Wageningen University
Agricultural Economics and Rural Policy Group

Machine Learning to predict grains future prices

Forecasting agricultural commodity prices using an LSTM Neural Network and traditional econometrics

2021

Management Economics and Consumer Studies
Economics and Governance

Master Thesis
961220-124020

Supervisors: dr.ir. C (Koos) Gardebroek
Prof. Paolo Sckokai

Wageningen University & Research

Date: January 2021

Acknowledgment

To God, who kept me and my family safe, and gifted us with the possibility of still being together after this long journey;

To my mother, who dispensed me with wise words, and provided me with the love to bear the distance;

To my father, who supported me with every mean he had, and on which I relied like a traveller on its staff;

To my brother, who proved my patience every day, strengthening my character to prepare me for the stormy days;

To my friends, who believed in me and kept close, even if I travelled far away;

To my professors, who lit my desire for knowledge and guided it over the years;

I express my sincerest gratitude, you have been the travel companions without which I could not stand where I am now.

Contents

Table of figures:

Abstract

In recent years, the doors of the social sciences have opened for the implementation of Machine Learning (ML) models. These models in fact grant the researcher an increased power to approximate the data generation process at study, and the possibility to take into consideration a larger number of variables. Given their structure optimized for prediction accuracy, Neural Networks, and in particular Long Term Short Memory (LSTM), are natural candidates to take upon the time series forecasting problem. They also have been found to outperform traditional econometrics, but only in specific settings. However, the functioning of these models in respect to traditional ones as well as in respect to time properties still remains unclear. This is particularly true when considering agricultural commodities future prices, as the implementation of Machine Learning in this sector cannot count on many studies.

This research therefore compares the functioning as well as the performances of ML models versus traditional ones. The comparison is carried out across different scenarios, so to provide a clearer picture of when the ML models are more appropriate. The different scenarios are built around different pre-processing techniques, different forecast horizons as well as different clusters of variables considered. In addition to this, LSTM's ability to pick up time elements of this dataset is assessed.

The results highlight how LSTM models outperform traditional econometrics when forecasting over a long horizon (30 days), while the opposite can be said for a short horizon (7 days). No significant differences were found across pre-processing procedures nor cluster of variables. The LSTM were found not to be able to pick up neither seasonality nor trends.

Chapter I – Introduction

1.1 *Introduction*

The tremendous increase in data availability in the agricultural and environmental sector, paired with an outstanding progress in the computational power of computers, has opened several possibilities for the application of Machine Learning (ML) techniques in agricultural economics (Storm et al., 2019). Machine Learning is a term that comprises several techniques, from decision trees to neural networks. These techniques are often better equipped than traditional statistical tools to deal with big data, and therefore can contribute significantly to the understanding of agricultural price time-series (Mullainathan and Spiess, 2017). In particular, these ML models have as a primary goal prediction accuracy, and could therefore greatly impact how forecasting is carried out in this field. To compare the performances of these new techniques to traditional econometric forecasting techniques, in recent years some researchers have competed in forecasting contests: the M-competitions series (Hyndman, 2020). The astonishing results of the last competition were that complex approaches were finally able to outperform more simple methods, in contrast with what happened in previous competitions. Such results undermine the long-standing preference for model parsimony, and, according to some authors, sounded the "death knell" for traditional econometrics models, such as the Autoregressive Integrated Moving Average (Gilland, 2020).

Nevertheless, one of the main unresolved issues pointed out by recent studies is the functioning of ML models as "black box", meaning that often it is not clear how they behave compared to traditional econometrics (Mullainathan and Spiess, 2017; Sheikh and Jahirabadkar, 2018; Storm et al., 2019). In order to produce accurate forecasts, the model has to be able to deal with a number of time elements: such as seasonality, trends or unit root. These elements introduce non-stationarity in the data, hindering the possibility of making proper forecasts. Moreover, there are also theoretical properties that need to be addressed when dealing with time series, for instance cointegration of two or more series. Cointegration implies that there is some long-run equilibrium relation among non-stationary time-series variables that often has a structural cause (Verbeek, 2017). This long-run relationship can be made explicit and exploited for forecasting purposes.

Dealing with all of these aspects largely depends on the pre-processing of the data, but there is a trade-off between transformation of the data and loss of information, and more or less flexible models are expected to react differently to this balance.

1.2 Research Objective

The objective of this research is to compare Machine Learning algorithms with time series econometrics models by analysing their differences and similarities and applying them on different pre-processed series of the same dataset. Particular focus is put on their forecasting performances as well as on which time elements are picked up by the Machine Learning models.

This general objective leads to four sub-questions to be answered:

- What are the characteristics of the selected ML and econometric models?
- In which aspects do the models differ and in which aspects are they similar?
- How do these models perform in forecasting agricultural commodities price series?

- How do ML models deal with time properties? How do they perform in respect to models able of picking up cointegration?

1.3 Background

We start from studies demanding for a deeper analysis of the functioning of ML learning models applied to economics (Ahmed et al., 2010; Mullainathan and Spiess, 2017; Storm et al., 2019), and decided to follow the most advanced procedure highlighted in studies concerning the forecasting of time-series data (Gilland, 2020; Bandara et al., 2019).

The subset of models that are discussed in this research are:

1. **Traditional econometric time series models**: Autoregressive Integrated Moving Average model (**ARIMA**), as it is the predominant model that has been used in the past (Storm et al., 2019). The acronym shows the key aspects of the model, which is composed by: an Autoregressive part (AR), capturing the relationship between a variable and its lagged values; the Integration order (I), representing the order of differencing needed to obtain a stationary series; the Moving Average part (MA), capturing the relationship between an observation and the errors of previous observations (Verbeek, 2017). Also its multivariate extensions, the **Vector Autoregressive** model and the **Vector Error Correction Model** will be presented and utilized.
2. **Machine Learning**: The machine learning models to be utilized in this research belong to the family of neural networks. The general idea governing neural networks is, starting from input data, to obtain the mapping of the outcomes through a series of layers building a chain-like structure of functions (Storm et al., 2019). In particular, our model is a deep Long Short Term Memory (**LSTM**) neural network.

The data set to be utilized is composed of time series of agricultural commodities prices retrieved from future markets, as explained in chapter III.

1.4 Methodological Design

Sub-questions 1 and 2 will be answered through a review of the existing scientific literature concerning the models. The third question will be answered running the models in Python, and then comparing the performances with the appropriate accuracy measures (Goodwin, 2020; Kolassa, 2020). Eventually, the last sub-question will be dealt with discussing the findings and results of the previous sub-questions.

1.5 Content Overview

After the introduction, in Chapter 2 the theory behind the models applied in this research is presented. Subsequently, the data set to be used is presented and the pre-processing procedures carried out are explained in detail in Chapter 3. Chapter 4 contains the methodology of the research: the specifications of the models will be reported; the clustering we applied to the time series will be discussed; and the accuracy measures chosen will be explained. The results will be reported in Chapter 5, highlighting their forecasting accuracy. Last, in Chapter 6 all the information gathered in the previous chapters is used to discuss the conclusions of the research and summarize its findings.

Chapter II – Theoretical Background

In this chapter we present the theoretical background of the models that will be utilized in the empirical analysis. First, a brief introduction to the traditional models is given, focusing mainly on what characterizes these models and contributed to their wide adoption. Then, a simple explanation of the NN and the LSTM architecture is provided, with particular focus on the algorithm used in this analysis.

2.1 Introduction

Forecasting has a long tradition in human history. If the first objective of science is to understand nature, the immediate following one is trying to predict natural events before their manifestation. This intention does not only concern nature, but also the social environment and in particular what relates to economy.

The most widely adopted model that was designed to take on this task is the Autoregressive Integrated Moving Average (ARIMA). This model tries to describe the relation between future and past observations of a certain variable, by modelling the patterns of the data generation process. To realize this model, the researcher has to look for seasonality, trends, structural breaks and lags.

What characterizes Machine Learning (ML) on the other hand is the development of algorithms able to **automatically learn by practice**. This means that its ability on a certain task (e.g. prediction), as measured by a certain performance (e.g. accuracy), improves the more experience (e.g. data) the algorithm is exposed to. Ideally, the broader and wider is the dataset that the learning algorithm is fed, the better its performance will be.

ML models are often called pure prediction algorithms (Efron, 2020) due to their nature. While traditional prediction methods are shaped by the original scientific goal of understanding nature, ML models are exclusively focused on high predictive accuracy, with no regards for estimation or attribution power (Efron, 2020). The natural data generation process is a "black box": also the artificial one is a "black box", but one that closely resembles the original one. Thanks to this resemblance, it can make very accurate predictions.

We focus on a specific class of ML algorithms: Neural Networks (NN). In particular Long Short Term Memory Networks (LSTM) are described, as they were specifically designed to learn from sequential data, as in the case of time-series.

2.2 Functioning of ARIMA models

The rise of ARIMA models can be explained by the disappointing results that initial structural macro-models bore in predictions. These models were based on theories coming from macro-economic (therefore structured), and often involved several variables to approximate the underlying data generation process. The failure of these models in the 1970s led to the development of unstructured, univariate models strongly data-driven: the ARIMA models. These models in fact tend to exploit the patterns in the data series, such as seasonality and trend, to make predictions for a single variable. To achieve this, ARIMA models combine autoregressive processes and moving average process, hence its name.

An important aspect to take into consideration when trying to forecast with ARIMA models is the **stationarity** of the data. Stationarity is a property of time series, and essentially it concerns the

stability of the data generation process over time: a stationary series is stationary if its probability distribution is independent of time; while the opposite can be said for non-stationary series. There are many elements that can give rise to non-stationarity, for instance a trend, seasonality or unit roots.

Stationarity is extremely important in forecasting, as we cannot reliably predict future observations if the law generating those observation changes as time passes. The ARIMA solution for this problem is to transform the time-series to be analysed, so to make it stationary. The most common transformations comprehend the pre-processing of the data to take care of the trend or the seasonality or first-differencing to get rid of units root. A series that becomes stationary after one first difference is said to be **Integrated of order one,** hence the "I" in ARIMA. For this to happen, there must be an exact unit root (Verbeek, 2017).

2.3 VAR and VECM models

ARIMA models can be extended to take into consideration also causal relationships with other time-series variable, as it may improve the forecasting accuracy. This is the case with Vector Autoregressive (VAR) models. There are several variants to this model, all with the key concept of exploiting dynamic relationships among the variables in the system.

The possible problem with this framework is that in case the variables are non-stationary, a **spurious regression** may occur. This is the scenario where two independent series, both containing a similar stochastic trend, seem to be related.

Nevertheless, a possible solution can be found if the non-stationary series in the multivariate model **have the same order of integration.** In this scenario, there might be a cointegrating vector such that the combination of the two non-stationary series is integrated of order zero (stationary). We refer to this property as **cointegration**: it implies a long-run relationship among the variables, that reciprocally tend to bring each other towards a certain distribution, and it often has a theoretical explainable basis.

To exploit this property Vector Error Correction Models (VECM) were designed. Following the Engle-Granger representation theorem (Engle and Granger, 1987), in a system of cointegrated variables integrated of order one there must be a valid Error Correction Model representation. In this representation, the long run relationships between the variables are made explicit, and it is possible to take advantage of them for making forecasts. In principle these long-term relations should lead to better forecasts.

2.4 Neural Networks

2.4.1 Origins and development

Neural Networks (NN) take their name from the functioning of biological neurons, but despite some similarities, the "artificial brain" and the natural one bear little resemblance to each other. Nevertheless, the analogy with a biological neuron is particularly suited to explain the original line of reasoning that led to the creation of the first Neuron model (McCulloch and Pitts, 1943] that later on became the Perceptron (Rosenblatt, 1958) which gave rise to the Neural Networks models.

A biological neuron is composed of three elements relevant for this example: the dendrites, which are the termination of the neuron from which stimuli and information (so the inputs) are received by the neuron; the nucleolus, which is the computational site of the neuron where the information is processed; and the axon, which is the terminal by which the response elaborated by the nucleolus (so the output) is transmitted to other neurons, or to the final destination of the message.

An artificial neuron (a Perceptron) follows the same logic: first, it receives the inputs from the outside; second, it processes the input by assigning them weights and biases (equivalent to coefficients in linear regression with biases equivalent to intercept), so to be able to filter or amplify information; last, the output, being the corrected weighted sum of inputs, is fed into an **activation function**. In biological neurons, every stimulus needs to overcome a certain threshold so to continue its transmission. It is possible to think about the activation function in the same way. Figure 1 provides an example in two dimensions for simplicity. The activation function f divides the plane in two regions. The algorithm only considers the inputs (x) whose output (y) belong to the green region (overcoming the established threshold).

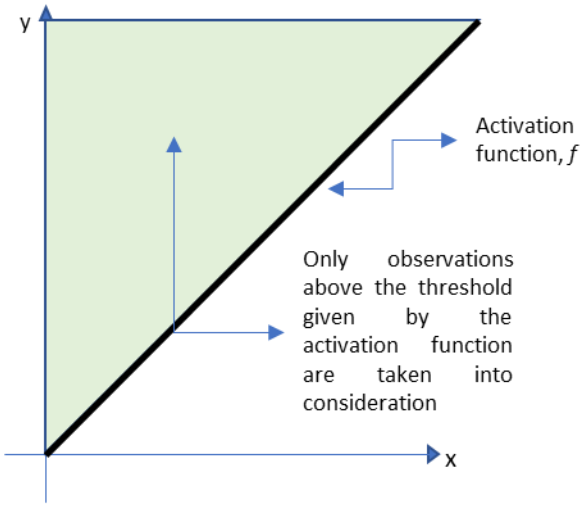

Figure 1: Activation function - simple visualization

Finally, NN are composed of layers – where each layer consist of several neurons – interconnected at various intensity among themselves. They are also composed of three types of layers: the input layer, receiving the input; the output layer, providing the final output; and the hidden layers, carrying out intermediate computations. Depending on how the neurons are connected within the layer, but also among layers as well as the type of connections, it is possible to obtain different kind of layers. By increasing the number of hidden layers, it is possible to increase the model ability to deal with complex functions, enhancing its flexibility. This data-driven model generalizability is what made NN particularly attractive to data scientist, as it enables to perform accurate prediction without being constrained by assumptions (Staudemeyer and Morris, 2019). In particular, neural networks can be used for classification and forecasting purposes.

2.4.2 The structure

There are four key elements for constructing a NN: the graph, the loss function, the optimizer and the initialization. In this section, a brief introduction to these elements is given while highlighting the main aspects of the algorithm functioning.

The graph:

The combination of connected neurons builds the **graph**, which is the approximation of the function we want to model. This final function will be conditional on all the parameters of the graph (weights and biases) as well as on the different activation functions in each node.

The loss function:

The weights and biases on which the graph depends have to be estimated. The **learning phase** of a neural network is the process by which the models choose the values of these variables so to ensure the best fit to the function we are trying to model. This in practice is an optimization problem, where the objective value is called a loss function as it represents the error (or distance) between our estimates and the real value we are aiming for. Depending on the task we are instructing our NN for (regression or classification) there are several loss functions that can be applied (Mean Square Error MSE, Mean Absolute Error MAE or Binary/Categorical Cross Entropy).

The optimizer:

Due to the high complexity of the function it is infeasible to obtain an analytical solution by computing the derivatives due to the huge number of parameters, and we have to rely on heuristics to find the optimal function. One of the most common heuristics is the Gradient Descent, which allows to find a minimum point following the direction in which the gradient is decreasing. The problem with this solution is that there is no guarantee of finding the global minima, and therefore we have to do with local minima. Moreover, the result is highly dependent on the starting point from which we compute the derivative. The Gradient Descent updates the parameters based on their previous value, the gradient and a parameter called **learning rate.** The setting of the learning rate is a critical point in constructing a NN, as a too low value would bring this process to converge in an excessive amount of time, while a too high value would make the process diverge. Equation (1) describes how the parameters for the model, i.e. weights (W) and biases (B) are updated via learning. This mechanism is what enables models to learn from the data, and it is one of the most relevant differences with traditional models.

$$[W,B]_{t+1} = [W,B]_t - \eta * 1/n \sum \nabla_{W,B} * loss[\mathbf{Y_i}, g(\mathbf{x_i}|W,B)] \qquad (1)$$

W=weights	η = learning rate	t = current time step	loss = loss function considered	
B= biases	1/n= average	$\mathbf{Y_i}$,$\mathbf{x_i}$ = vectors of data	$\nabla_{W,B}$ = gradient operator	g = the graph

In fact, also in econometrics we have to deal with loss functions (e.g. sum of squares), but given their relatively simplicity in terms of parameters to be estimated, there it is possible to compute the exact solution, in contrast with what happens with more complex models. When the training starts, a Neural Network assigns values to its parameters (W and B) to minimize the loss function, while at the same time computing the value for a certain **metric**. This metric is what allows to monitor the performances of the model. For instance when considering a forecasting task, the most common metric is the accuracy. The NN is then able to improve by iteratively updating these parameters, taking into consideration the minimization of the loss function as well as the improvement of the metric considered. Each iteration of the algorithm is called an **epoch**. The update of the parameters depends on the learning rate η and the gradient operator $\nabla_{W,B}$: these two elements govern the scouting of the model to improve its fit to the data. The larger the dataset available, the larger the pool from which the NN can extract its parameters' values. In this way, the model can learn from the data.

The most critical point that makes the implementation of NN feasible is **Backpropagation** (Rumelhart et al., 1986). This algorithm computes the values for the parameters of each layer starting from the last one and moving backwards, exploiting the chain rule of derivatives to avoid redundant calculations. Nevertheless this method generally did not bear great results, as it suffered from

overfitting or underfitting (Staudemeyer and Morris, 2019). To address this issue, the most common technique is the **k-fold cross validation**. The dataset is shuffled and divided in k groups. A group is selected and it is set as test group, and the remaining data is used for training the model. This process is iterated k-times, each time selecting a different sub-sample as a test set. After each training, the evaluation score is computed and retained and eventually the best model is selected.

The initialization:
In the Gradient Descent method, weights and biases need initial values so to start the computation of the gradient. These initial values are chosen or computed from a uniform or normal distribution.

2.5 Regularization and Overfitting

One of the key differences between the traditional methods and ML ones is the concept of **overfitting** and **regularization.** To produce accurate forecast in the future, the model should not approximate the underlying data generation process perfectly, picking up every aspect of it. Otherwise, we would be able to make reasonable prediction only in the case in which the future observations of a certain variable will have the same exact distribution. This concept, to some extent, related to non-stationarity, and represents a way by which NN address this issue: since we expect the observations in the future to vary from the past observations, we leave for room for flexibility in the model. The right balance between fitting the data and flexibility can be calibrated before forecasting on the training set.

Neural Networks, due to their high approximation power, can easily overfit, while this is not typically the case with simpler models such as the ARIMA. In order to overcome this problem, it is possible to recur to regularization strategies, aimed at decreasing the goodness of fit of the models. Classical strategies for NN are limiting the number of epochs; including dropout in layers, discarding at random a selected percentage of the units of the previous layer; including regularizers, that apply penalties to how parameters are computed. The models used in this analysis make use of all these strategies.

2.6 Recurrent Neural Networks

2.6.1 Approaching Time-series

Neural Networks proved to be extremely powerful models, able to achieve relevant results. Nevertheless, feedforward networks (going from an input layer to an output layer) do not take into consideration temporal states, and therefore they are limited to static tasks (for instance cross-sectional data or pattern recognition in images) (Nielsen, 2019). This is an extremely relevant problem for time-series forecasting, where it is key to be able to consider the whole evolution of the variable we are considering over time and to respect the order of observation over time.

To solve this issue Recurrent Neural Networks (RNN) were developed, addressing the problem in two ways. First, they are able to maintain an internal state carrying on the information processed previously. This is made possible by the presence of special hidden layers, commonly referred to as **memory states**, containing recurrently connected neurons. These layers accumulate the sequential information and maintain the knowledge acquired over time (Lazzeri, 2020) . Second, they iterate over the elements of the analysis: the networks utilize the output of the current step as part of the input for the next step during training. In this way, they obtain the ability to solve dynamic tasks (Lazzeri, 2020).

The most common way of representing RNN to explain the features above is like a chain of repeating modules of neural networks: in this way, it is clear that the final output depends on the whole history of the variables (figure 2):

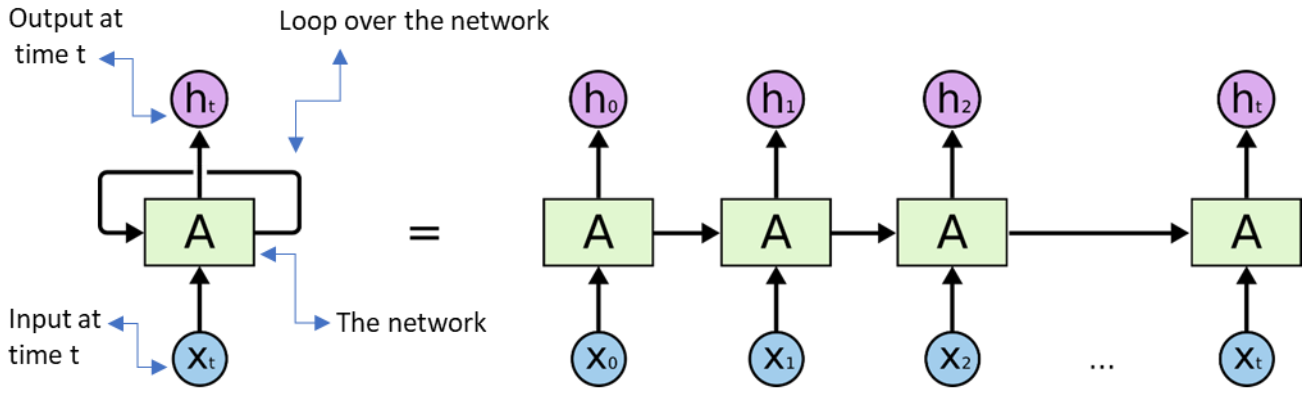

Figure 2: Unfolding the RNN

Despite solving the time-sequence issue, RNN still encountered two other problems in their application and were not efficient. The first problem regards the Backpropagation (explained in the section above), that in the time-series context is called Backpropagation trough time, that led to the problem of **Vanishing Gradients** (Nielsen, 2019)**,** hindering the learning phase which is not able to compute the optimal weights. The second problem is that the model has no control over what to remember. This is referred to as the **long-term dependency problem:** the inputs at the beginning of the time-series undergo a series of transformations (every time it is weighted and fed into the activation function), so that eventually is it impossible to assess which information has been taken into consideration and which has not.

2.6.2 Long Short Term Memory Networks

To address the two problems specified above, Long Short Term Memory (LSTM) networks were developed. LSTM are explicitly designed to control the vanishing gradient problem as well as learning effectively the long-term dependencies.

Also the LSTM can be represented via a chain-like structure, but in this case there are several differences. The first one is a standalone memory cell containing a recurrently self-connected linear unit, the **Constant Error Carousel (CEC)**. In this way, thanks to a state vector (Ct), it is possible for information to flow from cell state to cell state and to safeguard long-term dependencies (Staudemeyer and Morris, 2019). See figure 3.

In addition to this, the LSTM is provided with gates allowing it to filter or add information to the CEC. They output values ranging from 0 to 1, using a sigmoid function or the hyperbolical tangent, to carry out this process: a value of zero means that no information is added, while a value of 1 means that the whole information is added to the cell state.

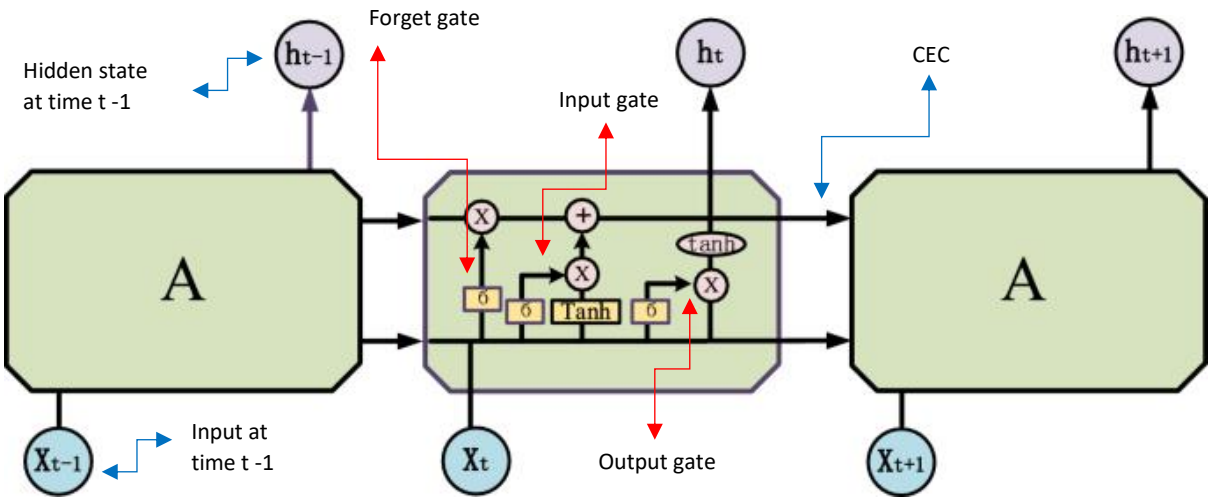

Figure 3: LSTM Network unfold

There are three types of gate (Staudemeyer and Morris, 2019):

- The forget gate: it determines how much of the information accumulated so far in the memory cell is to consider from now on

  $f_t = \sigma(U_f * x_t + W_f * h_{t-1} + b)$ *(2)*
- The input gate: it determines how much the new output can contribute to the memory cell. In particular, first it decides which values to update (3), and then it computes the new candidate values for the memory cell (4).

  $i_t = \sigma(U_i * x_t + W_i * h_{t-1} + b)$ *(3)*

  $\tilde{C}_t = \phi(U_c * x_t + W_c * h_{t-1} + b)$ *(4)*

  So that

  $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$ *(5)*
- The output gate: depending on the memory cell inner state, it controls how much information to use for generating the output.

  $o_t = \sigma(U_o * x_t + W_o * h_{t-1} + b)$ *(6)*

  $h_t = o_t * \phi C_t$ *(7)*

Legend:

U – input to hidden weights	f – forget vector
W – hidden to hidden weights	i – input vector
V – hidden to output weights	o –output vector
σ – sigmoid function	b – biases

ϕ – hyperbolic tangent function *(tanh)*

C – state vector of the CEC, preserving long term dependencies

H – is the hidden state: it is how the memory of the process is stored, and it is computed based on h_{t-1} and x_t. It represents the overall information accumulated so far by the model.

There are several variants to the LSTM model in respect to its inner structure, for instance the Gated Recurrent Unit (Cho, et al. 2014) version: in this model, the forget and input states are combined into a single update gate.

Chapter III – Data

In this chapter the data series used in the empirical analysis are presented and commented. In the first part, each time series is shown, and the most relevant information for each time series is summarized. In the second part, the focus shifts on the pre-processing procedures we applied on the data. More information about the dataset can be found in ANNEX A.

3.1 Data Description

The data utilized in this analysis are 13 time series measured at daily frequency. These series represent the stock quotations of future contracts traded on different market belonging to 6 categories: grains, financial indexes, animal products, metals, energies and currencies. The series were retrieved from Barchart (2020), a website gathering stocks and future contracts quotation while providing users insights on trends. Due to the fact that future contracts have a finite expiration date, in contrast to stocks that trade perpetually, nearest-series were used at each frequency so to obtain continuous series rolling from one contract to the next for the whole period considered.

Grains:

The first group of time series used in this analysis is represented by the future contract prices of grains, in particular corn, soybean, hard red wheat and spring wheat. This group of basic commodities is composed by the major cereals traded on the future markets, especially on the Chicago Mercantile Exchange and related. Figure 1 shows that these four price series follow a very similar trend, peaking and dropping simultaneously.

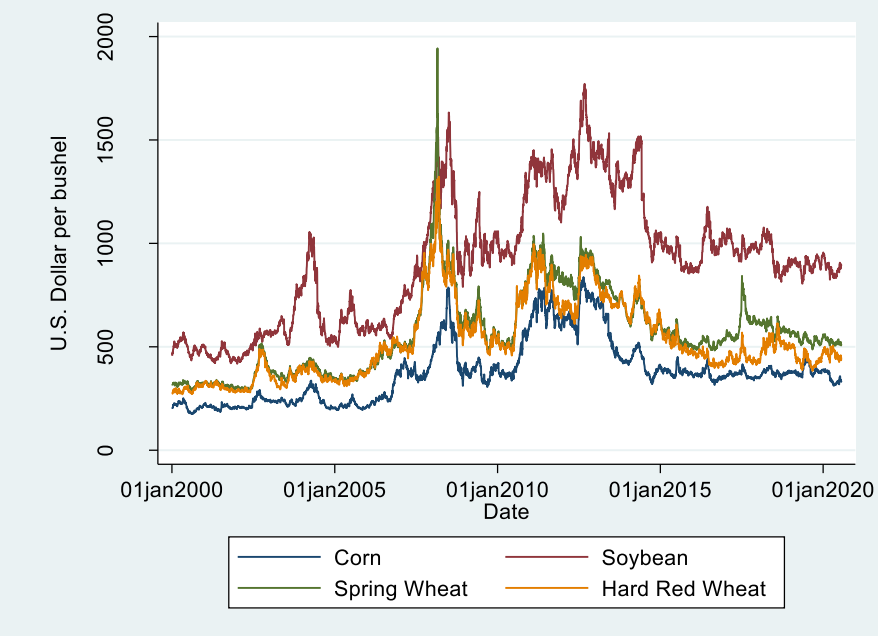

Figure 4: Grains future prices per unit from January 4th , 2000 till July 23rd , 2020

Financial Indexes:

The second group of time series used in this analysis contains two indexes of stock market movements: the E-mini futures for the Standard & Poor 500 and the Nasdaq 100. The S&P 500 is an index composed of the market-capitalization weighted average of the largest 500 publicly traded U.S. firms (CME Group, 2020). The Nasdaq 100 is also a market-capitalization weighted index, but composed of the

100 most actively traded U.S. companies listed on the Nasdaq stock exchange, excluding the ones belonging to the financial industry (CME Group, 2020). E-mini futures are electronically exchanged future contracts representing a fraction[1] of the value of the original future contract, introduced to make investments in indexes, such as the one presented here, affordable for small investors.

Figure 2 shows the development of these three indices for the period January 2000 – July 2020. In the first part of the graph the series are characterized by an initial drop followed by a stable trend. In the second part, all three series follow an upward sloping trend.

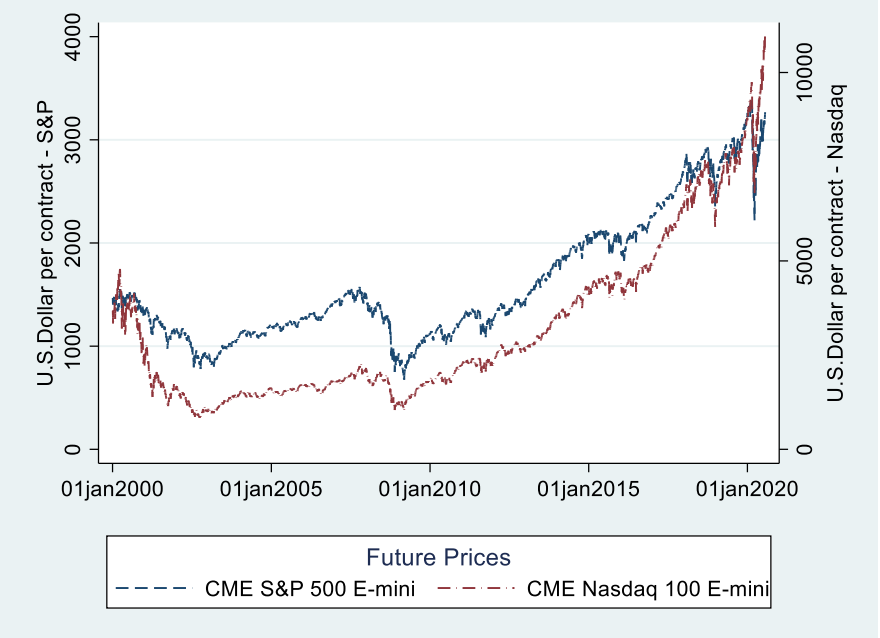

Figure 5: Financial indexes contract prices from January 4th , 2000 till July 23rd , 2020

Animal products:

The third group of time series consists in future contracts traded on the Chicago Mercantile Exchange related to animal production, namely Class III milk, feeder cattle and live cattle. Milk on future contracts is priced based on its final use: class III stands for the milk that is used for the production of hard cheese. The difference between feeder cattle and live cattle is in the production phase of the animal: feeders cattle are weaned calves usually weighting between 600 and 800 pounds that still needs to gain mass in order to be slaughtered, while live cattle are finished animals ready to be slaughtered and transformed (CME Group, 2020). Feeders cattle prices were selected as we expect to find a cointegrating relationship between this series and the ones for soy and corn, as these two highly energetic grains are the predominant feed used for the purpose of making cattle gain weight (Moreira et al., 2019).

Figure 3 shows that the three series present a similar trend, characterized by rapid fluctuations over the whole period considered.

[1] For the Nasdaq 100 E-mini and Standard & Poor 500 E-mini the value is set at one fifth of the value of the original contracs. (CME group, 2020)

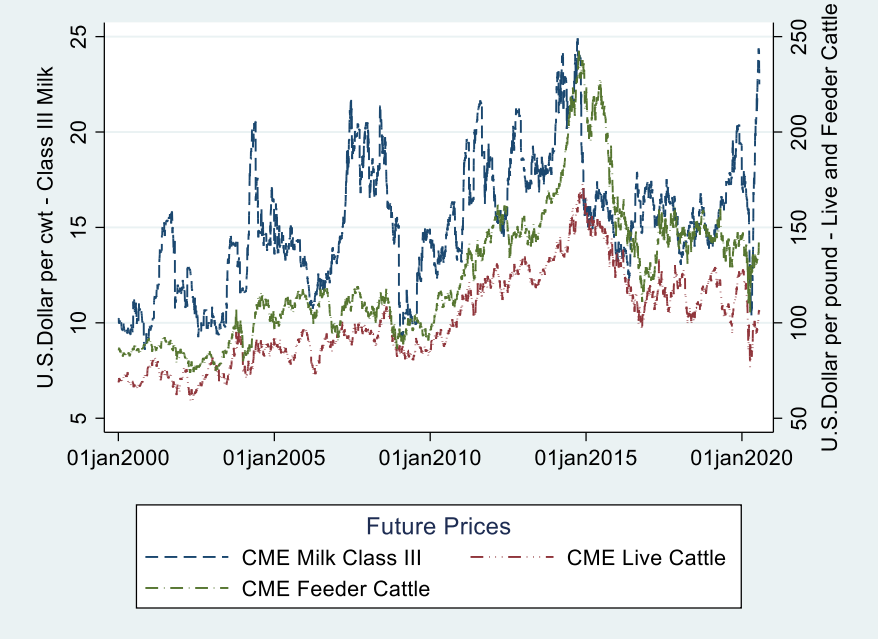

Figure 6: Animal products prices per unit from January 4th , 2000 till July 23rd , 2020

Metal:

The fourth group of time series is the one composed by metals; in our case only by gold. Gold, as a product whose value is acknowledged all over the world, has always been perceived as a way to safeguard investor's' money against unstable geopolitical and macro-economic conditions (Barchart, 2020).

This time series shows an upward trend for most of the time period considered. The gold price temporarily dropped in the period 2011-2016, after which it increased again.

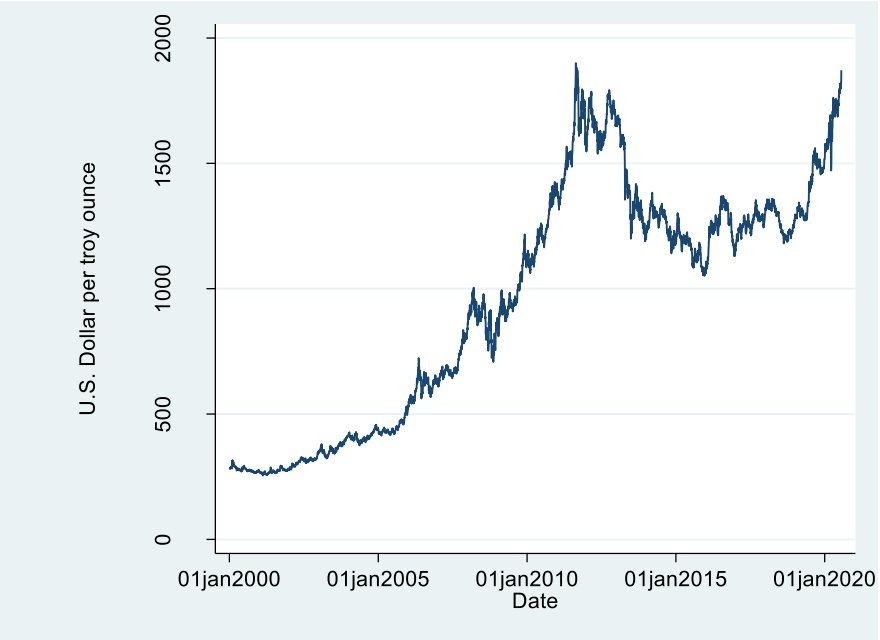

Figure 7: Gold price per unit from January 4th , 2000 till July 23rd , 2020

Energies:

The fifth group is the one concerning energies. For the purpose of this research, we have taken into consideration the future prices for the Crude Oil WTI, natural gas and Gasoline RBOB.

West Texas Intermediate Oil is an high quality oil sourced in Texas that is used as a global benchmark for oil, together with Brent crude oil and Dubai Crude (Barchart, 2020).

Reformulated Blend-stock for Oxygenate Blending gasoline is a derived of crude oil. Nevertheless, RBOB market has its own supply and demand factors, as its production is affected for instance by different taxations in different jurisdictions [NYMEX, 2020]. This can also be seen in the two series patterns, not particularly similar.

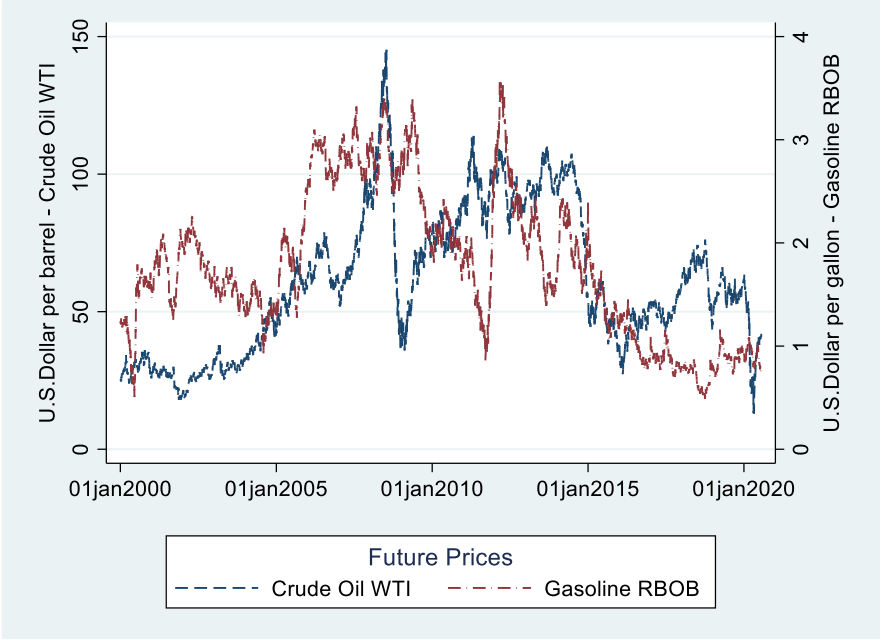

Figure 8: Crude Oil WTI, Natural Gas and Gasoline prices per unit from January 4th , 2000 till July 23rd , 2020

Currencies:

The last series is the U.S. dollar index. It is a geometrically averaged calculation of six currencies, adopted by the six most relevant U.S. trading partners, weighted against the U.S. dollar. It is calculated by taking into consideration the exchange rates among this currencies: as a result, the index provides guidance on the relative strength of the dollar versus the other currencies (Barchart, 2020). The development over time is given in figure 9.

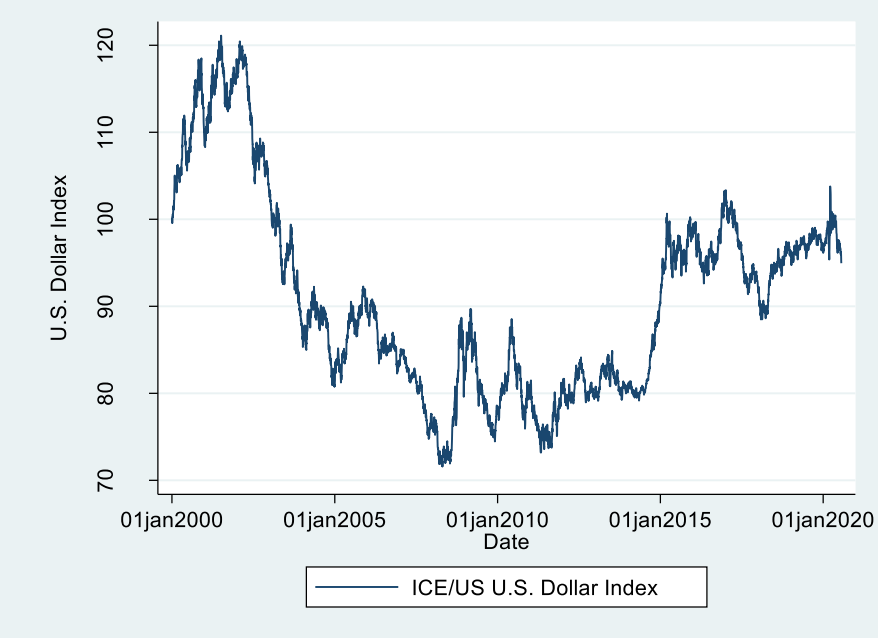

Figure 9: Euro FX and U.S. dollar index contract from January 4th , 2000 till July 23rd , 2020

3.2 Data Pre-Processing

In order to ensure the correct functioning of Neural Networks (NN), data need to be pre-processed. Different time series with different scales in fact might prevent the model from learning effectively, by making diverge the parameters during the computations. As stated in the introduction, pre-processing is also fundamental to deal with some time properties, such as seasonality, trends and unit roots: more in general, with non-stationarity.

3.2.1 Comparison between LSTM and traditional models

Rolling Window approach:

The Rolling Window (RW) procedure is based on the most advanced pre-processing procedures utilized for LSTM, and in particular from Bandara et al. (2019). Within this method, there are three elements to take care of: the stabilization of the variance, seasonality, and the trend.

The stabilization of the variance was performed first, through a logarithmic transformation of the raw data. This transformation in practice reduce the scale of the data.

The de-seasonalization was carried out using the STL package in R (Hyndman et al., 2015). This package divides the series into a trend, seasonal and remainder components, and allows for taking care through a robust procedure of deterministic seasonality (Bandara et al., 2019).

Last, we eliminated the trends from the series. To achieve this, we utilized a rolling window approach computing the trend through the Tukey's biweight estimator (Mosteller and Tukey, 1977; Hippke et al., 2019). The advantage of this technique, besides providing more accurate estimates of local trends, is that it does not introduces information from the training test to the test set, affecting the actual performances of the NN.

First Difference approach:

First-Differencing (FD) is the most common traditional approach to non-stationary series. This methods consist of subtracting from each observation of the series its previous observation, so to

obtain the difference between two consecutive observations. This difference, in case of a precise unit root, it's stationary.

Unprocessed data:
Raw data are included in the analysis as the problem with pre-processing is that it causes the loss of information. In this case, removing the non-stationarity from the data prevent us to exploit the long run cointegrating relationship that might exist between two or more variables.

On the databases obtained from the first two procedures, we decided to confront the performances of the LSTM versus the traditional models, VAR and ARIMA. In particular, we decided to compare the models first on a 30 days forecast horizon, and consequently on a 7 days forecast horizon. Both of these methods remove effectively non-stationarity from the series in our case, as tested with Augmented Dickey-Fuller and KPSS tests.

Then, we proceeded to confront the LSTM performances on these two datasets versus the performances of the VECMs based on the real data. The reason for this choice was to confront these two kind of models in a realistic setting.

The ultimate goal of this comparison is to understand how these two different kind of models relate and how they perform in respect to these different methods of pre-processing data. Different pre-processing in fact entails two different aspects that need to be considered: first, a different set of time elements of the series that need to be modelled or learned by the models; second, a different amount of information left in the series. Therefore, we also aim at testing whether these procedures are worth in term of loss of information, or if it is better to exploit those information.

3.2.2 Comparison across dataset for LSTM

Another important aspect of this thesis is understanding which time elements the NNs are able to pick up. To do so, starting from the raw data we proceeded to remove one time property at a time utilizing the same techniques described above, to see how the model would perform. In addition to those transformation the data were also standardized, as it is common practice when using NN. We obtained the following subsets:

- **Unprocessed data,** identified with the suffix "_un"
- **Trend data,** where the seasonality is removed and the trend is kept. The models run on this dataset are identified by the suffix "_t"
- **Seasonal data,** where the trend is removed and the seasonality is kept. The models run on this dataset are identified by the suffix "_s"

In Bandara et al. (2019) in fact it is also stated that de-seasonalization should not be necessary when the time-series in the analysis present calendar features and/or homogeneous seasonality pattern. Our series are sampled on a daily basis, and can be expected to have homogeneous seasonality as the grains production cycle is defined by the growth season.

By doing this, we should be able to understand which elements are preventing the LSTM from learning from the data.

The differently processed datasets used in the analysis are summarized in table 1:

Table 1: DatasetS used in the analysis

Dataset	Details	Identifier
Rolling window	*Logarithmic transformation* *Seasonality removed* *Trend removed*	_rw
First difference	*Unit root removed*	_fd
Unprocessed	*Standardization only for LSTM*	_n
Trend	*Seasonality removed* *Standardization*	_t
Season	*Trend removed* *Standardization*	_s

Chapter IV – Methodology

In this chapter the methodology we applied to carry out the analysis is reported. Starting from how the data were clustered, the procedures to construct the models are explained in detail, and the specifics of each model are shown. At the end of the chapter, the accuracy measures utilized to compare the performances are discussed.

This section, together with chapter II, provide some insights on the differences between the traditional econometrics and the Machine Learning models.

4.1 Data Clustering

Training the NN on a set of heterogeneous time-series could lead to a decrease in model performance. Therefore, Bandara et al. (2019) propose to cluster series as the initial step to take advantage of the similarities between the series. This clustering should be based on time properties of the series, such as frequency and seasonality. Clustering can also be based on a-priori knowledge that the researcher has on the data. For this reason, we decided to avoid this step in our analysis: the series we are considering all have the same frequency and relate to the same domain (commodity market).

Therefore, the sub-groups on which the analysis is carried out are based on the authors knowledge, dividing the series into groups relating from the agricultural and economical point of view. This also allow us to assess to what extent the different methods can take advantage from considering an increasing number of variables. The resulting models are summarized in table 2.

Table 2: Data clustering and resulting models

Models	Variables
LSTM tot **VAR tot**	*All variables are considered*
LSTM 6 **VAR 6** **VEC6**	*Considering only:* Corn – Soybean – Hard Wheat – Spring Wheat – Feeder Cattle – Live Cattle
LSTM 4 **VAR 4** **VEC 4**	*Considering only:* Corn – Soybean – Hard Wheat – Spring Wheat
ARIMA	*Only Corn is considered*

4.2 Building the Models

4.2.1 Traditional models

The following procedure was applied for constructing the VAR and ARIMA models both on the RW dataset as well as the FD one:

- Check for stationarity of each series, by means of the Augmented Dickey-Fuller test as well as the KPSS (Verbeek, 2017). All series were found to be integrated of order zero.

- The **univariate** model was determined by using the Autocorrelogram and Partial Autocorrelation graph, and consequent white-noise error tests. The choice for autoregressive processes or moving averages was guided by the obtainment of white noise errors, as well as parsimony, since there are no fundamental differences (Verbeek, 2017)
- For the **multivariate** cases we proceeded with a VAR model, establishing optimal number of lags with *varsoc.* The choice for the number of lags and the stability of the VAR after the estimation was then checked through *varwle* and *varstable* routines.

The following procedure was applied to construct the VEC models on the real dataset:

- **Cointegration** was tested for through the Johansen test (Verbeek, 2017). The optimal number of lags is based on Information Criteria such as AIC and BIC and Likelihood Ratio tests.

Once the models have been estimated, predictions for 30 and 7 days ahead were produced. The predictions are *dynamic* in the sense that the predictions for each time step are used recursively to predict the next step ahead. The reason we choose this option is that this is the most common method to make prediction when using ARIMA models, despite the risk of carrying over errors in consecutive predictions given an error in the beginning.

The resulting final models are summarized in table 3.

Table 3: Traditional models specifications

	Specifications	
	Rolling Window dataset:	First differenced dataset:
VAR tot	*2 lags, constant*	*2 lags, no constant*
VAR 6	*2 lags, constant*	*2 lags, no constant*
VAR 4	*3 lags, constant*	*3 lags, no constant*
ARIMA	*AR(1), I(0), constant*	*AR(1,3), I(3), MA(7,9), no constant*
	Unprocessed dataset	
VEC 6	*2 lags, 2 cointegrating vector, constant*	
VEC 4	*2 lags, 1 cointegrating vector, constant*	

4.2.2 Neural Networks

This thesis uses a specific training technique to overcome issues related with working with a sequenced dataset: the sliding window technique. K-cross fold validation in fact is not feasible in this context, as it will not preserve the chronological order of the observations.

A simple way to visualize the sliding window approach is by considering a zipper mechanism, where each tooth on the string can be considered as a time step. The first part in this procedure is designing a window covering a certain number of steps. The window is trained over a fixed proportion of those steps (the teeth below the left side of the slider) and validated over the remaining part. Starting from the beginning of the sequence, the window is applied and the first set of parameters is computed. Then, this window (the zipper) is rolled over the sequence (the string), each time updating the parameters. The parameters computed of the last window of data are the final parameters for the

algorithm. In this way it is possible to cross-validate the parameters while preserving the temporal state of the data.

Therefore, the first step for each of our models is the establishing of the sliding window. Considering the 30 days forecast horizon, we start from a window consisting of a total of 395 time steps, 365 of which are input steps and 30 output steps, sliding 180 time steps (roughly half a year) at time. This means that starting from the first observation, we feed the model a year of observations and validate the parameters by measuring the accuracy in predicting the following month. Then move to the 181st time step and repeat this operation, until reaching upon the observation 395 steps back from the last one. The same reasoning goes for the 7 days forecast horizon, where the sliding window has 7 output steps.

The machine learning model is a Recurrent Neural Network, consisting of 4 layers fully interconnected among them: 3 LSTM layers, each of 512 units, and one Dense layer. The prediction from these models are produced directly all of 30 forecast contemporaneously. The specification is summarized in table 4:

Table 4: LSTM specifications

	Layers	Units	Dropout rate	Kernel regularizer	Lambda	Loss function
LSTM	3 LSTM 1 Dense	512 [1]	0.3	L2	0.01	MSE

[1] *Equal to the number of variables considered times the number of forecast horizon steps*

The **LSTM tot, LSTM 6, and LSTM 4** have all the same architecture. What is different among them is the number of epochs on which they were trained.

After pre-processing of the data, we proceeded with the division of the dataset into a training, validation and test sample, maintaining the order of the sequence. The training set is the portion of the data on which the parameters of the models are computed; while the validation set is the portion of the data used as a reference to evaluate the performances of the model. We ran the models on the training set and measured their performances on the validation data to calibrate the **hyperparameters** of the models, such as the number of units or the number of epochs. Once the models were performing satisfactorily on the validation set, we fed them the test set and obtained the future predictions.

4.3 Accuracy Measures

To compare the performances of the traditional models versus the Machine Learning one, the Mean Absolute Error (MAE), the Root Mean Square Error (RMSE) and the Mean Absolute Percentage Error (MAPE) were chosen.

The choice of the MAE as well as the RMSE was largely dictated by their wide application in comparing different models on the same dataset. The first is computed as the arithmetic average of the sum of the absolute value of errors, defined as the difference between the real value and the prediction. The second is computed as the root of the quadratic difference between the observations and the predictions. RMSE in respect to MAE is suggested to be more susceptible to outliers (Hyndman and Koehler, 2006).

To provide a quick insight in the performances of the models, also the relative MAE is included in the table, using as the baseline performance the ARIMA model in each cluster. This was decided as the ARIMA model is traditionally the most commonly adopted model.

In addition to those scale-dependent measure, the MAPE was chosen. This measure of accuracy is scale-independent as the absolute differences between the observations and the predictions is divided by the actual values before computing the mean error.

Chapter V - Results

In this chapter the results are presented, divided as introduced in the previous chapters: first the comparison on the Rolling Window dataset; second on the First-Differenced dataset; third the comparison between LSTM and VEC; and last the comparison of the performances of the LSTM across datasets. The accuracy measures are explained, but only the table for the first analysis is presented with all the measures. This is done as the measures rank almost in identical order the models within each analysis. See ANNEX B for the whole set of complete tables. Only the graphs with the forecast predictions comparison of the models using all variables are shown in each section. See ANNEX C for the whole set of graphs on all the different clusters.

5.1 Results on Rolling Window Data

Table 5 presents the performance measures for the models based on the Rolling Window dataset with a 30 days forecast horizon. It can be seen that the performance decreases from the most complex to the simplest model.

Table 5: Performances comparison on the Rolling Window approach on the 30 days forecast horizon

RW – 30 days	**RMSE**	**MAE**	**MAPE**	**RelMAE**
LSTM tot	10.28	8.30	2.44	0.53
LSTM 6	10.75	8.36	2.45	0.53
LSTM 4	10.56	8.30	2.44	0.53
VAR tot	15.44	12.50	3.65	0.80
VAR 6	16.61	13.45	3.93	0.86
VAR 4	17.74	14.34	4.19	0.92
UNI	19.26	15.66	4.57	1.00

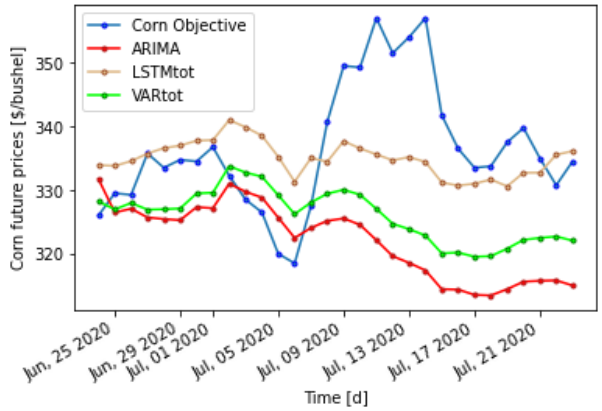

Figure 10: Forecast predictions - LSTM tot versus VAR tot - Rolling Window dataset on 30 days forecast

As stated in the introduction, one of the biggest advantage of NN is the ability to take into consideration an higher number of variables compared to traditional models. Nevertheless, from table 5 it is not possible to appreciate this fact. The performances of the LSTM models trained on different numbers of variables does not differ particularly in fact. On the other hand, surprisingly this seems the case with the VAR models.

The forecasts for the traditional models are surprising, since these are commonly known to flatten out rapidly after the initial predicted steps. In figure 10, what's preventing the series to flatten out is simply the fact that the models were run on the de-seasonalized and de-trended series. The seasonal component and the trend, eliminated during the pre-processing, were respectively added after the estimation, giving the impression that the predictions are better fitting the actual future values. In reality, this better fit is to be attributed to the pre-processing of the data.

Effectively therefore, the traditional forecast are flattening out with a downward trend, but this is partially obscured by the re-introduction of the time elements. The LSTM is able to perform better as it does not flatten out after the drop happening at July 6th . The fact that LSTM does not flatten out makes it suitable for longer forecast horizons.

RW – 7 days	MAE	RelMAE
LSTM tot	5.63	0.79
LSTM 6	7.19	1.00
LSTM 4	10.26	1.43
VAR tot	7.50	1.05
VAR 6	6.42	0.90
VAR 4	6.42	0.90
UNI	7.17	1.00

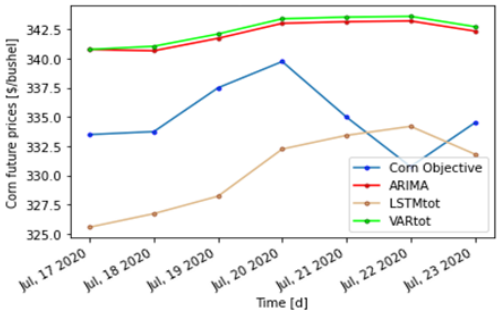

Figure 11: Performances comparison on the Rolling Window approach on the 7 days forecast horizon

Considering the 7 days forecast horizon, as expected the accuracy of the models increased in respect to the 30 days forecast (figure 11), and there are some other relevant changes. It is possible to appreciate how the LSTM improves their performances the more variables they are fed to. On the other hand, the VAR models perform better when given only variables closely related[2] to the one we are trying to predict. Moreover, it is possible to see how for shorter horizons more traditional models have better predictions.

5.3 Results on First Difference Data

FD – 30 days	MAE	RelMAE
LSTM tot	7.96	1.07
LSTM 6	6.93	0.93
LSTM 4	9.49	1.27
VAR tot	14.76	1.98
VAR 6	14.19	1.90
VAR 4	14.26	1.91
UNI	7.46	1.00

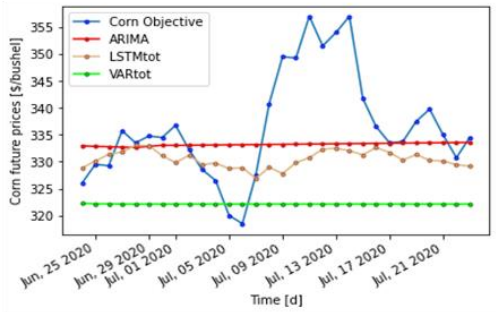

Figure 12: Performances comparison on the First Difference approach on the 30 days forecast horizon

Also in the case of using first differenced data the LSTM models are performing better than the traditional models (figure 12). Surprisingly, the **LSTM 6** model is the one performing better than a simple univariate **ARIMA** model.

The predictions from the classical models on this dataset are similar to what we would expect from them. In fact, after a few time steps they perfectly flatten out. The **ARIMA** model, given the fact that it only deviates slightly from the last observation more or less predicts the mean of these future observations. In fact, it displays the second higher performances within the group, but this is due to

[2] The regression of Soybean, Hard Wheat and Spring Wheat showed significant parameters; and these variables were found in the VECM to be in a long run relationship.

the fact that the last future prices are very close the initial ones. Also in this case it is possible to see that the LSTM predictions do not flatten out, allowing to forecast for a longer horizon.

FD – 7 days	MAE	RelMAE
LSTM tot	5.09	0.69
LSTM 6	4.44	0.60
LSTM 4	4.47	0.61
VAR tot	3.83	0.52
VAR 6	3.83	0.52
VAR 4	3.95	0.54
UNI	7.35	1.00

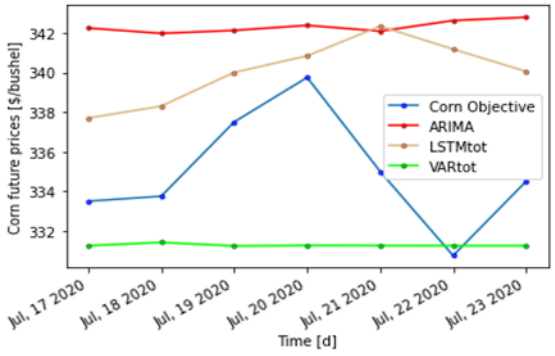

Figure 13: Performances comparison on the First Difference approach on the 7 days forecast horizon

On the shorter forecast horizon, the traditional models are performing better (figure 13). This scenario, with first-differencing is the situation where normally the VAR models would be used. This supports what was notable also in the Rolling Window dataset, that more rigid models perform better on shorter forecast horizons.

5.4 LSTM versus VECMs

Cointegration - 30 days	MAE	RelMAE
LSTM_rw	8.30	1
LSTM_fd	7.96	0.96
VEC 6	11.26	1.36
VEC 4	12.44	1.50

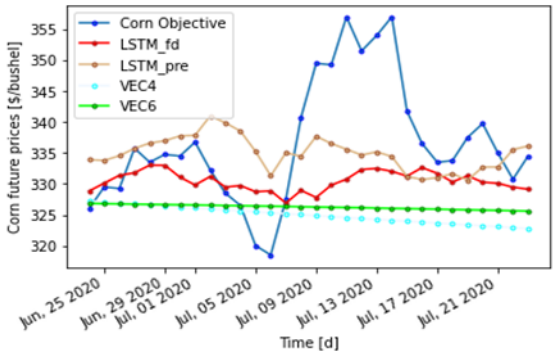

Figure 14: Performances comparison LSTM versus VECMs approach on the 30 days forecast horizon

Much like in the previous comparisons, on the longer forecast horizon the LSTM models are performing better than the traditional ones (figure 14). Also in this case, the VECMs are flattening out as expected. Their forecasts are not fitting the data, despite the increased amount of information that the VECMs could in theory exploit to make predictions by means of the cointegrtaing relations. This result is particularly disappointing as the coefficients of the estimated cointegration relationship were found to be statistically significant, and to some extent showcase the weaknesses of relying on these kind of methods to build the model (Breiman, 2004). While the LSTMs predictions manage to describe satisfactorily well the future observations, given the non-stationarity of the series, VECMs one are trending downward and getting further from their target.

Cointegration– 7 days	MAE	RelMAE
LSTM_rw	6.28	1.00
LSTM_fd	5.85	0.90
VEC 6	6.39	1.13
VEC 4	6.63	1.18

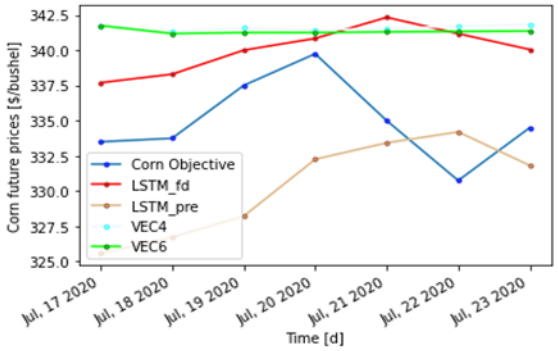

Figure 15: Performances comparison LSTM versus VECMs approach on the 7 days forecast horizon

In contrast with what happened in the previous comparisons, even with the shorter forecast horizon the VEC models do not outperform the LSTM (figure 15).

It is possible to notice that except for the First Difference case with a 7 days forecast horizon, the VECMs are performing better than the relative VARS. This finding support what stated in the introduction: there is a trade off between the pre-processing of the dataset, and therefore how accessible the dataset is made for the model, and the amount of information in the dataset.

However, this increased amount of information did not enable VECMs to outperform LSTMs. There is possible intuition as to why the increased amount of information did not enable the VECMs to outperform the LSTM models. It can be thought that the dynamics of the long-run relationships between the variables might be different than the ones contained in the forecast horizon, meaning that they would need a longer horizon to come into play and actually improve the forecast.

5.5 LSTM across Datasets:

Time elements	MAE	RelMAE
LSTM_rw	8.30	1.00
LSTM_fd	7.96	0.96
LSTM_n	249.17	30.02
LSTM_t	862.98	103.97
LSTM_s	34.45	4.15

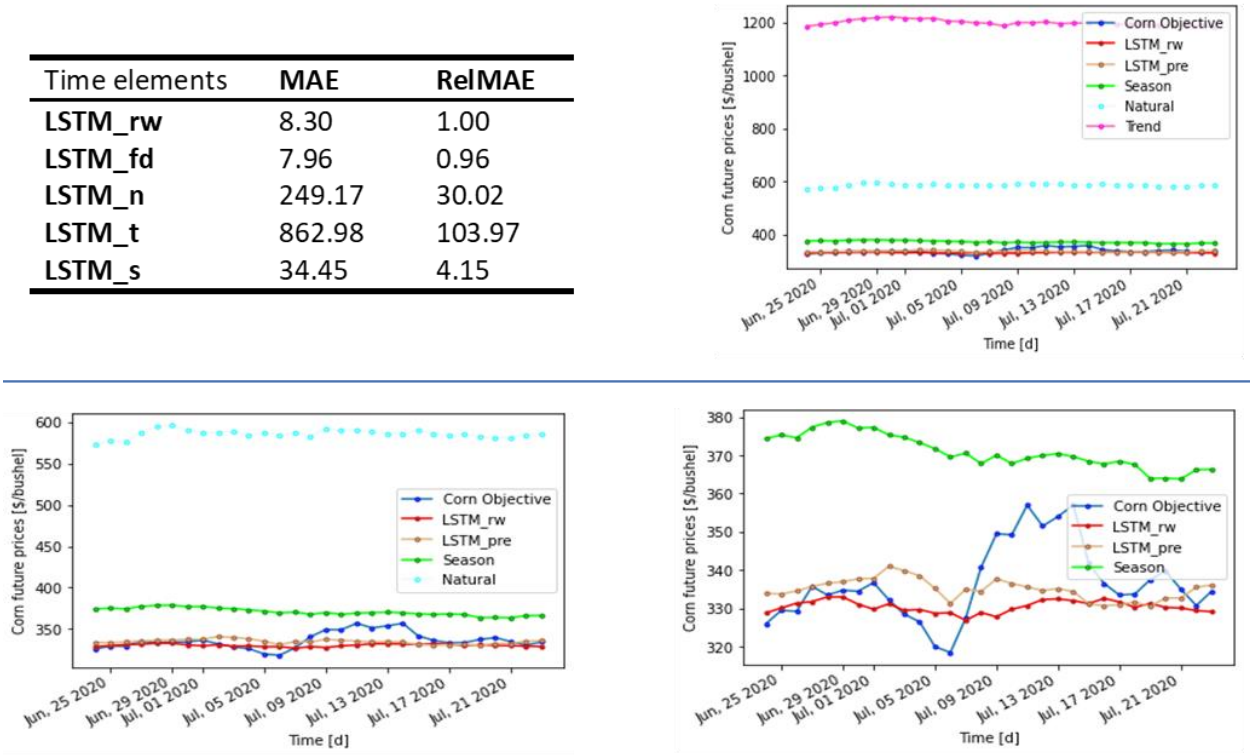

Figure 16: Performances comparison LSTM across datasets on the 30 days forecast horizon

From Figure 16 it is possible to understand how fundamental pre-processing is for Neural Network functioning. Beside the hindering of the learning phase if the dataset is not scaled properly, the LSTM fed on unprocessed data is not able to pick up certain time elements, with extreme consequences for the forecast accuracy.

When fed with the dataset where the seasonality was removed, but not the trend (figure 16, top right graph), the LSTM produces forecast which are completely out of scale. As suggested by Bandara et al. (2019), the trend in fact might be particularly difficult for the LSTM to pick up.

Also when fed with the raw data (figure 16, bottom left graph), that were only standardized to make the training possible, the LSTM produces results which are completely out of scale.

Last, when fed with the dataset where the trend was removed, but not the seasonality (figure 16, bottom right graph), the prediction by the models get closer to the actual observed data, but are still completely out of boundary. It is interesting to observe how the unprocessed data, containing both seasonalities and trends, is more suited for making predictions than the dataset containing only the trend (figure 16, table).

5.6 Discussion

In this section the results from all the analyses are charted together to provide a comparison of the effectiveness of the pre-processing and the importance of the number of variables as well as the forecast horizon. Figure 17 shows the MAE of all models divided by pre-processing procedures and forecast horizon.

Figure 17: LSTM versus traditional performances comparison across number of variables, datasets and forecast horizons

From figure 17, it is possible to draw some conclusions about the performance of the LSTM as well as the traditional models:

- The LSTM models outperform with almost double accuracy in the longer forecast horizon; However, for the shorter forecast horizon the picture is mixed
- The LSTM is able to take into consideration a large number of variables, but in this analysis it does not seems like it is able to capitalize on it, except for the third case. The same goes for the VAR models, as there are no significant difference between models comprehending a different number of variables
- Using the Rolling Window approach or the First Difference does not seems to have a great impact on the 30 days forecast horizon, where only a few models change their performances slightly. On the 7 days forecast horizon it makes a relevant difference which method to use, in particular for the VAR models.

To recap, both of the two pre-processing methods present advantages and disadvantages. Starting from the **Rolling Window** approach, it allows to use very different time series in the analysis, that could otherwise be very disruptive for the analysis due to their dissimilarities. This is particularly true for the LSTM. This justifies their widespread adoption for machine learning analysis in most of the scientific domains, but it might be more suited per application such as the one seen in the M-competitions (Hyndman, 2020) rather than forecasting agricultural future prices. These series in fact are characterized by having defined time series characteristics[3], and therefore do not need such heavy pre-processing.

To this end, after you perform this many pre-processing steps on the database, what it is basically left is almost white noise. In this case it was a perfect AR(1) model. This can be considered as a **loss of information**. Ideally, the model should be able to pick up all of these different elements and make predictions after them. The seasonality, the trend on more in general the non-stationarity are elements of the time-series and as such they are part of the prediction problem.

[3] For instance the frequency is dictated by the future market, while the seasonality dictated by nature.

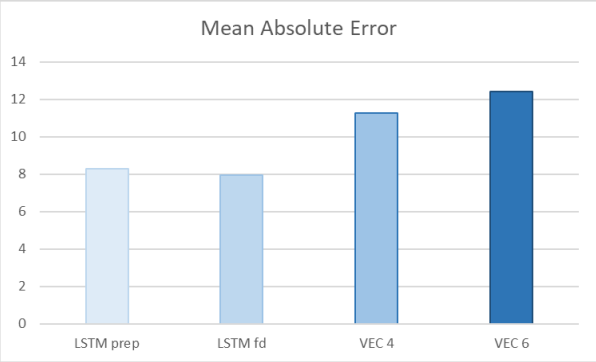

Figure 18: Performance comparison LSTM versus VECMs – 30 days forecast horizon

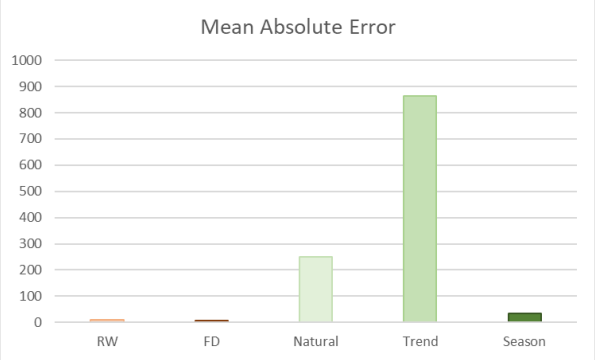

Figure 19: LSTM performance comparison across datasets

The increased amount of information contained in the natural dataset, in particular the long run relationships, allows the VECMs to score a lower MAE than the correspondents VAR models. The comparison between the LSTM and the VECMs does not really provide us final results. It is hard to believe that the neural network could be able to pick up the long term relationships among variables, especially given the pre-processing that the data underwent. Nevertheless, the machine learning models are outperforming the traditional ones including the VECM (figure 18).

Figure 19 provides an idea of how relevant it is for the LSTM to have properly processed data for the analysis.

Differently to what suggested in Bandara et al. (2019), in our case the LSTM was not able to pick up seasonality when trained on a set of similar time series.

Particularly interesting is that the data where only the trend was left performed worse than the raw dataset, also containing the trend. This element supports what is stated previously, that pre-processing of the data introduces bias and assumption, modifying what is actually modelled by the series.

Given the implications in this final section of the discussion, it is possible to understand how the procedures used to train the LSTM on non-stationary data, containing seasonality, trends or cointegrating relationships have still room for improvement.

Chapter VI – Conclusion and Implications

In this last chapter the answers from the analysis to the research questions are proposed. First, the conclusions and implications from the analysis are drawn. Then, critical reflections on the research are exposed and discussed, providing the limitations as well as the way forward from this analysis.

6.1 Conclusions

Machine Learning models and the traditional ones from econometrics display many differences. These differences can be found in the way they try to solve the forecasting problem; in the resulting structure; and in their building procedure. However, there are also some similarities that can be highlighted.

VAR models approach the forecasting problem by trying to describe reality. Consequently, the structure is shaped by the researcher who selects the variables that are considered relevant, the lags and every other specification. This is done by mean of statistical tests, and guided by parsimony. The problem with this **parametric approach** is that the reins are left to the researchers, who tries to represents an underlying truth that might not even exist (Efron, 2020), or that might be valid only for a specific range in time. Moreover, every test carried out in order to justify the choices of our model introduces assumptions, which eventually will make the forecasting only applicable in a specific frame which might be very distant from the real data generation process (DGP). In our analysis, the results of the sum of these elements is a model which has confused idea about what is making the prices move, as shown in [figure 20].

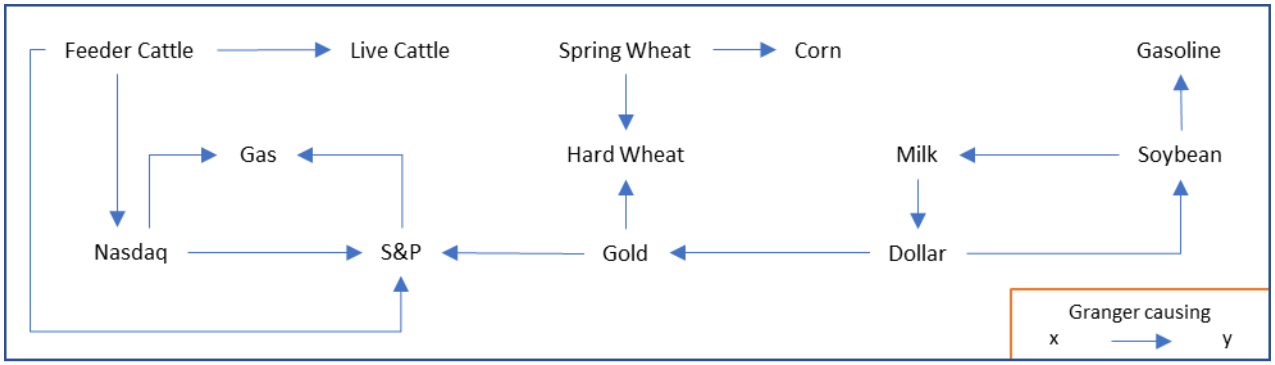

*Figure 20: Granger causality in **VAR tot** - First Difference dataset*

In this figure in fact it is possible to see that the model manages to represent some relationships that are reasonable, as the price for soybean Granger causing the price of class III milk, but many other representation are misrepresented, like the feeder cattle price Granger causing the two financial indexes.
Eventually, VAR models are relatively simple models, where the relationships among variables, although not perfectly accurate, can be interpreted. This relative simplicity, implying a lower fit on the training data, allows for a greater flexibility in the predictions, which will be less constrained by the past.

On the other hand, Neural Networks are highly complex **non-parametric** models, which abandon the goal of explaining the data generation process to instead mimic it. The ultimate goal is the one of prediction accuracy: the choice of the variables and the estimation of the parameters is guided by the performances, and not by significance tests or explanatory power. In doing this, the researcher is not called into question by the models, which automatically computes a very large number of parameters that eventually are not interpretable. As such they might be recreating a situation like in figure 20 or

even worse. The positive side of this complexity, which is the ability to fit the training data perfectly, is actually a downside in prediction: NN risks in fact of being too much constrained by the past data.

The linking chain between these two domains is the ARIMA model, which combines characteristics from both models. In fact, the ARIMA model like the other traditional models has a simple and interpretable structure, but in stark contrast it is not aimed at understanding the underlying DGP. Similarly to Neural Networks, it aims at exploiting the patterns in the series to make forecast.

For what concern the procedures, the main differences refer to the effort and time required to the researcher. The traditional models construction is pretty straightforward, much of the issues that might arise have been covered during the many years of these models implementation and overall, is not particularly time-consuming. The construction of the Neural Network architecture in contrast presents many pitfalls, and require much time for calibration and actual implementation of the model. Consider for instance the issue of regularization to avoid overfitting, an aspect which is completely absent in the traditional econometric due to the different functioning of the models.

The last relevant difference lies in the very structure of the models, and it plays a major role in determining the results of the comparisons. Traditional models are based on a **distribution** (also called **surface**) **plus error** formulation (Efron, 2020): this kind of structure is expected to encounter difficulties when modelling a non-stationary series, as this implies that distribution it is trying to model is not stable over time. On the other hand, Neural Networks are **adaptive** models, which are more suited in this situation thanks to their adjustable parameters (Efron, 2020; Badach, 1980). This difference, as a matter of fact, provides a solid advantage to the LSTM models.

In our comparisons the Neural Networks were able to outperform the traditional methods when applied both datasets, but only in the longer forecast horizon. Much of this success has also to be attributed the ability of the NN to not flatten out after few time steps. The shorter forecast horizon in contrast is dominated by the VARs, that being more rigid, and therefore less sensitive to noise, can still find application in settings like one presented in our analysis.

It is not possible from our analysis to draw definitive conclusions on how LSTM relate to cointegration. In particular, it is not inferable whether the neural network would be able in theory to pick up the long run relationship. What can be said is that in our analysis, it is not reasonable to assume that the LSTMs actually pick up the cointegration relationships, given the fact that they must be trained on pre-processed data where the non-stationarity was removed.

In fact, pre-processing has revealed to be of uttermost importance when training the LSTM. In the absence of an initial de-seasoning and de-trending the neural networks were not able to train effectively, producing forecast completely out of scale. These results showcase the difficulties of LSTM to learn these time properties, even in context where learning should be feasible (Bandara et al., 2019).

One more time, the highlight ultimately is on how the true question should not be which model performs better, but which model is more adequate to fit the problem. Eventually, the ultimate goal of a scientist should be to have a broad range of techniques, to solve the broad range of problems that are presented to him (Breiman, 2004).

6.2 Critical Reflections

The analysis carried out in this thesis presents many strong points and provides many interesting insights from which follow-up research may continue to explore the functioning of Machine Learning models in respect to agricultural time series and traditional econometrics. However, it also displays some limitations.

One important issue to consider when evaluating the performances reported is the different way by which the forecasts are outputted by the models. The LSTM we implemented is a **sequence to sequence** model, meaning that from an input sequence it **directly** inputs a sequence of outputs. In contrast, the traditional models make forecasts in a **dynamic** (or **autoregressive**) way, meaning that each forecast is used as an additional information for the following forecast. There are obliviously consequences from these different behaviours: it can be thought that the traditional models perform worse on the longer forecast horizon as they carry the errors from the previous forecasts all the way to the end; while they perform better on the shorter forecast horizon as they can take into consideration more information.

Another issue that needs to be considered is that the comparison was carried out on non-stationary series. Even though we take care of this element through pre-processing, the impacts on the final performances are not negligible, in particular given the different nature of the two models. As it is also stated in the conclusions, NN have an advantage over traditional models being adaptive models. To have a fair comparison, it would be needed to benchmark our results with a performance comparison on stationary series.

Another limitation concerns the amount of data available. What makes Machine Learning models interesting is their ability of taking into consideration a higher number of variables in respect to traditional models (Efron, 2020; Storm et. al.,2019), as well as an increased number of observations. Since it is suggested that NNs need time series of sufficient length to be trained effectively (Bandara et al., 2019), it would be useful to benchmark our results with the performances on the models on larger datasets. Also a comparison in the opposite direction would be interesting, decreasing the length of the series: this could be done both by truncation, but also by decreasing the frequency of the data. This would also allow us to determine how NN performs when faced with data at different frequency, and therefore noise.

The strengths of this research on the other hand lies in the procedure applied to pre-process the data, referring to the Rolling Window one, as well as the ML model utilized which are both the state of the art for what concerns time series forecasting. Even though there is still plenty to explore to frame in which scenarios LSTM models would be fit, they are certainly promising for forecasting agricultural prices.

The possible ways forward from this research move in two directions. The first is more practical, and concerns the expansion of the comparison on different dataset and with different pre-processing procedures, to confirm or deny the results of this analysis. The second is more theoretical, and it is directed at answering more technical questions such as the relationship between regularization and non-stationarity. It is in fact our opinion that regularization could be considered as a strategy implemented by NN to deal with non-stationarity, by allowing more flexibility in the forecasts in respect to the past.

References

Ahmed, N. K., Atiya, A. F., Gayar, N. E., and El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6), 594-621.

Badach, A. (1980) Adaptive Models in Econometric Forecasting. *IFAC Proceedings Volumes*. Volume 13, Issue 5,Pages 271-277,ISSN 1474-6670, https://doi.org/10.1016/S1474-6670(17)64881-X

Bandara, K., Bergmeir, C., and Smyl, S. (2019) Forecasting Across Time Series Databases using Recurrent Neural Networks on Groups of Similar Series: A Clustering Approach. *Expert Systems with Applications*, 140, 112896. https://doi.org/10.1016/J.ESWA.2019.112896

Efron, B. (2020) Prediction, Estimation, and Attribution. *Journal of the American Statistical Association*. 115:530, 636-655, DOI: 10.1080/01621459.2020.1762613

Engle, R., and Granger, C. (1987). Co-Integration and Error Correction: Representation, Estimation, and Testing. *Econometrica*, 55(2), 251-276. doi:10.2307/1913236

Friedman, J., Hastie, J., and Tibshirani, R. 2009. The elements of statistical learning. *Springer-Verlag New York*. DOI:10.1007/978-0-387-84858-7

Gers, F., Schraudolph, N., and Schmidhuber, J. (2002). Learning Precise Timing with LSTM Recurrent Networks. *Journal of Machine Learning Research*. 3. 115-143. 10.1162/153244303768966139.

Gilliland, M. (2020). The value added by machine learning approaches in forecasting, *International Journal of Forecasting*, Volume 36, Issue 1, Pages 161-166, ISSN 0169-2070, https://doi.org/10.1016/j.ijforecast.2019.04.016.

Goodwin P. (2020). Performance measurement in the M4 Competition: Possible future research, *International Journal of Forecasting*, Volume 36, Issue 1, Pages 189-190, ISSN 0169-2070, https://doi.org/10.1016/j.ijforecast.2019.02.015.

Hippke, M., David, T. J., Mulders, G. D., and Heller, R. 2019. Wōtan: Comprehensive Time-series Detrending in Python. *The Astronomical Journal*, 158(4), 143.

Hyndman R. J. (2020). A brief history of forecasting competitions, *International Journal of Forecasting*, Volume 36, Issue 1, Pages 7-14 ,ISSN 0169-2070 https://doi.org/10.1016/j.ijforecast.2019.03.015.

Hyndman, R. J., Wang E., and Laptev, N. 2015. Large-Scale Unusual Time Series Detection. 2015 IEEE *International Conference on Data Mining Workshop* (ICDMW), Atlantic City, NJ, 2015, pp. 1616-1619, doi: 10.1109/ICDMW.2015.104.

Hyndman R. J., and Koehler, A. B., 2006. Another look at measures of forecast accuracy, *International Journal of Forecasting*, Volume 22, Issue 4, Pages 679-688,ISSN 0169-2070, https://doi.org/10.1016/j.ijforecast.2006.03.001.

Kolassa S. (2020). Why the "best" point forecast depends on the error or accuracy measure, *International Journal of Forecasting*, Volume 36, Issue 1, Pages 208-211, ISSN 0169-2070, https://doi.org/10.1016/j.ijforecast.2019.02.017.

Kubat, M. 2017. An introduction to Machine Learning. Springer International Publishing AG 2017. https://doi-org.ezproxy.library.wur.nl/10.1007/978-3-319-63913-0

Lazzeri, F. 2020. Time Series Forecasting: An Applied Machine Learning Approach. O'REILLY MEDIA. ISBN: 1492049662, 9781492049661

McCulloch, W.S., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5, 115–133 (1943). https://doi.org/10.1007/BF02478259

Mosteller, F., and Tukey, J. 1977. Data Analysis and Regression: A Second Course in Statistics, *Addison-Wesley series in behavioral science, Volume 4854*. Addison-Wesley Publishing Company

Mullainathan, S., Spiess, J. (2017). Machine learning: an applied econometric approach. *Journal of Economic Perspectives*, 31(2), 87-106.

Nielsen, A. 2019. Practical Time Series Analysis: Prediction with Statistics & Machine Learning. O'Reilly Media. ISBN: ISBN: 9781492041658

Rosenblatt, F. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review. 1958.* Volume 65 6, pp. 386-408.
Rumelhart, D., Hinton, G. and Williams, R. Learning representations by back-propagating errors. *Nature* 323, 533–536 (1986). https://doi.org/10.1038/323533a
Sheikh R., Jahirabadkar S. (2018) An Insight into Theory-Guided Climate Data Science—A Literature Review. In: Kolhe M., Trivedi M., Tiwari S., Singh V. (eds) *Advances in Data and Information Sciences. Lecture Notes in Networks and Systems*, vol 38. Springer, Singapore
Smyl S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting, *International Journal of Forecasting*, Volume 36, Issue 1, Pages 75-85, ISSN 0169-2070, https://doi.org/10.1016/j.ijforecast.2019.03.017.
Staudemeyer, R.C., and Morris, E.R. 2019. Understanding LSTM--a tutorial into Long Short-Term Memory Recurrent Neural Networks. arXiv preprint arXiv:1909.09586.
Storm, H., Baylis, K., and Heckelei, T. (2019). Machine learning in agricultural and applied economics. *European Review of Agricultural Economics*. URl: https://doi. org/10.1093/erae/jbz033.
Verbeek, M. (2017) *A Guide to Modern Econometrics*. (5th edition). Croydon, UK: Wiley Custom

Websources:

Barchart (2020) *Historical data Futures* Retrieved from https://www.barchart.com/

CME Group (2020) *Chicago Mercantile Exchange historical data DataMine* Retrieved from https://datamine.cmegroup.com/#

ANNEX A

Table A1: Future contracts specifications for grains at daily frequency

Futures Contract	Symbol	Expiration	Exchange	Period	Number of Observations
Corn	ZCU	September 2020	CBOT	04/01/2000 – 23/07/2020	5174
Soybean	KEU	September 2020	KBCT	04/01/2000 – 23/07/2020	5174
Hard Red Wheat	ZSU	September 2020	CBOT	04/01/2000 – 23/07/2020	5174
Spring Wheat	MWU	September 2020	MGEX	04/01/2000 – 23/07/2020	5174

Table A2: Future contracts specifications for Currencies and Energies at daily frequency

Futures Contract	Symbol	Expiration	Exchange	Period	Number of Observations
U.S. Dollar Index	DXU	September 2020	ICE/US	04/01/2000 – 23/07/2020	5174
Natural Gas	NGZ	December 2020	NYMEX	04/01/2000 – 23/07/2020	5174
Crude Oil WTI	CVL	September 2020	NYMEX	04/01/2000 – 23/07/2020	5174

Table A3: Future contracts specifications for Gold and Financial indexes at daily frequency

Futures Contract	Symbol	Expiration	Exchange	Period	Number of Observations
Gold	GCV	October 2020	COMEX	04/01/2000 – 23/07/2020	5174
Nasdaq 100 E-mini	NQU		CME	04/01/2000 – 23/07/2020	5174
S&P 500 E-mini	ESZ	October 2020	CME	04/01/2000 – 23/07/2020	5174

Table A4: Future contracts specifications for Animal products at daily frequency

Futures Contract	Symbol	Expiration	Exchange	Period	Number of Observations
Milk Class III	GLX	November 2020	COMEX	04/01/2000 – 23/07/2020	5174
Feeder Cattle	DFU	September 2020	CBOT	04/01/2000 – 23/07/2020	5174
Live Cattle	LEV	October 2020	CME	04/01/2000 – 23/07/2020	5174

ANNEX B

Table B1: Performances comparison on the Rolling Window approach on the 7 days forecast horizon

RW – 7 days	**RMSE**	**MAE**	**MAPE**	**RelMAE**
LSTM tot	6.28	5.63	1.68	0.87
LSTM 6	8.03	7.19	2.14	1.11
LSTM 4	11.09	10.26	3.06	1.59
VAR tot	7.69	7.17	2.15	1.11
VAR 6	8.00	7.50	2.25	1.16
VAR 4	6.91	6.42	1.92	0.99
UNI	6.94	6.47	1.94	1.00

Table B2: Performances comparison on the First Difference approach on the 30 days forecast horizon

FD – 30 days	**RMSE**	**MAE**	**MAPE**	**RelMAE**
LSTM tot	11.17	7.96	2.31	1.07
LSTM 6	9.91	6.93	2.02	0.93
LSTM 4	13.60	9.49	2.75	1.27
VAR tot	17.42	14.76	4.31	1.98
VAR 6	16.85	14.19	4.15	1.90
VAR 4	16.91	14.26	4.16	1.91
UNI	10.31	7.46	2.18	1.00

Table B3: Performances comparison on the First Difference approach on the 7 days forecast horizon

FD – 7 days	**RMSE**	**MAE**	**MAPE**	**RelMAE**
LSTM tot	5.85	5.09	1.53	0.69
LSTM 6	5.20	4.44	1.33	0.60
LSTM 4	5.19	4.47	1.34	0.61
VAR tot	4.57	3.83	1.14	0.52
VAR 6	4.61	3.83	1.14	0.52
VAR 4	4.75	3.95	1.17	0.54
UNI	7.86	7.35	2.20	1.00

Table B4: Performances comparison LSTM versus VECMs on the 30 days forecast horizon

Cointegration	**RMSE**	**MAE**	**MAPE**	**RelMAE**
LSTM_rw	10.28	8.30	2.44	1
LSTM_fd	11.17	7.96	2.31	0.96
VEC 6	14.29	11.26	3.28	1.36
VEC 4	15.52	12.44	3.63	1.50

Table B5: Performances comparison LSTM versus VECMs on the 7 days forecast horizon

Cointegration	**RMSE**	**MAE**	**MAPE**	**RelMAE**
LSTM_rw	6.28	5.63	1.68	1.00
LSTM_fd	5.85	5.09	1.53	0.90
VEC 6	6.96	6.37	1.91	1.13
VEC 4	7.18	6.63	1.99	1.18

Table B6: Performances comparison for LSTM across datasets on the 30 days forecast horizon

Time elements	**RMSE**	**MAE**	**MAPE**	**RelMAE**
LSTM_rw	10.28	8.30	2.44	1.00
LSTM_fd	11.17	7.96	2.31	0.96
LSTM_n	249.35	249.17	74.17	30.02
LSTM_t	863.11	862.98	256.76	103.97
LSTM_s	36.45	34.45	10.36	4.15

ANNEX C

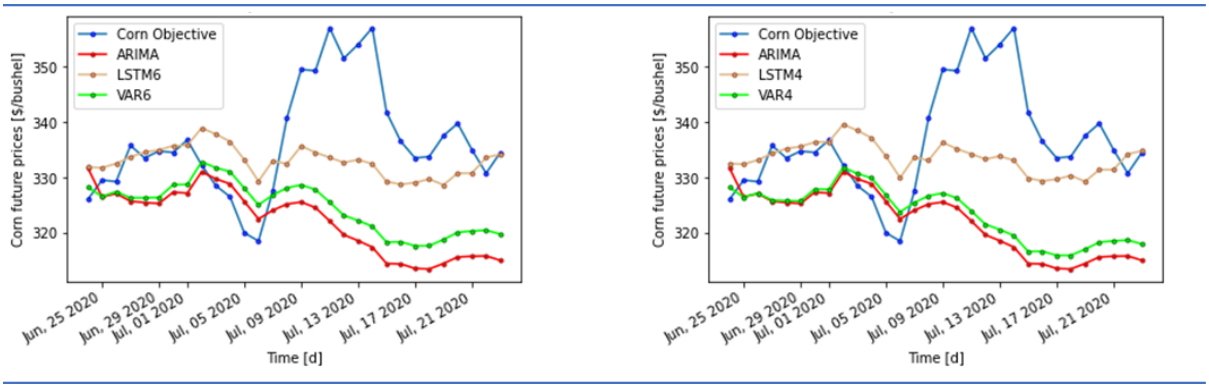

Figure C1: Forecast predictions - LSTM 6 vs VAR 6 & LSTM 4 vs VAR 4– Rolling Window dataset on 30 days forecast

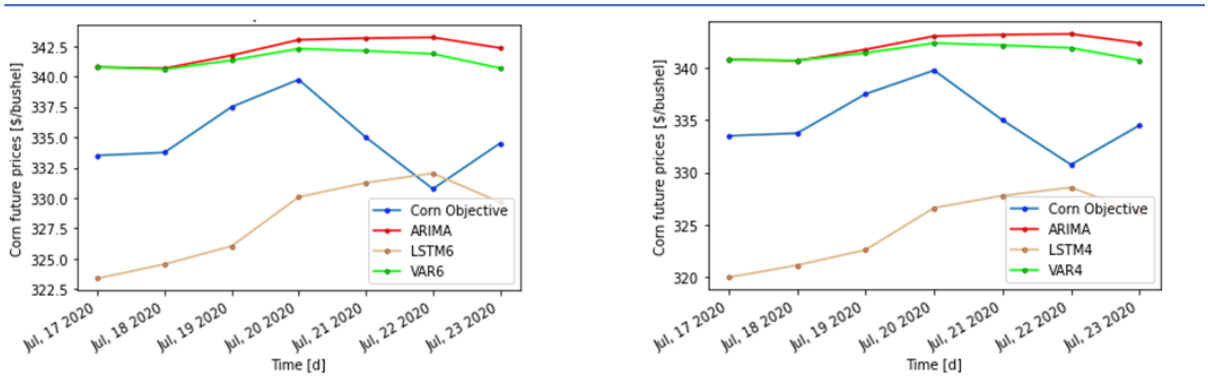

Figure C2: Forecast predictions - LSTM 6 vs VAR 6 & LSTM 4 vs VAR 4– Rolling Window dataset on 7 days forecast

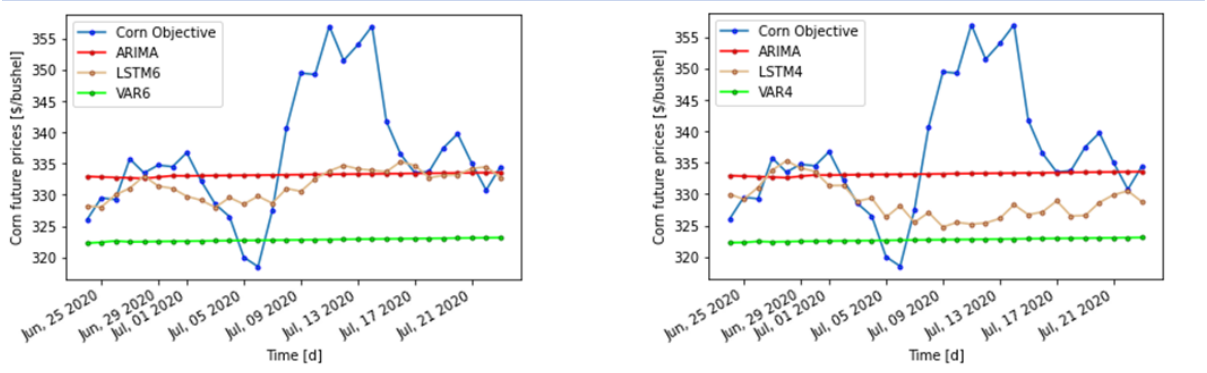

Figure C3: Forecast predictions - LSTM 6 vs VAR 6 & LSTM 4 vs VAR 4– First Difference dataset on 30 days forecast

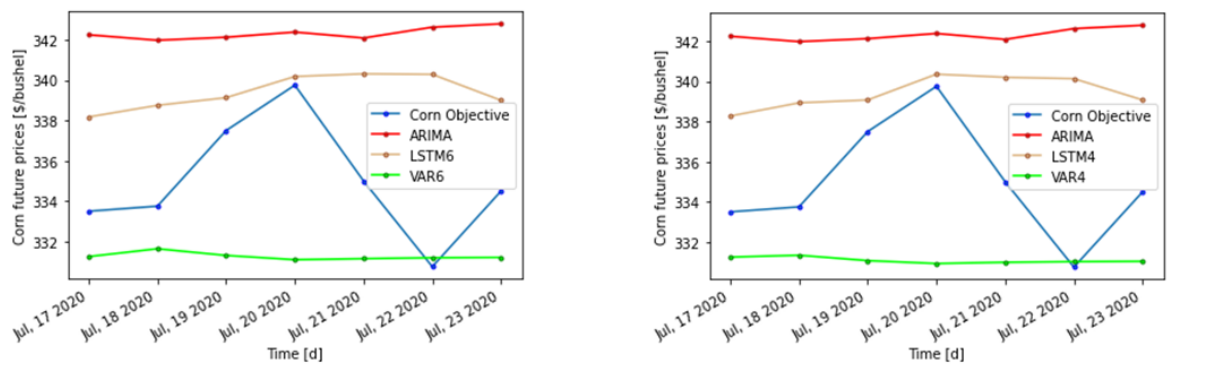

Figure C4: Annex C Figure 4: Forecast predictions - LSTM 6 vs VAR 6 & LSTM 4 vs VAR 4– First Difference dataset