analyzed within the current master thesis, and the idea behind boosting is slightly less intuitive compared with random forests. While random forests try to find the optimal forecasts by diversifying across multiple simple models, boosted algorithms aim to learn from past iterations of the model and improve with each consecutive one. Boosted algorithms are

distinguished by the ways they "learn", and two of the more popular ones are Adaptive Boosting and Extreme Gradient Boosting, both of which are used in this thesis.

The founding principles of Adaptive Boosting are outlined in Freund and Schapire (1996, 2012), and theoretical principles and further advancements were derived by Breiman (2004). The algorithm is the following:

1. Set a matrix of equal weights $w_i = I^{-1}$
2. For $m \in \{1, 2, \dots, M\}$, find the learner that minimizes the weighted loss function

$$\sum_{i=1}^{M} w_i L(l_m(x_i), y_i) \tag{7}$$

3. Calculate the weight of the learner

$$a_m = f_a(w, l_m(x), y) \tag{8}$$

4. Update the weights matrix

$$w_i \leftarrow w_i e^{f_w(l_m(x_i), y_i} \tag{9}$$

5. Normalize the weights so that they sum to one.

The output for each instance is thus a simple function of the following form:

$$\tilde{y}_i = f_y(\sum_{m=1}^{M} a_m l_m(x_i)) \tag{10}$$

The difference between simple decision trees and AdaBoost comes in points 2 and 3: in these parts of the algorithm, the model sequentially adapts the weight of each learner based on its performance, assigning larger weights to the more accurate learners. Then the model will change the weights of observations within the sample, giving preference to those where the learners perform poorly, i.e. generate the largest errors. Thus, not only do boosted decision tree algorithms take multiple learners into account, but they also aim at identifying the weak spots in their forecasting abilities.

3.3.3 Extreme Gradient Boosting (XGBoost)

XGBoost is a relatively young tree-based algorithm first introduced by Chen and Guestrin (2016); however, it has gained high popularity in all applications of ML, including finance. This algorithm builds upon the best features of AdaBoost and Random Forests, allowing for better regularization to avoid overfitting and reduce variance. Consistently with Gu et al. (2020), we will be using the Mean Squared Error loss function, for which the XGBoost seeks to optimize the following objective:

$$O = \sum_{i=1}^{I} \left(y_i - m_{J-1}(x_i) - T_J(x_i)\right)^2 + \sum_{j=1}^{J} \Omega(T_j) \tag{11}$$

Where T_j is the j^{th} tree under consideration, x_i is the instance, $m_{J-1}(x_i)$ and $T_J(x_i)$ are the aggregate forecasts of past learners and the currently examined tree, respectively. The first term penalizes large deviations from the observed values of the dependent variable, while the second term controls the complexity of the model, preventing it from overfitting. These terms are called the error term and the regularization term respectively. According to Hull (2020), one of the main features of XGBoost is that the algorithm "targets" residuals from the previous iterations and trains each learner consecutively. For instance, if the first tree produced a significant upward bias, the next one will be biased downwards, thus aiming to minimize the aggregate error of all learners (hence the name "gradient boosting" through additional learners). More detailed derivations and cases of different loss functions are described in Chen and Guestrin (2016) and Coqueret and Guida (2020).

3.4 Neural Networks

Artificial Neural Networks are arguably the most powerful machine learning technique that is used in a wide array of applications, such as natural language processing, image processing and recognition, automated game-playing, etc. (Gu et al, 2020). An important caveat is that neural networks are widely considered some of the most complex and heavily parametrized ML techniques, perceived by many as "black boxes". This way, the lack of transparency in their structure requires a carefully developed methodology and multiple iterations of different combinations of hyperparameters that will optimize the model for the task at hand.

Further we describe the main principles behind neural networks and outline the architectures used in the current thesis. We mainly refer to Gu et al. (2020), Dixon et al. (2020), Hull (2020), and Coqueret and Guida (2020) for the theoretical foundations behind neural networks.

3.4.1 Feedforward Neural Networks

Consistently with Gu et al. (2020), we start the analysis with simpler feedforward networks. Such networks consist of the input layer, hidden layers, and the output layer, and a non-linear activation function is used to transform the input data before it is passed to the next layer. Each layer consists of neurons which independent variables are passed to, and the values in the neurons are multiplied by the optimal weights estimated by the neural network. This step is called forward-propagation; however, the power of neural networks lies in their ability to iteratively improve weights of independent variables to minimize the targeted loss function. Aiming to minimize the assigned loss function, the neural network will estimate the sensitivities of the loss function to weights and biases through a process called backpropagation, obtaining the information on how these weights and biases are to be adjusted. The estimation of the required sensitivities starts at the output layer and goes all the way back to the input layer, hence the name "backpropagation". Next, the network will repeat the forecasting exercise, but with updated weights. This explains how the model "learns" and iteratively improves the forecasts based on the obtained value of the loss function. This step is regulated by the number of epochs, i.e. the number of times the model "is allowed" to re-estimate the forecasts with updated parameters and learn before it outputs the final forecast.

Intuitively, the choice of the loss function and the activation function is of utmost importance: in the end, these functions define what the model finds accurate or inaccurate, and how exactly the data will be transformed within the network before the final forecasts are produced. Thus, we closely follow the methodology by Gu et al. (2020) when choosing these functions. The loss function for all architectures of neural networks will be the Huber loss (Huber, 1964):

$$L_\delta \begin{cases} \frac{1}{2}(y - f(x))^2 & for\ |y - f(x)| \leq\ \delta, \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & otherwise. \end{cases} \tag{12}$$

Until a certain threshold δ, the Huber loss function is identical to the classical Mean Squared Error (MSE); however, the advantage of the former is that it is less sensitive to outliers in the data. If the error is above the threshold, the Huber loss function resembles the Mean Absolute Error, "punishing" the model less than the MSE would. This loss function is also consistent with the two forecast accuracy metrics we are using in this thesis (RMSFE and MAFE), allowing us to find a balance between the two during the optimization process.

Regarding the activation function, consistently with Gu et al. (2020) we will be using the rectified linear unit (ReLU) function that has recently gained high popularity in machine learning literature. Such studies as Jarrett et al. (2009), Nair and Hinton (2010), and Glorot et al. (2011) outline the superior performance of ReLU, showing that one of its main advantages is that by construction it leads to "sparsity in the number of active neurons", significantly reducing the computational requirements. The function is defined as follows:

$$ReLU(x) = \begin{cases} 0, & if\ x < 0 \\ x, & otherwise \end{cases} \tag{13}$$

This way, the recursive output formula for the neuron k in the layer $l > 0$ is the following:

$$x_k^{(l)} = ReLU(x^{(l-1)\prime}\theta_k^{(l-1)}) \tag{14}$$

Where x is the vector of values of the neurons within layer $(l-1)$ and θ_k is the matrix of weights associated with the neuron k.

The final output of the model has the following shape:

$$g(z;\theta) = x^{(L-1)\prime}\theta^{(L-1)} \tag{15}$$

Next, we would like to proceed with a discussion of one of the most important hyperparameters within neural networks, namely the learning rate. This hyperparameter regulates how much weight the model puts on new observations and which share of adjustment implied by the new observations will be used to update the weights of predictor

variables. Thus, an extremely large learning rate can deprive the model of its memory and continuous past "learning", making large adjustments to the parameters after each consecutive epoch. At the same time, an extremely small learning rate would significantly slow down the optimization process, making it difficult for the gradient to converge to the optimal value. Besides, another important issue emphasized by Gu et al. (2020) is that the learning rate has to be shrunk to zero as the gradient approaches zero in order to prevent the noise component from negatively affecting the gradient. The first issue is tackled by adding the learning rate as one of our tunable hyperparameters and choosing the optimal value based on out of sample performance of neural networks. In order to solve the second issue, we follow the approach suggested by Gu et al. (2020) and use the "learning rate shrinkage" algorithm called Adam, which was suggested by Kingma and Ba (2015).

Another issue that has to be considered is the fact that due to the stochastic nature of neural networks, two identical models will generate different forecasts because of randomly generated initial weights. More formally, as the weights are transformed within the neural network, their initial values are generated at random, and the shallower the neural network, the larger the impact of this weight initialization. This way, models can show high out of sample forecasting accuracy by chance, resulting in an apparent lack of robustness and use in real world applications. To account for this issue, we follow the ensemble methodology suggested by Gu et al. (2020). Ensemble allows us to account for the stochastic component of neural networks by training several networks with identical parameter values and averaging their forecasts to achieve the final result. We set different random seeds for each neural network in the ensemble, which eliminates the possibility of initializing identical weights. We also set identical random seeds for all models trained in the hyperparameter optimization stage and take the average forecasts of three runs of each model prior to assessing and ranking their out of sample forecasting performance. This allows us to increase the robustness of our results and eliminate the "luck" component in the initialization of weights.

3.4.2 Long Short-Term Memory (LSTM) Neural Networks

Introduced by Hochreiter and Schmidhuber (1997), the Long Short-Term Memory model is a Recurrent Neural Network (RNN) that has several advantages over the more conventional feedforward neural networks. The first advantage is that LSTM models tackle the so-called "Vanishing Gradient Problem": discussed in Hochreiter (1998), the vanishing gradient problem occurs when the inputs of the activation function approach extreme values. For example, the ranges of such common activation functions as the sigmoid are bounded between 0 and 1; thus, the closer the transformed values are to either of these extremes, the closer the partial derivatives within the gradient are to zero. The vanishing gradient problem thus significantly complicates the backpropagation process since the model lacks information on the sensitivities of the loss function to the parameters. While Wang (2019) show that the ReLU activation function is effective in tackling this issue, it cannot completely eliminate the exposure of the model to the vanishing gradient problem by itself. LSTM in turn tackles this issue by using a "unique additive gradient structure", which we describe in more detail further.

While it is true that the vanishing gradient problem is not a too dangerous for shallower neural networks, there is another important property of LSTM that is significantly more important in the context of forecasting asset returns. Namely, by construction LSTM models are capable of learning and, what is more important, remembering long-term dependencies in the data. As argued in Wilmott (2019), recurrent neural networks are distinguished by feedback loops, i.e. not only is the output from any layer passed to the next one, but it is also used to update that same layer further in the learning process. The LSTM model builds upon this framework by adding the so-called "gates" – devices used to selectively let information through the network. The first gate within the LSTM layer is the forget gate which controls whether the memory is reset to zero:

$$f(t) = \sigma(b^f + W^f h(t-1) + U^f x(t)) \tag{16}$$

Next, the input gate controls the updates of the memory cell and what new information will be stored in the current cell state:

$$g(t) = \sigma(b^g + W^g h(t-1) + U^g x(t)) \quad (17)$$

Finally, the output gate is used to determine the next hidden state h_t (which is the output of the layer):

$$q(t) = \sigma(b^q + W^q h(t-1) + U^q x(t)) \quad (18)$$

$$h_t = q(t) \times \tanh(C_t) \quad (19)$$

An important detail is that all these gates have the same activation function (sigmoid); however, the weights and biases within each gates are individual, which allows the optimal gating procedure to be captured from the data and learned, rather than recreated using a predefined pattern. At the same time, the cell state, which is the internal state of the layer that does not directly produce any output but instead modifies the hidden state in further iterations, is modified by the vector $\bar{C}$:

$$\bar{C}^{(t)} = \tanh(W^C \left[h^{(t-1)}, x^{(t)}\right] + b^C \quad (20)$$

To sum up, the LSTM can "decide" on the optimal way to update the cell state and the hidden state based on this gating system, and its construction allows it to tackle both the vanishing gradient problem and the tendency of feedforward neural networks to "forget" long-term dependencies in the data. This way, the construction of LSTM makes these networks ideally suitable for long sequences of data, including financial time series. This class of networks is an extension to the methodology suggested by Gu et al. (2020), and it has been widely used in modern applications of machine learning, including finance. For example, we refer to Huang et al. (2021) where LSTM networks are used in the context of portfolio optimization, and Ouyang et al. (2019) where the networks are applied to forecast returns of commodity futures.

3.5 Hyperparameter Optimization

A crucial distinction between the more conventional methods from financial econometrics and machine learning techniques is that the latter require a much more careful approach to the choice of model parameters. While such common models as, for example, the Vector Autoregression (VAR) can be optimized by minimizing information criteria for different lag

lengths and experimenting with variable ordering to produce more realistic impulse response functions, the range of parameters is significantly larger in machine learning. Apart from tuning the parameters that regulate how fast or how slow the model "learns" the dependencies within the data, the researcher is free to decide how much time and "freedom" they are willing to give the model to learn. Besides, one has to remember that even the shallowest models will quickly overfit if they are given the freedom to, so multiple regularization techniques and parameters have to be considered in order to avoid this issue. All these considerations are taken into account through the so-called model hyperparameters, and multiple studies have shown that the choice of these hyperparameters can have an immense impact on how the models perform both in and out of sample.

Technically, one can come up with an infinite number of potential models based on their hyperparameter values; however, estimating such a large number of models is neither computationally feasible nor effective. This way, preliminary research is required to narrow down the hyperparameter search space to make the potential array of models estimable for optimization algorithms. Our approach to hyperparameter optimization consists of two stages: during the first stage we narrow down the search space based on previous academic research in this field, while in the second stage we use the Keras Tuner algorithm to evaluate different model specifications and test their out of sample predictive accuracy. Besides, the algorithms will not be heavily restricted by the hyperparameter range: both shallow and deep, regularized and unregularized models will be tested. Hyperparameter ranges for each class of models can be found in Table 5.

As we mentioned previously, an important contribution of the current master thesis is that we are aiming to review the approach to hyperparameter optimization in the context of asset pricing suggested by Gu et al. (2020) and Struck and Cheng (2020). The structure of our data set allows for a more rigorous optimization process and model hyperparameter search, and we will combine tuning algorithms with the findings from previously published academic research in order to deliver the optimal model specification. This way, the first step of our hyperparameter search is to use findings from Gu et al. (2020) to narrow down the range of model hyperparameters that are expected to deliver optimal performance. By narrowing

down the range of potential hyperparameters we reduce the computational costs, which will allow us to conduct an exhaustive cross-evaluation of model parameters within the pre-specified range. As opposed to Gu et al (2020) who worked with significantly larger data set and tested fewer model specifications due to high computational costs, we can afford to run more models over a significantly smaller data set without any extreme computational power requirements.

3.6 Regularization

In this section we outline regularization techniques, which are additional measures used to prevent neural networks from overfitting. Consistently with Gu et al. (2020), we use the l1 and l2 norms as our first technique. Gu et al. (2020) argue that early stopping can be implemented instead of l2 regularization in order to sacrifice marginal improvements in accuracy for a dramatically lower computational time. While this decision is justified by a significantly larger size of the sample, the decrease in computational requirements would be miniscule in our case. Thus, we implement both the l1 and l2 regularization in order to obtain more accurate estimates of optimal hyperparameters. We proceed with a more formal description of these regularization techniques, referring to Hull (2020) and Oppermann (2020) for the technical details. Further details and derivations can be found in Deisenroth et al. (2020).

Formally, l1 and l2 norms add penalizing terms to the loss function of the neural network:

$$\Omega(W) = ||W||_1 = \sum_i \sum_j |w_{ij}| \tag{21}$$

in the case of l1 regularization, and

$$\Omega(W) = ||W||_2^2 = \sum_i \sum_j w_{ij}^2 \tag{22}$$

in the case of l2 regularization. These regularization terms are used to adjust the gradient of the initial loss function:

$$\nabla_W \hat{L}(W) = \alpha \times \text{sign(W)} + \nabla_w L(W) \tag{23}$$

$$\nabla_W \hat{L}(W) = \alpha \mathrm{W} + \nabla_w L(W) \tag{24}$$

in the case of l1 and l2 regularization respectively.

This way, similar to conventional Lasso and Ridge regressions, l1 and l2 regularization forces the weights of neurons in hidden layers to converge to zero if their contribution to the reduction of loss is insignificant. This "insignificance" is in turn regulated by the alpha parameter of each norm: the larger the alpha, the higher the required "contribution" of the neuron. By construction, an incorrect choice of the alpha parameter can either make the model too simple and shallow by leaving too few hidden neurons or keep the model unregularized as all hidden neurons will qualify as sufficiently strong. Thus, the l1 and l2 alphas will enter our hyperparameter optimization process as two separate parameters.

Another important regularization technique used in this thesis is the dropout ratio, introduced and discussed in Srivastava et al. (2014). The idea behind the dropout is self-explanatory: part of the neurons in each layer is deactivated at random, which reduces the complexity of the model. The tunable parameter can be thus interpreted as a probability that a neuron within a layer is deactivated during the forward-propagation process. Thus, a dropout ratio of 0.5 implies that one half of neurons in each layer is expected to be deactivated. Apart from reducing the complexity of the model, the dropout ratio prevents the network from "focusing" on a particular subset of neurons: by introducing a noise component, the dropout forces the neural network to use all available neurons. This way, the model becomes more robust as the network is forced to increase weights of all neurons, but not only those that lead to a good in-sample fit.

Finally, another regularization technique consistent with Gu et al. (2020) is early stopping: this is a callback that prevents the model from further training if there are no reductions observed in the validation loss function. We will not be using early stopping in our hyperparameter optimization phase in order to obtain the most accurate values of parameters; however, early stopping will be implemented when we evaluate the models on test data. The parameter associated with early stopping is called patience – this parameter sets the maximum number of consecutive epochs during which the model is "free" to show stability or increases in the value of the loss function. For example, a patience of 5 would

imply that the callback will stop the model from further training if it has not shown decreases in training/validation loss during five consecutive epochs. Apart from significantly reducing estimation time, early stopping allows us to cut off unnecessary epochs and prevent the model from overfitting in case the optimal out of sample performance was achieved before the pre-set number of epochs expired. The patience parameter will be set to 5 in all forecasting exercises and will not be tunable, which is consistent with Gu et al. (2020).

3.7 Keras Tuner and Bayesian Optimization

After narrowing down the ranges of potential hyperparameters, we rely on algorithms to find the optimal combinations. Specifically, we will be using an adjusted version of the Bayesian Optimization tuner, which is part of a Python package called Keras. By default, the Bayesian Optimization tuner is not capable of optimizing such parameters as the number of epochs, number of hidden layers, and batch size; however, due to the flexibility of the algorithms within the package the tuner can be easily adjusted to take these parameters into account.

The basis of each tuner is consecutive testing of multiple model specifications, followed by a comparison of out of sample performance. The simplest forms of tuners are Random Search and Grid Search: the former selects n random combinations of hyperparameters and returns the best performing one(s), while the latter relies on an exhaustive, "grid" search of all possible combinations. The main drawback of these tuners is that they require an extreme number of iterations to converge to optimal hyperparameters. For example, with just five hyperparameters and four potential values for each, the Grid Search would have to test 1,024 neural networks which, given a more complex architecture, can take weeks. Thus, more optimal ways to tune hyperparameters and find the optimal values are required to make the task computationally feasible, reducing the number of required iterations.

For this reason, we will be using the Bayesian Optimization tuner, referring to the works by Shahriari et al. (2015) and Dewancker et al. (n.d.) to describe its main advantages and the key principles behind the tuner. The tuner consists of two main components: a probabilistic surrogate model and a loss function that describes the optimality of hyperparameter

combinations. The former is used to describe the distributions of hyperparameters, serving as an approximator of the objective function which gives the tuner guidance on which parameters can yield higher accuracy. The latter evaluates the chosen hyperparameters and provides an estimate of how the model performs on validation data. This way, the main advantage of Bayesian optimization is that we use the knowledge from past iterations to choose combinations of hyperparameters in the next iterations. This allows us to outstandingly decrease the number of iterations required to achieve optimal values, significantly reducing the computational costs.

While the Bayesian tuner will be used to find optimal hyperparameters for our neural networks, tree-based methods will be optimized using the Grid Search algorithm, which is consistent with Gu et al. (2020). The reason for this choice is that tree-based methods imply a much smaller number of tunable hyperparameters, and they are significantly easier to estimate and handle than neural networks. Thus, despite the large number of model specifications tested by the Grid Search, the computational costs are even lower than those of neural network tuning via Bayesian optimization.

3.8 Contributions to the Literature

Despite the fact that a significant part of the current thesis is an attempt to combine the methodologies by Gu et al (2020) and Guidolin and Pedio (2020), there are several important deviations from these studies that have to be mentioned. The first one is the training/validation/test data split: while Guidolin and Pedio (2020) use a 50/50 split and Gu et al (2020) choose the 30/20/50 one, the training/validation/ test data split is 60/15/25 in the current thesis. The rationale behind this choice is the fact that machine learning models require more data compared with more conventional econometric techniques (see for example van der Ploeg et al. (2014)). Given that we work with monthly data, a 50/50 split would yield a training subsample of only 189 observations, which would be insufficient to correctly train the employed machine learning models and capture the complex dependencies between the returns of futures and predictor variables.

Another deviation from Gu et al. (2020) is that we do not follow the pyramid rule suggested by Masters (1993) when determining the optimal number of neurons in our neural networks. While multiple studies confirm the efficiency of this approach, our goal is to give the optimization algorithms as much freedom as possible when determining the optimal hyperparameters, including the number of hidden layers and neurons. However, we also aim to control this freedom by adding more ways to regularize neural networks. For example, Gu et al. (2020) claim that l2 regularization is neglected in favor of early stopping, which allows them to dramatically reduce computational requirements at a cost of marginal decreases in accuracy. Computational costs are a significantly smaller problem in this thesis, so we will use both l1 and l2 regularization, combined with the dropout parameter.

4. Data

In this section we describe the structure of the dataset, explaining three categories of predictor variables used in the current thesis. We closely follow the study by Guidolin and Pedio (2020) who analyze monthly series of commodity futures returns, borrowing and extending their list of variables and returns series.

4.1 Commodity Series

The futures contracts under consideration include the following set of commodities: Light Crude Oil, Corn, Soybeans, Wheat, Coffee, Cocoa, Sugar, Cotton No.2, Gold, Silver, Platinum, Frozen Orange Juice, Lumber, and Live Cattle. We will be using monthly data on the returns of these commodity futures, and our sample window spans the period from January 31, 1989 to June 30, 2020. The data was downloaded from Thomson Reuters Eikon.

Consistently with Guidolin and Pedio (2020) and Fuertes et al. (2015), we follow the common approach that focuses on fully collateralized positions. This allows us to make the series of futures returns consistent with those of other asset classes, where long positions imply an initial outflow of money for an investor. Besides, the absence of leverage eliminates the risk of unintentional liquidation of investors' positions. This sample period captures the

start of the COVD-19 pandemic, which had a significant impact on financial markets and led to black swan-like price changes. As this period will be included in the test subsample, the ability of trained models to accurately predict such dramatic price changes is of particular interest. Next, we will describe the predictor variables, broadly divided into three categories: Macroeconomic Factors, Commodity-Specific Factors, and Miscellaneous Factors.

4.2 Macroeconomic Factors

The first class of predictor variables is Macroeconomic Factors: 132 macroeconomic variables reduced to seven factors using Principal Component Analysis were borrowed from Ludvigson & Ng (2009). The underlying variables represent different indicators of macroeconomic health and stability of the U.S. economy, such as personal income, industrial production indices, treasury bill rates of different maturities, etc. The choice of these factors is consistent with Guidolin and Pedio (2020), and macroeconomic factors have been used in the analysis of commodity futures returns in such studies as, for example Daskalaki et al. (2013). The data is freely available on Sydney Ludvigson's official website.

4.3 Commodity-Specific Factors

The set of commodity-specific factors is borrowed from Guidolin & Pedio (2020), and a similar set of factors (with the exception of Net Trading) was analyzed in Daskalaki et al. (2013). These factors are the Hedging Pressure (HP), Basis, and the Net Trading (NT). The data used to construct these factors was downloaded from Bloomberg and weekly "Commitment of Trader" (COT) reports which publish the data on the number and size of positions across commodity futures in the United States. The COT classification allows us to distinguish between different categories of trades, such as commercial and non-commercial.

Guidolin and Pedio (2020) analyze potential increases in forecasting power for returns of commodity futures from these commodity-specific factors. The authors find that models that incorporate these factors do not show improvements in forecasting power when compared with models that only assume macroeconomic factors. While this conclusion might make the inclusion of commodity-specific factors appear redundant, the authors also find that

strategies that assume commodity-specific predictors outperform other strategies in high-volatility regimes. This suggests that commodity-specific factors can improve the analyzed models during the period of market downturns, which gives us an important reason to include these factors in our analysis.

Hedging Pressure (HP) is defined as the total number of short positions for hedging purposes, less the total number of long positions for hedging purposes, divided by the total number of hedge positions at time t. However, consistently with Guidolin and Pedio (2020), the number of total hedge positions is replaced by total open interest at time t.

$$HP_{j,t} = \frac{\#shorthedgeposition_{j,t} - \#longhedgeposition_{j,t}}{Total\#hedgeposition_{j,t}} \tag{25}$$

Basis is calculated as the relative difference between the prices of two portfolios of commodity futures j: one of these portfolios has a negative basis while the other has a positive one. This way, if the prices of futures contracts increase with time to maturity, the basis is positive. The futures curves used to construct this factor were downloaded from Bloomberg.

$$Base_{j,t} = \frac{F_{j,t} - F_{j,t+1}}{F_{j,t}} \tag{26}$$

The NT factor is the ratio of change in net long positions in a futures contract over open interest. This factor allows us to capture demand for commodity futures contracts: a high NT would indicate that the number of long positions is growing over time.

$$NT_{j,t} = \frac{\#netlongposition_{j,t} - \#netlongposition_{j,t-1}}{OpenInterest_{j,t-1}} \tag{27}$$

Finally, the Momentum Factor is constructed by estimating a series of AR(p), $p \in \{1, 2, \dots, 12\}$ models over the series constructed by taking the simple average returns of all commodities analyzed within this thesis. This allows us to consider the aggregate momentum of commodity futures over different horizons, capturing both long- and short-term momentum patterns.

These factors are calculated both separately for each commodity and on an aggregate level. We include these factors along with the average return of the fourteen futures as predictor variables.

4.4 Miscellaneous Factors

The first set of factors we include is the Fama-French 5 Factors, introduced in Fama and French (2015). Multiple studies have explored the links between commodity markets and equities, yielding ambiguous results. For example, while Chong and Miffre (2010) argue that there has been a decrease in correlations between commodity futures and equity indices, Creti et al. (2013) and de Boyrie and Pavlova (2016) contradict this result by showing a high degree of variation and time-dependency in such correlations. While the results are ambiguous, the literature has shown that the links between equities and commodities do exist and vary in strength over time. This way, in line with Blocher et al. (2016), we will be using the 5 factors by Fama and French (2015) to capture the explanatory power of equities and to further explore the links between commodity futures and equity markets.

Another factor is the lagged value of returns, which is an additional proxy for momentum. As mentioned previously, momentum has already been taken into account by running a family of AR(p), $p \in \{1, 2, \dots, 12\}$ models on the series of average returns of the fourteen commodity futures analyzed in this thesis, consistently with Guidolin & Pedio (2020). However, consistently with Struck and Cheng (2020), we take the lagged return of each commodity futures contract to add a more commodity-specific momentum component. While moving averages of several lags are also suggested by the authors, the limited sample size of this study forces us to use predictor variables that would not lead to significant losses in the number of available observations.

4.5 Transformations of Data

The only transformation used in the current thesis is scaling; according to Bhandari (2020), scaling significantly improves the performance of ML methods and thus cannot be neglected.

The contribution of scaling is most apparent in models that rely on gradient descent (such as neural networks):

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \tag{28}$$

The values of independent variables x will affect the step size of the gradient descent; thus, the step sizes will vary unless the independent variables are of the same scale, which can slow down the convergence to optimal parameter values at best, and make this convergence impossible at worst. More intuitively, algorithms will give preference to independent variables of larger magnitude and variation, and this preference will not be guided by fundamental reasons and/or dependencies observed in the data. This way, the distribution of weights of independent variables will be suboptimal, leading to lower forecasting ability of the chosen model.

For this reason, we will standardize all the predictor variables using the StandardScaler() function from the Python package called scikit-learn:

$$X_i' = \frac{X_i - \bar{X}}{\sigma_s} \tag{29}$$

However, a common mistake is to scale the variables prior to splitting the data into the training/test subsamples. Since our goal is to make the forecasting conditions as realistic as possible, our models have to be isolated from any possible foresight into the test part of the sample. This condition can be easily violated by scaling the predictor variables using the full sample mean and variance of these variables, which will make the experiment unrealistic. To avoid this issue, the test subsample will be scaled using the mean and the variance of variables from the training subsample, which is a common practice in ML.

Elaborating on the topic of preventing the model from having a foresight, we use the values available to investors at the beginning of each month to forecast the next month's returns. This way, all models used in the current thesis share the same structure:

$$\tilde{y}_t = f(X_{t-1}) \tag{30}$$

4.6 Simulation Exercises

Prior to applying the algorithms to forecast the returns of commodity futures, we run a simulation exercise that allows us to make sure that the algorithm does not contain any logical errors and is capable of capturing simple dependencies in the data despite the limited size of the sample. We generate five random variables and create the following dependency between them:

$$y = x_1 - 5x_2 * 0.5x_3 + 1.5x_4 + a{\sim}U(0,1)x_5 \quad (31)$$

Where $x_1 \sim$ U(0,1), x_2~N(5,2), x_3 is the inverse of the cumulative probability density function of the standard normal distribution, where a uniformly distributed random variable acts as "probability", x_4~N(10, 4), and x_5~U(0,5). x_5 is also multiplied by a uniformly distributed random variable to add a noise component to the equation.

We find that after hyperparameter tuning, both the Feedforward neural networks and LSTM models capture this simple dependency almost perfectly (Figure 1 and Figure 2), which allows us to conclude that the algorithms do not contain logical errors and can be tested on more complex dependence structures. It is also worth mentioning that the Bayesian Tuner algorithm found optimal hyperparameters after only 15 runs or less, while we will be using 125-150 runs of the tuner when optimizing the models for commodity series.

5. Empirical Results

The purpose of this section is to describe the performance of our hyperparameter tuning algorithms and to assess the out of sample forecasting performance of the optimal ML models. We also describe the results of our portfolio exercise and summarize the findings of our approach to dimensionality reduction.

5.1 Out of Sample Accuracy and Model Specifications

In this section we summarize the out of sample forecasting performance of the analyzed machine learning models and compare it with the AR(1) benchmark. We run three different tree-based methods on three separate sets of predictor variables ("kitchen sink", five principal components, and ten principal components) and two types of neural networks on four sets of predictor variables ("kitchen sink, five principal components, ten principal components, and five key variables selected by the Decision Tree Regressor). Thus, we are analyzing the performance of 17 models, excluding the benchmark AR(1).

Our first finding is consistent with the one by Gu et al. (2020): according to the Keras Tuner algorithm, most commodity series are best predicted by shallower models with fewer parameters and a relatively relaxed regularization. Indeed, most of our optimal neural networks contain one or two hidden layers, and the optimal tree-based methods are significantly below the upper bound of allowed complexity. An interesting result is that some optimal neural networks do not even include a hidden layer, which makes them identical to simple linear regressions. These results are also consistent with our initial expectations since the complexity of the model directly correlates with the size of the analyzed dataset. The robustness of this finding is confirmed by a relatively large range of regularization parameters: one could argue that a more complex model would imply heavier regularization, and the algorithm simply chose shallower models in order to avoid overfitting. However, since the range of potential regularizing terms was large, the algorithm had the "freedom" to test deep and heavily regularized models, and the properties of the Bayesian tuner allowed it to adjust regularization to avoid both overfitting and underperformance. Given that the search space spanned the set of values that would successfully prevent the deeper models from overfitting, we can conclude that shallower models do indeed perform better in the context of forecasting commodity futures returns, and this result is consistent with the one found by Gu et al. (2020) in the case of equities.

As in Guidolin and Pedio (2020), we do not find any evidence of consistent outperformance of ML techniques: most of our forecasts converge to the unconditional mean, indicating that most of the time the models cannot capture the complex dependencies between commodity

futures and explanatory variables, given the adopted research design. Despite the fact that some models do not converge to the long-run mean, their forecasts fluctuate in a relatively small range, exhibiting a significantly lower degree of volatility compared with the targeted series of returns. Considering the fact that the models did capture simpler dependencies within our simulation exercise, we conclude that data-hungry ML techniques are not suitable for this kind of analysis, and more conventional econometric techniques that rely on a significantly smaller number of parameters can show better performance.

According to the RMSFE metric, the benchmark AR(1) model showed the lowest forecast error in three out of fourteen cases – the best performance of any individual model under our consideration. Out of the 11 remaining commodity futures series, tree-based methods showed the lowest forecast error in 4 cases, and neural networks in 7. It is also worth mentioning that apart from the benchmark AR(1), only two models showed the lowest forecast error in more than one case: these models are Random Forests with the full set of predictor variables and LSTM with 5 principal components. Given our discussion about the curse of dimensionality, this performance of random forests might appear surprising, since dimensionality reduction techniques were not used. However, it is worth mentioning that random forests rely on variable selection algorithms: a tunable number of variables is selected at random to train each separate tree within the forest. This way, a large number of predictor variables was an advantage rather than a disadvantage for the random forest, since it allowed for higher diversification among learners and a larger set of available information that did not overcomplicate the overall model. In the case of LSTM, fewer variables allowed the network to focus on understanding the dependency rather than optimizing the weights of all predictor variables, which was highly beneficial given the limited size of the sample.

The results suggested by the MAFE metric are slightly different: although the AR(1) still shows the smallest error in 3 cases out of 14, the same overall performance is achieved by AdaBoost with 10 principal components. Two more models return the lowest error more than once in the case of MAFE: Feedforward Neural Network with 5 variables selected by the decision tree regression, and LSTM with 5 principal components. A full summary of model performance with respect to the accuracy metrics can be found in Table 8 and Table 9.

5.2 Portfolio Exercise

After evaluating the out of sample forecast accuracy metrics, we proceed with an application of the best performing models on a portfolio level. Our experiment borrows from the one conducted by Gu et al. (2020): the forecasted returns are sorted from highest to lowest, and we assume a long position in the top four commodities and a short position in the bottom four commodities. Given a generally positive correlation between commodity futures (Table 2), we also place a restriction that the long position is only assumed if the forecasted value is positive, and vice versa. Next, we create the forecast tables for two portfolios. The first (ML) portfolio combines the forecasts of the best-performing models (including AR(1)), while the second portfolio is based solely on AR(1) forecasts.

We find that the ML portfolio significantly outperforms the benchmark in terms of cumulative return. According to our experiment, the ML portfolio yields a 63.60% return over the period from September 30, 2012 to April 30, 2020 as opposed to a 6.57% return generated by the benchmark portfolio over the same period. Despite a better performance of the ML portfolio, there are several issues related to the robustness of the given experiment, and the main one lies in sample selection. Since most forecasts converged to the long-term mean, some commodities were only selected because of their better long-term performance, but not accurate forecasts. One example of such a commodity is lumber: this futures contract landed among the "top-performers" in most cases; however, the "optimal" model failed to capture the volatility of its returns. Besides, given the fact that we fail to provide undisputable evidence that any particular ML method can consistently outperform the benchmark, future outperformance of the ML portfolio is questionable. Thus, despite the fact that the ML portfolio yielded a higher out of sample return compared with the benchmark, the lack of robustness and the disability of some optimal models to capture the volatility patterns make the strategy practically inapplicable. A comparison of the performance of these portfolios can be found in Table 10 and Figure 3.

6. Conclusion

The purpose of this thesis was to test the out of sample forecasting power of machine learning models applied to model commodity futures returns. Our goal was also to analyze the approach to hyperparameter optimization suggested by Gu et al. (2020) and investigate whether modern tuning algorithms can suggest a better approach to choosing the most suitable architectures of ML models. Moreover, we extended the proposed set of models by adding Long Short-Term Memory networks that were proven successful in modeling various time series, including financial data. Finally, consistently with Gu et al. (2020), we constructed a portfolio of commodity futures based on the forecasts of the optimal ML models for each commodity and compared the out of sample performance of this portfolio with that of a portfolio suggested by the benchmark AR(1) model.

Our first finding is that each commodity requires separate modelling and hyperparameter tuning. Despite the fact that commodity futures are mostly driven by common factors and the correlation among the returns is generally positive, our experiment showed that different factors have a different impact on different commodities, and thus different model specifications and predictors are required to accurately forecast returns. This can be detected not only in the fact that the optimal models are generally different for each commodity, but also in the fact that the optimal predictor variables suggested by the decision tree regression do differ. Indeed, we find that commodities such as corn and silver are more forecastable from momentum and aggregate commodity factors, while light crude oil and lumber are mostly anticipated by the general macroeconomic outlook. A surprising finding is that the Fama-French factors were chosen by the decision tree regression as one of the five key variables in 6 cases out of 14, indicating that the linkages between commodity markets and equities are indeed strong, which can be used for forecasting purposes. While the academic research still debates about the strength and consistency of the linkages between the two asset classes (see Chong and Miffre, 2010 and Boyrie and Pavlova, 2016), we argue that equity factors improve the forecasting power in our case.

Similar to Guidolin and Pedio (2020), we find that neither tree-based methods nor neural networks are capable of consistently outperforming the benchmark AR(1). Given the

complexity of the dynamic dependence patterns between commodity futures and the predictors analyzed in this thesis, we believe the sample window is not sufficiently large to efficiently capture this dependence, which leads a large number of models to mean revert to the unconditional mean (or to values extremely close to it) as their implied forecast. Despite this fact, ML methods outperform the benchmark in 11 cases out of 14 according to both accuracy metrics; however, this outperformance is rarely (if ever) robust and we cannot recommend using our methodology in real world investment applications. Despite the fact that the ML methods allowed us to construct a portfolio whose cumulative out of sample return is around 57 percentage points larger than that of the benchmark portfolio, the robustness of this outperformance remains questionable.

Despite our failure to let a consistently better out of sample forecasting accuracy emerge, we believe that our thesis extends the existing body of research by introducing a consistent approach to hyperparameter tuning and model selection, compared with those suggested in Gu et al. (2020) and Struck and Cheng (2020). Algorithms such as the Keras tuner allow us to test the out of sample performance of a significantly larger number of architectures, which brings us closer to the optimal model specification. While the one "optimal" ML model is impossible to find, tuning algorithms allow us to make a significantly more educated choice of model specifications, ensuring the strongest possible out of sample performance. When structured and applied correctly, optimization algorithms find the right balance between shallow and deep, highly regularized and relaxed neural networks, taking their out of sample performance as the main criterion. This is confirmed by the results of our simulation exercise (Figures 1 and 2), which show that even despite the limited sample size the models can capture simple dependencies in the data after only fifteen tests conducted by the tuner.

We believe that further tests of our approach on larger sets of data would be beneficial, since the probability to find higher portfolio gains and forecasting power using this approach is by definition larger when compared against the approach that assumes default hyperparameters that do not imply any optimization with respect to the analyzed dataset. Indeed, our approach targets optimal out of sample performance based on the patterns observed in the training subsample, while models that assume default hyperparameters do not use the training subsample to identify the optimal hyperparameters, which makes these

models suboptimal. The sole drawback of the tuner algorithms is their high computational costs: the average computer can spend weeks searching for the optimal model if an unreasonably large search space and a large amount of data is analyzed. However, as we showed in this thesis, guided tuning algorithms such as the Bayesian tuner, combined with search space limitations suggested by earlier research can make the task computationally feasible, especially for new, fast machines.

A useful potential extension of this thesis is an adaptation of the suggested methodology to a larger sample and a different frequency of observations. While neural networks are known to be "data-hungry" and to perform best on significantly larger datasets, the low frequency of observations is arguably the main reason why the models failed to deliver accurate out of sample forecasts and capture the volatility patterns exhibited by the commodity series. While it is true that some predictors (such as commodity specific factors, for example) can be measured weekly at most, a larger focus could be put on predictor variables generated from the series of returns themselves, such as momentum and volatility estimates with longer rolling windows, as suggested in the study by Struck and Cheng (2020). While higher frequency data is also characterized by more noise and short-term fluctuations (discussed e.g. by Shiller, 2013), it also gives ML techniques more opportunities to learn and capture patterns, which can arguably mitigate the effect of short-term volatility.

Besides, a higher frequency would also enable the researcher to study interactions between predictors which, as argued by Gu et al. (2020), lead to higher predictive power in the case of equities. Another potential extension would be to test the same methodology on a classification problem: instead of forecasting the magnitude of returns, neural networks and hyperparameter optimization algorithms could be used to forecast the probabilities of upward/downward movements in futures prices. Leung et al. (2000) find that conventional classification models such as logit and probit lead to higher utility gains compared with models that aim to predict the magnitude of returns, and Dixon et al. (2016) find potential risk-adjusted portfolio gains from using deep neural networks to forecast the sign of asset returns.

End of Master Thesis I

MASTER THESIS II

7. Extension: Classification Problem

In this section we extend the employed methodology by reformulating the objective of the thesis into a classification problem. In other words, instead of forecasting the exact scale of price deviations of commodity futures, we will now focus on the direction of price changes, transforming the returns into a set of binary variables that take the value of one if the return is nonnegative and zero otherwise. As we mentioned previously, multiple studies (such as Leung et al., 2000) show increased efficiency of classification models when compared against regression-based approaches, both in terms of forecasting power and potential portfolio-level gains.

Given the inability of our models to consistently outperform the AR(1) benchmark when forecasting the exact scale of commodity futures price changes, it is still uncertain whether the same classes of models can outperform such benchmark classification models as logit and probit. Thus, we will compare the forecasting accuracy of benchmark logit models with that of the two types of neural networks used in the thesis: feedforward neural networks and LSTM. Although the analysis will be limited to neural networks, we will be using the same sets of predictor variables, which gives us a total of eight networks.

7.1 Introduction and Adjustments to the Methodology

Consistently with Leung et al. (2000), we will be using logit as our benchmark model; however, in order to make the comparison as full and transparent as possible, we will be using four versions of logit models based on the sets of predictor variables. This way, the performance of neural networks will be compared against that of four benchmark logit models with the following sets of predictor variables: full set, five first principal components, ten first principal components, and five key features selected by decision tree regressor. The same sets of predictor variables will be used in feedforward and LSTM neural networks for consistency, which gives us a total of four benchmark models and eight neural networks. We will rely on studies by Leung et al. (2000) and Dixon et al. (2016) when adapting our methodology to the classification case. However, the core methodology, set of variables, and

design of the portfolio exercise will remain consistent with Gu et al. (2020) and Guidolin and Pedio (2020).

As the first step of the analysis, we implement several adjustments to the methodology in order to approach the classification problem in a more consistent and accurate way. Namely, we start by substituting the Huber loss function used in the original exercise with binary cross-entropy, which is a more common choice in classification exercises (Dixon et al., 2020).

According to Godoy (2018), binary cross-entropy is a loss function of the following form:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log\left(p(y_i)\right) + (1 - y_i)\log\left(1 - p(y_i)\right) \tag{32}$$

Where y_i is the value of the dependent variable on date i and $p(y_i)$ is the probability that the dependent variable takes the value of 1 on date i. This loss function uses natural logarithms of forecasted probabilities to penalize the model for large deviations from the actual, observable value of the dependent variable, which makes binary cross-entropy a more suitable choice for classification exercises than the classic MSE or Huber loss.

Besides, we use the sigmoid activation function in the output layer of each of our neural networks: this bounds the output of neural networks between 0 and 1, allowing us to interpret the output as probabilities of upward futures price movements. Given that Tensorflow and Keras use the linear activation function by default, the use of sigmoid allows us to get a smoother distribution of forecasts in the tails, since the linear activation function can return both negative "probabilities" and "probabilities" above one.

$$S(x) = \frac{1}{1 + e^{-x}} \tag{33}$$

Finally, we use the ratio of correct forecasts to total forecasts (hereafter Accuracy) as our key performance metric, as opposed to RMSFE and MAFE in the original exercise: the choice of this metric is consistent with Leung et al. (2000) and Dixon et al. (2016). Probabilities of 0.5 and above will be considered as upward movement forecasts, and these forecasts will be compared with the binary dependent variables from the test subsample.

The first step of the analysis is to re-run the set of optimization algorithms in order to find the optimal hyperparameters for the set of analyzed neural networks. Given the set of adjustments implemented in the methodology and the overall goal of the exercise, it would be unreasonable to expect the initial sets of hyperparameters to deliver an optimal performance, so re-estimating hyperparameter values is a vital step in ensuring that the neural networks are optimized for the new task. Thus, we use 100-125 runs of the Bayesian optimization tuner in order to find new optimal hyperparameters for each neural network (see Table 11 for optimal hyperparameter combinations). Consistently with our initial methodology, we are using an ensemble of five neural networks to estimate forecasted probabilities: this allows us to minimize the contribution of randomness inherent to all network-based methods.

After estimating the probability forecasts, we reconstruct the portfolio exercise discussed by Gu et al. (2020), adapting it to the new, classification problem. Namely, we retrieve the probabilities of upward price movements from the best-performing model for each commodity futures, assuming a long position in the top four commodities and a short position in the bottom four commodities based on the retrieved probabilities. Thus, forecasted probabilities determine the choice of assets to be included in the portfolio, and the preference is given to the more "certain" forecasts. The performance of this ML portfolio is compared with that of four benchmark portfolios constructed based on the forecasted probabilities retrieved from the four benchmark logit models. Besides, given the generally positive correlation between commodity futures (Table 2), the original restriction is still in place: a long (short) position can be assumed only if the forecasted probability of an upward price movement is above (below) 0.5. Finally, we assume that all commodity futures contracts within each portfolio have equal absolute weights.

7.2 Empirical Results

In this section we describe the out of sample performance of the analyzed neural networks relative to the benchmark logit models and discuss the results of our portfolio exercise.

One of the very first observations is that the learning process of our neural networks is far from smooth. Well-trained neural networks are expected to consistently maximize validation accuracy and minimize validation loss as the number of epochs increases; however, in our case these indicators exhibit an unreasonable degree of volatility, suggesting that models cannot capture the complex dependencies between the observable time series. Similarly to our initial exercise, a significant portion of networks converges to long-run mean probability forecasts, failing to adapt the forecasts to changes in predictor variables. This finding is consistent with our initial regression-based exercise, which suggests that the analyzed data frequency does not provide a sufficient number of observations to ensure a smooth training process.

We also find that consistently with the original regression exercise, ML models fail to regularly outperform the benchmark models based on out of sample forecast accuracy metrics, which can be seen from Table 12. Logit models maximize the out of sample forecast accuracy in five cases out of fourteen, suggesting that more conventional econometric techniques cannot be neglected in modelling return series of commodity futures. It is also worth mentioning that all best-performing models show a consistent accuracy of at least 50%, suggesting that a combination of the analyzed methods can generate positive expected returns in real-world investing applications; however, further robustness checks are required.

Despite the failure to consistently outperform the benchmark models in terms of forecast accuracy, the results of our portfolio exercise show evidence of a significant outperformance of the ML portfolio. Namely, the ML portfolio generates a 37.9% return over the testing period from September 2012 to June 2020, while the best-performing benchmark portfolio generates the return of 19.64%. The best-performing benchmark portfolio is based on the forecasts of logit models that use five optimal DTR variables as the set of predictors; this portfolio is followed by the ten principal component benchmark portfolio, the "kitchen sink" benchmark portfolio, and finally the five principal component portfolio which is the only portfolio that generated a negative cumulative return over the testing period. Surprisingly, while the ML portfolios outperform the benchmark in both regression and classification stages of the analysis, we find that the classification-based portfolio generates a lower

absolute return over the testing period. Indeed, the returns generated by the regression-based ML portfolio are larger by 25.7 percentage points, which is inconsistent with the study by Leund et al. (2000), who find higher portfolio level gains from the use of classification-based models.

All other findings are expectedly consistent with our initial regression-based exercise. Namely, our hyperparameter tuning algorithms still rely on shallower model specifications, and the set of optimal predictor variables suggested by the decision tree regression is similar to the one used in the case of regression-based methods.

7.3 Conclusion

The purpose of this extension was to compare the out of sample performance of ML algorithms with that of a set of benchmark models to see whether consistent benchmark outperformance and increased portfolio-level gains could be obtained. The extension focuses on binary classification and forecasted probabilities of price movements rather than the magnitude of returns. Thus, we compared the performance of two classes of neural networks (feedforward networks and LSTM) with a set of benchmark logit models using both out of sample forecast accuracy metrics and aggregate portfolio performance.

Our main finding is that the set of analyzed ML techniques cannot consistently outperform conventional benchmark models in both cases: while the AR(1) benchmark model was selected as the optimal model in three cases out of fourteen in the regression part of the thesis, different kinds of logit models outperform the ML techniques in five cases out of fourteen in the classification part. The lack of training consistency indicates that the current sample size is insufficient to accurately capture the complex dependencies between commodity futures and predictors, allowing us to conclude that higher frequency data has to be analyzed in order to ensure that the training process of data-hungry ML algorithms is conducted smoothly. The biggest empirical problem was the observed tendency of neural networks to converge to their long-run forecasts, exhibiting minor deviations from these long-run probabilities. This problem is similar to the one observed in the regression part of the thesis, where forecasts would converge to sample mean returns, rather than

probabilities of upward price movements. Considering a highly volatile training process of the analyzed neural networks, we believe that using a larger sample of higher frequency data (as suggested by Struck and Cheng, 2020) could be useful in solving this issue. Besides, higher frequency data would allow us to focus on rolling window volatility and mean return estimates that were proven successful in forecasting returns of commodity futures (Struck and Cheng, 2020). Finally, Gu et al. (2020) argue that interactions between predictor variables carry information useful for forecasting returns of equities, and higher data frequency data would enable us to test similar interactions in the case of commodity futures.

Despite the failure to consistently outperform the benchmark, we find significant portfolio-level gains from the use of ML models in both regression and classification parts of the thesis (Figure 3 and Figure 4 respectively). Surprisingly, we find smaller portfolio level gains in the classification part of the thesis, which contradicts the finding by Leung et al. (2000) who claim that classification-driven investment decisions lead to higher potential portfolio-level returns. However, it is important to mention that given the observed failure to consistently outperform the benchmark models and the tendency of the analyzed ML models to converge to their ergodic mean forecasts, these potential gains have to be treated with caution, and their application in real-world investment decisions is limited.

This way, a final potential extension would be the application of the analyzed ML techniques on higher frequency data, which would provide the data-hungry neural networks with a more suitable environment to operate in. Despite covering a relatively large time period (namely, from 1989 to 2020), we use monthly frequency which gives us a total of 377 observations, which is shown to be insufficient to ensure a smooth training process of the analyzed neural networks. Thus, it would be useful to test the applied ML algorithms on higher frequency data to see whether a larger number of observations can increase the forecasting power of the analyzed neural networks and ensure a smoother training process.

References

Akyildirim, E., Goncu, A., & Sensoy, A. (2020). Prediction of Cryptocurrency Returns Using Machine Learning. *Annals of Operations Research*, 297(1), 3-36.

Algieri, B., & Kalkuhl, M. (2019). Efficiency and Forecast Performance of Commodity Futures Markets. *American Journal of Economics and Business Administration*, 11(1), 19-34.

Athey, S., & Imbens, G. W. (2019). Machine Learning Methods That Economists Should Know About. *Annual Review of Economics*, 11(1), 685-725.

Bali, T. G., Goyal, A., Huang, D., Jiang, F., & Wen, Q. (2020). "The Cross-Sectional Pricing of Corporate Bonds Using Big Data and Machine Learning". Swiss Finance Institute, working paper.

Bhandari, A. (2020). *Feature Scaling for Machine Learning: Understanding the Difference Between Normalization vs. Standardization.* Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/

Blocher, J., Cooper, R., & Molyboga, M. (2016). Benchmarking Commodity Investments. *Journal of Futures Markets*, 38(3), 340-358.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification And Regression Trees.* CRC Press.

Breiman, L. (2004). Population Theory for Boosting Ensembles. *The Annals of Statistics*, 32(1), 1-11.

Chong, J., & Miffre, J. (2010). Conditional Correlation and Volatility in Commodity Futures and Traditional Asset Markets. *Journal of Alternative Investments*, 12(3), 61-75.

Coqueret, G., & Guida, T. (2020). *Machine Learning for Factor Investing.* CRC Press.

Cramer, S., Kampouridis, M., Freitas, A. A., & Alexandridis, A. K. (2017). An Extensive Evaluation of Seven Machine Learning Methods for Rainfall Prediction in Weather Derivatives. *Expert Systems With Applications*, 85(1), 169-181.

Creti, A., Joets, M., & Mignon, V. (2013). On the Links Between Stock and Commodity Markets' Volatility. *Energy Economics*, 37, 16-28.

Daskalaki, C., Kostakis, A., & Skiadopoulos, G. (2013). Are There Common Factors in Individual Commodity Futures Returns? *Journal of Banking & Finance*, 40, 346-363.

de Boyrie, M. E., & Pavlova, I. (2016). "Linkages Between Equity and Commodity Markets: Are Emerging Markets Different?". New Mexico State University, working paper.

Deisenroth, M. P., Faisal, A., & Ong, C. (2020). *Mathematics for Machine Learning.* Cambridge University Press.

Dewancker, I., McCourt, M., & Clark, S. (n.d.). *Bayesian Optimization Primer.* Retrieved from https://static.sigopt.com/b/20a144d208ef255d3b981ce419667ec25d8412e2/static/pdf/SigOpt_Bayesian_Optimization_Primer.pdf

Dixon, M. F., & Halperin, I., 2019. "The Four Horsemen of Machine Learning". Illinois Institute of Technology, working paper.

Dixon, M. F., Halperin, I., & Bilokon, P. (2020). *Machine Learning in Finance: From Theory to Practice.* Springer Nature Switzerland AG.

Dixon, M. F., Klabjan, D., & Bang, J. (2016). "Classification-Based Financial Markets Prediction Using Deep Neural Networks". Northwestern University, working paper.

Fama, E. F., & French, K. R. (2015). A Five-Factor Asset Pricing Model. *Journal of Financial Economics*, 116(1), 1-22.

Freund, Y., & Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference* (pp. 148-156). International Conference on Machine Learning.

Fuertes, A.-M., Miffre, J., & Fernandez-Perez, A. (2015). Commodity Strategies Based on Momentum, Term Structure, and Idiosyncratic Volatility. *Journal of Futures Markets*, 35(3), 274-297.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. *Proceedings of Machine Learning Research*, 15, 315-323.

Godoy, D. (2018, November 21). *Understanding binary cross-entropy/log loss: a visual explanation.* Retrieved from Towards Data Science: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a.

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

Guestrin, C., & Chen, T. (2016). XGBoost: A Scalable Tree Boosting System. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794). Knowledge Discovery and Data Mining International Conference.

Guidolin, M., & Pedio, M. (2020). Distilling Large Information Sets to Forecast Commodity Retunrs: Automatic Variable Selection or Hidden Markov Models? BAFFI CAREFIN Working Paper No. 20140.

Guidolin, M., & Pedio, M. (2020). Forecasting Commodity Futures Returns with Stepwise Regressions: Do Commodity-Specific Factors Help? *Annals of Operations Research,* 299(1), 1317-1356.

Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2), 107-116.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735-1780.

Huang, X., Guidolin, M., Platanakis, E., & Newton, D. P. (2021). "Dynamic Portfolio Management with Machine Learning". University of Bath – School of Management and Bocconi University, working paper.

Huber, P. J. (1992). Robust Estimation of a Location Parameter. *Breakthroughs in Statistics. Springer Series in Statistics (Perspectives in Statistics)*.

Hull, I. (2020). *Machine Learning for Economics and Finance in TensorFlow 2.* Apress.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., & LeCun, Y. (2009). What is the Best Multi-Stage Architecture for Object Recognition? *IEEE 12th International Conference on Computer Vision* (pp. 2146-2153). IEEE.

Kaminsky, G., & Kumar, M. S. (1990). Efficiency in Commodity Futures Markets. IMF *Staff Papers*, 37, 670-699.

Kellard, N., Newbold, P., Rayner, T., & Ennew, C. (1999). The Relative Efficiency of Commodity Futures Markets. *Journal of Futures Markets*, 19, 413-432.

Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *The International Conference on Learning Representations.* ICLR.

Kira, K., & Rendell, L. A. (1992). A Practical Approach to Feature Selection. *Proceedings of the Ninth International Workshop on Machine Learning* (pp. 249-256).

Leung, M. T., Daouk, H., & Chen, A.-S. (2000). Forecasting Stock Indices: a Comparison of Classification and Level Estimation Models. *International Journal of Forecasting*, 16(2), 173-190.

Ludvigson, S. C., & Ng, S. (2009). Macro Factors in Bond Risk Premia. *Review of FInancial Studies*, 22(12), 5027-5067.

Masters, T. (1993). *Practical Neural Network Recipes in C++.* Academic Press, Inc.

Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In J. Furnkranz and T. Joachims (Eds.), *International Conference on Machine Learning* (pp. 807-814). ICML.

Ni, T. (2019). Data Analysis and Machine Learning: on Long Memory Commodity Time Series (Unpublished master's dissertation). University of Liverpool, Liverpool, United Kingdom.

Oppermann, A. (2020, February 19). *Regularization in Deep Learning - L1, L2, and Dropout.* Retrieved from Towards Data Science: https://towardsdatascience.com/regularization-in-deep-learning-l1-l2-and-dropout-377e75acc036.

Ouyang, H., Wei, X., & Wu, Q. (2019). Agricultural Commodity Futures Prices Prediction via Long- and Short-term Time Series Network. *Journal of Applied Economics*, 22(1), 468-483.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L., & Hinton, G. (2017). Regularizing Neural Networks by Penalizing Confident Output Distributions. *International Conference on Learning Representations.* Toulon: ICLR.

Sadorsky, P. (2021). A Random Forests Approach to Predicting Clean Energy Stock Prices. *Journal of Risk and Financial Management*, 14(2), 1-20.

Schapire, R. E., & Freund, Y. (2012). *Boosting Foundations and Algorithms.* Cambridge, London: The MIT Press.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & de Freitas, N. (2015). Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of The Institute of Electrical and Electronics Engineers, 104(1), 148-175.*

Shiller, R. J. (2013, December 8). Speculative Asset Prices. *Nobel Prize Lecture.*The Nobel Prize. Retreived from https://www.nobelprize.org/uploads/2018/06/shiller-lecture.pdf

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56), 1929-1958.

Struck, C., & Cheng, E. (2020). The Cross Section of Commodity Returns: A Nonparametric Approach. *The Journal of Financial Data Science*, 2(3), 86-103.

Swanson, N. R., & White, H. (1995). A Model-Selection Approach to Assessing the Information in the Term Structure Using Linear Models and Artificial Neural Networks. *Journal of Business & Economic Statistics*, 13(3), 265-275.

Trippi, R. R., & Desieno, D. (1992). Trading Equity Index Futures With a Neural Network. *Journal of Portfolio Management*, 19(1), 27-33.

van der Ploeg, T., Austin, P. C., & Steyerberg, E. W. (2014). Modern Modelling Techniques are Data Hungry: a Simulation Study for Predicting Dichotomous Endpoints. *BMC Medical Research Methodology*.

Wang, C.-F. (2019, January 8). *The Vanishing Gradient Problem: The Problem, Its Causes, Its Significance, and Its Solutions.* Retrieved from Towards Data Science: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

Wilmott, P. (2019). *Machine Learning: An Applied Mathematics Introduction.* Panda Ohana Publishing.

Ye, T., & Zhang, L. (2019). Derivatives Pricing via Machine Learning. *Journal of Mathematical Finance*, 9(3), 561-589.

Table 1

Descriptive Statistics for Commodity Futures Returns and Factors

Panel A **Commodity Futures Returns**

	Mean	Median	Standard Deviation	Skewness	Excess Kurtosis
Light Crude Oil	0.0068	0.0077	0.0961	0.6137	6.8628
Corn	0.0031	0.0000	0.0739	0.2241	0.7525
Soybeans	0.0025	-0.0002	0.0676	-0.2984	1.2123
Wheat	0.0035	-0.0020	0.0802	0.6784	2.5456
Coffee	0.0038	-0.0096	0.1021	1.0520	3.2504
Cocoa	0.0044	0.0000	0.0827	0.4072	0.8372
Sugar	0.0038	-0.0020	0.0855	0.2799	1.0797
Cotton	0.0030	0.0044	0.0756	0.1008	0.3717
Gold	0.0049	0.0004	0.0440	0.1609	1.2003
Silver	0.0061	-0.0017	0.0796	0.1742	0.9695
Platinum	0.0034	0.0037	0.0610	0.1511	6.9707
Frozen Orange Juice	0.0034	-0.0001	0.0903	0.5575	1.1142
Lumber	0.0055	0.0030	0.0820	0.2845	0.4934
Live Cattle	0.0018	0.0028	0.0468	-0.6000	2.8585

Panel B **Hedging Pressure**

	Mean	Median	Standard Deviation	Skewness	Excess Kurtosis
Light Crude Oil	0.0660	0.0512	0.1060	0.1091	-0.6767
Corn	0.0178	0.0247	0.1295	-0.1814	-0.6744
Soybeans	0.0931	0.1124	0.1627	-0.1701	-0.7748
Wheat	0.0168	-0.0080	0.1514	0.4941	-0.6627
Coffee	0.1218	0.1252	0.1629	-0.0093	-1.0584
Cocoa	0.1313	0.1338	0.1565	-0.0764	-0.5274
Sugar	0.1553	0.1705	0.1885	0.0184	-0.6801
Cotton	0.0908	0.1093	0.2226	-0.2890	-0.8171
Gold	0.2145	0.3031	0.2778	-0.6497	-0.8482
Silver	0.3871	0.3930	0.1598	-0.1588	-0.4378
Platinum	0.4488	0.4864	0.2349	-0.6253	-0.4085
Frozen Orange Juice	0.1828	0.2256	0.2606	-0.3952	-0.6611
Lumber	0.0712	0.0498	0.1988	0.0950	-0.7914
Live Cattle	0.0701	0.0629	0.1072	0.0136	-0.9274

Table 1 (continued)

Descriptive Statistics for Commodity Futures Returns and Factors

Panel C **Basis**

	Mean	Median	Standard Deviation	Skewness	Excess Kurtosis
Light Crude Oil	-0.0020	-0.0023	0.0231	-2.5719	19.3927
Corn	-0.0200	-0.0240	0.0307	3.5431	21.0297
Soybeans	-0.0020	-0.0077	0.0218	2.7195	9.4695
Wheat	-0.0204	-0.0248	0.0306	1.5734	4.1569
Coffee	-0.0170	-0.0211	0.0293	1.8336	5.9854
Cocoa	-0.0108	-0.0098	0.0201	0.0424	1.2058
Sugar	-0.0024	-0.0055	0.0492	-0.0141	4.2124
Cotton	-0.0102	-0.0146	0.0389	2.0637	12.8607
Gold	-0.0025	-0.0023	0.0018	-0.6308	-0.1248
Silver	0.0008	-0.0026	0.0502	13.5710	182.8021
Platinum	0.0360	-0.0007	0.1758	4.9210	23.2381
Frozen Orange Juice	-0.0089	-0.0095	0.0288	0.7264	0.7839
Lumber	-0.0205	-0.0162	0.0451	-0.0972	0.5998
Live Cattle	0.0030	0.0004	0.0362	0.4075	-0.0789

Panel D **Net Trading**

	Mean	Median	Standard Deviation	Skewness	Excess Kurtosis
Light Crude Oil	0.0074	0.0043	0.0506	0.2870	1.2997
Corn	0.0060	-0.0002	0.0746	-0.0267	8.2348
Soybeans	0.0074	0.0094	0.0819	-0.3725	3.9093
Wheat	0.0071	0.0005	0.0910	0.1013	4.4926
Coffee	0.0112	0.0206	0.0981	0.2754	0.5015
Cocoa	0.0072	0.0099	0.0784	-0.1107	1.0623
Sugar	0.0078	0.0062	0.0782	-0.1134	0.2407
Cotton	0.0098	0.0119	0.1024	0.1827	1.3654
Gold	0.0073	0.0003	0.0944	0.4213	1.2585
Silver	0.0056	0.0032	0.0814	0.4144	2.6158
Platinum	0.0093	0.0086	0.1081	0.3042	1.1899
Frozen Orange Juice	0.0056	0.0020	0.1095	0.4082	0.7570
Lumber	0.0085	-0.0054	0.1506	1.8085	11.0145
Live Cattle	0.0074	0.0031	0.0754	0.1642	0.3888

Table 1 (continued)

Descriptive Statistics for Commodity Futures Returns and Factors

Panel E **Aggregate Commodity Factors**

	Mean	Median	Standard Deviation	Skewness	Excess Kurtosis
Hedging Pressure	0.0986	0.1004	0.0754	-0.3121	-0.2061
Basis	-0.0053	-0.0074	0.0145	2.0420	6.8304
Net Trading	0.0043	0.0036	0.0413	-1.5402	15.5081

Table 2

Correlation Matrix of Commodity Series

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton No.2	Gold	Silver	Platinum	Orange Juice	Lumber	Live Cattle
Light Crude Oil	**1.00**	0.10	0.14	0.07	0.04	0.15	0.11	0.16	0.18	0.25	0.27	0.04	0.17	0.12
Corn	0.10	**1.00**	0.67	0.62	0.18	0.17	0.12	0.32	0.17	0.22	0.13	0.15	0.11	-0.01
Soybeans	0.14	0.67	**1.00**	0.45	0.19	0.11	0.10	0.36	0.17	0.17	0.17	0.13	0.16	-0.01
Wheat	0.07	0.62	0.45	**1.00**	0.18	0.10	0.09	0.19	0.16	0.12	0.09	0.12	0.13	0.05
Coffee	0.04	0.18	0.19	0.18	**1.00**	0.14	0.12	0.10	0.17	0.19	0.16	0.15	0.06	-0.06
Cocoa	0.15	0.17	0.11	0.10	0.14	**1.00**	0.14	0.14	0.16	0.20	0.16	0.06	0.08	0.00
Sugar	0.11	0.12	0.10	0.09	0.12	0.14	**1.00**	0.12	0.11	0.13	0.23	0.07	0.11	0.09
Cotton No.2	0.16	0.32	0.36	0.19	0.10	0.14	0.12	**1.00**	0.09	0.14	0.28	0.06	0.20	0.02
Gold	0.18	0.17	0.17	0.16	0.17	0.16	0.11	0.09	**1.00**	0.72	0.46	0.07	0.12	-0.04
Silver	0.25	0.22	0.17	0.12	0.19	0.20	0.13	0.14	0.72	**1.00**	0.56	0.06	0.17	0.01
Platinum	0.27	0.13	0.17	0.09	0.16	0.16	0.23	0.28	0.46	0.56	**1.00**	0.03	0.18	0.10
Orange Juice	0.04	0.15	0.13	0.12	0.15	0.06	0.07	0.06	0.07	0.06	0.03	**1.00**	0.02	0.02
Lumber	0.17	0.11	0.16	0.13	0.06	0.08	0.11	0.20	0.12	0.17	0.18	0.02	**1.00**	0.09
Live Cattle	0.12	-0.01	-0.01	0.05	-0.06	0.00	0.09	0.02	-0.04	0.01	0.10	0.02	0.09	**1.00**

Table 3

Shares of In-Sample Variance Explained by the First 10 Principal Components

	1	2	3	4	5	6	7	8	9	10
Light Crude Oil	0.2549	0.3443	0.4232	0.4884	0.5409	0.5903	0.6291	0.6664	0.7014	0.7343
Corn	0.2557	0.3504	0.4278	0.4960	0.5499	0.5995	0.6412	0.6810	0.7169	0.7507
Soybeans	0.2561	0.3466	0.4215	0.4891	0.5410	0.5892	0.6296	0.6687	0.7053	0.7406
Wheat	0.2556	0.3425	0.4157	0.4795	0.5300	0.5766	0.6181	0.6575	0.6936	0.7286
Coffee	0.2557	0.3421	0.4137	0.4786	0.5354	0.5856	0.6266	0.6651	0.6995	0.7315
Cocoa	0.2560	0.3402	0.4133	0.4787	0.5324	0.5796	0.6205	0.6591	0.6931	0.7267
Sugar	0.2551	0.3382	0.4107	0.4749	0.5258	0.5732	0.6160	0.6550	0.6885	0.7205
Cotton	0.2549	0.3402	0.4147	0.4806	0.5349	0.5833	0.6230	0.6606	0.6950	0.7274
Gold	0.2570	0.3602	0.4402	0.5040	0.5551	0.6041	0.6428	0.6803	0.7133	0.7452
Silver	0.2559	0.3464	0.4185	0.4830	0.5340	0.5813	0.6220	0.6606	0.6947	0.7270
Platinum	0.2557	0.3492	0.4290	0.4962	0.5519	0.5989	0.6391	0.6761	0.7103	0.7411
Orange Juice	0.2552	0.3404	0.4133	0.4767	0.5293	0.5777	0.6190	0.6568	0.6914	0.7227
Lumber	0.2557	0.3425	0.4214	0.4839	0.5355	0.5828	0.6244	0.6615	0.6966	0.7276
Live Cattle	0.2546	0.3370	0.4091	0.4720	0.5234	0.5728	0.6158	0.6545	0.6894	0.7213

Table 4

Optimal Variables Selected by Decision Tree Regressor

The table reports the five predictor variables that carry the highest explanatory power according to the Decision Tree Regressor, ranked from highest to lowest. CMA, RMW, Mkt-Rf, SMB are Fama-French factors, F# are macroeconomic factors (Ludvigson and Ng, 2009), L(commodity) are lagged returns of respective commodity futures, _aggr implies that the aggregate commodity factors are used.

	1	2	3	4	5
Light Crude Oil	CMA	F7	basis	HP_aggr	AR(2)
Corn	AR(10)	AR(11)	Mkt-Rf	NT	basis_aggr
Soybeans	basis	F2	AR(11)	AR(10)	F1
Wheat	AR(5)	NT_aggr	RMW	L(Wheat)	basis_aggr
Coffee	F1^3	HP	F4	AR(5)	SMB
Cocoa	AR(10)	AR(11)	basis	F6	AR(3)
Sugar	L(Sugar)	basis_aggr	AR(12)	AR(9)	F1^3
Cotton No.2	NT_aggr	basis	F2	F1	HP_aggr
Gold	F3	F1	AR(5)	avg_rend	AR(1)
Silver	AR(10)	AR(5)	basis_aggr	SMB	F4
Platinum	NT_aggr	F1	basis	AR(11)	F1^3
Orange Juice	Basis_aggr	AR(4)	NT_aggr	AR(3)	F6
Lumber	NT_aggr	F7	AR(7)	F6	F2
Live Cattle	CMA	AR(7)	F5	F7	AR(12)

Table 5

Search Space for Hyperparameter Optimization

Neural Networks

Hyperparameter	Range/Value
Learning Rate	(1e-6, 1e-1)
Batch Size	{1, 2, 3, …, 7}
Number of Epochs	{min=3, max=103, step=4}
Dropout Rate	{0, 0.25, 0.5}
Number of Layers*	{1, 2, …, 5}[1]
Number of Neurons per Layer	{min=3, max=33, step=3}
L1 rate	(1e-8, 1e-3)
L2 rate	(1e-8, 1e-3)
Loss Function	Huber
Optimizer	Adam
Activation Function	ReLU
Ensemble	5
Early Stopping Patience	5
Tuner Simulations	{125, 150}

Random Forests

Hyperparameter	Range/Value
Maximum Depth	{1, 2, 3 ,4, 5, 10}
Maximum Number of Features	{auto, sqrt, 3, 5, 7, 10, 20, 34}
Minimum Samples per Leaf	{2, 4, 8, 12}
Number of Estimators	{50, 100, 200, 250}

Boosting Algorithms

Hyperparameter	Range/Value
Number of Estimators	{20, 35, 50, 100, 150}
Learning Rate	{0.1, 0.25, 0.5, 0.75, 1}

[1] Number of layers including the input layer but excluding the output layer. Only the networks estimated on five principal components and five optimal predictor variables include 1 layer as a potential hyperparameter value. Other networks are assumed to have at least one hidden layer.

Table 6
Optimal Hyperparameters: Tree-Based Methods

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton	Gold	Silver	Platinum	Orange Juice	Lumber	Live Cattle
Panel A	**Random Forests, Full Set of Variables**													
Max. Depth	1	1	1	1	1	1	3	1	2	2	10	2	3	4
Max. Features	sqrt	20	auto	34	5	34	sqrt	3	5	7	3	sqrt	7	5
Min. Samples/Leaf	4	2	4	12	4	12	2	4	12	4	8	12	12	2
Estimators	50	50	250	100	100	50	50	50	50	50	50	200	50	100
Panel B	**Random Forests, 5 Principal Components**													
Max. Depth	1	1	2	2	1	1	5	1	1	3	2	1	3	1
Max. Features	sqrt	auto	sqrt	sqrt	sqrt	sqrt	sqrt	sqrt	sqrt	sqrt	sqrt	sqrt	sqrt	sqrt
Min. Samples/Leaf	8	8	12	4	12	8	8	12	8	12	8	8	12	4
Estimators	100	50	50	100	200	50	100	100	50	50	100	100	100	100
Panel C	**Random Forests, 10 Principal Components**													
Max. Depth	1	2	1	1	1	1	5	1	5	1	1	1	3	2
Max. Features	sqrt	sqrt	sqrt	sqrt	sqrt	sqrt	sqrt	sqrt	sqrt	sqrt	auto	sqrt	sqrt	sqrt
Min. Samples/Leaf	12	8	8	12	12	12	4	4	12	12	4	8	12	4
Estimators	50	50	50	100	50	100	100	50	100	50	200	100	100	100
Panel D	**AdaBoost, Full Set of Variables**													
Learning Rate	0.1	0.5	0.1	0.5	0.25	0.25	0.25	0.1	1	0.1	0.1	0.5	0.1	0.1
Estimators	20	35	20	20	50	20	20	50	35	50	150	150	100	50
Panel E	**AdaBoost, 5 Principal Components**													
Learning Rate	0.1	0.25	0.25	0.1	0.5	0.1	0.25	0.25	0.1	0.1	1	0.1	0.1	0.25
Estimators	35	20	100	50	35	50	100	20	150	100	35	35	35	20
Panel F	**AdaBoost, 10 Principal Components**													
Learning Rate	0.25	0.1	0.1	0.25	0.25	0.25	0.1	0.5	0.1	0.25	1	0.1	0.1	0.1
Estimators	50	50	50	20	35	20	20	50	20	50	100	20	20	20
Panel G	**XGBoost, Full Set of Variables**													
Learning Rate	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.25	0.1	0.1	0.1
Estimators	20	20	20	20	20	20	20	20	20	20	35	20	20	20

Table 6 (continued)

Optimal Hyperparameters: Tree-Based Methods

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton	Gold	Silver	Platinum	Orange Juice	Lumber	Live Cattle
Panel H	**XGBoost, 5 Principal Components**													
Learning Rate	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
Estimators	10	10	10	10	10	10	10	10	10	10	10	10	10	10
Panel I	**XGBoost, 10 Principal Components**													
Learning Rate	0.0001	0.0001	0.0001	0.1	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.25
Estimators	1	50	150	1	1	1	150	1	1	1	1	150	1	1

Table 7
Optimal Hyperparameters: Neural Networks

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton	Gold	Silver	Platinum	Orange Juice	Lumber	Live Cattle
Panel A	**Feedforward Neural Network, Full Set of Variables**													
Batch Size	2	1	1	1	5	6	1	1	2	6	1	7	3	5
Epochs	67	83	19	95	23	99	91	103	99	83	99	79	95	31
Dropout Rate	0.5	0.25	0.25	0	0.25	0.25	0.5	0.5	0.5	0	0.25	0.25	0	0
Learning Rate	0.0067	0.0033	0.0028	0.0008	0.0115	0.0004	0.0048	0.0027	0.0024	0.0002	0.0014	0.0126	0.0064	0.0026
Layers	2	3	2	2	4	3	3	2	2	5	2	5	2	3
Units (1)	3	3	3	12	3	3	6	3	3	6	15	3	6	21
Units (2)	15	3	3	15	6	12	18	24	27	9	27	6	12	33
Units (3)	-	6	-	-	3	3	27	-	-	27	-	33	-	30
Units (4)	-	-	-	-	24	-	-	-	-	15	-	3	-	-
Units (5)	-	-	-	-	-	-	-	-	-	9	-	6	-	-
L1 Rate	0.000406	0.000417	8.25E-05	0.000386	0.000277	2.58E-07	0.000118	4.62E-06	0.000421	0.000134	0.000147	5.9E-07	0.000174	4.97E-05
L2 Rate	0.000153	0.000359	0.000925	0.000353	0.000346	0.000698	0.000723	0.000377	0.000275	2.93E-05	0.000573	4.04E-05	0.000644	0.00052
Panel B	**Feedforward Neural Network, 5 DTR Features**													
Batch Size	6	4	7	2	6	7	4	5	7	3	3	1	7	2
Epochs	91	35	67	59	91	39	63	99	103	47	51	47	67	59
Dropout Rate	0	0.5	0.25	0.5	0	0	0	0	0	0	0	0.5	0	0.25
Learning Rate	0.059777	0.069973	0.005635	0.038596	0.075986	0.048298	0.023968	0.002334	0.037206	0.015455	0.005227	0.014833	0.032392	0.007549
Layers	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Units (1)	3	21	24	21	21	3	21	18	3	3	6	6	15	6
Units (2)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (3)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (4)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (5)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
L1 Rate	1.48E-05	1.47E-05	8E-05	2.07E-05	1.21E-05	6.07E-05	9.75E-05	0.000141	3.32E-05	2.14E-05	4.93E-05	1.23E-05	1.59E-05	0.000123
L2 Rate	0.000322	0.000168	0.000373	0.000543	0.000962	0.000168	0.000139	0.000577	0.000189	9.93E-05	0.000522	0.000596	0.000626	0.000473

Table 7 (continued)
Optimal Hyperparameters: Neural Networks

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton	Gold	Silver	Platinum	Orange Juice	Lumber	Live Cattle
Panel C	**Feedforward Neural Network, 5 Principal Components**													
Batch Size	7	4	7	7	2	7	7	6	7	3	7	4	5	1
Epochs	103	95	75	99	99	67	103	95	99	67	83	103	91	103
Dropout Rate	0.5	0	0.5	0.25	0	0.25	0	0.25	0	0.5	0	0.25	0	0
Learning Rate	0.041703	0.024614	0.062995	0.021979	0.011303	0.080743	0.038358	0.056008	0.012172	0.029922	0.011816	0.011317	0.067003	0.049049
Layers	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Units (1)	33	6	30	6	3	3	3	6	9	9	21	33	6	24
Units (2)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (3)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (4)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (5)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
L1 Rate	0.000316	0.000267	0.000129	0.000393	0.000169	7.82E-06	0.000916	3.25E-07	0.000262	7.61E-05	0.000157	0.000208	4.59E-05	4.53E-05
L2 Rate	1.24E-07	2.26E-05	0.000607	0.000702	0.000787	0.000728	9.85E-05	0.000703	0.000626	0.000188	0.000219	0.000309	0.000533	0.000742
Panel D	**Feedforward Neural Network, 10 Principal Components**													
Batch Size	6	5	6	1	6	7	3	7	3	5	1	3	2	5
Epochs	35	39	63	27	91	99	67	47	83	87	11	99	63	11
Dropout Rate	0.5	0.5	0.5	0.25	0	0	0.25	0	0	0.5	0.5	0.5	0	0.5
Learning Rate	0.012269	0.007363	0.02419	0.001111	0.08209	0.047543	0.005024	0.004107	0.028876	0.0405	0.021503	0.068404	0.017511	0.017593
Layers	4	2	2	3	3	2	2	2	2	2	2	3	3	2
Units (1)	3	30	27	3	6	15	6	6	3	9	12	21	9	3
Units (2)	27	3	18	6	3	3	6	3	18	9	9	9	12	18
Units (3)	6	-	-	6	6	-	-	-	-	-	-	30	3	-
Units (4)	33	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (5)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
L1 Rate	6.74E-06	3.86E-05	3.02E-07	0.00037	2.18E-06	1.62E-06	4.83E-05	0.000229	1.68E-05	1.86E-05	2.07E-06	8.94E-08	6.39E-06	4.03E-05
L2 Rate	0.000296	0.000402	0.00072	0.000888	0.000156	0.000925	0.000656	0.000843	0.000958	0.000603	0.000189	0.000108	0.000133	0.000768

Table 7 (continued)
Optimal Hyperparameters: Neural Networks

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton	Gold	Silver	Platinum	Orange Juice	Lumber	Live Cattle
Panel E	**LSTM, Full Set of Variables**													
Batch Size	1	1	4	1	1	7	1	1	2	1	1	2	7	1
Epochs	3	3	71	7	83	103	3	7	91	3	103	11	103	99
Dropout Rate	0.5	0	0.5	0.5	0	0	0	0.5	0	0.25	0.5	0.5	0.5	0
Learning Rate	0.043119	0.005572	0.006083	0.000503	0.016084	0.028182	0.004613	0.002188	0.002704	0.01497	0.018451	0.002798	0.000656	0.003354
Layers	2	2	2	4	2	4	2	2	2	2	2	2	2	2
Units (1)	3	3	3	3	3	3	3	3	6	3	3	3	12	3
Units (2)	6	12	3	9	3	3	9	3	33	30	3	33	3	3
Units (3)	-	-	-	3	-	3	-	-	-	-	-	-	-	-
Units (4)	-	-	-	18	-	3	-	-	-	-	-	-	-	-
Units (5)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
L1 Rate	2.17E-05	0.000194	0.000195	0.000863	3.01E-05	2.4E-05	0.000102	0.00028	1.39E-05	3.21E-05	6.49E-05	0.000157	0.000444	2.9E-05
L2 Rate	0.000764	0.000766	0.000469	0.000818	0.000643	0.00053	0.000955	0.000223	4.36E-05	0.000895	0.000577	0.000238	0.000384	0.000225
Panel F	**LSTM, 5 DTR Features**													
Batch Size	2	2	7	2	6	7	7	7	7	1	5	6	7	3
Epochs	99	19	27	75	39	67	83	7	11	3	15	87	75	99
Dropout Rate	0	0.25	0.5	0	0.5	0.5	0.5	0.5	0.25	0.25	0.5	0.5	0.5	0.25
Learning Rate	0.004456	0.021682	0.019293	0.002263	0.022597	0.022494	0.092257	0.028752	0.047458	0.014502	0.006075	0.011711	0.002071	0.005718
Layers	3	1	1	1	1	1	1	1	1	1	1	1	1	1
Units (1)	6	6	3	3	6	3	15	3	3	3	3	9	6	15
Units (2)	21	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (3)	3	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (4)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (5)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
L1 Rate	4.66E-05	1.04E-05	1.03E-05	0.000682	3.09E-05	4.87E-05	1.13E-05	3.62E-05	3.41E-05	2.34E-05	0.000342	3.6E-05	7.77E-05	2.46E-05
L2 Rate	0.000787	0.000544	0.000814	0.000726	0.000923	0.000461	0.000415	0.000713	0.000406	0.000844	0.000202	0.000195	0.000739	0.000507

Table 7 (continued)

Optimal Hyperparameters: Neural Networks

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton	Gold	Silver	Platinum	Orange Juice	Lumber	Live Cattle
Panel G	**LSTM, 5 Principal Components**													
Batch Size	6	3	1	7	6	1	1	6	6	1	7	6	1	2
Epochs	103	99	19	99	67	47	103	103	87	103	19	47	103	63
Dropout Rate	0.25	0.5	0.5	0	0.5	0	0	0	0	0.25	0	0.5	0.5	0.5
Learning Rate	0.011969	0.040869	0.041703	0.044532	0.09315	0.011587	0.004446	0.095951	0.008284	0.09862	0.053503	0.019933	0.026645	0.001143
Layers	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Units (1)	15	9	3	3	33	3	3	33	12	3	3	3	3	24
Units (2)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (3)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (4)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (5)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
L1 Rate	0.000132	0.000013	2.13E-05	0.000276	3.02E-07	0.000494	0.000462	1.87E-06	5.45E-05	6.06E-06	0.000116	1.36E-05	4.64E-05	9.23E-05
L2 Rate	0.000252	0.000286	1.24E-07	0.000812	0.000105	0.000495	0.000849	0.000226	0.000719	0.000995	1.06E-05	0.00063	0.000146	6.46E-05
Panel H	**LSTM, 10 Principal Components**													
Batch Size	1	2	7	6	2	1	1	3	3	1	1	1	4	7
Epochs	3	31	91	3	47	103	11	91	67	87	59	103	3	103
Dropout Rate	0	0	0.5	0	0	0	0	0.5	0	0	0	0.5	0.5	0
Learning Rate	0.023557	0.002516	0.014143	0.042705	0.00036	0.057369	0.018593	0.0097	0.00442	0.008642	0.012524	0.04099	0.010541	0.007191
Layers	3	5	2	2	2	2	2	2	2	2	2	2	2	5
Units (1)	3	3	3	3	3	3	3	9	3	3	3	3	3	3
Units (2)	3	3	12	3	12	3	12	3	6	3	3	3	3	6
Units (3)	3	3	-	-	-	-	-	-	-	-	-	-	-	3
Units (4)	-	3	-	-	-	-	-	-	-	-	-	-	-	6
Units (5)	-	18	-	-	-	-	-	-	-	-	-	-	-	3
L1 Rate	0.000107	0.000185	1.32E-05	1.99E-05	0.000875	5.88E-06	6.64E-06	1.49E-05	0.000282	1.36E-05	4.64E-05	3.46E-05	2.39E-05	1.47E-05
L2 Rate	1.24E-07	0.000812	0.000706	0.000495	0.000984	0.000198	8.56E-05	0.000259	7.73E-05	0.00063	0.000989	0.000339	0.000685	0.000351

Table 8

Root Mean Square Error Out of Sample Forecast Performance (RMSFE)

This table reports the out of sample RMSFE of the analyzed models against the AR(1) benchmark. The OOS forecasts cover the period from September 30, 2012 to June 30, 2020.

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton	Gold	Silver	Platinum	Orange Juice	Lumber	Live Cattle
Panel A							**Benchmark**							
AR(1)	**0.1157**	0.0627	0.0600	0.0765	0.0838	**0.0675**	0.0699	0.0634	0.0420	0.0734	0.0604	**0.0854**	0.0868	0.0559
Panel B							**Tree-Based Methods**							
Random Forests (Full)	0.1167	0.0631	0.0595	**0.0757**	0.0837	0.0689	**0.0679**	0.0627	0.0413	0.0734	0.0622	0.0866	0.0879	0.0563
Random Forests (5 PC)	0.1177	0.0630	0.0600	0.0767	0.0843	0.0691	0.0684	0.0633	0.0416	0.0741	0.0612	0.0869	0.0900	0.0557
Random Forests (10 PC)	0.1170	0.0634	0.0591	0.0768	0.0838	0.0695	0.0699	0.0630	0.0421	0.0732	0.0609	0.0871	0.0877	0.0554
AdaBoost (Full)	0.1173	0.0628	0.0620	0.0775	0.0908	0.0677	0.0690	0.0627	**0.0408**	0.0731	0.0614	0.0900	0.0892	0.0559
AdaBoost (5 PC)	0.1174	0.0638	0.0620	0.0775	0.0943	0.0695	0.0695	0.0632	0.0419	0.0748	0.0603	0.0892	0.0910	0.0552
AdaBoost (10 PC)	0.1171	0.0660	0.0616	0.0778	0.0880	0.0694	0.0696	0.0627	0.0410	0.0754	0.0608	0.0890	0.0882	**0.0548**
XGBoost (Full)	0.1193	0.0639	0.0604	0.0766	0.0859	0.0696	0.0723	0.0635	0.0412	0.0741	0.0631	0.0864	0.0887	0.0572
XGBoost (5 PC)	0.1168	0.0655	0.0588	0.0773	0.0846	0.0682	0.0697	0.0633	0.0413	0.0733	0.0619	0.0876	0.0878	0.0555
XGBoost (10 PC)	0.1170	0.0629	0.0601	0.0770	0.0837	0.0691	0.0692	0.0633	0.0414	0.0731	0.0608	0.0866	0.0864	0.0560
Panel C							**Neural Networks**							
Feed-Forward NN (Full)	0.1165	0.0624	0.0594	0.0771	0.0842	0.0691	0.0689	**0.0616**	0.0413	0.0732	0.0607	0.0869	0.0866	0.0555
Feed-Forward NN (5 PC)	0.1178	**0.0609**	0.0629	0.0765	0.0867	0.0724	0.0689	0.0632	0.0423	0.0721	0.0675	0.0882	0.0887	0.0557
Feed-Forward NN (10 PC)	0.1173	0.0621	0.0591	0.0771	0.0866	0.0705	0.0687	0.0632	0.0412	0.0790	0.0610	0.0867	0.0883	0.0555
Feed-Forward NN (5 Var.)	0.1163	0.0621	**0.0581**	0.0813	0.0839	0.0695	0.0689	0.0634	0.0430	0.0729	0.0619	0.0900	0.0868	0.0563
LSTM (Full)	0.1164	0.0621	0.0608	0.0769	0.0835	0.0689	0.0691	0.0632	0.0412	0.0721	**0.0602**	0.0867	0.0864	0.0556
LSTM (5PC)	0.1171	0.0650	0.0591	0.0804	0.0837	0.0690	0.0687	0.0647	0.0422	**0.0720**	0.0611	0.0872	**0.0863**	0.0555
LSTM (10PC)	0.1164	0.0639	0.0609	0.0769	0.0837	0.0784	0.0696	0.0632	0.0412	0.0729	0.0622	0.0873	0.0875	0.0559
LSTM (5 Variables)	0.1181	0.0625	0.0643	0.0777	**0.0833**	0.0695	0.0729	0.0634	0.0412	0.0721	0.0604	0.0869	0.0865	0.0555

Table 9

Mean Absolute Error Out of Sample Forecast Performance (MAFE)

This table reports the out of sample MAFE of the analyzed models against the AR(1) benchmark. The OOS forecasts cover the period from September 30, 2012 to June 30, 2020.

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton	Gold	Silver	Platinum	Frozen Orange Juice	Lumber	Live Cattle
Benchmark														
AR(1)	**0.0770**	0.0479	0.0463	0.0595	0.0612	**0.0542**	0.0542	0.0497	0.0333	0.0574	0.0494	**0.0661**	0.0677	0.0424
Tree-Based Methods														
Random Forests (Full)	0.0784	0.0486	0.0465	**0.0581**	0.0616	0.0559	0.0535	0.0493	0.0330	0.0582	0.0511	0.0665	0.0687	0.0432
Random Forests (5PC)	0.0787	0.0482	0.0473	0.0585	0.0622	0.0559	0.0530	0.0496	0.0332	0.0586	0.0501	0.0667	0.0690	0.0424
Random Forests (10PC)	0.0786	0.0480	0.0461	0.0592	0.0613	0.0560	0.0544	0.0491	0.0338	0.0574	0.0498	0.0671	0.0685	0.0423
AdaBoost (Full)	0.0784	0.0477	0.0485	0.0593	0.0713	0.0557	0.0535	0.0488	0.0324	0.0578	0.0504	0.0693	0.0699	0.0428
AdaBoost (5PC)	0.0788	0.0491	0.0490	0.0596	0.0744	0.0565	0.0536	0.0496	0.0333	0.0593	0.0495	0.0691	0.0699	0.0420
AdaBoost (10PC)	0.0787	0.0508	0.0491	0.0600	0.0645	0.0561	0.0542	0.0485	**0.0324**	0.0603	**0.0490**	0.0680	0.0685	**0.0415**
XGBoost (Full)	0.0793	0.0492	0.0473	0.0586	0.0659	0.0574	0.0579	**0.0484**	0.0328	0.0592	0.0500	0.0677	0.0699	0.0437
XGBoost (5PC)	0.0782	0.0497	0.0462	0.0597	0.0621	0.0550	0.0543	0.0496	0.0327	0.0572	0.0507	0.0670	0.0682	0.0424
XGBoost (10PC)	0.0786	0.0480	0.0465	0.0593	0.0612	0.0557	0.0537	0.0497	0.0328	0.0573	0.0498	0.0667	0.0675	0.0428
Neural Networks														
Feed-Forward NN (Full)	0.0783	0.0476	0.0458	0.0594	0.0623	0.0557	0.0531	0.0489	0.0327	0.0568	0.0497	0.0671	0.0692	0.0422
Feed-Forward NN (5PC)	0.0792	**0.0471**	0.0492	0.0589	0.0648	0.0580	0.0532	0.0497	0.0334	0.0556	0.0564	0.0669	0.0691	0.0427
Feed-Forward NN (10PC)	0.0789	0.0473	0.0455	0.0594	0.0662	0.0564	0.0525	0.0495	0.0325	0.0649	0.0500	0.0667	0.0695	0.0423
Feed-Forward NN (5 Var.)	0.0784	0.0473	**0.0446**	0.0629	**0.0587**	0.0572	0.0524	0.0491	0.0336	0.0566	0.0507	0.0694	0.0681	0.0426
LSTM (Full)	0.0781	0.0473	0.0472	0.0593	0.0607	0.0558	0.0534	0.0499	0.0327	**0.0551**	0.0491	0.0668	0.0675	0.0423
LSTM (5PC)	0.0781	0.0503	0.0455	0.0621	0.0610	0.0562	**0.0521**	0.0502	0.0338	0.0552	0.0501	0.0669	**0.0672**	0.0424
LSTM (10PC)	0.0781	0.0491	0.0472	0.0593	0.0611	0.0633	0.0543	0.0498	0.0324	0.0571	0.0512	0.0672	0.0687	0.0424
LSTM (5 Variables)	0.0793	0.0477	0.0505	0.0599	0.0595	0.0558	0.0583	0.0497	0.0324	0.0551	0.0494	0.0671	0.0676	0.0424

Table 10

Comparison of Performance of the ML Portfolio Against the Benchmark, by Month

Date	Return (ML)	Return (Benchmark)	Portfolio Value (ML)	Portfolio Value (Benchmark)	Cumulative Return (ML)	Cumulative Return (Benchmark)
9/30/2012	-0.2424%	-0.0686%	997.58	999.31	-0.2424%	-0.0686%
10/31/2012	1.4492%	-0.9858%	1012.03	989.46	1.2032%	-1.0537%
11/30/2012	0.6804%	-0.4484%	1018.92	985.03	1.8918%	-1.4974%
12/31/2012	2.7448%	0.3342%	1046.89	988.32	4.6885%	-1.1683%
1/31/2013	-1.0313%	0.5914%	1036.09	994.16	3.6088%	-0.5837%
2/28/2013	1.0074%	-0.2015%	1046.53	992.16	4.6526%	-0.7840%
3/31/2013	-0.1302%	-0.8006%	1045.16	984.22	4.5163%	-1.5783%
4/30/2013	1.4535%	-1.1489%	1060.35	972.91	6.0354%	-2.7091%
5/31/2013	-0.0989%	-1.8203%	1059.31	955.20	5.9306%	-4.4801%
6/30/2013	2.0270%	-1.1103%	1080.78	944.59	8.0778%	-5.5406%
7/31/2013	3.6344%	2.0107%	1120.06	963.59	12.0058%	-3.6413%
8/31/2013	-2.5600%	0.8851%	1091.38	972.12	9.1385%	-2.7884%
9/30/2013	1.6073%	-0.4240%	1108.93	967.99	10.8927%	-3.2006%
10/31/2013	0.6029%	0.1281%	1115.61	969.23	11.5612%	-3.0766%
11/30/2013	2.2535%	0.5035%	1140.75	974.11	14.0753%	-2.5885%
12/31/2013	-0.9407%	-0.7192%	1130.02	967.11	13.0022%	-3.2891%
1/31/2014	-0.6943%	0.0387%	1122.18	967.48	12.2177%	-3.2517%
2/28/2014	-2.4536%	-2.3957%	1094.64	944.31	9.4643%	-5.5695%
3/31/2014	0.0661%	-0.9805%	1095.37	935.05	9.5366%	-6.4953%
4/30/2014	-1.8513%	-0.6213%	1075.09	929.24	7.5087%	-7.0762%
5/31/2014	2.0445%	-0.2323%	1097.07	927.08	9.7067%	-7.2921%
6/30/2014	1.3607%	-0.2561%	1112.00	924.71	11.1995%	-7.5295%
7/31/2014	2.7008%	-0.1546%	1142.03	923.28	14.2028%	-7.6724%
8/31/2014	1.1303%	-0.2921%	1154.94	920.58	15.4937%	-7.9421%
9/30/2014	1.8744%	-1.6576%	1176.59	905.32	17.6585%	-9.4680%
10/31/2014	-1.4300%	1.1241%	1159.76	915.50	15.9760%	-8.4503%
11/30/2014	1.9619%	1.0492%	1182.51	925.10	18.2514%	-7.4897%
12/31/2014	1.7603%	1.7572%	1203.33	941.36	20.3329%	-5.8641%
1/31/2015	-0.1596%	0.9726%	1201.41	950.51	20.1408%	-4.9486%
2/28/2015	0.8886%	-0.5069%	1212.08	945.70	21.2084%	-5.4304%
3/31/2015	0.9141%	1.3278%	1223.16	958.25	22.3164%	-4.1748%
4/30/2015	-1.1885%	-2.4559%	1208.63	934.72	20.8627%	-6.5282%
5/31/2015	0.3161%	-0.5053%	1212.45	930.00	21.2447%	-7.0004%
6/30/2015	1.1512%	-1.1275%	1226.41	919.51	22.6405%	-8.0490%
7/31/2015	-2.3334%	-0.1162%	1197.79	918.44	19.7789%	-8.1559%
8/31/2015	-1.1302%	-0.9961%	1184.25	909.29	18.4251%	-9.0707%
9/30/2015	0.1759%	-0.7801%	1186.33	902.20	18.6334%	-9.7801%
10/31/2015	2.4092%	2.2020%	1214.91	922.07	21.4915%	-7.7934%
11/30/2015	-0.4394%	-1.8630%	1209.58	904.89	20.9577%	-9.5112%
12/31/2015	0.9369%	0.7651%	1220.91	911.81	22.0910%	-8.8188%
1/31/2016	0.0173%	-0.7056%	1221.12	905.38	22.1121%	-9.4622%
2/29/2016	-0.6078%	-1.1424%	1213.70	895.04	21.3699%	-10.4965%
3/31/2016	1.4204%	2.9370%	1230.94	921.32	23.0938%	-7.8678%
4/30/2016	2.8583%	3.9320%	1266.12	957.55	26.6122%	-4.2451%
5/31/2016	3.3834%	3.8695%	1308.96	994.60	30.8959%	-0.5399%
6/30/2016	-5.2088%	0.8554%	1240.78	1003.11	24.0778%	0.3109%
7/31/2016	-1.2343%	-1.9381%	1225.46	983.67	22.5463%	-1.6333%
8/31/2016	0.7401%	-0.5665%	1234.53	978.10	23.4533%	-2.1905%
9/30/2016	1.7098%	2.7087%	1255.64	1004.59	25.5641%	0.4588%
10/31/2016	-0.8309%	-2.1285%	1245.21	983.21	24.5208%	-1.6794%
11/30/2016	1.8476%	-0.3979%	1268.22	979.29	26.8215%	-2.0707%
12/31/2016	0.7000%	-0.5331%	1277.09	974.07	27.7092%	-2.5928%
1/31/2017	-2.4172%	-0.3866%	1246.22	970.31	24.6222%	-2.9694%
2/28/2017	0.2981%	-1.4613%	1249.94	956.13	24.9938%	-4.3873%

Table 10 (continued)

Comparison of Performance of the ML Portfolio Against the Benchmark, by Month

Date	Return (ML)	Return (Benchmark)	Portfolio Value (ML)	Portfolio Value (Benchmark)	Cumulative Return (ML)	Cumulative Return (Benchmark)
3/31/2017	1.0799%	0.6969%	1263.44	962.79	26.3435%	-3.7209%
4/30/2017	2.0644%	-0.0334%	1289.52	962.47	28.9518%	-3.7530%
5/31/2017	0.1057%	1.1242%	1290.88	973.29	29.0881%	-2.6710%
6/30/2017	2.7981%	0.6783%	1327.00	979.89	32.7001%	-2.0109%
7/31/2017	-1.2715%	0.5629%	1310.13	985.41	31.0128%	-1.4593%
8/31/2017	1.6063%	-0.6851%	1331.17	978.66	33.1173%	-2.1344%
9/30/2017	1.9932%	0.4111%	1357.71	982.68	35.7705%	-1.7321%
10/31/2017	1.1248%	-0.0119%	1372.98	982.56	37.2977%	-1.7438%
11/30/2017	0.0475%	0.3056%	1373.63	985.57	37.3630%	-1.4435%
12/31/2017	0.0685%	0.5427%	1374.57	990.91	37.4570%	-0.9086%
1/31/2018	2.2251%	1.3951%	1405.16	1004.74	40.5156%	0.4738%
2/28/2018	0.8375%	-1.3307%	1416.92	991.37	41.6924%	-0.8632%
3/31/2018	-3.1069%	-1.4588%	1372.90	976.91	37.2901%	-2.3093%
4/30/2018	1.4462%	0.3672%	1392.76	980.49	39.2757%	-1.9506%
5/31/2018	0.9880%	-0.8704%	1406.52	971.96	40.6517%	-2.8040%
6/30/2018	0.1567%	-0.2555%	1408.72	969.48	40.8720%	-3.0524%
7/31/2018	-0.4700%	-0.7397%	1402.10	962.31	40.2100%	-3.7695%
8/31/2018	1.4146%	-0.3750%	1421.93	958.70	42.1933%	-4.1304%
9/30/2018	0.9576%	0.7151%	1435.55	965.55	43.5550%	-3.4448%
10/31/2018	0.6266%	-2.8508%	1444.54	938.03	44.4545%	-6.1974%
11/30/2018	2.3776%	1.7610%	1478.89	954.54	47.8890%	-4.5455%
12/31/2018	1.1481%	2.5025%	1495.87	978.43	49.5868%	-2.1568%
1/31/2019	0.6958%	-0.9453%	1506.28	969.18	50.6276%	-3.0817%
2/28/2019	0.9289%	0.3883%	1520.27	972.95	52.0269%	-2.7054%
3/31/2019	-0.0405%	0.0735%	1519.65	973.66	51.9653%	-2.6339%
4/30/2019	0.1090%	-0.0865%	1521.31	972.82	52.1309%	-2.7181%
5/31/2019	0.3744%	0.8702%	1527.01	981.28	52.7005%	-1.8716%
6/30/2019	1.3567%	1.4065%	1547.72	995.09	54.7722%	-0.4914%
7/31/2019	-1.0819%	0.1600%	1530.98	996.68	53.0977%	-0.3322%
8/31/2019	-0.3191%	-0.9483%	1526.09	987.23	52.6092%	-1.2773%
9/30/2019	1.6885%	3.1565%	1551.86	1018.39	55.1860%	1.8389%
10/31/2019	0.7455%	0.8812%	1563.43	1027.36	56.3429%	2.7363%
11/30/2019	0.3934%	1.2190%	1569.58	1039.89	56.9579%	3.9887%
12/31/2019	0.2205%	0.6995%	1573.04	1047.16	57.3040%	4.7161%
1/31/2020	0.4824%	0.3764%	1580.63	1051.10	58.0627%	5.1103%
2/29/2020	-0.5032%	0.6095%	1572.67	1057.51	57.2673%	5.7510%
3/31/2020	3.4015%	1.5500%	1626.17	1073.90	62.6168%	7.3901%
4/30/2020	3.7081%	4.0529%	1686.47	1117.43	68.6468%	11.7426%
5/31/2020	-4.4465%	-5.8269%	1611.48	1052.31	61.1480%	5.2314%
6/30/2020	1.5236%	1.2683%	1636.03	1065.66	63.6033%	6.5661%

Table 11
Optimal Hyperparameters: Neural Networks, Classification Problem

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton	Gold	Silver	Platinum	Orange Juice	Lumber	Live Cattle
Panel A	**Feedforward Neural Networks, Full Set of Features**													
Batch Size	1	1	3	7	1	7	1	4	1	1	1	1	1	1
Epochs	99	103	103	103	103	103	103	103	103	99	103	51	103	47
Drop Rate	0	0.5	0.5	0.5	0	0.5	0.5	0	0.25	0.5	0.5	0	0.5	0
Learning Rate	0.0814	0.0603	0.0508	0.0587	0.0419	0.0417	0.0803	0.0594	0.0587	0.0419	0.0339	0.0355	0.0462	0.0802
Layers	5	1	1	1	5	1	1	5	5	5	1	5	1	5
Units (1)	3	33	3	3	3	3	3	3	6	3	33	3	3	3
Units (2)	3	-	-	-	3	-	-	3	12	18	-	6	-	3
Units (3)	12	-	-	-	3	-	-	3	3	27	-	3	-	3
Units (4)	3	-	-	-	3	-	-	3	3	3	-	3	-	3
Units (5)	30	-	-	-	3	-	-	3	3	33	-	18	-	12
L1 Rate	1.4E-05	0.000996	9.18E-05	0.000865	0.000446	0.000968	0.000739	0.000311	0.000166	6.86E-05	0.000155	5.6E-05	4.14E-06	0.000104
L2 Rate	1.24E-07	0.000116	0.000437	0.000378	0.000693	1.24E-07	0.000116	0.000437	0.000561	0.00076	0.000722	0.000708	0.000896	0.000288
Panel B	**Feedforward Neural Networks, 5 Principal Components**													
Batch Size	1	1	7	4	1	7	1	2	1	7	7	7	7	7
Epochs	39	99	99	103	103	103	103	103	103	103	103	103	103	103
Drop Rate	0.5	0	0	0	0.5	0.5	0	0.25	0.25	0.25	0.5	0.5	0	0
Learning Rate	0.007202	0.025242	0.058352	0.032399	0.023809	0.060678	0.019776	0.023165	0.079134	0.059837	0.026645	0.032043	0.094235	0.009855
Layers	1	1	1	1	1	1	1	1	5	1	1	1	1	1
Units (1)	9	33	12	6	33	3	3	33	3	3	3	33	3	33
Units (2)	-	-	-	-	-	-	-	-	3	-	-	-	-	-
Units (3)	-	-	-	-	-	-	-	-	3	-	-	-	-	-
Units (4)	-	-	-	-	-	-	-	-	6	-	-	-	-	-
Units (5)	-	-	-	-	-	-	-	-	3	-	-	-	-	-
L1 Rate	0.000175	0.000477	0.000894	0.000465	0.000504	0.000699	0.000443	7.34E-05	1.24E-05	1.36E-05	4.64E-05	0.000363	1.89E-05	0.000136
L2 Rate	0.000874	0.000812	0.000706	0.000495	0.000258	0.000474	0.000689	0.000616	0.000736	0.000357	0.000146	5.51E-05	0.000334	1.24E-07

Table 11 (continued)
Optimal Hyperparameters: Neural Networks, Classification Problem

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton	Gold	Silver	Platinum	Orange Juice	Lumber	Live Cattle
Panel C	**Feedforward Neural Networks, 10 Principal Components**													
Batch Size	5	7	5	7	7	6	1	7	1	1	7	1	7	1
Epochs	103	103	99	103	103	75	103	103	103	27	103	71	87	103
Drop Rate	0	0.5	0	0	0	0	0	0	0.25	0	0.5	0	0	0.5
Learning Rate	0.067838	0.022287	0.027246	0.043727	0.091023	0.033562	0.035775	0.08433	0.056282	0.08612	0.099373	0.033059	0.085143	0.065114
Layers	2	2	2	2	2	2	2	2	2	2	2	5	2	5
Units (1)	3	3	3	3	3	6	3	3	3	3	3	6	3	3
Units (2)	9	18	6	3	3	24	3	3	3	3	33	3	3	3
Units (3)	-	-	-	-	-	-	-	-	-	-	-	3	-	3
Units (4)	-	-	-	-	-	-	-	-	-	-	-	9	-	3
Units (5)	-	-	-	-	-	-	-	-	-	-	-	3	-	3
L1 Rate	0.00072	0.00021	0.000493	0.000863	3.01E-05	0.000273	0.000576	0.00042	0.000314	0.000199	0.000243	2.46E-05	0.000313	2.9E-05
L2 Rate	1.24E-07	0.000524	0.000469	0.00079	3.69E-05	0.000814	0.000813	0.000692	0.000625	0.000634	0.000888	0.000426	0.000232	0.000652
Panel D	**Feedforward Neural Networks, 5 DTR Features**													
Batch Size	2	6	1	7	1	7	6	7	1	1	1	3	5	3
Epochs	39	99	99	91	83	51	103	35	103	103	103	91	103	95
Drop Rate	0	0	0	0	0	0	0	0.5	0	0	0	0.5	0	0
Learning Rate	0.007198	0.007056	0.074026	0.008354	0.066448	0.009882	0.002582	0.037287	0.058316	0.007174	0.008146	0.05308	0.018667	0.00206
Layers	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Units (1)	9	9	9	21	3	24	33	6	3	3	6	6	6	12
Units (2)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (3)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (4)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (5)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
L1 Rate	0.000751	0.000185	0.000311	0.000259	0.000281	0.000733	0.00024	0.000484	0.000254	0.000611	0.00018	0.000267	0.000811	9.75E-05
L2 Rate	0.000355	0.000808	0.000395	0.000846	2.78E-05	7.25E-07	0.000135	0.00097	0.000161	0.00066	0.000812	0.000807	0.000836	0.000907

Table 11 (continued)
Optimal Hyperparameters: Neural Networks, Classification Problem

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton	Gold	Silver	Platinum	Orange Juice	Lumber	Live Cattle
Panel E	**LSTM, Full Set of Features**													
Batch Size	1	1	4	1	1	7	1	1	2	1	1	2	7	1
Epochs	3	3	71	7	83	103	3	7	91	3	103	11	103	99
Drop Rate	0.5	0	0.5	0.5	0	0	0	0.5	0	0.25	0.5	0.5	0.5	0
Learning Rate	0.043119	0.005572	0.006083	0.000503	0.016084	0.028182	0.004613	0.002188	0.002704	0.01497	0.018451	0.002798	0.000656	0.003354
Layers	2	2	2	4	2	4	2	2	2	2	2	2	2	2
Units (1)	3	3	3	3	3	3	3	3	6	3	3	3	12	3
Units (2)	6	12	3	9	3	3	9	3	33	30	3	33	3	3
Units (3)	-	-	-	3	-	3	-	-	-	-	-	-	-	-
Units (4)	-	-	-	18	-	3	-	-	-	-	-	-	-	-
Units (5)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
L1 Rate	2.17E-05	0.000194	0.000195	0.000863	3.01E-05	2.4E-05	0.000102	0.00028	1.39E-05	3.21E-05	6.49E-05	0.000157	0.000444	2.9E-05
L2 Rate	0.000764	0.000766	0.000469	0.000818	0.000643	0.00053	0.000955	0.000223	4.36E-05	0.000895	0.000577	0.000238	0.000384	0.000225
Panel F	**LSTM, 5 Principal Components**													
Batch Size	7	1	7	7	1	5	1	1	1	1	1	3	1	7
Epochs	47	3	103	103	103	35	103	103	103	103	103	19	103	39
Drop Rate	0.5	0	0	0	0.5	0.5	0.5	0	0	0	0.5	0	0.5	0.5
Learning Rate	0.060144	0.022287	0.049586	0.043727	0.02688	0.04882	0.027211	0.084637	0.017472	0.067778	0.010384	0.051351	0.012481	0.065114
Layers	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Units (1)	9	3	33	3	3	3	3	3	18	18	3	3	33	3
Units (2)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (3)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (4)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (5)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
L1 Rate	0.000691	0.000194	0.000508	0.000113	3.01E-05	0.000536	0.000576	0.000186	0.000394	0.000119	0.000275	1.66E-05	0.000444	2.9E-05
L2 Rate	1.24E-07	0.000562	0.000386	0.000956	0.000643	0.000674	0.000392	0.000877	0.00026	0.000904	0.000888	2.75E-05	0.000629	0.000103

Table 11 (continued)
Optimal Hyperparameters: Neural Networks, Classification Problem

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton	Gold	Silver	Platinum	Orange Juice	Lumber	Live Cattle
Panel G	**LSTM, 10 Principal Components**													
Batch Size	6	5	6	6	3	2	7	7	7	1	7	1	7	1
Epochs	11	103	71	103	99	27	103	71	103	103	7	103	103	83
Drop Rate	0	0.5	0	0.5	0	0.5	0.5	0.5	0	0.5	0.5	0.5	0.25	0.5
Learning Rate	0.008897	0.000681	0.00861	0.000226	0.02688	0.06871	0.014154	0.008891	0.031355	0.068276	0.083542	0.006701	0.043727	0.001758
Layers	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Units (1)	30	30	9	3	3	24	33	30	33	33	27	33	33	30
Units (2)	6	3	30	9	33	33	33	33	3	33	33	33	33	24
Units (3)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (4)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Units (5)	-	-	-	-	-	-	-	-	-	-	-	-	-	-
L1 Rate	2.66E-05	0.000194	0.000126	0.000751	3.01E-05	0.000629	0.000315	0.000436	0.000314	0.00072	0.000414	0.000508	0.000889	0.000305
L2 Rate	0.000188	0.000766	0.000643	0.00068	0.000521	0.000373	0.000576	0.000223	0.00026	1.24E-07	0.00059	0.000469	0.000251	3.69E-05
Panel H	**LSTM, 5 DTR Features**													
Batch Size	3	3	4	4	3	1	6	3	1	7	2	7	7	6
Epochs	11	87	95	91	39	3	51	87	3	103	55	103	103	99
Drop Rate	0.5	0	0.5	0	0.25	0.25	0.5	0	0.5	0.5	0	0	0.5	0
Learning Rate	0.009637	0.02016	0.005664	0.009733	0.043727	0.065647	0.003848	0.011836	0.095114	0.012857	0.009294	0.010384	0.015412	0.033473
Layers	1	1	1	1	1	4	1	1	5	1	1	1	1	1
Units (1)	30	3	9	6	24	15	21	9	9	21	21	3	33	6
Units (2)	-	-	-	-	-	3	-	-	3	-	-	-	-	-
Units (3)	-	-	-	-	-	3	-	-	3	-	-	-	-	-
Units (4)	-	-	-	-	-	3	-	-	3	-	-	-	-	-
Units (5)	-	-	-	-	-	-	-	-	3	-	-	-	-	-
L1 Rate	0.000507	0.000231	0.000438	0.000443	0.000863	0.000189	0.000949	0.000828	0.000839	0.000394	0.000421	0.000275	0.00075	0.000149
L2 Rate	0.000143	0.000401	3.76E-05	0.000469	0.000994	0.000645	0.000869	0.000625	0.000475	0.000625	0.000327	0.000577	0.000723	0.000719

Table 12
Out of Sample Forecast Accuracy (Classification)

This table reports the out of sample forecast accuracy of the analyzed neural networks against the four logit benchmark models. The OOS forecasts cover the period from September 30, 2012 to June 30, 2020.

	Light Crude Oil	Corn	Soybeans	Wheat	Coffee	Cocoa	Sugar	Cotton	Gold	Silver	Platinum	Orange Juice	Lumber	Live Cattle
Panel A							Benchmark Logit Models							
Logit (Full)	37.23%	48.94%	39.36%	**60.64%**	56.38%	52.13%	48.94%	**58.51%**	45.74%	36.17%	44.68%	50.00%	44.68%	46.81%
Logit (5PC)	46.81%	47.87%	40.43%	54.26%	53.19%	47.87%	42.55%	50.00%	51.06%	39.36%	48.94%	46.81%	39.36%	47.87%
Logit (10PC)	**53.19%**	42.55%	47.87%	51.06%	48.94%	54.26%	55.32%	57.45%	50.00%	37.23%	41.49%	46.81%	43.62%	48.94%
Logit (5 Var.)	45.74%	52.13%	46.81%	51.06%	54.26%	**57.45%**	54.26%	53.19%	38.30%	45.74%	44.68%	**52.13%**	40.43%	45.74%
Panel B							Neural Networks							
Feedforward NN (Full)	48.94%	48.94%	45.74%	52.13%	**58.51%**	47.87%	**58.51%**	46.81%	50.00%	**55.32%**	44.68%	48.94%	**50.00%**	51.06%
Feedforward NN (5PC)	50.00%	**56.38%**	46.81%	42.55%	39.36%	48.94%	40.43%	47.87%	50.00%	40.43%	44.68%	42.55%	44.68%	**53.19%**
Feedforward NN (10PC)	50.00%	50.00%	51.06%	56.38%	40.43%	51.06%	42.55%	53.19%	50.00%	**55.32%**	44.68%	48.94%	45.74%	51.06%
Feedforward NN (5 Var.)	48.94%	46.81%	**54.26%**	48.94%	**58.51%**	51.06%	57.45%	53.19%	50.00%	54.26%	44.68%	48.94%	44.68%	48.94%
LSTM (Full)	48.94%	43.62%	53.19%	52.13%	40.43%	45.74%	43.62%	51.06%	54.26%	51.06%	44.68%	45.74%	41.49%	47.87%
LSTM (5PC)	50.00%	55.32%	**54.26%**	51.06%	**58.51%**	45.74%	40.43%	46.81%	**55.32%**	**55.32%**	44.68%	51.06%	36.17%	51.06%
LSTM (10PC)	**53.19%**	44.68%	45.74%	52.13%	**58.51%**	45.74%	43.62%	51.06%	50.00%	**55.32%**	44.68%	48.94%	44.68%	52.13%
LSTM (5 Var.)	48.94%	43.62%	**54.26%**	52.13%	**58.51%**	45.74%	42.55%	53.19%	50.00%	**55.32%**	**55.32%**	48.94%	44.68%	48.94%

Figure 1

Simulation Exercise: Performance of Feedforward Neural Network

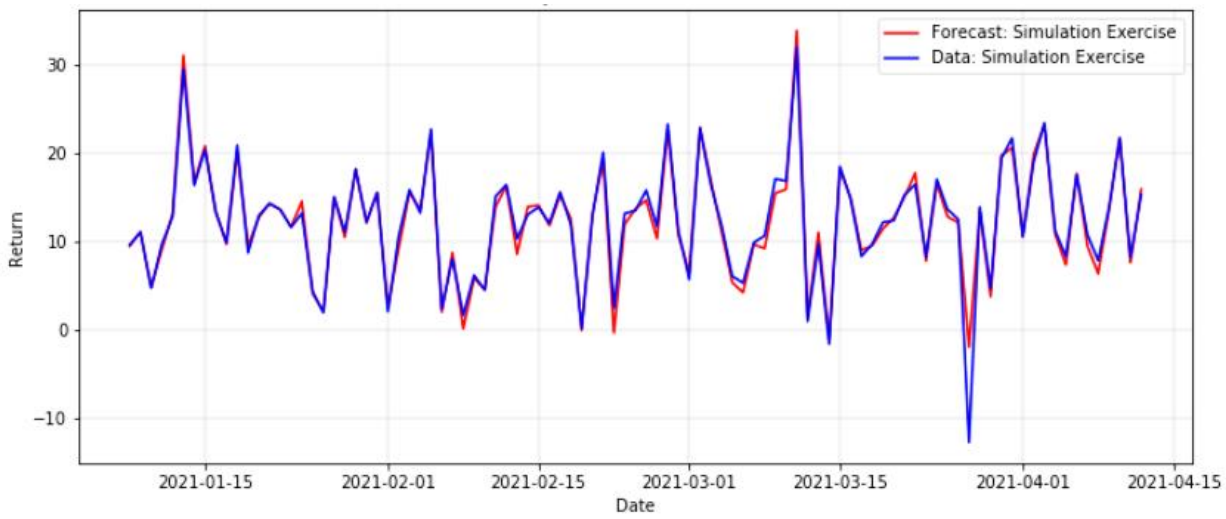

Figure 2

Simulation Exercise: Performance of Long Short-Term Memory Network

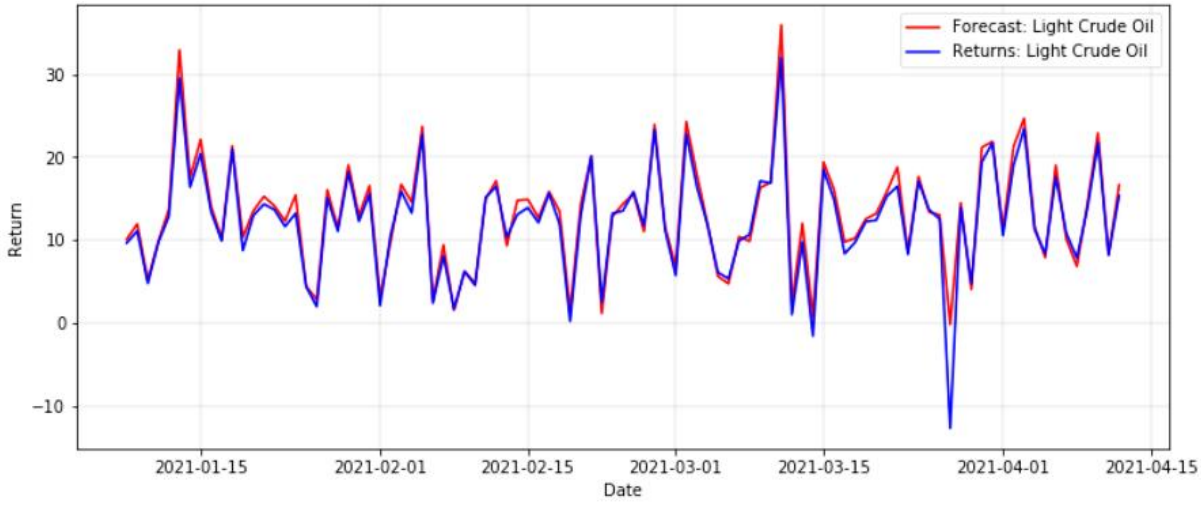

Figure 3

Out of Sample Performance: ML Portfolio and Benchmark Portfolio

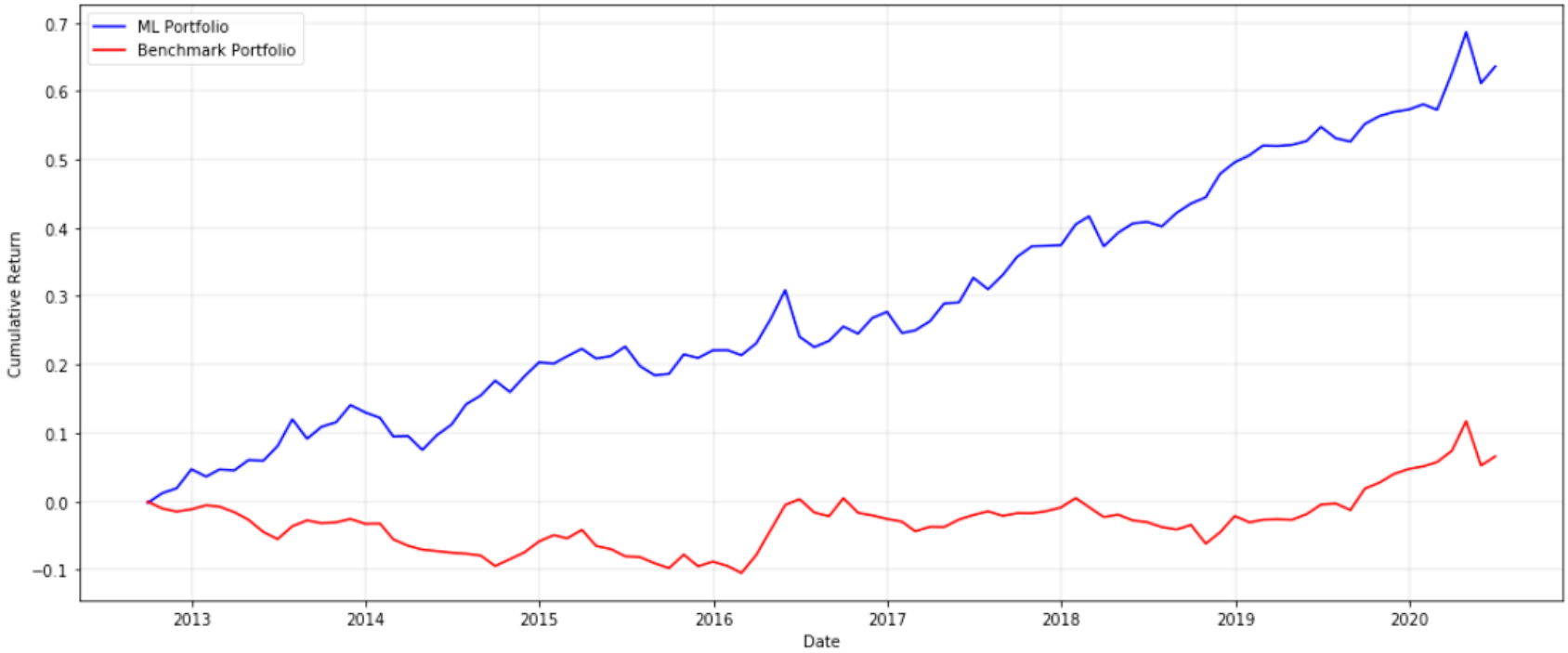

Figure 4

Out of Sample Performance: ML Portfolio and Benchmark Portfolios (Classification Problem)