Comparative Study of Vector Autoregression and Recurrent Neural Network Applied to Bitcoin Forecasting

Autor: Zakariae El-Abdelouarti Alouaret
Director: Alfonso Mateos Caballero
Comparative study of Vector Autoregression and Recurrent Neural Network applied to Bitcoin forecasting, © July 2017
9.2 Accuracy measures ........................................ 60
9.3 Errors analysis ............................................. 62
9.4 Results for 2013 onwards .............................. 65

VI DISCUSSION .................................................. 67
10 DISCUSSION .................................................. 69

BIBLIOGRAPHY ................................................... 71
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bitcoin days destroyed chart</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>Cost per transaction chart</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>Cost per transaction percent chart</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>Estimated transaction volume chart</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>Difficulty chart</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>Euro price in USD chart</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>Market price chart</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>Median confirmation time chart</td>
<td>22</td>
</tr>
<tr>
<td>9</td>
<td>Number of transactions chart</td>
<td>23</td>
</tr>
<tr>
<td>10</td>
<td>Output volume chart</td>
<td>23</td>
</tr>
<tr>
<td>11</td>
<td>Standard &amp; poors 500 chart</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>Total bitcoins chart</td>
<td>26</td>
</tr>
<tr>
<td>13</td>
<td>Trade volume chart</td>
<td>27</td>
</tr>
<tr>
<td>14</td>
<td>Transaction Fees chart</td>
<td>28</td>
</tr>
<tr>
<td>15</td>
<td>Transaction fees USD chart</td>
<td>28</td>
</tr>
<tr>
<td>16</td>
<td>Transactions/trade ratio chart</td>
<td>29</td>
</tr>
<tr>
<td>17</td>
<td>Wikipedia trend for bitcoin chart</td>
<td>30</td>
</tr>
<tr>
<td>18</td>
<td>Explained variance of PCA dimensions</td>
<td>34</td>
</tr>
<tr>
<td>19</td>
<td>One layer artificial neural network</td>
<td>41</td>
</tr>
<tr>
<td>20</td>
<td>Two layer artificial neural network</td>
<td>42</td>
</tr>
<tr>
<td>21</td>
<td>Recurrent Neural Network</td>
<td>45</td>
</tr>
<tr>
<td>22</td>
<td>Unfolded RNN for BPTT</td>
<td>46</td>
</tr>
<tr>
<td>23</td>
<td>Weight restrictions in unfolded RNN</td>
<td>46</td>
</tr>
<tr>
<td>24</td>
<td>Time series cross validation example</td>
<td>51</td>
</tr>
<tr>
<td>25</td>
<td>First experiments RNN topology</td>
<td>56</td>
</tr>
<tr>
<td>26</td>
<td>Second experiment RNN topology</td>
<td>56</td>
</tr>
<tr>
<td>27</td>
<td>Third experiment RNN topology</td>
<td>57</td>
</tr>
<tr>
<td>28</td>
<td>Prediction models and MarketPrice true values in separate charts</td>
<td>59</td>
</tr>
<tr>
<td>29</td>
<td>Prediction models and MarketPrice true values in the same chart</td>
<td>60</td>
</tr>
<tr>
<td>30</td>
<td>Histogram of RNN absolute errors and VAR absolute errors</td>
<td>61</td>
</tr>
<tr>
<td>31</td>
<td>Error graphs</td>
<td>63</td>
</tr>
<tr>
<td>32</td>
<td>Normal fitted to prediction errors</td>
<td>64</td>
</tr>
<tr>
<td>33</td>
<td>Prediction models and MarketPrice in separate plots for 2013 onwards</td>
<td>66</td>
</tr>
<tr>
<td>34</td>
<td>Prediction models and MarketPrice for 2013 onwards</td>
<td>66</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Table 1</td>
<td>BTC features selected by Garcia and Schweitzer, 2015</td>
<td>4</td>
</tr>
<tr>
<td>Table 2</td>
<td>BTC features selected by Madan, Saluja, and Zhao, 2014</td>
<td>5</td>
</tr>
<tr>
<td>Table 3</td>
<td>Comparisons between ANN model/s in financial prediction and planning in Bahrammirzaee, 2010</td>
<td>7</td>
</tr>
<tr>
<td>Table 4</td>
<td>16 example sets of variables used by different articles extracted from Atsalakis and Valavanis, 2009</td>
<td>8</td>
</tr>
<tr>
<td>Table 5</td>
<td>Reviewed papers classified by machine learning technique by Krollner, Vanstone, and Finnie, 2010</td>
<td>9</td>
</tr>
<tr>
<td>Table 6</td>
<td>Reviewed papers classified by forecasting timeframe by Krollner, Vanstone, and Finnie, 2010</td>
<td>9</td>
</tr>
<tr>
<td>Table 7</td>
<td>Reviewed papers classified by input variables by Krollner, Vanstone, and Finnie, 2010</td>
<td>9</td>
</tr>
<tr>
<td>Table 8</td>
<td>Reviewed papers classified by machine learning technique by Krollner, Vanstone, and Finnie, 2010</td>
<td>10</td>
</tr>
<tr>
<td>Table 9</td>
<td>Comparison between ANNs and traditional techniques by Beiranvand, Abu Bakar, and Othman, 2012</td>
<td>11</td>
</tr>
<tr>
<td>Table 10</td>
<td>Comparison between ANNs and other methods of portfolio management in Beiranvand, Abu Bakar, and Othman, 2012</td>
<td>11</td>
</tr>
<tr>
<td>Table 11</td>
<td>Comparison between ANNs and other methods of forecasting and planning in Beiranvand, Abu Bakar, and Othman, 2012</td>
<td>12</td>
</tr>
<tr>
<td>Table 12</td>
<td>Statistical values for Cost per transaction</td>
<td>17</td>
</tr>
<tr>
<td>Table 13</td>
<td>Statistical values for Cost per transaction percent</td>
<td>18</td>
</tr>
<tr>
<td>Table 14</td>
<td>Statistical values for Market price</td>
<td>21</td>
</tr>
<tr>
<td>Table 15</td>
<td>Statistical values for Standard &amp; poors 500</td>
<td>26</td>
</tr>
<tr>
<td>Table 16</td>
<td>Statistical values for Transaction fees USD</td>
<td>29</td>
</tr>
<tr>
<td>Table 17</td>
<td>Statistical values for Wikipedia trend for bitcoin</td>
<td>30</td>
</tr>
<tr>
<td>Table 18</td>
<td>Forecast accuracy measures</td>
<td>60</td>
</tr>
<tr>
<td>Table 19</td>
<td>Normality test results</td>
<td>64</td>
</tr>
<tr>
<td>Table 20</td>
<td>Forecast accuracy measures for 2013 onwards</td>
<td>66</td>
</tr>
</tbody>
</table>
LISTINGS

<table>
<thead>
<tr>
<th>Listing</th>
<th>Code Type</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Listing 1</td>
<td>TSCV pseudo code</td>
<td>51</td>
</tr>
<tr>
<td>Listing 2</td>
<td>WFSS pseudo code</td>
<td>52</td>
</tr>
</tbody>
</table>
ACRONYMS

AI    Artificial Intelligence
ANN   Artificial Neural Network
APN   Arrayed probabilistic network
ARIMA Autoregressive integrated moving average
ATR   Average true range
BFGS  Broiden-Fletcher-Goldfarb-Shanno
BPNN  Backpropagation neural networks
BTC   Bitcoin
CBR   Case-based reasoning
CFLANN Cascaded functional link ANN
CHF   Swiss Franc
CNY   Chinese Renminbi
DA    Discriminant Analysis
DRPNN Dynamic ridge polynomial neural network
ECB   European Central Bank
EMA   Exponential moving average
FA    Factor analysis
FLANN Functional link ANN
FSI   Financial Stability Institute
FSS   Feature Subset Selection
FTSE  Financial Times Stock Exchange
GA    Genetic Algorithms
GLM   General Linear Models
GRU   Gated Recurrent Unit
HM    Hierarchical Model
LDA   Linear Discriminant Analysis
LMS  Least mean square
LM   Levenberg-Marquardt
LR   Logistic regression
LS-SVM Least Squares Support Vector Machine
LSTM Long Short-Term Memory
MACD moving average convergence/divergence
MAE Mean Absolute Error
MDA Multivariable Discriminant Analysis
MLP Multilayer Perceptron
MSPE Mean Squared Prediction Error
NASDAQ National Association of Securities Dealers Automated Quotations
NN  Neural network
NYSE New York Stock Exchange
OLS Ordinary least square
PCA Principal Component Analysis
PNN Probabilistic Neural Network
PSO Particle Swarm Optimization
RMSE Root Mean Square Error
RNN Recurrent neural networks
ROC Rate of change
RPROP Resilient propagation
RSI Relative strength index
RS  Rough Sets
SMA Simple moving average
SP500 Standard & Poor’s 500
SVM Support Vector Machine
TXN Transaction/s
USD United States Dollar
VARIMA  Vector Autoregressive Integrated Moving Average
VARMA  Vector Autoregressive Moving-Average Models
VAR   Vector Autoregression
WFSS  Wrapper Feature Subset Selection
AIC   Akaike’s Information Criteria
BIC   Bayesian Information Criteria
FPE   Final Prediction Error
HQC   Hannan–Quinn Information Criterion
Part I

INTRODUCTION

Where we introduce the problem we want to solve, explore the state-of-the-art of related articles and get an overview of how to tackle the solution.
INTRODUCTION

In this thesis we are exploring the prediction of next day Bitcoin (BTC) price through the usage of Recurrent Neural Networks (RNN). Our aim is, by using state-of-the-art techniques, to predict the price of BTC with higher accuracy than the previous works in the literature. This thesis uses up to 27 time series, spanning from 03/01/2009 to 28/04/2016, with a granularity of 1 data point per day.

The thesis is organized as follows: Chapter 1 describes the aim of the thesis and the state-of-the-art of BTC prediction and financial forecasting, in Chapter 2 and Chapter 3 we describe the variables gathered and how they where selected, then Chapter 4, Chapter 5 and Chapter 6 describe RNN and its suitability to time-series forecasting, Chapter 7 shows the tools implemented to adapt RNN to the particular problem of BTC price prediction, Chapter 8 and Chapter 9 reviews the experiments executed and the data produced as output and, finally, Chapter 10 discusses the results obtained where we draw some conclusions and then propose some improvements to the work done in this thesis.

There are recent works predicting the price of BTC, or the fluctuations of its price. Madan, Saluja, and Zhao, 2014 forecast the sign of the price of BTC for periods of 24h (phase one) and 10 minutes and 10 seconds (phase two), obtaining an accuracy of 98.7% for phase one and 50% – 55% for phase two. In phase one they used binomial general linear models (GLM), Support Vector Machine (SVM) and Random Forest. Binomial GLM shows higher precision and accuracy that the other two algorithms. For phase two they used only Binomial GLM and Random Forest. When comparing 10 second and 10 minute prediction with Binomial GLM with 10 minute prediction for Random Forest, it turns out that Random Forest shows higher accuracy and precision than GLM in both 10 seconds and 10 minutes interval prediction. Garcia and Schweitzer, 2015 provide a consistent approach that integrates various data-sources in the design of algorithmic traders. They applied multidimensional model of vector auto-regression to design different types of trading algorithms. The analysis performed reveals that increases in opinion polarization and exchange volume precede rising BTC prices, and that emotional valence precedes opinion polarization and rising exchange volumes. They chose a number of BTC related variables based on the work of other researchers that can be seen in Table 1.

When it comes to variables used for predicting the BTC price, Madan, Saluja, and Zhao, 2014 uses for phase one 16 features (shown in Ta-
Table 1: BTC features selected by Garcia and Schweitzer, 2015

<table>
<thead>
<tr>
<th>NAME OF VARIABLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(t) )</td>
<td>Price</td>
</tr>
<tr>
<td>( \text{Ret}(t) )</td>
<td>Return</td>
</tr>
<tr>
<td>( \text{FXVol}(t) )</td>
<td>Trading volume</td>
</tr>
<tr>
<td>( BCTra(t) )</td>
<td>Transaction volume in the Blockchain</td>
</tr>
<tr>
<td>( \text{Dwn}(t) )</td>
<td>Amount of downloads of the most important BTC client</td>
</tr>
<tr>
<td>( S(t) )</td>
<td>Level of search volume in Google for the term “bitcoin”</td>
</tr>
<tr>
<td>( \text{TN}(t) )</td>
<td>Amount of tweets containing BTC-related terms</td>
</tr>
<tr>
<td>( \text{TVal}(t) )</td>
<td>Emotional valence</td>
</tr>
<tr>
<td>( \text{TPol}(t) )</td>
<td>Opinion polarization expressed in the tweets</td>
</tr>
</tbody>
</table>

relating to the BTC price and payment network over the course of five years, recorded daily. The features were selected manually based on research of the significance to the problem of forecasting. For the phase two the data consisted of 10-second and 10-minute interval BTC price. There is no particular feature selection technique used by Madan, Saluja, and Zhao, 2014 given that they manually selected the features to use to perform the prediction.

On the other hand Georgoula et al., 2015 used time-series analysis to study the relationship between BTC prices and fundamental economic variables, technological factors and measurements of collective mood derived from Twitter feeds. They conclude that BTC price is positively associated with the number of BTCs in circulation and negatively associated with the Standard and Poor’s 500 stock market index.

They used regression over time-series to forecast the BTC price and SVM to classify the tweet feeds sentiment, suggesting the next conclusions.

- \( \log\text{wiki} \) (representing the degree of public recognition or interest in BTCs) and \( \log\text{hash} \) (measuring the mining difficulty) have a positive impact on BTC price.
- The exchange rate between the USD and the Euro relationship is negative.
- The sentiment ratio of Twitter users, and the number of Wikipedia search queries positively affects the price of BTCs
- The stock of BTCs has a positive long-run impact on its price
- The Standard and Poor’s 500 index was found to have a negative impact on BTC prices in the long run

Kristoufek, 2015 proposes another approach to find potential drivers of BTC price by means of wavelet coherence analysis of different variables.
Table 2: BTC features selected by Madan, Saluja, and Zhao, 2014

<table>
<thead>
<tr>
<th>Feature</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Confirmation Time</td>
<td>Average time to accept transaction in block</td>
</tr>
<tr>
<td>Block size</td>
<td>Average block size in MB</td>
</tr>
<tr>
<td>Cost per transaction percent</td>
<td>Miners revenue divided by the number of transactions</td>
</tr>
<tr>
<td>Difficulty</td>
<td>How difficult it is to find a new block</td>
</tr>
<tr>
<td>Estimated Transaction Volume</td>
<td>Total output volume without change from value</td>
</tr>
<tr>
<td>Hash Rate</td>
<td>BTC network giga hashes per second</td>
</tr>
<tr>
<td>Market Capitalization</td>
<td>Number of BTCs in circulation * the market price</td>
</tr>
<tr>
<td>Miners Revenue</td>
<td>(number of BTC mined/day * market price) + transaction fees</td>
</tr>
<tr>
<td>Number of Orphaned Blocks</td>
<td>Number of blocks mined/day not off blockchain</td>
</tr>
<tr>
<td>Number of Transactions (TXN) per block</td>
<td>Average number of transactions per block</td>
</tr>
<tr>
<td>Number of TXN</td>
<td>Total number of unique BTC transactions per day</td>
</tr>
<tr>
<td>Number of unique addresses</td>
<td>Number of unique BTC addresses used per day</td>
</tr>
<tr>
<td>Total BTCs</td>
<td>Historical total Number of BTCs mined</td>
</tr>
<tr>
<td>TXN Fees Total</td>
<td>BTC value of transactions fees miners earn/-day</td>
</tr>
<tr>
<td>Trade Volume</td>
<td>USD trade volume from the top exchanges</td>
</tr>
<tr>
<td>Transactions to trade ratio</td>
<td>Relationship of BTC transaction volume and USD volume</td>
</tr>
</tbody>
</table>

The conclusions after the analysis are:

- The Trade Exchange Ratio has a strong, but not statistically significant at the 5% level, relationship at high scales. The price and the ratio are negatively correlated in the long term.

- BTC appreciates in the long run if it is used more for trade, i.e., non-exchange transactions, and the increasing price boosts the exchange transactions in the short run.

- The money supply works as a standard supply, so that its increase leads to a price decrease.

- For the trade transactions, the increasing usage of bitcoins in real transactions leads to an appreciation of the bitcoin in the long run.

- For the trade volume the relationship changes in time to offer any strong conclusion.

- Both measures of the mining difficulty (Hash and Difficulty) are positively correlated with the price in the long run.
• In the short run, BTC price and both Hash and Difficulty relationship becomes negative.

• The interest (Google and Wikipedia) in BTC appears to have an asymmetric effect during the bubble formation and its bursting. During the bubble formation, interest boosts the prices further, and during the bursting, it pushes them lower.

• For the FSI, apart from the Cypriot crisis, there are no longer-term time intervals during which the correlations are both statistically significant and reliable.

• Gold price apparently has no relationship with BTC price

• Although the USD and Chinese Renminbi (CNY) markets are tightly connected, there is no clear evidence that the Chinese market influences the USD market.

We can also focus our attention in the financial prediction methods in general, which can be useful to the BTC price prediction.

Bahrammirzaee, 2010 surveyed several forecasting and classification techniques and compared to three important Artificial Intelligence (AI) techniques, namely, artificial neural networks, expert systems and hybrid intelligence systems. They didn’t focus on BTC forecasting but predict another variables shown in Table 3.

I want to include the next quote from the article because describes very well how NNs behave in a financial scenario: “Regarding NNs, the empirical results of these comparative studies indicate that the success of NNs in financial domain, especially in financial prediction and planning, is very encouraging. (However, while NNs often outperform the more traditional and statistical approaches but this is not always the case. There are some studies in which other traditional methods, or intelligent approach outperforms NNs.) This success is due to some unique characteristics of NNs in financial market like their numeric nature, no requirement to any data distribution assumptions (for inputs) and model estimators and finally, their capability to update the data. Despite this success, this paper could not conclude that NNs are very accurate techniques in financial market because at the first, among these studies, BPNN is the most popular NN training technique. However, BPNN suffers from the potential problems. One of the problems of BPNN is local minimum convergence. Because the gradient search process proceeds in a point-to-point mode and is intrinsically local in scope, convergence to local rather than global minima is a very possibility. Also BP training method is very slow and takes too much time to converge. Besides, it can over fit the training data. Secondly, it is difficult, if not impossible, to determine the proper size and structure of a neural net to solve a given problem. Therefore, the architectural parameters are generally designed by researchers and via trial and errors and since these parameters determine outputs of NNs, their accuracy and performance are subject to numerous errors. Also sometimes NNs are incapable to recognize patterns and establish
relevant relationships between various factors, which are important reasons to reduce their performance. Finally, NNs learn based on past occurrences, which may not be repeated, especially in financial markets and in current financial crisis.”

Another survey, which include 100 studies applied to stock market forecasting with a focus on neural networks and neuro-fuzzy techniques was made by Atsalakis and Valavanis, 2009. This paper enumerates the modeling techniques used to forecast, the stock markets whose information are used (not very useful for this thesis because we are interested in BTC where the information source is clear), the input variables, which we summarize in Table 4, whether they perform data processing, sample sizes, network layers, membership functions, validation set and training method. They also show the methods used to compare with for each article without giving the conclusions made in each article. Another very interesting information are the model performance measures where the measures are classified as statistical and non-statistical measures: “Statistical measures include the root mean square error (RMSE), the mean absolute error (MAE) and the mean squared prediction error (MSPE), statistical indicators like the autocorrelation, the correlation coefficient, the mean absolute deviation, the squared correlation and the standard deviation. [...] Non-statistical performance measures include measures that are related with the economical side of the forecast. The
most common used performance measure is the so called Hit Rate that measures the percentage of correct predictions of the model. Another two measures that deal with the profitability of the model are the annual rate of return and the average annual profit of the model.”

Table 4: 16 example sets of variables used by different articles extracted from Atsalakis and Valavanis, 2009

<table>
<thead>
<tr>
<th>VARIABLE SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eight input variables (NASDAQ and 7 indexes)</td>
</tr>
<tr>
<td>Opening, lowest, highest and closing index price</td>
</tr>
<tr>
<td>Index values of five previous days</td>
</tr>
<tr>
<td>Ten variables (five Technical Analysis factors, price of last 5 days)</td>
</tr>
<tr>
<td>Eight fundamental analysis indicators</td>
</tr>
<tr>
<td>Change of stock prices in last two sessions</td>
</tr>
<tr>
<td>Three technical analysis indexes</td>
</tr>
<tr>
<td>Volume, opening, lowest, highest and closing index price</td>
</tr>
<tr>
<td>Fifteen technical analysis factors</td>
</tr>
<tr>
<td>Changes in stock prices</td>
</tr>
<tr>
<td>Volume ratio, relative strength index, rate of change, slow %D</td>
</tr>
<tr>
<td>Daily closing value</td>
</tr>
<tr>
<td>Technical Analysis variables</td>
</tr>
<tr>
<td>FTSE index, exchange rate, futures market etc</td>
</tr>
<tr>
<td>Beta, cap, b/m</td>
</tr>
<tr>
<td>Seven economical indicators</td>
</tr>
</tbody>
</table>

As a conclusion, they mention that: “The observation is that neural networks and neuro-fuzzy models are suitable for stock market forecasting. Experiments demonstrate that soft computing techniques outperform conventional models in most cases. They return better results as trading systems and higher forecasting accuracy. However, difficulties arise when defining the structure of the model (the hidden layers the neurons etc.). For the time being, the structure of the model is a matter of trial and error procedures.”

Following the surveying articles is the one introduced by Carpinheiro et al., 2012 which gives a comparison of three forecasting models: a Multilayer Perceptron (MLP), a SVN, and a Hierarchical Model (HM). After performing six experiments running through time series of the stock market fund on Bank of Brasil, they concluded that the performance of HM is better than SVM and MLP. As a possible explanation, they suggest that the superiority of HM can be due to its hierarchical architecture.

Krollner, Vanstone, and Finnie, 2010 performs a meta-survey, with 46 forecasting studies. They divide the papers into Artificial Neural Network (ANN) based and evolutionary and optimisation based techniques. Related to this division they show the amount of studies using each technique, which we reproduce in Table 5.

When it comes to the prediction periods, they categorised it in one day, one week, and one month ahead predictions, shown in Table 6
Table 5: Reviewed papers classified by machine learning technique by Krollner, Vanstone, and Finnie, 2010

<table>
<thead>
<tr>
<th>TECHNOLOGY</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN based</td>
<td>21</td>
</tr>
<tr>
<td>Evolutionary and optimisation</td>
<td>4</td>
</tr>
<tr>
<td>Multiple / hybrid</td>
<td>15</td>
</tr>
<tr>
<td>Other</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 6: Reviewed papers classified by forecasting time-frame by Krollner, Vanstone, and Finnie, 2010

<table>
<thead>
<tr>
<th>TIME-FRAME</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day</td>
<td>31</td>
</tr>
<tr>
<td>Week</td>
<td>3</td>
</tr>
<tr>
<td>Month</td>
<td>3</td>
</tr>
<tr>
<td>Multiple / Other</td>
<td>9</td>
</tr>
</tbody>
</table>

The 75% of papers reviewed use some sort of lagged index data as input variables. Most commonly used parameters are daily opening, high, low and close prices. Also several technical indicators are used, been the more important: Simple moving average (SMA), Exponential moving average (EMA), Relative strength index (RSI), Rate of change (ROC), Moving average convergence/divergence (MACD), William’s oscillator and Average true range (ATR). Table 7 displays the number of studies using the different input variables.

Table 7: Reviewed papers classified by input variables by Krollner, Vanstone, and Finnie, 2010

<table>
<thead>
<tr>
<th>INPUT</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lagged Index Data</td>
<td>35</td>
</tr>
<tr>
<td>Trading volume</td>
<td>4</td>
</tr>
<tr>
<td>Technical Indicators</td>
<td>13</td>
</tr>
<tr>
<td>Oil Price</td>
<td>4</td>
</tr>
<tr>
<td>SP500 / NASDAQ / Dow Jones (non US studies)</td>
<td>4</td>
</tr>
<tr>
<td>Unemployment Rate</td>
<td>1</td>
</tr>
<tr>
<td>Money Supply</td>
<td>3</td>
</tr>
<tr>
<td>Exchange Rates</td>
<td>3</td>
</tr>
<tr>
<td>Gold Price</td>
<td>3</td>
</tr>
<tr>
<td>Short &amp; Long Term Interest Rates</td>
<td>6</td>
</tr>
<tr>
<td>Others</td>
<td>10</td>
</tr>
</tbody>
</table>

The evaluation methods are diverse, and the categories used are buy & hold, random walk, statistical techniques, other machine learning techniques, and no benchmark model. The relationship between categories and the studies is reproduced in Table 8.
An interesting quote made by Krollner, Vanstone, and Finnie, 2010 is: “Over 80% of the papers report that their model outperformed the benchmark model. However, most analysed studies do not consider real world constraints like trading costs and slippage.”

As one of the conclusions of the review they state that ANN are the dominant techniques for the field of financial forecasting.

Beiranvand, Abu Bakar, and Othman, 2012 review ANNs compared with meta-heuristics, such as genetic algorithms (GA) and particle swarm optimization (PSO), and compare their performance in financial domain, dividing, as previous papers did, in three subdomains: portfolio management, financial forecasting and planning, and credit evaluation. ANNs prove to be a good method when compared to traditional techniques, and to note that, they say: “In 2008, two Artificial NNs were developed by Angelini et al.. One of them was based on a regular feed-forward model and the other one had specific purpose architecture. The performance of the system was assessed using real world data, acquired from small corporations in [Italy]. They proved that if data analysis and pre-processing are done carefully, and suitable training is carried out, then Artificial NNs will demonstrate a powerful performance in estimating and learning the default direction of a borrower.”, as shown in Table 9.

When applied to portfolio management, ANNs has better performance, and is particularly true for BPNNs. Table 10 shows a brief comparison made by Beiranvand, Abu Bakar, and Othman, 2012. The same goes for forecasting and planning, where Table 11 shows the comparison with other methods.

As for GAs and PSOs there aren’t as much evidence of their suitability for financial domain apart from this quote: “Mahfoud and Mani address the general problem of predicting future performances of individual stocks. They compare a genetic-algorithm-based system to an established neural network system on roughly 5000 stock-prediction experiments. According to their experiments, both systems significantly outperform the B&H strategy”

Their conclusion is that “the performance and accuracy of the above mentioned artificial intelligence techniques are considerably higher, compared to the traditional statistical techniques, particularly in nonlinear models. Nevertheless, this superiority is not true in all cases.”
Table 9: Comparison between ANNs and traditional techniques by Beiranvand, Abu Bakar, and Othman, 2012

<table>
<thead>
<tr>
<th>Domain</th>
<th>Author(s)</th>
<th>Approaches compared</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credit evaluation</td>
<td>Malhotra and Malhotra, 2003</td>
<td>Back-propagation Neural Network (BPNN) compared with MDA</td>
<td>ANN has a better performance</td>
</tr>
<tr>
<td>Credit scoring</td>
<td>Abdou, Pointon, and El-Masry, 2008</td>
<td>PNN and MLP compared with DA and LR</td>
<td>PNN and MLP have a better performance</td>
</tr>
<tr>
<td>Credit measurement</td>
<td>Bennell et al., 2006</td>
<td>BPNN with OPM</td>
<td>ANN has a better performance</td>
</tr>
<tr>
<td>Bond scoring</td>
<td>Kaplan and Urwitz, 1979</td>
<td>BPNN compared with LR</td>
<td>ANN has a better performance</td>
</tr>
<tr>
<td>Credit scoring</td>
<td>Baesens et al., 2003</td>
<td>BPNN compared with MDA, LR, and LS-SVMs</td>
<td>LS-SVMs and BPNN have better performance</td>
</tr>
</tbody>
</table>

Table 10: Comparison between ANNs and other methods of portfolio management in Beiranvand, Abu Bakar, and Othman, 2012

<table>
<thead>
<tr>
<th>Domain</th>
<th>Author(s)</th>
<th>Approaches compared</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mortgages choice decision</td>
<td>Hawley, Johnson, and Raina, 1990</td>
<td>NN compared with probit</td>
<td>Artificial NN has a better performance</td>
</tr>
<tr>
<td>Portfolio optimization</td>
<td>Wong and Selvi, 1998</td>
<td>B&amp;H strategy</td>
<td>Artificial NN has a better performance</td>
</tr>
<tr>
<td>Portfolio optimization</td>
<td>Holsapple, Tam, and Whinston, 1988</td>
<td>Back-Propagation NN with mean-variance model</td>
<td>Artificial NN has a better performance</td>
</tr>
</tbody>
</table>

Verikas et al., 2010 introduce another techniques in his review, namely hybrid and ensemble-based soft computing techniques applied to bankruptcy. The focus is on techniques and not on obtained results, given that almost all authors demonstrate that the technique they propose outperforms some other methods chosen for the comparison. They do some observations after the study: “fair comparison of results obtained in the different studies is hardly possible” due to the different data-set used, and different sizes of these data-sets. “Nonetheless the difficulty in comparing the reviewed techniques, one can make a general observation that ensembles, when properly designed, are more accurate than the other techniques. This is expected, since an ensemble integrates several predictors. In a successful ensemble design, a tradeoff between the ensemble accuracy and the diversity of ensemble members is achieved. [...] However, transparency of ensemble-based techniques is rather limited, when compared to RS or IF-THEN rules- based approaches.”
When referencing to feature set used for predicting bankruptcy, they mention: “Different studies indicate that the bankruptcy prediction accuracy can be increased substantially by including non-financial features into the modeling process and a trend is on using non-financial features, for example macroeconomic indicators and qualitative variables, in addition to financial ratios. […]. Genetic algorithms and RS are the two most popular approaches to feature selection in hybrid and ensemble-based techniques for bankruptcy prediction.”

As a major downside of soft computing techniques their conclusion is: “The non-linear nature of hybrid and ensemble-based models and the lack of a widely accepted procedures for designing such models are major factors contributing to pitfalls in applications of these technologies. Model building, model selection and comparison are the designing steps where the most common pitfalls occur due to small sample sizes, model over-fitting or under-fitting, the sensitivity of solutions to initial conditions. The problem of selecting a model of appropriate complexity is often forgotten when developing soft computing techniques for bankruptcy prediction.”

Table 11: Comparison between ANNs and other methods of forecasting and planning in Beiranvand, Abu Bakar, and Othman, 2012

<table>
<thead>
<tr>
<th>Domain</th>
<th>Author(s)</th>
<th>Approaches Compared</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selecting financial variables</td>
<td>Markowitz, 1959</td>
<td>ANN models that uses constant relevant variables compared to LR, BSH strategy, and random walk</td>
<td>Artificial NN has a better performance</td>
</tr>
<tr>
<td>Exchange rates prediction</td>
<td>Lam, 2004</td>
<td>ANN compared to linear autoregressive and random walk models</td>
<td>PNN and MLP have a better performance</td>
</tr>
<tr>
<td>Post-bankruptcy resolutions</td>
<td>Atsalakis and Valavanis, 2009</td>
<td>5 various models of ANN</td>
<td>ANN has a better performance</td>
</tr>
<tr>
<td>Bankruptcy forecasting</td>
<td>Mochón et al., 2008</td>
<td>ANN with MDA</td>
<td>ANN has a better performance</td>
</tr>
<tr>
<td>Classification of financial data</td>
<td>Sivanandam and Deepa, 2007</td>
<td>ANN with LDA and LR</td>
<td>ANN has a better performance</td>
</tr>
<tr>
<td>Asset value prediction</td>
<td>Prasad, 2008</td>
<td>BPNN with regression models</td>
<td>ANN has a better performance</td>
</tr>
<tr>
<td>Consumer price index forecasting</td>
<td>Atiya, 2001</td>
<td>ANN with random walk model</td>
<td>ANN has a better performance</td>
</tr>
</tbody>
</table>
Part II

VARIABLES TO PREDICT THE BTC PRICE

Where we explain the variables selected to study, their interest for this project, and how we selected them.
STATISTICAL VARIABLE ANALYSIS

This section describes the basic properties of each feature we are using in this particular thesis. Among them we have the count of all variables, mean, standard deviation, min value, max value, quartiles. We also add the histogram to have an idea of how the distribution of the variable works and the boxplot that can help us determine if there are outliers in the data-set.

There are 16 variables descriptions, two of them are sets of variables, namely, bitcoins days destroyed, which groups two individual variables, and Standard & Poors 500, which groups another two individual variables. Therefore in the data-set, every individual member of said sets, would count as an individual feature for prediction purposes.

The data-sets were obtained from a variety of sources, but mostly from blockchain.info, excluding Euro price in USD that was obtained from the European Central Bank web site, and the Standard & Poors 500 that was obtained from Yahoo! Finance.

While all the data-set have daily data points for their values, they have different spanning dates, been Euro price in USD and Standard & Poors 500 that span the most, because they have more history than Bitcoin. After analyzing the date span of each variable, we conclude that the intersection of all the sets is in the range between 2009-01-03 and 2016-04-28.

BITCOIN DAYS DESTROYED

Bitcoin days destroyed is a weighted measure of aggregate economic activity, placing value on transacted coins in proportion to the time they have spent idle on the Bitcoin blockchain. For any given transaction, Bitcoin days destroyed is calculated by multiplying its estimated transaction value by the number of days since the coins within the transaction were last spent. It is a useful proxy for measuring growth in real value transacted on the Bitcoin blockchain over time, since it controls for rapid movement of coins between wallets (potentially owned by just one entity). One integer data point each day, at 18:15:05, spanning from 03/01/2009 to 28/04/2016.

To better understand this variable we include a quote extracted from Bitcoin Beta | Stack Exchange:

“The idea of "bitcoin days destroyed" came about because it was realized that total transaction volume per day might be an inappropriate measure of the level of economic activity in Bitcoin. After all, someone could be sending
the same money back and forth between their own addresses repeatedly. If you sent the same 50 BTC back and forth 20 times, it would look like 1000 BTC worth of activity, while in fact it represents almost nothing in terms of real transaction volume.

With “bitcoin days destroyed”, the idea is instead to give more weight to coins which haven’t been spent in a while. To do this, you multiply the amount of each transaction by the number of days since those coins were last spent. So, 1 bitcoin that hasn’t been spent in 100 days (1 bitcoin * 100 days) counts as much as 100 bitcoins that were just spent yesterday (100 bitcoins * 1 day). Because you can think of these “bitcoin days” as building up over time until a transaction actually occurs, the actual measure is called “bitcoin days destroyed”. This is believed to give a better indication of how much real economic activity is occurring on the bitcoin network.

So how well does it work? Well, it’s still not perfect, because the other day I moved some coins out of a wallet they’ve been in for several months without spending them or giving them away. And some genuine businesses have very rapid turnover in bitcoins, so they’re not being measured well by this method. But it does do a good job of filtering out the “noise” of bitcoins that are just “bouncing around” without really going anywhere. The graph of overall bitcoin days destroyed is believed to show that the genuine level of activity in the Bitcoin economy is continually increasing—it’s not just one person experimenting by rapidly sending the same coins back and forth, flooding the network with meaningless chatter. Looks pretty good, hey?

In Figure 1 we show data over time for Bitcoin days destroyed.

Figure 1: Bitcoin days destroyed chart

**Cost per Transaction**

This variable shows miners revenue (in USD) divided by the number of transactions. Miners revenue are a result of the transaction fees plus the reward for block discovery. The block discovery reward is
fixed by the community, and is up to the miner to accept any transaction regardless of the fee in the transaction. A miner can choose not to add any transaction at all to the block, or add transactions with a 0 BTC fee.

Figure 2 shows that in the end of 2010 the Cost per transactions started to increase and had a local maximum followed by another local maximum in mid 2011. This two peaks took place in a low price of Bitcoin scenario, hence probably the blocks were introduced with a small amount of transactions, and with a subtle increase in the Bitcoin price the ratio would increase.

![Figure 2: Cost per transaction chart](image)

On the contrary, the two peaks of 2014 are in the highest price of the Bitcoin history, where the number of transactions would make little effect. Price in that period fluctuates between 700 USD and 1100 USD approximately.

Table 12 shows us that the median revenue is 6.199964 USD and the mean is 9.731184 USD. This information is later increased in Section 2.3.

<table>
<thead>
<tr>
<th>MEASURE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>2673</td>
</tr>
<tr>
<td>mean</td>
<td>9.731184</td>
</tr>
<tr>
<td>std</td>
<td>13.280920</td>
</tr>
<tr>
<td>min</td>
<td>0</td>
</tr>
<tr>
<td>25%</td>
<td>1.295504</td>
</tr>
<tr>
<td>50%</td>
<td>6.199964</td>
</tr>
<tr>
<td>75%</td>
<td>10.413847</td>
</tr>
<tr>
<td>max</td>
<td>92.202095</td>
</tr>
</tbody>
</table>
COST PER TRANSACTION PERCENT

This variable shows miners revenue as percentage of the transaction volume. Figure 3 shows is that the miners revenue has been near to zero all the time, excluding few exceptions.

Figure 3: Cost per transaction percent chart

As of 2016, miners are more concerned about the reward obtained by discovering new blocks in the Bitcoin network. In the future the median percentage is 3.297002 (see Table 13) will probably increase, because the sole income for miners will be transaction fees.

Table 13: Statistical values for Cost per transaction percent

<table>
<thead>
<tr>
<th>MEASURE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>2673</td>
</tr>
<tr>
<td>mean</td>
<td>2.364769e+04</td>
</tr>
<tr>
<td>std</td>
<td>8.113123e+05</td>
</tr>
<tr>
<td>min</td>
<td>0</td>
</tr>
<tr>
<td>25%</td>
<td>1.583523</td>
</tr>
<tr>
<td>50%</td>
<td>1.297002</td>
</tr>
<tr>
<td>75%</td>
<td>9.613778</td>
</tr>
<tr>
<td>max</td>
<td>3.65e+07</td>
</tr>
</tbody>
</table>

ESTIMATED TRANSACTION VOLUME

The total estimated value of transactions on the Bitcoin blockchain (does not include coins returned to sender as change). The transaction volume represented in Figure 4 is very slowly increasing, meaning that more transactions in Bitcoin are processed. However, it doesn’t represent the actual value of transactions, because people think in their local currency when trading or shopping, and Bitcoin’s value is volatile. The next variable, Estimated Transaction Volume USD, is
more appropriate for measuring the transactions volume as it gives us the value of transactions in USD. The peak that happened in 2012 wouldn’t occur easily today due to the current value of Bitcoin.

**Figure 4: Estimated transaction volume chart**

### Difficulty

This variable depicts a relative measure of how difficult it is to find a new block. The *Difficulty* is adjusted periodically as a function of how much hashing power has been deployed by the network of miners. **Figure 5** displays the difficulty values over time. It can be noticed the staggered nature of the values, that is because the *Difficulty* is adjusted automatically every 2016 blocks, and changes equally for the entire *Bitcoin* network. Until 2014, the difficulty of mining was very low, it started to raise probably because the trade volume in the end of 2013 and start of 2014 was approximately 70,000,000$, which attracted professional miners with a greater processing power, those creating more and more blocks and augmenting the difficulty of mining.

### Euro Price in USD

*Euro price in USD* provided by the *European Central Bank (ECB)* is a variable that shows the *Euro* price in USD. We have found latent values which have been interpolated in order to fill them.

As shown in **Figure 6** the value of *Euro* has been above that of the *USD* from its creation, been the period of 2000 through 2003 its closer price to each other. From 2003 to 2007 the price of the *Euro* increased, coinciding with an increase on the interest imposed by the *ECB*. This increase in the interest of the credits is followed by the collapse of the housing bubble in 2007, where the price of the *Euro* kept increasing, although the interest was maintained by the *ECB* until 2009 where the
ECB started to lower it. The increase in the Euro price is probably due to strategies of the Federal Reserve to lower the price of the USD. Finally in 2015, the ECB lowered the interests of credits approaching the 0.0%, and that’s reflected with a decrease in the price of the Euro with respect to USD.

Market Price

The USD value of Bitcoin, as calculated by the daily average market price across major exchanges. This variable is related to many others, starting with the number of transactions, that can explain the fluctuations in the Bitcoin price and different events. In March 9, 2011, the Bitcoin reaches parity with USD which is shown in a timid growth in market price followed by a decrease (see Figure 7).
After that there isn’t a big growth until 2013, where several things happen. Mega, the cloud storage service, starts accepting Bitcoins, Internet Archive starts accepting Bitcoins, a new food service PizzaForCoins.com accepts Bitcoins as a payment for food, CoinDesk is launched by Spotify investor and Coinbase receives 5 million USD in funding. This, and several other events increase the market price of Bitcoin.

After that there is a decrease in market price of Bitcoin, maybe because MtGox, the largest exchange operator at the time, was seized by The United States Department of Homeland Security.

In the second half of 2013, various events happen that can be the cause of the raise of Bitcoin market price, first, in August 6th, Bitcoin is ruled currency by a Texas judge, then in August 20th, Bitcoin is ruled as private money in Germany, then in August 28th, RoboCoin, a Bitcoin ATM manufacturer, starts accepting orders. This can be the cause of the huge peak at the end of 2013 and start of 2014, where it reaches its highest value (see Table 14). There are no clear events in the Bitcoin history that explain the period after 2014, which leads us to think that the price has been fluctuating due to trading strategies.

Table 14: Statistical values for Market price

<table>
<thead>
<tr>
<th>Measure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>2673</td>
</tr>
<tr>
<td>mean</td>
<td>155.375541</td>
</tr>
<tr>
<td>std</td>
<td>218.398495</td>
</tr>
<tr>
<td>min</td>
<td>0</td>
</tr>
<tr>
<td>25%</td>
<td>0.199</td>
</tr>
<tr>
<td>50%</td>
<td>12.28</td>
</tr>
<tr>
<td>75%</td>
<td>270</td>
</tr>
<tr>
<td>max</td>
<td>1151</td>
</tr>
</tbody>
</table>
Is important to note how volatile is the Bitcoin average price, ranging from nearly 0 USD to 1151 USD in just a year (see Table 14), and now, in 2016 fluctuating in a range bigger than 10 USD on average. These fluctuations, if predicted, can be very profitable for the trader.

MEDIAN CONFIRMATION TIME

The median time for a transaction to be accepted into a mined block and added to the public ledger (note: only includes transactions with miner fees). Until the end of 2011, the confirmation time is negligible. There is an important event that can be related to the growth of the median confirmation time, and that is the largest Bitcoin fee in a single transaction, in December 12, 171 Bitcoins was paid as fee in block 157235.

If we look at Figure 8 there is a peak in mid 2012, that can coincide with the increase in output volume seen in Figure 10.

![Median confirmation time chart](image)

NUMBER OF TRANSACTIONS

Here we analyze number of transactions, which is defined as “The number of daily confirmed Bitcoin transactions”. The number of transactions represented in Figure 9 show a moderated growth compared to that of the market price. That may be because this variable doesn’t depend on the inflation of the Bitcoin, and there aren’t exaggerated fluctuation. We see that even the price of the Bitcoin is not at its peak in 2016, the number of transactions are at its higher point so far in the Bitcoin history.
The total value of all transaction outputs per day (includes coins returned to the sender as change). There are a lot of events that appear in output volume (Figure 10) that also appear in market price (Figure 7) showing that there is a strong relationship between the two measures.

It’s easier to understand the meaning of output and change returned to the sender in the Bitcoin world with an explanation given by a user nicknamed DeathAndTaxes in Bitcointalk.org:

“Bitcoin can only create transactions by using as the input an ENTIRE prior unspent output. The most important thing to realize is that Bitcoin tx have inputs and outputs. The “value” of your wallet is an abstraction. It is simply your client (software which analyzes the wallet) taking a SUM of all the unspent outputs which you have private keys for. The input of a tx is the
output of a PRIOR tx. You can only use unspent outputs in a new tx. Once they are part of a tx they can never be used again ("spent").

Say you I send you 50 BTC (for simplicity lets assume this is compromised of a single 50 BTC output). No matter how you spend that the input for the tx will be 50 BTC.

Want to spend 20 BTC? Input: 50 BTC Output: 20 BTC + 30 BTC “change” back to an address you control. Want to spend 1 BTC? Input: 50 BTC Output: 1 BTC + 49 BTC “change” back to an address you control.”

Output can be due to three circumstances, large amount of transactions, transactions of big amount of Bitcoins, and big amount of change returned to sender. The peak of 2013 is clearly due to change received by sender as DeathAndTaxes explains: “In the early days of Bitcoin there really was nothing to “spend” it on so most tx tended to be accumulation. This resulted in most addresses having very large unspent outputs. As people started “breaking” up those unspent outputs in tx involving smaller amounts most of the volume WAS change.”

STANDARD & POORS 500

From the Wikipedia:

“The Standard & Poor’s 500, often abbreviated as the S&P 500, or just “the S&P”, is an American stock market index based on the market capitalizations of 500 large companies having common stock listed on the NYSE or NASDAQ. The S&P 500 index components and their weightings are determined by S&P Dow Jones Indices. It differs from other U.S. stock market indices, such as the Dow Jones Industrial Average or the Nasdaq Composite index, because of its diverse constituency and weighting methodology. It is one of the most commonly followed equity indices, and many consider it one of the best representations of the U.S. stock market, and a bellwether for the U.S. economy. The National Bureau of Economic Research has classified common stocks as a leading indicator of business cycles.”

We can see the description of certain events of Figure 11 also from Wikipedia:

“In mid-2007, the subprime mortgage crisis spread to the wider U.S. financial sector. The resulting situation became acute in September 2008, ushering in a period of unusual market volatility, encompassing record 100-point swings in both directions and reaching the highest levels since 1929. On November 20, 2008, the index closed at 752.44, its lowest since early 1997. A modest recovery the following day still left the index down 45.5% for the year. This year-to-date loss was the greatest since 1931, when the broad market declined more than 50%. The index closed the year at 903.25, for a loss of 38.5%. The market continued to decline in early 2009, surrounding the financial crisis of 2008. The index reached a nearly 13-year low, closing at 676.53, on March 9, 2009.

On March 23, 2009, the S&P 500 marked a 20% gain when it hit 822.92. The Dow Jones Industrial Average soon followed. The close for 2009 was
1,115.10, making it the second-best year of the decade. On April 14, 2010 the index broke 1200 closing at 1210.65, but by July 2, 2010 it had closed at 1022.58. On April 29, 2011, the index closed at 1363.61, but it had a sharp drop in August and briefly broke 1100 in October (with the VIX hitting 40). Gains continued despite significant volatility amid electoral and fiscal uncertainty, and the 2012 close of the S&P 500 following QE3 was its third-highest ever, at 1,426.22 points. Many people hated the bull market. On March 28, 2013, it closed above the closing high from 2007. On April 10, 2013, it also closed above the intraday high from 2007.

On May 3, 2013—more than 13 years since its first close above 1,500—the S&P 500 closed above 1,600 for the first time, at 1,614.42. This would be the first of three 100-point milestones in 2013: 1,600 on May 3, 2013; 1,700 on August 1, 2013; and 1,800 on November 22, 2013. The S&P 500 closed out 2013 at a record high, finishing the December 31, 2013, trading day at 1,848.36. On May 23, 2014, the index for the first time closed above 1,900, at 1,900.53. On August 26, 2014, the index closed above 2,000 for the very first time, and on December 22 the S&P 500 climbed to 2078, an all-time high. The index closed on December 29 at 2,090.57 with a closing of 2,058.90 at the end of 2014. This was a gain of 85% (in price return, and 105% in total return) for the five years 2010-2014. On February 17, 2015, the index first closed above 2,100, closing at 2,100.34. On February 25, 2015 it reached 2,119.59 during mid-day, and on the following day it closed at record high of 2,115.48. On May 21, 2015, the index closed at 2,130.82, its high point for the year. At the close of 2015, the index hit 2,043.94, down 0.73% for the year.”

Variable close has a linear increasing trend which can probably be explained because of the inflation, while the volume is a random walk, following no apparent trend or seasonality. To complete the information and better understand the graphics shown the reader can analyze Table 15.
### Table 15: Statistical values for Standard & poors 500

<table>
<thead>
<tr>
<th>Measure</th>
<th>SP500-CLOSE</th>
<th>SP500-VOLUME</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>2673</td>
<td>2673</td>
</tr>
<tr>
<td>mean</td>
<td>1503.979515</td>
<td>4.000942e + 09</td>
</tr>
<tr>
<td>std</td>
<td>395.164083</td>
<td>1.122921e + 09</td>
</tr>
<tr>
<td>min</td>
<td>676.330029</td>
<td>5.362e + 05</td>
</tr>
<tr>
<td>25%</td>
<td>1178.359966</td>
<td>3.299395e + 09</td>
</tr>
<tr>
<td>50%</td>
<td>1406.290039</td>
<td>3.78247e + 09</td>
</tr>
<tr>
<td>75%</td>
<td>1898.293335</td>
<td>4.50811e + 09</td>
</tr>
<tr>
<td>max</td>
<td>2130.820068</td>
<td>1.061781e + 10</td>
</tr>
</tbody>
</table>

### Total Bitcoins

The total number of bitcoins that have already been mined; in other words, the current supply of bitcoins on the network. First we have to know that, according to the Wikipedia “The bitcoin protocol specifies that the reward for adding a block will be halved every 210000 blocks (approximately every four years). Eventually, the reward will decrease to zero, and the limit of 21 million bitcoins will be reached circa 2140; the record keeping will then be rewarded by transaction fees solely.” As shown in Figure 12 the growth of bitcoin amount will decrease every time the difficulty increases, with 21 million been the upper limit. This change in difficulty depend on the blocks discovered by miners.

![Total bitcoins chart](image)

**Figure 12: Total bitcoins chart**

### Trade Volume

The total USD value of trading volume on major Bitcoin exchanges.

By comparing trade volume(Figure 13) with other figures as market price(Figure 7) it’s clear that there is a big bound between trade volume and market price of Bitcoin. There are peaks in mid 2011, March
of 2013, December of 2013, and the end of 2015 in both variable, showing the aforementioned relationship.

![Trade volume chart](image)

Figure 13: Trade volume chart

Trade volume doesn’t really exhibit the real activity in the Bitcoin network, because with this variable we don’t see the transactions between users. There is also another pattern that can be seen in the trade volume, is that when the market price is maintained the trade volume decreases, meaning that people are interested in the Bitcoin exchange when its price fluctuates.

**TRANSACTION FEES**

The total value of all transaction fees paid to miners (not including the coinbase value of block rewards) in BTC. This variable depends on the amount of transactions and the amount of fee paid per transaction. If we look at transactions fees(Figure 14) we can see that even there is an increase in the number of transactions near the peaks, the height of those peaks is not comparable, in consequence we can assume that at least those peaks are not due entirely to the number of transactions made. We can see that there is some relationship with market price(Figure 7), reflecting how “eager” is the user to have the transaction accepted. This relationship is not in the sense that the peaks appear simultaneously in the charts, but the peaks of transaction fees precede those of market price. This can also be due to the low price of Bitcoin in those transaction fees peaks, because people think in USD mostly, so they fix the price of the fee in USD instead of BTC producing those rises.
TRANSACTION FEES USD

The total value of all transaction fees paid to miners (not including the coinbase value of block rewards) in USD. Transaction fees USD (Figure 15) shows how the fees measured in USD are more stable in time than fees measured in BTC shown in transaction fees(Figure 14).

There highest fees prices have been paid in the last half of 2013 and start of 2014, falling together with the highest price of Bitcoin, this can be explained because the users would want their transactions to be processed before others. There is another peak, very localized because it’s only one day where the transactions fees reach 157676.252 USD (see Table 16), this can be because the transactions/trade volume ratio is at its highest point since 2011, which means that Bitcoin transaction volume is even higher than January of 2014 when the price of Bitcoin
transactions/trade ratio (Figure 16) presents the relationship between BTC transaction volume and USD exchange volume. This is another chart that can gives us an idea of the transactions inside the Bitcoin network, excluding the trading.

High values of this variable can mean two different things. One is that the value of Bitcoin is very low, as in 2011, and the transactions between users, no matter how high they are, measured in Bitcoin, are going to have low values measured in USD. The other explanation is that there are a lot of transactions inside the Bitcoin network compared with the USD exchange volume, even if the Bitcoin price is high, which would probably mean that there is a high usage of Bitcoin as a currency between individuals, this may be the cause of the raise of this ratio since the end of 2015.
WIKIPEDIA TREND FOR BITCOIN

The particular trend used in this case is the page views per day of the Bitcoin article in the English version of Wikipedia. The daily visits to this page has been as high as 923659 (see Table 17), and a mean of 8168.301160.

Table 17: Statistical values for Wikipedia trend for bitcoin

<table>
<thead>
<tr>
<th>MEASURE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>2673</td>
</tr>
<tr>
<td>mean</td>
<td>8168.301160</td>
</tr>
<tr>
<td>std</td>
<td>25977.892265</td>
</tr>
<tr>
<td>min</td>
<td>0</td>
</tr>
<tr>
<td>25%</td>
<td>86</td>
</tr>
<tr>
<td>50%</td>
<td>4173</td>
</tr>
<tr>
<td>75%</td>
<td>8585</td>
</tr>
<tr>
<td>max</td>
<td>923659</td>
</tr>
</tbody>
</table>

For a better understanding of this variable we can rely on the ubiquitous market price since Bitcoin’s allure is it’s price and fluctuation with respect to more stable currencies like USD or EUR. There are a number of events that can produce the spike produced in second half of 2015 (see Figure 17).

Figure 17: Wikipedia trend for bitcoin chart

In January the New York Stock Exchange invests 75M USD in Coinbase. In March the results of the UK Treasury’s call for information on digital currency are announced. In May the 19th Ross Ulbricht, operator of the Silk Road marketplace, is sentenced to life in prison. In June the 3rd, New York state releases the BitLicense, a set of customized rules meant to regulate Bitcoin and digital currency businesses that serve customers located in New York state. In July the 1st, two federal agents plead guilty to Silk Road theft during their active investigation of the marketplace. In August the 1st, Mark Karpeles, the
CEO of the failed *Bitcoin* exchange Mt. Gox, was arrested in Japan on charges of fraud and embezzlement in relation to collapse of the exchange. *Bitcoin Core* developers Mike Hearn and Gavin Andresen released a separate version of the *Bitcoin* client software, called *Bitcoin XT*. The release illustrates an ongoing controversy in the *Bitcoin* development community: *What limit should be placed on the size of Bitcoin’s blocks?*. In October the 22nd, The *European Court of Justice* ruled that the exchange of *Bitcoin* and “virtual currencies” is not subject to *value added tax (VAT)* in the *European Union*. In October, the 31st, *Bitcoin* is featured on front page of The Economist. In November, the 3rd, *Bitcoin* sign is accepted into Unicode. This are all events that may have produced many visits to the *Bitcoin* Wikipedia page.
The process of feature subset selection (FSS) used in this thesis is composed of two steps. One where we choose a set of variables with principal component analysis (PCA) (Pearson, 1901) which is going to be our feature super-set for all the models. Later on, for selecting per-model subsets a wrapper FSS (WFSS) technique (Kohavi and John, 1997) is used.

This approach to FSS has been taken because wrapper methods select an optimum (that can be local) set of features (Inza et al., 2004; Kumari and Swarnkar, 2011) and, at the same time, they are more computationally expensive than filter methods. We came out with the mixed approach of selecting a super-set with PCA, which is computationally cheap and then we selected the final subsets with a wrapper method.

WFSS is a greedy algorithm, also called a forward feature selection. In each iteration a model is produced with a subset which initially is composed of one variable. In each iteration of the algorithm the variable that minimizes the error produced is added to selected features. If in a iteration a variable selected doesn’t minimize the error of the selected features the algorithm has finished.

The features presented in Chapter 2 where obtained by running a PCA over 41 variables selecting the 18 that added more information to the prediction.

We can see in Figure 18 an ordered representation of the explained variance of the dimensions obtained by performing PCA over the original data-set. This information is what led us to choose 17 as the dimensions that has some meaningful information to predict the Bitcoin price.

The results of WFSS and the parameters used are explained in Chapter 8.
Figure 18: Explained variance of PCA dimensions.
Part III

METHODS & TECHNIQUES

Where we talk about the models used to predict the Bitcoin price and their characteristics.
INTRODUCTION

In the present study we decided to compare how a traditional prediction model (vector autoregression (VAR)) compares to a relatively new model (recurrent neural networks (RNN)) applied to the prediction of the price of Bitcoin.
**VECTOR AUTOREGRESSION**

Vector autoregressive models (VAR) is a multivariate prediction technique where in order to predict a value for each variable is necessary to take into account the past values of the variable and of other variables involved in the model. In VAR all variables are interdependent, which means that all variables depend on the other variables involved in the multivariate system. This interdependent systems are known in econometric as *endogenous*.

VAR is a generalization of Autoregressive Models (AR) applied to a vector of time series. Let’s analyze the mathematical expression of a first order VAR, or simply VAR(1) for a bivariate system defined in **Equation 1**.

\[
y_{1,t} = c_1 + \phi_{11}y_{1,t-1} + \phi_{12}y_{2,t-1} + \epsilon_{1,t} \\
y_{2,t} = c_2 + \phi_{21}y_{1,t-1} + \phi_{22}y_{2,t-1} + \epsilon_{2,t}
\]  \hspace{1cm} (1)

In this particular example \(y_{1,t}\) describes the variable \(i\) in the temporal instant \(t\). This equations define the relationship between the two variables involved. As the name suggests VAR(1) takes into account only the first lag of each variable, where \(c_i\) is a constant offset and \(\phi_{ij,l}\) is the influence of variable \(y_j\) on variable \(y_j\) in the \(l\)-th lag.

VAR models need two parameters for fitting. One is the number of variables, denoted by \(K\), and the other one is the number of lags or \(p\). The number of parameters to be estimated in a VAR is \(K + pK^2\) or \(1 + pK\) per equation. For **Equation 1** there are two variables, i.e. \(K = 2\), and only the first lag is used, i.e. \(p = 1\), that makes a total of parameters to be estimated of \(2 + 1 \times 2^2 = 6\).

The generalized VAR(\(p\)) expression would be **Equation 2**.

\[
y_t = c + \sum_{i=1}^{p} \phi_{iy_{t-i}} + \epsilon_t
\]  \hspace{1cm} (2)

If the time series is stationary, we can predict their values by directly fitting a VAR to the data. On the other hand, if the series is non-stationary we take differences to transform the time series into a stationary one, and then, fit a VAR to the differenciated data. In both cases the model parameters are estimated by *ordinary least squares (OLS)* per equation.

The predictions are generated recursively. VAR generates a prediction for each one of the variables involved in the model. Let’s take the VAR(1) described in **Equation 1**. VAR would estimate the parameters...
\( \phi_{i,j,l} \) and \( c_i \) for \( y_{i,t} \) such that \( i \in \{1, 2\} \) and \( t = T \). Once the parameters has been estimated by OLS is possible to create the prediction equations. **Equation 3** describes the prediction equation for \( h = 1 \)

\[
\begin{align*}
\hat{y}_{1,T+1|T} &= \hat{c}_1 + \hat{\phi}_{11,1} y_{1,T} + \hat{\phi}_{12,1} y_{2,T} \\
\hat{y}_{2,T+1|T} &= \hat{c}_2 + \hat{\phi}_{21,1} y_{1,T} + \hat{\phi}_{22,1} y_{2,T}
\end{align*}
\]  

Equation 3 is very similar to **Equation 1** except that the parameters has been replaced by its estimated values and the error has been replaced by zero.

If the horizon we want to use is \( h = 2 \), the equation would be **Equation 4** where the values for past values of the variable are replaced by predictions of those values.

\[
\begin{align*}
\hat{y}_{1,T+2|T} &= \hat{c}_1 + \hat{\phi}_{11,1} \hat{y}_{1,T+1} + \hat{\phi}_{12,1} \hat{y}_{2,T+1} \\
\hat{y}_{2,T+2|T} &= \hat{c}_2 + \hat{\phi}_{21,1} \hat{y}_{1,T+1} + \hat{\phi}_{22,1} \hat{y}_{2,T+1}
\end{align*}
\]  

**Equation 4**
Artificial neural networks are a learning model based on the human brain’s neural network structure. It represents graphically an ideal structure that resembles that of the neuron and its connections with another neurons.

A simple ANN (artificial neural network) can be seen in Figure 19. We can see in the first layer, called input layer, the nodes that represent the inputs, represented as \( x_0, x_1, x_2, \) and \( x_3 \), where \( x_1 \) through \( x_3 \) are actual inputs, and \( x_0 \) is a bias term. This bias term is always equal to 1.

Next we have the directed edges, which represent how the information flow goes, in this case, the information flows from \( x_0, x_1, x_2, x_3 \) to \( a^{(2)}_1 \), which is in the second layer. In other words, \( a^{(2)}_1 \) has \( x_0, x_1, x_2, x_3 \) as inputs. This edges has associated weights, which multiply the output of the origin node and their purpose. In this particular network the weights are represented by \( w_0, w_1, w_2 \) and \( w_3 \). The term \( a^{(2)}_1 \) means the first neuron in the second layer. In the next layer, called output layer, we have the output neuron represented by \( a^{(2)}_0 \), mentioned before. The nodes in this layer have an activation function which can perform different computations with the inputs and a vector of weights \( w \). In this case is the sigmoid function represented by \( \sigma \) with the expression in Equation 7. The expression \( h(x) = \sigma(w^T x) \) means that the input vector \( x \) is multiplied by the transpose of the vector of weights \( w \) and then the result of this operation is passed as input to the activation function \( \sigma \).

As you can imagine, the network can have different topologies, multiple hidden layers, different connection architectures, and different activation functions. Let’s take a look at a two layer network in Figure 20. Just as in Figure 19 the first layer is the input layer, with
bias term $x_0$ and inputs $x_1, x_2$ and $x_3$. Next to the input layer is the hidden layer, which also contains a bias term called $a_0^{(2)}$ and hidden neurons called $a_1^{(2)}, a_2^{(2)}$ and $a_3^{(2)}$. In this neural network architecture, called multilayer perceptron (MLP), each layer is fully connected with the next, therefore each neuron of the hidden layer except for the bias term have all the neurons from the input layer as inputs. Last we have the output layer, which contains a single neuron in MLP called $a_1^{(3)}$. This neuron gives the output of the neural network which is $h(x)$.

![Two layer artificial neural network](image)

Figure 20: Two layer artificial neural network

Here we will try to write the relationship between input, output and the parameters. Let’s start by writing the mathematical expression of the second layer in Equation 5. Where $W_{jk}^{(i)}$ is the weight controlling the mapping between layer $j$ and $j+1$ for the $k$-th input of the $i$-th neuron in layer $j$, and $\sigma$ is the activation function.

$$
\begin{align*}
    a_1^{(2)} &= \sigma(W_{10}^{(1)} x_0 + W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3) \\
    a_2^{(2)} &= \sigma(W_{20}^{(1)} x_0 + W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3) \\
    a_3^{(2)} &= \sigma(W_{30}^{(1)} x_0 + W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3)
\end{align*}
$$

Equation 5

Following with the same procedure we can build the expression for the output in Equation 6.

$$
\begin{align*}
    h(x) &= a_1^{(3)} = \sigma(W_{10}^{(2)} a_0^{(2)} + W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + W_{13}^{(2)} a_3^{(2)})
\end{align*}
$$

Equation 6

We mentioned before the activation function. This activation function takes the inputs of the neuron and outputs a value to the next layer. This value can be a two state value ($\{0, 1\}$), or a linear one. In classification the two-state value are used, and for this outputs the expression that the activation function takes is frequently a sigmoid function or logistic function, represented in Equation 7. We have been using a sigmoid function in all our activation functions.

$$
\sigma(z) = \frac{1}{1 + e^{-z}}
$$

Equation 7
6.2 Backpropagation Algorithm

Backpropagation (BP) was developed in the context of control theory (Kelley, 1960) and later applied to artificial intelligence (Werbos, 1974 and Rumelhart, Hinton, and Williams, 1988).

We are going to use gradient descent to optimize the error function in terms of the weights of the network. First we need to find the gradient of the error function in with respect to the weights, i.e., \( \frac{\partial E}{\partial W^{(l)}_{ij}} \).

In Equation 8 we calculate the derivative of the sigmoid function because we are going to use it all over the backpropagation explanation.

\[
\frac{d}{dz} \sigma(z) = \frac{d}{dz} \frac{1}{1 + e^{-z}} = \frac{e^{-z}}{(1 + e^{-z})^2} = \frac{1 + e^{-z}}{(1 + e^{-z})^2} - \left( \frac{1}{1 + e^{-z}} \right)^2
\]

\[
= \sigma(z) - (\sigma(z))^2 = \sigma(z)(1 - \sigma(z))
\]

Now let’s define the error function in Equation 9. Let’s also define, for the purpose of this explanation, three disjoint sets of neurons, i.e., the set of neurons in the input layer which is called I and is indexed by \( i \) such as \( i \in I \), the set of neurons in a hidden layer which is called J and is indexed by \( j \) such as \( j \in J \), and the set of neurons in the output layer called K and is indexed by \( k \) such as \( k \in K \).

\[
E = \frac{1}{2} \sum_{k \in K} (h_k - y_k)^2
\]

To compute the gradient of the error function we are going to compute it for two of the three disjoint sets that we have established before. Those sets are the output layer, and the hidden layer. There’s no need to compute the error for the input layer because there isn’t an error associated with the inputs.

In the calculation of the gradient of the output layer in Equation 10 we have \( W_{jk} \) meaning a weight between a neuron from the hidden layer to a neuron in the output layer, \( h_i, h_j \) and \( h_k \) are outputs of a neuron in the input layer, hidden layer or output layer respectively.

\[
\frac{\partial E}{\partial W_{jk}} = \frac{\partial}{\partial W_{jk}} \frac{1}{2} \sum_{k \in K} (h_k - y_k)^2
\]

\[
= (h_k - y_k)h_k(1 - h_k)h_j
\]

**substitute** \( \delta_k = (h_k - y_k)h_k(1 - h_k) \) ⇒

\[
\Rightarrow \frac{\partial E}{\partial W_{jk}} = h_j \delta_k
\]
Now for a hidden layer, the calculations of the gradient would be the ones shown in Equation 11.

\[
\frac{\partial E}{\partial W_{ij}} = \frac{1}{2} \sum_{k \in K} (h_k - y_k)^2
\]

\[
= h_j (1 - h_j) h_i \sum_{k \in K} (h_k - y_k) h_k (1 - h_k) W_{jk}
\]

substitute \( \delta_k = (h_k - y_k) h_k (1 - h_k) \implies \)

\[
\implies \frac{\partial E}{\partial W_{ij}} = h_j (1 - h_j) h_i \sum_{k \in K} \delta_k W_{jk}
\]

substitute \( \delta_j = h_j (1 - h_j) \sum_{k \in K} \delta_k W_{jk} \implies \)

\[
\implies \frac{\partial E}{\partial W_{ij}} = h_i \delta_j
\]

Once we come up with the gradient results we can present the backpropagation algorithm in Algorithm 1 to obtain those gradients.

**Algorithm 1** Backpropagation algorithm

1: Let the training set be \( \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\} \)
2: Set \( \Delta_{ij}^{(l)} = 0 \) for all \( l, i, j \)
3: for \( i = 1 \) to \( m \) do
4: \( a^{(1)} = x^{(1)} \)
5: Propagate values to compute \( a^{(l)} \) for \( l = 2, 3, \ldots, L \)
6: Using \( y^{(l)} \) compute \( \delta^{(L)} = a^{(L)} - y^{(l)} \)
7: Compute \( \delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)} \)
8: \( \Delta_{ij}^{(1)} := \Delta_{ij}^{(1)} + a^{(1)}_{j} \delta^{(1+1)}_{i} \)
9: \( D_{ij}^{(1)} := \frac{1}{m} \Delta_{ij}^{(1)} + \lambda W_{ij}^{(1)} \) if \( j \neq 0 \)
10: \( D_{ij}^{(1)} := \frac{1}{m} \Delta_{ij}^{(1)} \) if \( j = 0 \)

In the forward pass, backpropagation computes the values of each neuron until getting the outputs, and in the backward pass, backpropagation computes the error gradients and updates the weights and biases of the network.

**Recurrent Neural Network**

The general idea behind recurrent neural networks is the ability to use information from other elements of a sequence. Thus, the input of RNNs are considered sequences. Whether they are time sequences, or notes from a musical composition or words in a sentence, RNNs can take advantage of this sequential nature of the input.

A recurrent neural network (Rumelhart et al., 1986) is a class of artificial neural network where connections between units form a directed
cycle as shown in Figure 21. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. Unlike feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented connected handwriting recognition or speech recognition.

![Recurrent Neural Network](image)

Figure 21: Recurrent Neural Network

The two equations that govern the behavior of this type of neural nets are described in Equation 12. This equation

\[
\begin{align*}
    h(t) &= \sigma(w_{ij}x(t) + w_{jj}h(t-1) + b_h) \\
    \hat{y}(t) &= \sigma(w_{jk}h(t) + b_y)
\end{align*}
\]

- \( w_{ij} \leftarrow \) Vector of weights between inputs and hidden layer
- \( w_{jj} \leftarrow \) Vector of recurrent weights between hidden layer and itself
- \( b_h, b_y \leftarrow \) Biases to learn offsets

**Backpropagation through time**

*Backpropagation through time (BPTT)*, introduced by Werbos, 1990 is the use of *backpropagation* over an RNN. To be able to apply *BPTT* there are some issues that must be taken into account.

First the RNN need to be unfolded through time in a *feed forward network (FFN)*. This is done by creating a finite set of new nodes that represent the same neuron in successive time steps. This step is performed for every neuron in a hidden layer. This is done because *backpropagation* can’t be run over a network with cycles. Figure 22 shows the 3-step unfolded version of the RNN shown in Figure 21.

Second, as a consequence of the unfolding, we need to reproduce the edges connecting those new nodes, so we establish a restriction by which a set of edges in the FFN that are reproduced from one single edge in the RNN need to update their weights by the same amount. This can be enforced by averaging \( \Delta W \) over all the set each time is computed. Figure 23 shows the weight restriction that would apply to the unfolded RNN of Figure 22.
Figure 22: Unfolded RNN for BPTT

Figure 23: Weight restrictions in unfolded RNN
Part IV

IMPLEMENTATION

Where we explain what tools did we built and what third party ones did we use in order to complete the experiments designed.
IMPLEMENTATION

Before dealing with the models themselves we have been trying to find the best way to select a feature subset for the prediction of the Bitcoin price. We used two methods. First principal component analysis (PCA), an then wrapper FSS (WFSS) both presented in Chapter 3.

For PCA we used a Python implementation included in the library scikit-learn. On the other hand, for WFSS we have built it to embed the models inside it in order to select the right subset of features.

To run the needed experiments to compare the two models studied we searched for a way to build both recurrent neural networks (RNN) and vector autoregressions (VAR), for which we have used Keras and vars package. Keras is an open source neural network library written in Python. Designed to enable fast experimentation with deep neural networks, it focuses on being minimal, modular and extensible. The vars package is a library of programming functions for the R programming language which provides an implementation of vector autoregressive, structural vector autoregressive and structural vector error correction models as well as functions for diagnostic testing, estimation of a restricted models, prediction, causality analysis, impulse response analysis and forecast error variance decomposition.

For the assessment of the two models we developed time series cross-validation (TSCV) which is an adaptation of cross-validation for time series prediction assessment.

TIME SERIES CROSS VALIDATION

Time series cross validation (TSCV) is a model validation technique for assessing how the results generalize to an independent data set. It derives from cross-validation which is also a model validation technique that is used for cross-sectional data. Cross-validation partitions the given dataset in two disjoint subsets of the examples present in the dataset. They are called train partition and a test partition. The prediction model is built with the train partition elements and later it’s accuracy is measured using the test partition elements. This operation can be performed many times with selecting different examples for the train-test partitions, to later average the accuracy measures obtained in each iteration of model prediction and model assessment. This techniques helps us avoid overfitting and to have an insight on how will the model generalize to an independent dataset with unknown data.
A particular case of cross-validation that TSCV is based on is called leave-one-out cross-validation (LOOCV). The distinctive feature of LOOCV is that the test partition is composed of only one element of the dataset, while the train partition is composed of \( D - 1 \) elements, being \( D \) the size of the dataset. This technique is used when there is a need to reduce computations due to the possible combinations of train-test partitions, because LOOCV builds \( D \) models with their respective validation.

Now we get to time-series cross-validation which was proposed by Hart, 1994 as a way to get the same properties that cross-validation offers for cross-sectional data applied to time-series data. Time-series data is characterized by the sequence that its elements form, the elements of a time-series are always present in the same order, they have sequential dependency one with the previous and next elements. Another feature of time-series as an ordered sequence of events is that there isn’t a point in predicting a past event given. In other words, given a \( y(t) \) there is an interest in predicting \( y(t+k), k \in \{1, \ldots, N\} \) but not much in predicting \( y(t-k), k \in\{1, \ldots, N\} \). Added to that, the models that deal with time-series take advantage of their property by enforcing the sequential dependencies in their formulation, hence we can not present the elements of the dataset in any sequence, we have to present them in the correct order by which they are created or observed.

Given all the constraints, i.e. that we want to maintain the right order of the samples, and use LOOCV, partition the data again in train partition and test partition. The difference with standard LOOCV is that the train partition is composed of contiguous time-series elements, usually starting from the 1-st and ending with the \( (T-i) \)-th element, where \( T \) is the size of the time-series and \( i \) is a decreasing value, which initial value is a parameter that depends on the problem domain and the model been assessed. Then the train partition is a one element set composed by the \( (T-i+1) \)-th element. Another difference that arose from this explanation, is that in TSCV there are certain train-test partition configuration where part of the elements of the time-series are not been selected.

Let’s look at the example of Figure 24 assuming that is reflecting the initial partition. The size of the time-series is \( T = 41 \), composed of train partition represented by the red dots, test partition represented by the green triangle and non selected elements represented by the blue dots. Here the train partition is composed of all the elements that go from the 1-st to the 10-th. Here \( i \) has an initial value of 31, so the last element of the train partition is \( T-i = 41-31 = 10 \)-th. The test partition is the \( T-i+1 = 41-31+1 = 11 \)-th element. This way, the train-test partition would incrementally include more and more
elements of the time-series until reaching the last one as shown in Equation 13.

\[
\begin{align*}
\{1, \ldots, 10\} &\rightarrow \{11\} \\
\{1, \ldots, 11\} &\rightarrow \{12\} \\
    &\vdots \\
\{1, \ldots, 39\} &\rightarrow \{40\} \\
\{1, \ldots, 40\} &\rightarrow \{41\}
\end{align*}
\]

Figure 24: Time series cross validation example

In Listing 1 we summarize in a Python-like pseudo-code the behavior of TSCV.

```python
def tscv(dataset, model):
    overall_error = 0
    start = 0
    end = 10
    test_pos = 11

T = 41

    for i in range(train_end, T):
        train_part, test_part = dataset.get_parts(start, end,
                                                 test_pos)
        result = model.fit(train_part)
        error = assess_model(test_part, result)
        overall_error = aggregate_error(overall_error, error)

        end = i
        test_pos = i + 1

    return overall_error
```

Listing 1: TSCV pseudo code
WRAPPER FEATURE SUBSET SELECTION

Feature subset selection techniques are divided in two categories filter techniques and wrapper techniques. Filter techniques are applied over the dataset to extract the features and give the same feature subset as result regardless of the model used in the prediction process. One example of this category is principal component analysis. On the other hand, wrapper techniques are applied embedding the model in the assessment process of the feature subset selected. Basically a feature subset is selected, then a model is fitted to that subset. Then the model is assessed and compared to another feature subset.

We have used wrapper feature subset selection (WFSS) introduced by Kohavi and John, 1997. We used a variant of WFSS that reduces the search space by using a greedy strategy to find the feature subset selection. This version of WFSS doesn’t guarantee to get the subset with the best performance, however it gets a good result in less time. Listing 2 describes with a Python-like pseudo code how the version of WFSS we’ve implemented works.

```python
1 def wfss(dataset, model):
2     not_sel = dataset.get_features() # Not selected features
3     sel = [] # Selected features
4     overall_error = infinity # Assign a big number

5     while not is_empty(not_sel):
6         # Select candidate
7         cand_error = infinity # Assign a big number
8         for cand in not_sel:
9             new_error = assess_features(concat(sel, cand))
10            if cand_error > new_error:
11                selected_candidate = cand
12                cand_error = new_error
13
14            if not overall_error > new_error:
15                # Stop if the new candidate doesn’t
16                # improve the assessment of the
17                # previously selected candidates
18                break
19            else:
20                overall_error = new_error
21                sel.append(selected_candidate)
22                not_sel.remove(selected_candidate)

23     return sel
```

Listing 2: WFSS pseudo code
Part V

EXPERIMENT

Where we present the results of the experiments performed and then analyze and compare the model’s performance.
EXPERIMENTAL SETUP

For the experimentation phase we have used an Intel Core i7-6700HQ CPU @ 2.60GHz, with 16GB of RAM, and a Cuda enabled Nvidia GeForce GTX 960M graphic card.

The data-set previously described in Chapter 2 has been processed prior to using it to fit the models. This process consisted in narrowing down the data to the same date range. Later the missing values were filled with interpolated values. Finally the data-set has been normalized using a standard score described in Equation 14.

\[
X_{\text{norm}} = \frac{X - \mu}{\sigma}
\]

\(X_{\text{norm}}\) ← Normalized instance

\(X\) ← Original instance

\(\mu\) ← Feature mean

\(\sigma\) ← Feature standard deviation

After predicting the values of the Bitcoin (BTC) price, the values are still normalized. In order to correctly interpret them and compute the error measures, this prediction has been de-normalized using the inverse function of Equation 14, which is represented in Equation 15.

\[
\hat{X} = \hat{X}_{\text{norm}} \times \sigma + \mu
\]

\(\hat{X}_{\text{norm}}\) ← Normalized prediction

\(\hat{X}\) ← De-normalized prediction

\(\mu\) ← Feature mean

\(\sigma\) ← Feature standard deviation

We have conducted three different experiments for the neural network prediction model, each one with it’s own layout, which we are going to explain below.

For the first experiment, the topology is the one shown in Figure 25. We used all 17 features selected by PCA which are fed to the RNN neuron. This 17 features are fed to the RNN neuron, which also feeds itself with it’s output. The output of the RNN neuron is the output of the first experiment’s recurrent neural network.

For the second and the third experiment, the features used are those selected by WFSS, which are total bitcoins and market price. The second experiments layout which is shown in Figure 26 has as input layer two neurons for each timestep. In the second layer, the first hidden layer, there are eight RNN neurons, one per timestep. Each one of
those RNN neurons represent a timestep and is fed with the two corresponding inputs and the previous timestep, which is the previous RNN neuron. After the RNN hidden layer, there is an ANN hidden layer, with four neurons, fed by each one of the eight RNN neurons. Finally there is a single ANN which gives the output.

Just as in the second experiment, in the third we have four layers in total, with one for the input, another one for the output and two hidden layers. They are distributed similarly as those of the second experiment. In the first layer there are two neurons for each timestep, but in this third experiment we have 20 RNN neurons, fed by the two corresponding input neurons and the previous timestep. After the RNN hidden layer comes another ANN layer with 10 fully connected neurons. Each one of the ANN neurons are fed by all the RNN neurons of the previous layer. Finally there is the output ANN neuron. The layout can be seen in Figure 27.
After the pre-processing, and to avoid over-fitting we have used time series cross-validation proposed by Hyndman, 2010. The particular implementation of this technique used uses a test partition of size 1, which we call $x_{t+1}$ and a train partition composed of all the elements from the 1095-th to the t-th.

Regarding vector autoregression (VAR) we had to choose two parameters to build the model, one is the number of variables or features and the other is the lags of each variable. For the features used, as mentioned in Chapter 3, we used a principal component analysis (PCA) to select a subset of 17 features out of the original 41. After that, using wrapper feature subset selection (WFSS) we chose a subset of those 17 features previously selected. The size and the particular features selected in the last subset depends on the model used and the parameters chosen. Considering this, we will explain first how did we choose the lag parameter and then how that choice affected WFSS and the result of these process.

To set the lag parameter we have built an automated system to choose from several information criteria available in the \textit{vars} package. This information criteria are, Akaike's information criteria (AIC) by Akaike, 1974, bayesian information criteria (BIC) by Schwarz et al., 1978, final prediction error (FPE) by Akaike, 1973 and Hannan–Quinn information criterion (HQC) by Hannan and Quinn, 1979. Once we have the lag parameter chosen by the aforementioned information criteria, we order in ascending order the set of lags obtained and select the first one, then perform a Portmanteau's test to check that the residuals are uncorrelated. If the null hypothesis of no serial correlation is rejected we choose the next lag obtained. This process continues until the null hypothesis is not rejected.

Once we have the lag parameter we perform a WFSS to obtain a feature subset. In the specific case of VAR, the FSS obtained comprises
the variables *market price, Euro price in USD, cost per transaction percent, S&P 500 volume, estimated transaction volume and S&P 500 close.*

In the case of RNN, there are multiple parameters to choose, one of theme is the number of *epochs.* Each *epoch* is a single pass on the entire data set to update the weights of the network. In *neural networks* the training process can be stopped when the solution (weights and biases) has converged or after a certain number of *epochs.* In our case we chose to stop when the solution has converged. But due to the computational cost of each experiment, for the WFSS phase we chose a fixed amount of 10 *epochs,* which distorted the results, as the parameters for WFSS are different from the parameters used in the final model building. This was done this way because we estimated the time it would took to run all the WFSS experiments in a single computer would be 72 days, which is beyond the scope of this project. Although the results has been distorted, they are good enough to take them into the study as good approximations.

Another parameters are the number of *hidden layers* and the number of neurons in each *hidden layer.* There isn’t a heuristic to choose the number of layers and neurons, and the more layers and neurons you have, the more computational power you need to learn the network parameters. As we shown before, we have chosen to experiment with three different layouts to see how they compare and observe how the changes in the model accuracy with the change of certain parameters.

Learning rate is another parameter that controls the size of weight and bias changes in learning of the training algorithm. *Keras* framework lets us add a regularization term to the cost function in order to control the learning rate, we have added an *L2 or least square error* with parameter 0.001.

For the feature selection we also used the 17 features obtained with *PCA* and *WFSS* for getting an even smaller feature subset. In this case the subset was computed, as mentioned before, by using a number of *epochs* equal to 10, in order to run the experiment in a reasonable amount of time. The subset obtained is comprised of two variables, namely *market price* and *total Bitcoins.*
EXPERIMENTAL RESULTS

RESULTS INTRODUCTION

In this section we present the results for four experiments to compare the performance of two different models recurrent neural network (RNN) and vector autoregression (VAR), being RNN configured in three different ways. The first RNN configuration, previously introduced in Chapter 8, called simply RNN, is a RNN with the 17 features selected by PCA as input which feed one single recurrent neuron. The second is an RNN with 8 recurrent neurons and uses only the two features selected by WFSS as input, this configuration is called RNN 8. The third configuration used is called RNN 20, with 20 recurrent neurons and also uses the two features selected by WFSS.

Both Figure 28 and Figure 29 represent a day ahead prediction of the Bitcoin price, compared to the real Bitcoin price, shown in the variable market price. It is the same information presented in two different ways to allow the viewer to see the particular values of each one in Figure 28, and, on the other side, to be able to compare closely the values of the two models and market price in Figure 29.

Although the shapes of the charts are similar, it can be noticed how RNN takes more instances to learn certain patterns. We can see that RNN performs better than RNN 8. In this particular case, more features help to learn more rapidly from the data. But when we increase the number of recurrent input neurons, which is the same as saying that we increase the window of the input time series, we get better results without resorting to the use of more features. There is a comparison
of forecast accuracy measures that reflect the intuitions observed before in Table 18.

RNN, with the exception of RNN 20, has a tendency to predict lower values than those predicted by VAR and MarketPrice. When the change in the price is high RNN doesn’t change at the same pace while VAR follows the tendency better. This behavior is better perceived in the upper-right and lower-left chart of Figure 29.

![Figure 29: Prediction models and MarketPrice true values in the same chart.](image)

### ACCURACY MEASURES

#### Table 18: Forecast accuracy measures

<table>
<thead>
<tr>
<th>Measure Type</th>
<th>RNN Value</th>
<th>RNN 8 WSS Value</th>
<th>RNN 20 WSS Value</th>
<th>VAR Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean absolute error (MAE)</td>
<td>5.4</td>
<td>6.53</td>
<td>4.51</td>
<td>7.62</td>
</tr>
<tr>
<td>Mean squared error (MSE)</td>
<td>638.55</td>
<td>1282.43</td>
<td>189.5</td>
<td>325.60</td>
</tr>
<tr>
<td>Mean absolute percentage error (MAPE)</td>
<td>3.19</td>
<td>2.39</td>
<td>3.7</td>
<td>2.79</td>
</tr>
<tr>
<td>Theil’s U statistic</td>
<td>0.47</td>
<td>0.08</td>
<td>0.01</td>
<td>0.03</td>
</tr>
</tbody>
</table>

The main accuracy measure used is mean absolute error (MAE), shown in Equation 16. MAE averages the absolute error produced for every prediction done with a particular model. This error measure has the characteristic of representing quantities in the same unit as the original variable and that is easily understandable.

When the fluctuation of the BTC price are low, RNN is closer to market price than VAR, which can be seen in Figure 30.

\[
\text{Mean Absolute Error: MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|.
\] (16)
The next accuracy measure is mean squared error (MSE), shown in Equation 17, which doesn’t present the prediction errors in the same unit, but has the feature of giving more importance to bigger errors than smaller ones. To see how this two measures gives us different information about the same data we can see in Table 18 that, while RNN’s MAE is lower than VAR’s MAE, the RNN’s MSE for RNN and RNN 8 is bigger than VAR’s. That is because for RNN and RNN 8 big errors are bigger and at the same time it has less small errors. That can be clearly seen in Figure 30.

Figure 30: Histogram of RNN absolute errors and VAR absolute errors.

\[
\text{Mean Squared Error: } \text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \tag{17}
\]

Another accuracy measure used is mean absolute percentage error (MAPE), defined in Equation 18. This measure is useful when there are several scales, for example if the models are run over different data-sets. This measure have the disadvantage of being undefined for \( y_i = 0 \), and that it puts a heavier penalty on negative errors than on positive errors.

\[
\text{Mean Absolute Percentage Error: } \text{MAPE} = \frac{100}{n} \sum_{i=1}^{n} \frac{y_i - \hat{y}_i}{y_i}. \tag{18}
\]

Finally we include Theil’s U statistic which is defined in Equation 19. This measure compares a forecast model to the actual model. Is the ratio of the 1-step-ahead MSE for a given forecast relative to that of a random walk forecast. If Theil’s U statistic is less than 1 then the forecasting technique is better than guessing, if is exactly 1 then is
about as good as guessing and if it is more than 1 the forecasting technique is worse than guessing.

\[
\text{Theil's U statistic} = \sqrt{\frac{\sum_{t=1}^{n-1} \left( \frac{\hat{y}_{t+1} - y_{t+1}}{y_t} \right)^2}{\sum_{t=1}^{n-1} \left( \frac{y_{t+1} - y_t}{y_t} \right)^2}}.
\]  

We can see in Table 18 that the value of Theil's U statistic for all the models is less than one, hence we can assume that this models are better forecasting techniques than guessing.

**RNN 8** is the closest to one followed by **RNN**, then **VAR** and finally **RNN 20**. We can observe here, as well as in the charts and previous forecasting accuracy measures, that the number of features selected and the number of *recurrent input neurons* are two parameters that have a big impact on the performance of the RNN model.

**Errors Analysis**

*Figure 31* shows an important information to be able to understand how do the two prediction models studied behave.

First off we can see the comparison between the MAE’s of each model and the fluctuation (formally know as first difference or \( \text{Diff}(1) \)) of the BTC price. It’s important to note that for the two models the MAE is mostly beneath the daily fluctuation of BTC, which tells us the the error is relatively small to predict the *market price*.

In the two lower graphs of *Figure 31* we see a comparison of model’s MAE and absolute errors. The errors of RNN are located around two events of huge change of the *market price*. There the errors of **RNN 20** are lower than those of **VAR**, but those of **RNN** and **RNN 8** are larger. That’s the reason why **RNN** and **RNN 8** have larger *MSE* than **VAR**, while **RNN 20** has lower *MSE* than **VAR**, because bigger errors are penalized. Besides, in the rest of the time series the errors of **VAR** exceed those of **RNN**, which is the reason why **RNN**’s MAE is lower than **VAR**’s.

Regarding the errors probability distribution we would benefit from it to be normal, because that way we would be able to establish a confidence interval with the normal distribution parameters with the expression in *Equation 20*.

\[
\text{Confidence interval} = [\hat{x} - 2\sigma, \hat{x} + 2\sigma].
\]  

In *Figure 32* we can see the shape of the error’s histograms plotted along a fitted normal distribution. Visually the shape of the normalized histograms resembles that of the fitted normal distributions but
we need more evidence. Hence we’ve run several normality tests in order to ensure that the distribution of the errors can be considered to be normal.

In Table 19 we can see the results of all the normality tests performed, namely skewness test, kurtosis test, and D’Agostino and Pearson’s “omnibus” test.

Skewness (D’Agostino, 1970) - Is a measure of the asymmetry of the probability distribution of a random variable about it’s mean. In other words, skewness tells you the amount and direction of skew (departure from horizontal symmetry). The skewness value can be positive or negative, or even undefined. If skewness is 0, the data are perfectly symmetrical, although it is quite unlikely for real-world data. As a general rule of thumb:

• If skewness is less than -1 or greater than 1, the distribution is highly skewed.

• If skewness is between -1 and -0.5 or between 0.5 and 1, the distribution is moderately skewed.
• If skewness is between -0.5 and 0.5, the distribution is approximately symmetric.

The mathematical expression of the skewness measure is defined in Equation 21.

\[
\text{Skewness Measure} = \frac{m_3}{m_2^{3/2}}, \text{ where } m_k = \frac{1}{n} \sum_{i=1}^{n} (e_i - \bar{e})^k
\]

Table 19: Normality test results

<table>
<thead>
<tr>
<th>Type of Test</th>
<th>Statistic Test Value</th>
<th>P-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAR Skewness test</td>
<td>-24.11</td>
<td>1.69e - 128</td>
</tr>
<tr>
<td>RNN vs WFSS Skewness test</td>
<td>38.29</td>
<td>0.0</td>
</tr>
<tr>
<td>VAR Kurtosis test</td>
<td>22.39</td>
<td>4.57e - 111</td>
</tr>
<tr>
<td>RNN vs WFSS Kurtosis test</td>
<td>24.55</td>
<td>3.84e - 133</td>
</tr>
<tr>
<td>VAR D’Agostino and Pearson omnibus test</td>
<td>1083.04</td>
<td>6.61e - 236</td>
</tr>
<tr>
<td>RNN vs WFSS D’Agostino and Pearson omnibus test</td>
<td>2069.63</td>
<td>0.0</td>
</tr>
</tbody>
</table>

We can see by the results of the test in Table 19 that the distribution of the errors of the two models are skewed.

**Kurtosis** (Anscombe and Glynn, 1983) - This measure is informative about the tail behavior of a series. Kurtosis tells the height and sharpness of the central peak, relative to that of a standard bell curve. A distribution with positive kurtosis has heavier tails and a higher peak than the normal, whereas a distribution with negative kurtosis has lighter tails and is flatter. Why are tailedness and peakedness both components of kurtosis? It is basically because kurtosis represents a movement of mass that does not affect the variance. Consider the case of positive kurtosis, where heavier tails are often accompanied by a higher peak. Note that if mass is simply moved from the shoulders of a distribution to its tails, then the variance will also be larger.
To leave the variance unchanged, one must also move mass from the shoulders to the center, which gives a compensating decrease in the variance and a peak. For negative kurtosis, the variance will be unchanged if mass is moved from the tails and center of the distribution to it’s shoulders, thus resulting in light tails and flatness. The mathematical expression is shown in Equation 22.

\[ \text{Kurtosis Measure} = \frac{m_4}{m_2^2} - 3, \text{ where } m_k = \frac{1}{n} \sum_{i=1}^{n} (e_i - \bar{e})^k \]  

D’Agostino and Pearson omnibus (D’Agostino, 1971; D’Agostino and Pearson, 1973) - This test is a combination of skewness and kurtosis, called “omnibus” test. The mathematical expression is Equation 23. The distribution of the “omnibus” measure is approximately a \( \chi^2 \) distribution with two degrees of freedom under the null hypothesis that the sample was drawn from a population with normally distributed values.

\[ \text{Omnibus Measure} = (\frac{m_3}{m_2^{3/2}})^2 + (\frac{m_4}{m_2^2})^2, \text{ where } m_k = \frac{1}{n} \sum_{i=1}^{n} (e_i - \bar{e})^k \]  

Given that the null hypothesis is that the samples in the errors set are drawn from a normal distribution and the p-values are less than 0.01, the aforementioned null hypothesis gets rejected. In light of this results we can’t conclude that the errors are normally distributed.

**RESULTS FOR 2013 ONWARDS**

It’s interesting to look at the results of the range of all predictions made after 2013, there are barely fluctuations of the BTC price.

Both Figure 33 and Figure 34 are basically the same figures as Figure 28 and Figure 29 without the values before 2013.

Because this range of the time series has more fluctuations, the predictions have larger error measures than the predictions made with the whole time series considered in Section 9.1.

In Table 20 it can be seen how the error measures have increased and how the Theil’s U statistic has decreased, been farther from 1, which means that RNN and VAR are better predictors for the current range than for the original range in Table 20.
Figure 33: Prediction models and *MarketPrice* in separate plots for 2013 onwards.

Figure 34: Prediction models and *MarketPrice* for 2013 onwards.

Table 20: Forecast accuracy measures for 2013 onwards

<table>
<thead>
<tr>
<th>Measure Type</th>
<th>RNN 20 WESS Value</th>
<th>VAR Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean absolute error (MAE)</td>
<td>5.67</td>
<td>9.85</td>
</tr>
<tr>
<td>Mean squared error (MSE)</td>
<td>246.13</td>
<td>423.19</td>
</tr>
<tr>
<td>Mean absolute percentage error (MAPE)</td>
<td>2.22</td>
<td>3.04</td>
</tr>
<tr>
<td>Theil’s U statistic</td>
<td>0.01534</td>
<td>0.03588</td>
</tr>
</tbody>
</table>
Part VI

DISCUSSION

Where we discuss what we have learned in this project and what future works can be derived from the present.
DISCUSSION

In this project we had the purpose of comparing two predictive models, i.e. vector autoregression (VAR) traditionally used in econometric and recurrent neural network (RNN) developed in the area of artificial intelligence that is increasingly been used in a wider spectrum of fields. The two prediction techniques have been applied to the one day ahead prediction of the Bitcoin price to measure their performance.

Once we have finished the experimentation we observe that the behavior of VAR in pronounced changes of the Bitcoin's market price is similar to that of RNN. Although RNN compares better to VAR when fed with more elements of the time-series.

Is worth mentioning that RNN improves significantly as the number of inputs neurons increases in such a way that some error measures as mean squared error (MSE) and Theil’s U statistic move from performing worse than VAR to performing better.

RNN are black-box models. There is no explicit form to explain and analyze the relationship between inputs and outputs. This causes difficulty in interpreting results from the networks. Besides there are no structured methods today to identify what network structure can best approximate the function, mapping the inputs to outputs. Hence, the tedious experiments and trial-and-error procedures are often used.

The results obtained in the experiment support the intuition that non-linear models perform better than linear ones in prediction. At the same time linear models have less computational cost than non-linear models, this is because the number of different functions that can be represented by linear models is far less than the ones that non-linear models can represent. In terms of computational cost VAR performs approximately 10 times better.

As to the features that better predict the Bitcoin’s price, we observed that the only two variables that where selected by the different feature subset selection techniques and models where total bitcoins and market price. It has to be taken into account that wrapper feature subset selection (WFSS) is an heuristic technique that doesn’t guarantee the optimum subset of features because the algorithm can get stuck in local optima.

Possible ideas that arise from this study and the experiments that we have run would be:

- Include new features, trying to explore new currencies or maybe include social media information through sentiment analysis.
- Another approach to obtain better performance results can be reached by using different variants of the models used, such as...
vector autoregressive moving-average models (VARMA) or vector autoregressive integrated moving average (VARIMA) variants of VAR, or long short-term memory (LSTM) and gated recurrent units (GRU) variants of VAR. Even new models can be tested in order to check how they predict the Bitcoin price.


Markowitz, H (1959). “Portfolio Selection: Efficient Diversification of Investments.” In:


