



SESSIONAL PAPER

Time series analysis of GSS bonds

Part 1 – Introductory analysis of S&P Green Bond Index

[IFoA Data Science, Sustainability & Climate Change Working Party, November 2023]

D. Dey

Institute and Faculty of Actuaries, London, WC1V 7QJ, UK

Email: dkdey@deloitte.co.uk

Abstract

The market for green bonds, and environmentally aligned investment solutions, is increasing. As of 2022, the market of green bonds exceeded USD 2 trillion in issuance, with India, for example, having issued its first-ever sovereign green bonds totally R80bn (c.USD1bn) in January 2023. This paper lays the foundation for future papers and summarises the initial stages of our analysis, where we try to replicate the S&P Green Bond Index (i.e. this is a time series problem) over a period using non-traditional techniques. The models we use include neural networks such as CNNs, LSTMs and GRUs. We extend our analysis and use an open-source decision tree model called XGBoost. For the purposes of this paper, we use 1 day's prior index information to predict today's value and repeat this over a period of time. We ignore for example stationarity considerations and extending the input window/output horizon in our analysis, as these will be discussed in future papers. The paper explains the methodology used in our analysis, gives details of general underlying background information to the architecture models (CNNs, LSTMs, GRUs and XGBoost), as well as background to regularisation techniques specifically L2 regularisation, loss curves and hyperparameter optimisation, in particular, the open-source library Optuna.

Keywords: GSS Bonds; Green Bonds; Neural Networks; Time Series; CNN; LSTM; GRU; XGBoost

Disclaimer: The views expressed in this publication are those of invited contributors and not necessarily those of the Institute and Faculty of Actuaries (IFoA). The Institute and Faculty of Actuaries do not endorse any of the views stated, nor any claims or representations made in this publication and accept no responsibility or liability to any person for loss or damage suffered as a consequence of their placing reliance upon any view, claim or representation made in this publication. The information and expressions of opinion contained in this publication are not intended to be a comprehensive study, nor to provide actuarial advice or advice of any nature and should not be treated as a substitute for specific advice concerning individual situations. On no account may any part of this publication be reproduced without the written permission of the Institute and Faculty of Actuaries. This paper expresses the views of the individual author and not necessarily those of their employers.

1. Executive summary

We are pleased to publish our first paper as a Working Party using data science techniques to look at sustainability and climate change-related issues. In this paper, we summarise the first stage of

© The Author(s), 2024. Published by Cambridge University Press on behalf of Institute and Faculty of Actuaries. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (<https://creativecommons.org/licenses/by/4.0/>), which permits unrestricted re-use, distribution, and reproduction in any medium, provided the original work is properly cited.

our analysis, where we introduce data science techniques to construct a time series analysis of the *Standard & Poor's (S&P) Green Bond Index*.

1.1. Scope of this Paper

The aim of this paper is to lay out the foundations for a time series analysis of green, social and sustainability (GSS) bond indices, and is not intended to be a definitive guide. We have deliberately **excluded stationarity considerations and restricted this paper to a univariate analysis**. We will include stationarity considerations and expand our examination to a multivariate analysis in subsequent papers.

For the purposes of this paper, we have focussed on the *S&P Green Bond Index* and performed various univariate time series analyses using a range of models, which include neural networks. This paper focusses on using a rolling window approach of one prior day's index value to predict today's index value.

In particular, this paper discusses (arranged as per the following Sections):

- Section 2: **Introduction**
 - Background to GSS bonds and a brief explanation of the analysis covered in this paper.
- Section 3: **Data**
 - Insight into the data used in our analysis along with summary information on the training/validation/test splits.
- Section 4: **Summary of models used**
 - A high-level summary of model architectures used in our analysis (i.e. neural networks and a decision tree) with supplemental background information, grouped into five model categories.
- Section 5: **Training the models**
 - Background information on the loss history, Adam optimiser, regularisation techniques, and hyperparameter optimisation techniques used in our analysis.
- Section 6: **Results**
 - Summary tables and graphs for the best-performing model per model category.
- Section 7: **Conclusions and next steps**
 - Summary of conclusions from our analysis and potential areas of analysis for subsequent papers.

Please note that Sections 4 and 5 have been included in this paper to assist with the general understanding of underlying model architecture, and the training process of neural networks.

1.2. Summary of Analysis in this Paper

1.2.1. Aim of the analysis

This paper focusses on the initial stages of our time series analysis of GSS bonds, specifically focussing on the daily values from the *S&P Green Bond Index* and whether we can create accurate prediction models using, for example, neural networks. This paper is the foundation for future analysis, where we hope to develop a model that can assist with GSS-bond index prediction, which will have wider applications such as index price modelling and investment portfolio analyses for actuaries and non-actuaries alike.

For the purpose of this paper, we are looking to predict a rolling 1-day value of the index, based on the prior day's index value over the period 2013 to 2023 inclusive. A rolling window approach is a typical approach for developing a time series model, where we assume prior history is used to imply a future stock index price. For example, we assume values of the prior x days influence the value of future value y days. The simplest approach of this type is to use 1 day prior to predict

today's value (i.e. a rolling 1-day window) to repeat this process over a period of time. We discuss this in more detail in Section 3.2.

We set the baseline model such that today's value equals yesterday's value over the course of the full date range of 31 January 2013 to 17 February 2023 (referred to as 'Baseline model' in this paper). Applying a similar approach, we aimed to see if we can accurately create a time series model with non-traditional methods such as neural networks and a decision tree model (XGBoost). Please see Section 4 for further details.

Analyses using stationarity and multivariate techniques have been deferred to later papers. Similarly, we extend our analysis by looking at using the prior x index values (*window*) to predict the next y days in the future (*horizon*) in subsequent papers.

1.2.2. Data and method

We analysed the *S&P Green Bond Index* values between 31 January 2013 and 17 February 2023, splitting the data using 70%/20%/10% splits for training/validation/test data.

The machine learning models analysed can be categorised into the following model categories:

Deep Neural Network (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) neural network architectures. These architectures and their distinctions are covered further in Section 4. A decision tree model, i.e. XGBoost, was also included as the final category in our analysis. Again, we discuss this in further detail in Section 4.

A prior, simplified examination was performed during the preliminary stages of our analysis, where we adjusted, for example, the number of hidden layers up to five hidden layers, to see if this produced material improvements in the model outputs. We do not address this part of the analysis further in this report. Following this stage, we narrowed down the model architecture per model category for further analysis which we present in this paper. Please see Appendix C for more details.

Though we have analysed different model architectures as described in Appendix C in our analysis, using the techniques described in Section 5, we have presented the best-performing models per category when discussing the results and conclusions in this paper (Sections 6 and 7, respectively).

The loss function used during model training was set to Mean Absolute Error (MAE) for all models, with an Adam optimiser used to update the weights in the neural network and L2 regularisation to reduce any overfitting across all models. Hyperparameter tuning of all models was completed via the open-source library *Optuna*, using the Bayesian optimisation algorithm Tree-structure Parzen Estimator (TPE). These are discussed in more detail in Section 5.

1.2.3. Results and conclusions

Although we analysed a range of models per model category (see Appendix C for more details) e.g., by varying the number and type of hidden layers, we have presented the best-performing model per model category in this paper.

The results of our analysis were inconclusive: the models from each category produced comparable results to the Baseline model with differences in Mean Absolute Percentage Error (MAPE) of up to approximately $\pm 0.1\%$ and hence with no material outperformance. The DNN and LSTM models marginally outperformed the Baseline model, based on the data range and parameters in our analysis. Given the initial approach of a 1-day rolling window, in effect we are potentially not sharing sufficient historical information or correlated information e.g., as per a multivariate analysis, for our models to learn underlying material information and patterns in the data to result in a model that materially outperforms the Baseline model. We aim to address this in future papers (see Sections 1.2.4 and 7 for more details).

1.2.4. Next steps

For future papers, we will expand our analysis to include the following:

1. Stationarity considerations by introducing, for example, an autoregressive, integrated, moving average model by widening the data input *window* and output *horizon*. A variation of this has been explored for example Peters et al. (2022), which explores time series and green bonds.
2. Use of more complex models e.g. a variation of an LSTM model known as a Complete Ensemble Empirical Mode Decomposition with Adaptive Noise-LSTM (CEEMDAN-LSTM) model and Neural Basis Expansion Analysis for Interpretable Time Series (N-BEATS) model. The CEEMDAN-LSTM model is explored by Wang et al. (2022) on green bonds (see 4 below). The N-BEATS model is a time series model and is explored by Oreshkin et al. (2019). It outperformed the Makridakis time series M4 competition model winner by 3%. Both models are discussed further in Section 7 of this paper.
3. Expanding the analysis to general GSS bonds. The analysis in this paper is based on a single green bond index. We will look to expand our analysis to the wider GSS-bond universe and over differing date ranges for the data to see if there are general underlying patterns.
4. Expanding the analysis to include any potential relationships with the general market such as stock market and oil prices i.e. move to a multivariate analysis in subsequent papers. This is examined in Wang et al. (2022) which, when coupled with the CEEMDAN-LSTM model, seems to produce materially improved model predictions based on green bond time series data.

As mentioned, this paper lays the foundation for later papers and introduces a range of model architectures. We expect specific features such as *convolutions* for a CNN model, and iterations via LSTM and GRU cells to have a larger influence on results once we start expanding the input *window* and output *horizon* size in later papers.

1.2.5. Additional disclaimers

Please note the following:

- a. Information within this paper is **valid up to 30 September 2023**. Hence, there may be updates beyond this date that are not reflected in this paper e.g., changes to any legislation mentioned or updates to any open-source code/programming libraries used.
- b. This paper is not intended to be a comprehensive audit of models. Neither is this paper recommending or promoting one approach over another, nor promoting any of the sources or references stated in this paper. Any user of this paper should still reference the underlying legislation, reference any standard mentioned in this paper, and should there be any conflict, the underlying information in the relevant standard, reference or legislation supersedes any information presented in this paper.
- c. Though the work in this paper does not fall under the Financial Reporting Council's Technical Actuarial Standards, this paper has been reviewed both within the Working Party and by the Institute and Faculty of Actuaries' Data Science Practice Board.

2. Introduction

2.1. Background to Green Bonds

A green bond is a type of *fixed-income instrument that is specifically earmarked to raise money for climate and environmental projects* (Segal, 2022). The key difference between green bonds versus

conventional bonds is that green bonds are issued to finance projects that have a positive impact on the environment. Examples of such projects include renewable energy and clean transportation. For more background on green bonds, please see for example Investopedia's post (Segal, 2022) along with recent posts in *The Actuary* (Mitchell, 2022), which discuss the concept of a greenium.

The first ever green bonds were issued in 2007 by the European Investment Bank (EIB) totalling approximately USD 807 million (CFI Team, n.d.). Since then, the market for green bonds has increased, with global cumulative green bond issuance passing the USD 1 trillion mark in 2020 (Climate Bonds Initiative, 2020), and stands at just over \$2 trillion in 2022 (Statistica Research Department, 2023).

Please see immediate chart below from Climate Bonds, as well as related information, from their website (Climate Bonds Initiative, 2021) for details of this trend.

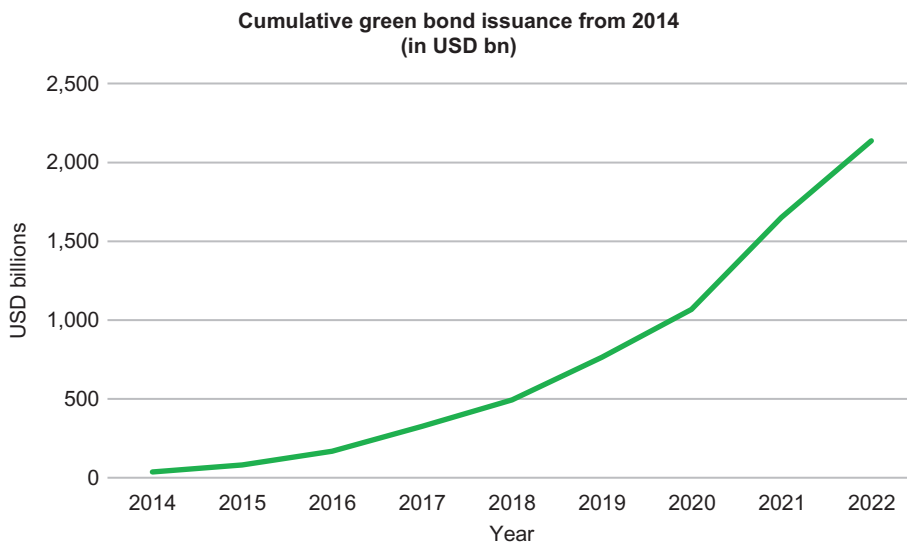


Figure 1. Cumulative global green bond issuance.

Please note that Figure 1 ignores any bonds issued prior to 2014.

Currently, issuers of green bonds range from supranational institutions, public entities, and private companies (Iberdrola, n.d.).

In 2022, although Europe continued to lead the way in terms of issuing green bonds and represented approximately 47% of the green bonds issued that year (Climate Bonds Initiative, n.d.), China and the US accounted for approximately 18% and 13% respectively of the green bond market (Michetti et al., 2023).

The number of countries issuing green bonds continues to expand. Recently, for example, India issued its first-ever sovereign green bonds totalling R80 billion (approximately USD1 billion) in January 2023 (Reed, 2023). Similarly, Israel issued its first-ever sovereign green bonds at the start of 2023 approximately USD2 billion (Wrobel, 2023).

Please note that green bonds are part of the wider GSS-bond universe, which also cover social- and sustainability-aligned investments. For more details on GSS bonds, please see, for example, JP Morgan Asset Management (2023) and The World Bank (2023).

With the proposed legislative EU Green Bond Standard, we expect the trend of increased green bond issuance to continue. The Standard was originally submitted by the European Council back in 2021 (European Commission, 2021). The legislation received agreement between the EU Parliament and the EU Council on 28 February 2023 (European Commission, 2023a). Currently, this will be a voluntary regime which is intended to be the “gold standard” for green bonds (Latham & Watkins, 2023) and will be subject to supervision by competent authorities (as set out in the *European Securities and Markets, Authority 2017 Prospectus Regulation*). Some of the requirements include issuers to publish a prospectus and a green bond fact sheet.

See European Commission (2023b) for further details on the Standard, and Latham & Watkins (2023) for more general commentary.

2.2. Introduction to the S&P Green Bond Index and Our Time Series Analysis

There are several green bond indices in existence, including the *Bloomberg Morgan Stanley Capital International (MSCI) Green Bond Indices*, *Financial Times Stock Exchange Green Impact Bond Index Series* and the *S&P Green Bond Index*, where the index is designed to track the global green bond market.

For the purposes of this paper, we have focussed on the *S&P Green Bond Index* and aim to replicate the index using various models over a certain time period. We include data from the end of January 2013 to mid-February 2023 in our analysis.

The aim of this paper is to introduce some of the ideas from the initial stages of our analysis. Hence, we focus on a univariate time series model, where our model uses the prior day’s value to predict the index value one day ahead (i.e. today’s value). We aim to address more complex univariate time series and build on models discussed in papers such as Peters et al. (2022) (where a hybrid Seasonal Auto-Regressive Integrated Moving Average with Exogenous factors, SARIMAX and LSTM model is used) in future papers. Similarly, we aim to analyse any interaction between green bonds and the general stock market (i.e., multivariate time series) in a future paper, widening our analysis to cover other GSS bonds as well.

For the purposes of this paper, we will aim to produce a sufficiently accurate predictive model, where the majority of the models we analyse are based on a neural network architecture. Data from 31 January 2013 to around mid-February 2022 will be used to train and validate the models. These models are then used to predict daily index values on unseen data from mid-February 2022 to mid-February 2023 (the test data set). The difference in predicted values from our models to actual daily index figures will be used to gauge the accuracy of the proposed models.

3. Data

3.1. Background to S&P Green Bond Index Data Used

For the purposes of this paper, we analyse the *S&P Green Bond Index* (Total Performance, in US dollars, from 31 January 2013 to 17 February 2023 inclusive). Part of the driver for this decision was to use an index that is freely available. For details of this index, please see the *S&P Green Bond Index* methodology paper published in February 2023, available via the main S&P website (S&P, 2023).

Figure 2 shows the index value over this period.

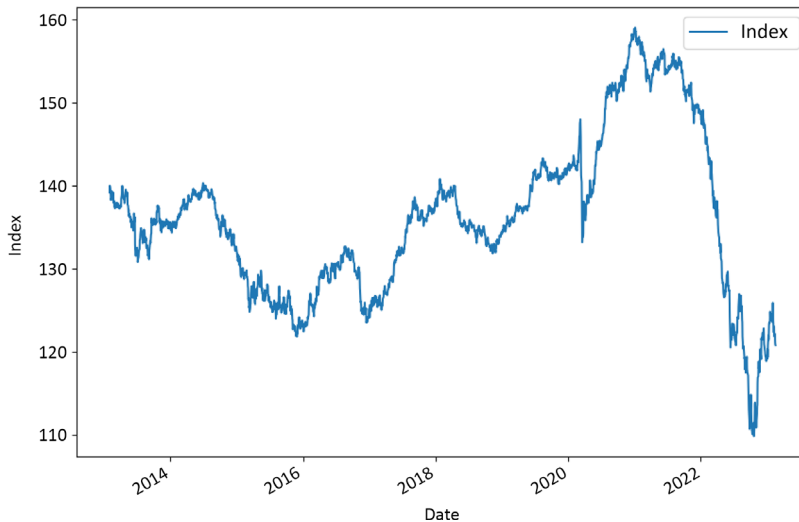


Figure 2. S&P Green Bond Index from 2013 to 2023.

The underlying data is daily data taken from the main S&P website: <https://www.spglobal.com/spdji/en/indices/esg/sp-green-bond-index/>. Please note that the data is based on work days. We downloaded the data from the main S&P website and performed basic checks such as checking for blanks in the data, ensuring that there were no duplicate date entries, comparing the chart against the S&P website charts and other websites that displayed this index, as well as comparable charts from other research papers which analysed this index. No adjustments were made to the data.

One interesting item of note is how the index behaves after early 2020: there is a general upward trend in the index, increasing close to approximately 160 on 5 January 2021 and decreasing to approximately 110 on 21 October 2022. This may be reflective of the underlying impact of COVID on general markets from the start of this period, though this will be explored in more detail in subsequent papers.

3.2. Overview of Time Series Methodology

For the purposes of this paper, we have based our time series analysis on a simple sliding window technique, where a subperiod of historic data (*window*) is input into the model and used to predict the future period (*horizon*), where both the window and horizon are of duration 1 day. We then slide this window along the data by 1 day to predict the following day, and so on. Please see Brownlee (2020) for the background to this approach. Alternative methods such as a cross-validation window approach (Shrivastava, 2020), which varies the window input length, are not analysed in this paper. We aim to look at more complex windowing techniques in future papers.

For the purposes of our analysis, the data set was split as follows: 70%/20%/10% between training/validation/test data sets. The first two splits are used to train/tweak our model (using the training/validation splits of data). We then tested the model against unseen data (test data) and compared the predicted outputs against actual observed data to gauge the model's accuracy.

3.3. Summary of Data

The splits mentioned above are shown in Figure 3. Given the nature of the analysis (a time series analysis), we have proportioned these splits in chronological order, so that we can build models to infer some form of prior/time dependency based on the underlying data and have not randomly allocated the data between these splits across the full data range.

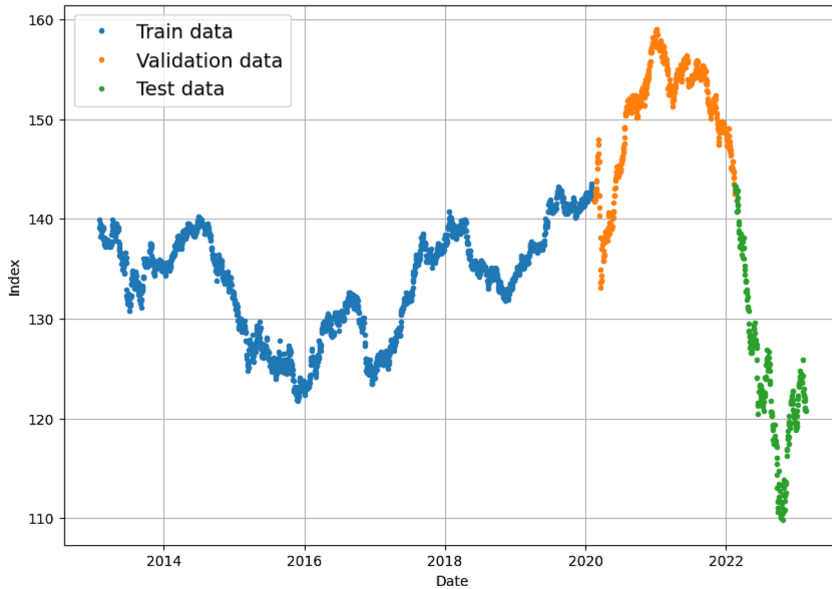


Figure 3. S&P Green Bond Index data with training/validation/test splits highlighted.

The splits correspond as follows: to 16 February 2020, to 16 February 2022 and to 17 February 2023, inclusive. Table 1 details further each data split.

Table 1. Summary information of the full data used, as well as the training, validation and test data sets

	Full Data	Training Data	Validation Data	Test Data
<i>Start Date</i>	31 Jan 2013	31 Jan 2013	17 Feb 2020	17 Feb 2022
<i>End Date</i>	17 Feb 2023	16 Feb 2020	16 Feb 2022	17 Feb 2023
<i>Number of Index Entries</i>	2,615	1,830	523	262
<i>Index Minimum</i>	109.80	121.78	133.14	109.80
<i>Index Maximum</i>	158.99	143.59	158.99	143.34
<i>Index Average (2 d.p.)</i>	136.00	133.60	150.46	123.96
<i>Index Standard Deviation (2 d.p.)</i>	9.65	5.32	5.70	8.04

In summary, there is greater volatility in the test data set range compared to the training and validation data sets. Hence, it will be interesting to see how our models cope given that they will be built on less volatile training and validation data.

Please note that, for the purposes of our analysis, we have not adjusted the data further i.e. no normalisation of the index (fixing data to lie within a scale from 0 to 1, a technique typically used to result in quicker convergence to a solution for a model) and no log transformation (which can be used to potentially dampen any impact of seasonality). Such techniques may be discussed in later papers.

4. Summary of Models Used

4.1. Introduction

In this section, we give an overview of the underlying model architecture used in our analysis. For the purposes of this paper, we have mainly used models based on neural networks. In summary, these can be grouped as follows:

1. The **Baseline** model where we assume today's value is consistent with yesterday's, and then move our projection along by 1 day. The aim of a baseline model is to start our analysis off with something simplistic and act as a reference marker for other models to (ideally) beat (and hence justify any additional complexity in model design).
2. A **Deep Neural Network (DNN)** model, which is a feedforward artificial neural network with 1, 2 or 3 hidden layers. (Please note that strictly speaking, a deep neural network has 2 or more hidden layers. For consistency and simplicity, we have retained the same labelling and categorisation approach between the models in this group.)
3. A **Convolutional Neural Network (CNN)**, where a CNN has a convolutional layer that effectively filters down information, stripping out noise to find an underlying pattern in the data.
4. A **Long Short-Term Memory (LSTM)** model, which is a type of Recurrent Neural Network (RNN) aimed at resolving the vanishing gradient problem.
5. A **Gated Recurrent Unit (GRU)** model, which is a type of RNN and may be seen as a simplified version of LSTM.
6. A decision tree/ensemble gradient-boosting library i.e. **XGBoost**.

We go into more detail on the architecture for the above models in Section 4.2. Please note that in our analysis we have used variations of models within each category above, for example by varying the number of hidden layers. For details of the models analysed, see Appendix C. However, for the purposes of this paper, we have presented the best-performing models per category in the results section (Section 6).

4.2. General Background to Neural Networks

The inspiration for the underlying design of neural networks is the design of the human brain. See IBM (n.d.) for a basic overview.

A traditional approach to tackling a problem such as modelling time series data is to take a prescriptive approach, for example, to consider a mathematical formula and build this formula based on observed data, where we analyse and assume some form of understanding of the underlying mechanics of a problem. The appeal of using neural networks is their potential to learn any form of problem. A neural network can train itself based on sufficient data, once the model's architecture has been decided upon. One positive of this approach is that we do not need to fully possess a mathematical model, or have even true underlying understanding, to tackle a problem.

4.3. Summary Comparison of Models Used

Below is a summary of the model architectures used in our analysis, with some explanatory comments on the underlying nature of each type of architecture. We have grouped our models into five categories in addition to the Baseline model. All models predict one day's value of the index, based on the prior day's observed index value over a time period.

Table 2. Summary Table Showing Categories of Models Used in this Paper

Model Category	Model Architecture	Neural Network?	Overview	Link	Typically Used to Solve Which Problem?
0	Baseline	N	Index equal to prior day's value	-	-
1	DNN	Y	Feedforward neural networks with 1 to 3 hidden layers	<ul style="list-style-type: none"> • Overview 	Wide range of problems
2	CNN	Y	A CNN uses a <i>convolutional layer</i> to extract underlying <i>features</i> from the data. Specifically, we focus on a <i>Conv1D</i> architecture in this paper.	<ul style="list-style-type: none"> • Overview • Paper 	Image, Speech, Audio
3	LSTM	Y	LSTM is a subset of RNNs, used to address the <i>vanishing gradient</i> problem which exists in RNNs. First published in 1997.	<ul style="list-style-type: none"> • Overview • Paper 	Sequential data e.g. Sentiment analysis, Language modelling, Speech recognition
4	GRU	Y	Can be seen as a simplified version of LSTM and sits within the RNN family. First published in 2014.	<ul style="list-style-type: none"> • Overview • Paper 	Sequential data e.g. Sentiment analysis, Language modelling, Speech recognition
5	XGBoost	N	Gradient boosting open-source library initially released in 2014.	<ul style="list-style-type: none"> • Overview • Paper 	Classification, Ranking, Regression

Table 2 highlights typical problems we would normally associate with each category of model. However, each model category can be extended to other areas such as time series, subject to the model producing sufficiently accurate results. Sections 4.4–4.8 detail further general model architectures for each category in Table 2.

4.4. Deeper Dive into Model Architectures – DNN

Figure 4 is an example of a deep neural network, with an input layer (in green), fully connected hidden layers (in blue) and an output layer (in red). Each individual unit (circle below) is a *neuron*.

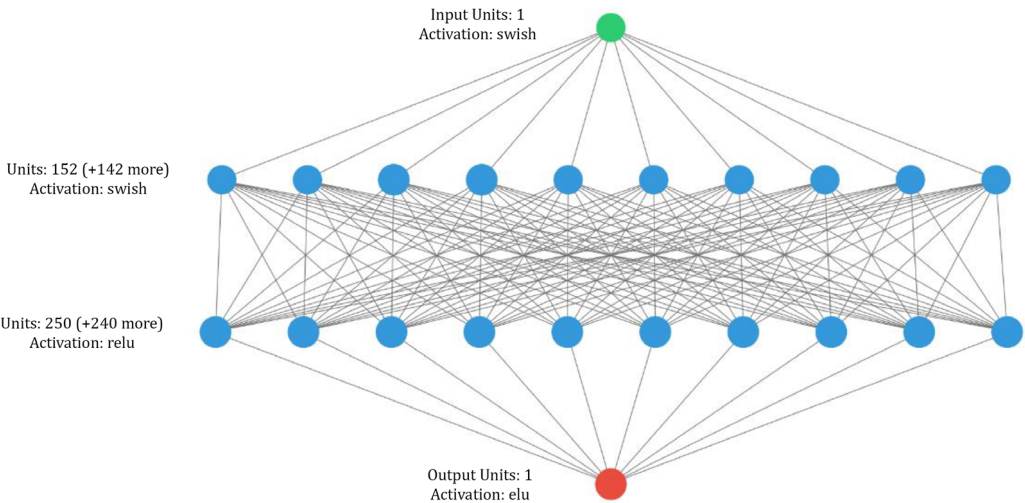


Figure 4. Example DNN architecture.

The underlying mechanics of a neural network are as follows: different weights and biases are applied to a value from a prior layer before an activation function is applied and then this value is passed on to the next layer.

This is shown further in Figure 5, where we show the various components of a neuron for a hidden layer, where x_i represents inputs, w_i the weight and b the bias.

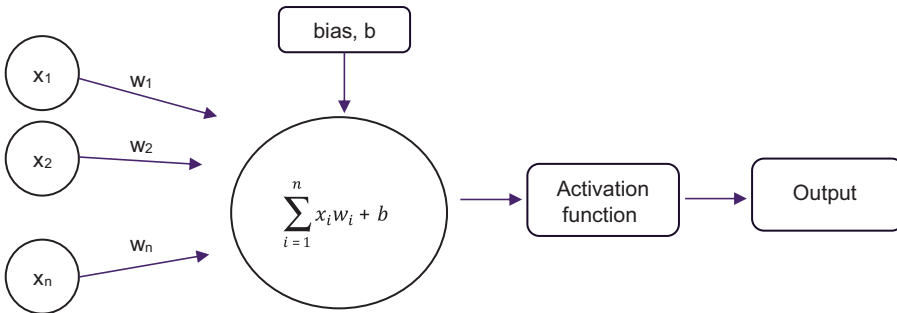


Figure 5. Components of a neuron in a hidden layer.

The aim of an *activation function* in a neural network is to introduce non-linearity to a regression model, such as a time series analysis. Please see Section 5.8 for more information on *activation functions*.

4.5. Deeper Dive into Model Architectures – CNN

CNNs are a type of neural network architecture that can work with two-dimensional (2D) data, for example, to classify images, though that can be extended to one-dimensional (1D) data such as time series and to three-dimensional (3D) data such as that used for video classification or medical image segmentation.

With a CNN, the underlying idea and aim of the architecture is to effectively summarise input information and extract underlying features or patterns in the data before performing further analysis. Typically, the information becomes quite large to handle and hence additional layers are introduced in the neural network to reduce this information whilst retaining important characteristics of the underlying data. CNNs are typically made up of:

- A *convolutional layer* uses *filters* and *kernels* to extract features (i.e. underlying patterns) in the data. For the purposes of our analysis, the kernel can be viewed as the length of the input window of our time series, whilst the filter size represents the number of features in the data, with a single filter responsible for learning a single underlying pattern in the data.
- A *pooling layer* further reduces the information produced by prior convolutional layers whilst still retaining important information.

Figure 6 shows an example structure of a CNN architecture for a time series.

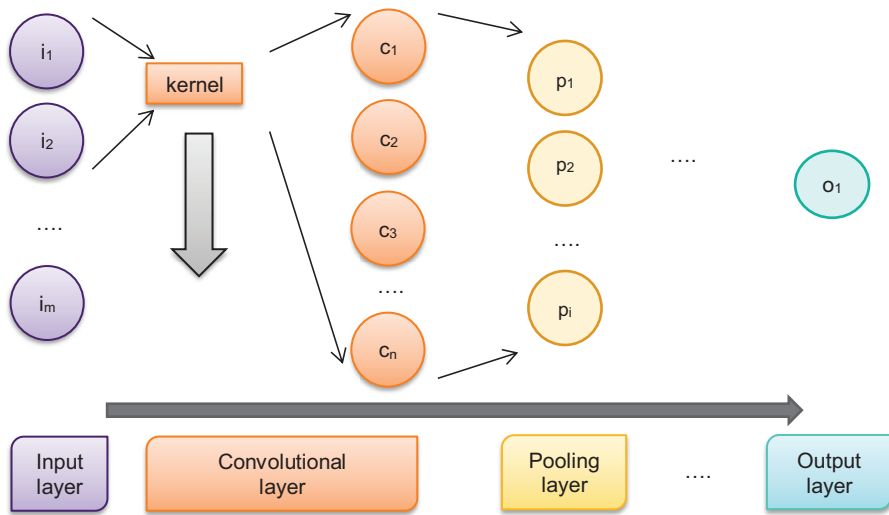


Figure 6. Example convolutional 1D architecture.

We have used a 1D CNN in this paper to deal with time series, as the *kernel* in effect moves in one direction (i.e., increases in time). Further, for the purposes of this paper, we have not included a *pooling layer* given the initial nature of our analysis.

4.6. Deeper Dive into Model Architectures – LSTM

LSTM was introduced in 1997 and builds on RNNs (where information is passed through the same layer multiple times before moving on to the next layer), with the aim of introducing some form of longer-term memory compared to an RNN, as well as addressing the *vanishing gradient* problem present in RNNs. An LSTM layer is a neural network layer that contains an LSTM cell. These cells have additional structures called *gates* to control the flow of information. Figure 7 shows the internal structure of an LSTM cell, where these three different gates are represented by the shaded, grey areas, though please note that there are many variants of an LSTM cell:

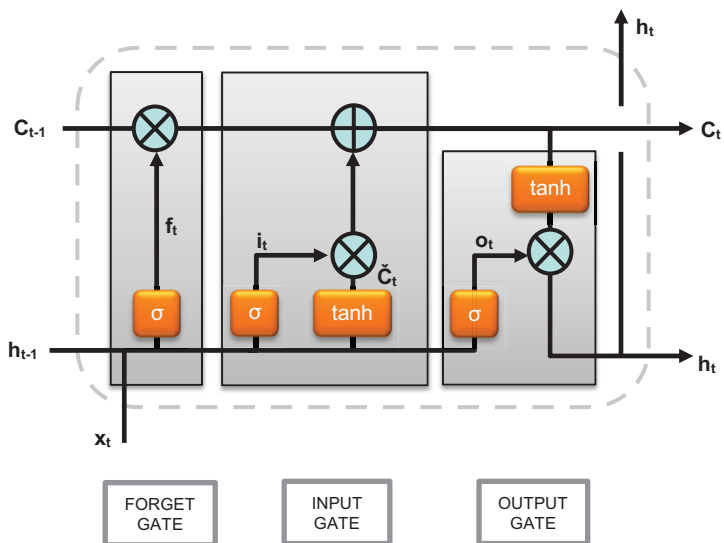


Figure 7. Inside an LSTM cell (Source: Ryan, 2020).

The components of an LSTM cell include:

- The *cell state* C_t at time t which represents the long-term memory component:
- $C_t = f_t \odot C_{t-1} + i_t \odot \check{C}_t$ where $\check{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$ and C_{t-1} is the *cell state* at time $t-1$.
- The *forget gate* f_t at time t decides which long-term memory component in the *cell state* is no longer needed and hence can be removed: $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$.
- The *input gate* i_t at time t decides which new information to add to the long-term component in the cell state: $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$.
- The *output gate* o_t at time t determines the value of the next hidden layer: $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$.

In the above, W and b represent weight and bias vectors for each respective component, x_t is the input vector at time t , and h_t and h_{t-1} are the hidden state vectors at times t and $t-1$ respectively.

4.7. Deeper Dive into Model Architectures – GRU

Introduced in 2014, GRUs are similar to LSTMs where the flow of information is via internal gates. The internal GRU cell architecture is, however, simpler to an LSTM cell, with only two internal gates. This makes it potentially quicker to train compared to an LSTM model. Figure 8 shows the internal structure of a GRU cell, showing these two different gates (though please note that there are many variants of a GRU cell):

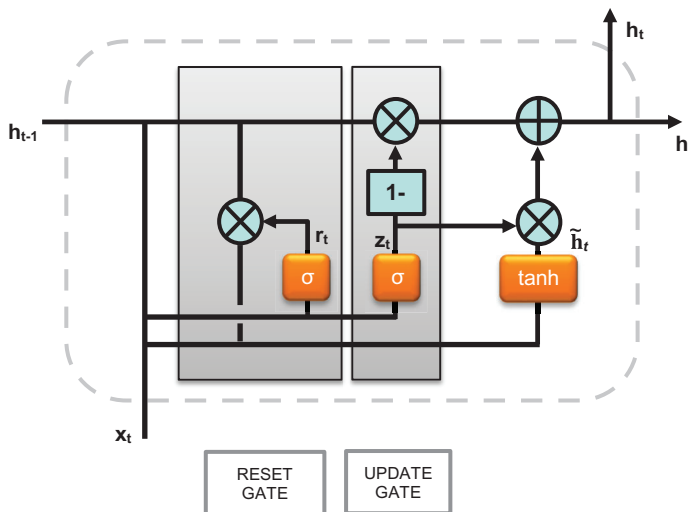


Figure 8. Inside a GRU cell (Source: Phi, 2018).

The components of a GRU unit include:

- The *reset gate* r_t at time t decides which past information to forget: $r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$.
- The *update gate* z_t at time t acts similarly as a combined *forget* and *input* gate of an LSTM unit i.e. decides which information to delete and which new information to add: $z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$.

In the above, W and b represent weight and bias vectors for each respective component, x_t is the input vector at times t , and h_t and h_{t-1} are the hidden state vectors at times t and $t-1$ respectively:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t.$$

This uses the *candidate activation vector* $\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h)$.

4.8. Deeper Dive into Model Architectures – XGBoost

XGBoost uses a *gradient-boosting* algorithm that uses *weak learners* as building blocks. *Weak learners* can be viewed as simplified models that are improved upon during the iterative training process (see below). In the case of a time series, the *weak learners* are *regression trees* i.e., decision trees that output continuous variables.

XGBoost is an *ensemble* method, which takes the aggregate of results from multiple smaller *weak learner* models. *Boosting* refers to the fact that the model is built *sequentially* i.e. a model using values from prior model iterations to produce improved subsequent model iterations. *Gradient* refers to the fact that a *gradient descent* algorithm is used to reduce errors in sequential models.

Broadly, the process underlying a gradient-boosting algorithm is (Analytics Vidhya, 2018) and summarised in Figure 9 below:

- i. An initial model, $F_1(X)$, is defined to predict a target variable, which will result in associated residuals, $r_1 = \text{actual value } (y) - \text{predicted value } (\hat{y})$.
- ii. A new model is fit to the residuals from the prior step, $F'_1(X)$.
- iii. The models from steps i and ii are combined to produce an improved model, $F_2(X)$.
- iv. We repeat this process (steps i to iii) but using model $F_2(X)$ as the starting model, and so on.

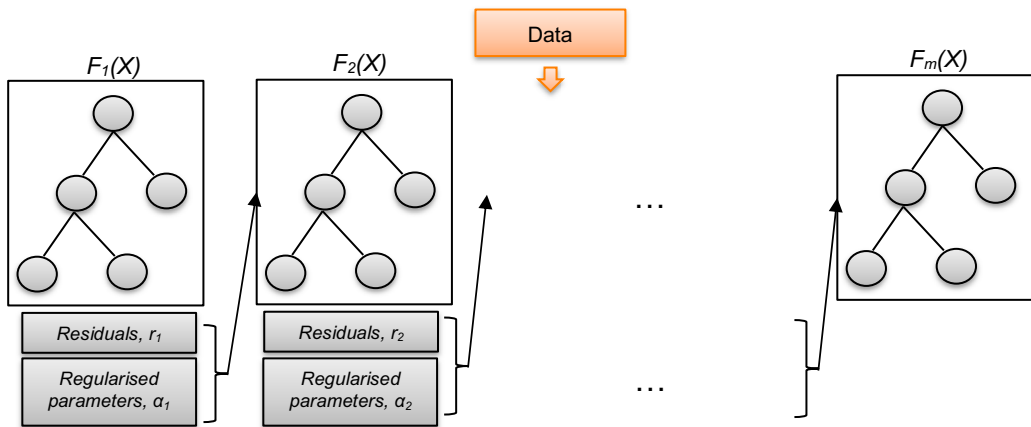


Figure 9. Overview of how XGBoost works (Source: Amazon Web Services, 2023).

The final model prediction $F_m(X) = F_{m-1}(X) + \alpha_m h_m(X, r_{m-1})$, where α_i and r_i are the regularisation parameters and residuals for the i^{th} tree, respectively, and h_i is a function to predict residuals. XGBoost also incorporates parallel processing, tree pruning, handling missing values and regularisation to avoid overfitting (Morde, 2019).

Launched in 2014, XGBoost won the *Higgs Machine Learning Challenge* (ATLAS Collaboration, 2014). Further, the recent popularity of XGBoost is increasing, with implementations of the approach having won several Kaggle awards (Kaggle, n.d.). Though XGBoost is predominantly used for classification problems, it can also be extended to regression problems including time series analysis.

5. Training the Models

5.1. A Note on the Code

We have mainly used Google's *Tensorflow* framework (see <https://www.tensorflow.org/>) and functions taken from the *Keras* library (see <https://keras.io/>) to build our neural networks, as well as Python-based libraries such as *Matplotlib* (see <https://matplotlib.org/>), *Numpy* (see <https://numpy.org/>) and *Pandas* (<https://pandas.pydata.org/>). Similarly, we have used open-source libraries for *XGBoost* (see <https://xgboost.ai/>) for the decision tree model analysis and *Optuna* (see <https://optuna.org/>) for hyperparameter optimisation. The code was run in *Google Colab* (see <https://colab.research.google.com/>).

5.2. Input Windows & Output Horizons

As mentioned earlier, for the purposes of this paper, we will use a *window* of 1 day to predict a *horizon* of 1 day i.e., 1 day prior to predict today's value and repeat this method over the data set (broadly 2013 to 2023 inclusive). In subsequent papers, we will adjust both the *window* and *horizon* size, as well as introduce stationarity.

5.3. Loss History

The underlying variables for a neural network model architecture can be grouped into *parameters* and *hyperparameters*. *Parameters* are adjusted by the model itself during the training process, while *hyperparameters* are set/adjustable by the modeller. We go into further detail for both below.

We have used the *Keras* default *Glorot Uniform Initialisation* method to initialise the weights and biases in the neural network. Glorot and Bengio (2010) provide further details. These weights and biases are then updated during the training process via a combination of *backpropagation* and *optimisation*, with the overall aim of minimising the *loss function* and hence producing a better-fitting model. *Backpropagation* calculates the gradient of the loss for each weight and bias in the network, and the *optimiser* subsequently uses these gradients to update any weights and biases.

When training models, we aim to reduce the size of the *loss function* that measures how accurate/ inaccurate the model outputs are for, in our case, a batch of data. The loss information is indirectly fed back into the model via *backpropagation* for a neural network, and weight/biases are updated via an Adam optimiser (Brownlee, 2021a) for the purposes of this paper, for batch sizes of 128 consecutive index values. Any updates to parameters are made without intervention from the modeller. This process is then repeated a number of times, with the aim of producing an improved model, with an overall reduced *loss function*.

The *loss function* used in our analysis was set to MAE. The MAE takes the average of the absolute differences between actual versus predicted values for each batch of data i.e. $\frac{1}{n} \sum_{i=1}^n |y - \hat{y}|$,

where y is the actual value and \hat{y} is the predicted value.

For *hyperparameter* optimisation, we used *Optuna* (see <https://optuna.org/>). The *hyperparameters* optimised include:

- The *number of units* (or neurons) in the neural network hidden layers.
- The number of iterations which information is passed through an LSTM cell or a GRU cell before moving on to the next layer.
- The *activation function* used in each layer.
- The *learning rate* applied to the Adam optimiser.
- *L2 regularisation* applied to the weights (i.e. *kernel regularisation*).
- *Filter size* for CNN models.

- *Return sequence* for LSTM and GRU, which indicates if a single value or sequential information is outputted to the next layer. Please note that given the analysis in this paper (effectively 1 day in, 1 day out) we do not expect this *hyperparameter* to have any material impact on the results.

For XGBoost, the following *hyperparameters* were included in our search space whilst tuning via *Optuna*:

- *Eta*, which represents the learning rate.
- *Gamma*, which is used to prune the branches.
- *Max depth*, which represents the maximum depth of each decision tree within the ensemble.
- *L2 regularisation* applied via a parameter called *reg_lambda* to leaf weights of individual decision trees.

Regularisation, Adam optimiser, *Optuna* and activation functions will be discussed in further detail in Sections 5.5–5.8.

5.4. Loss History Curves

Figure 10 is an example of the training outputs from one of the models, showing the training and validation loss of MAE after each *epoch*. An *epoch* represents the period when the entire data set has been passed through a neural network during the training/validation process.

As a model trains and converges to a solution, we would expect the loss to reduce after each *epoch* and hence the curve to decrease, as shown in Figure 10.

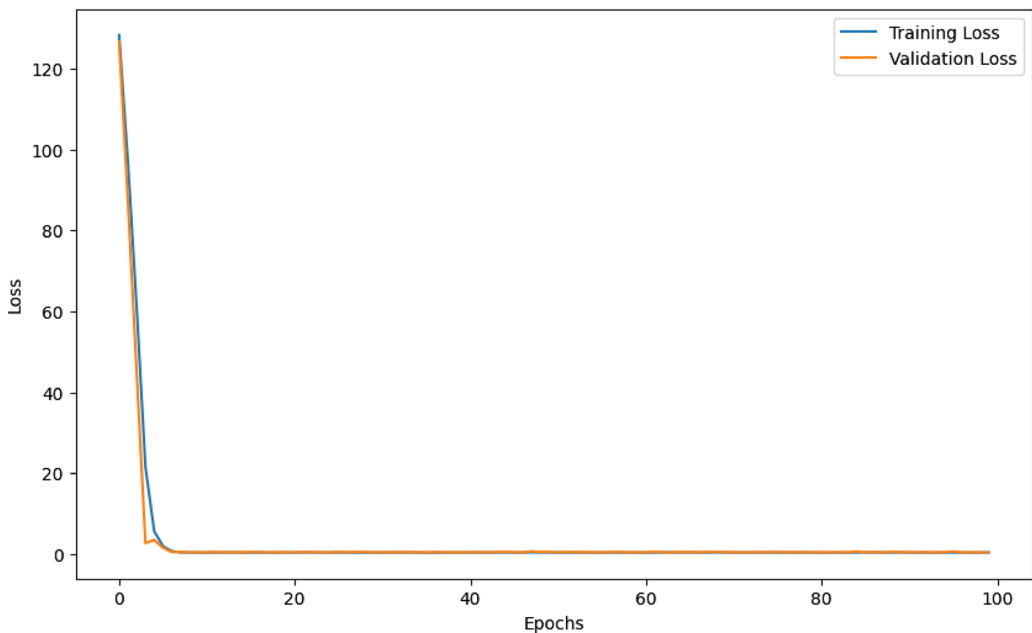


Figure 10. Sample loss history curve from training one of the models.

5.5. Adam Optimisation in More Detail

Optimisers are algorithms that are used to update the *parameters* of a neural network (weights and biases) during the training process, with the overall aim of minimising a set *loss function*, utilising gradients calculated via the process called *backpropagation*. Optimisers include *stochastic gradient descent*, *adaptive moment estimation* (Adam), *root mean square propagation* and Adagrad, *adaptive gradient algorithm* (Duchi et al, 2011).

For our analysis, we have used an Adam optimiser as it potentially converges to a solution more efficiently than other methods such as *stochastic gradient descent*.

Introduced in 2014 (Kingma and Ba, 2014), the Adam optimiser is *an algorithm for first-order gradient-based optimisation of stochastic objective functions*, based on *adaptive estimates of lower-order moments*.

The Adam optimiser combines the advantages of two other gradient descent methods (GeeksforGeeks, 2018) and (Musstafa, 2021):

1. *momentum* – storing exponentially weighted moving average of past gradients, to help overcome local minima and speed up convergence; and
2. *root mean square propagation* - storing the exponentially moving average of the past squared gradients, which helps in adapting the learning rates for each *parameter* individually.

The moving averages are used to calculate specific (*adaptive*) learning rates for each *parameter* (weight and bias).

Using similar notation as per the original published in 2014: $\Theta_{n+1} = \Theta_n - \alpha_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}$, where Θ_n and Θ_{n+1} are the *parameters* (a weight or bias in the neural network), at iteration n and $n+1$ respectively; \hat{m}_t is the bias-corrected first moment estimate; \hat{v}_t is the bias-corrected second raw moment estimate; α_t is the learning rate, and ϵ is a small value used to prevent division by zero.

Expanding for the terms in the formulae above:

- $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$, where β_1 is a *forgetting factor* and used for decaying the running average of the gradient; g_t is the gradient at time t along the parameter Θ . m_t is the exponential average of gradients along the parameter Θ .
- $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot (g_t)^2$, where β_2 is a *forgetting factor* and used for decaying the running average square of the gradient; and g_t is the gradient at time t along the parameter Θ . v_t is the exponential average of squares of gradients along the *parameter* Θ .
- $\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$ is the bias-corrected first moment estimate.
- $\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$ is the bias-corrected second raw moment estimate.

5.6. L2 Regularisation in More Detail

We have to be conscious of *overfitting* when training a neural network i.e., when the neural network in effect memorises or closely matches the training data set to the point that it is unable to predict effectively on unseen test data. To avoid this, we can use regularisation techniques to dampen this effect. Figure 11 shows different categories of regularisation techniques which we can apply to reduce any overfitting when training our models.

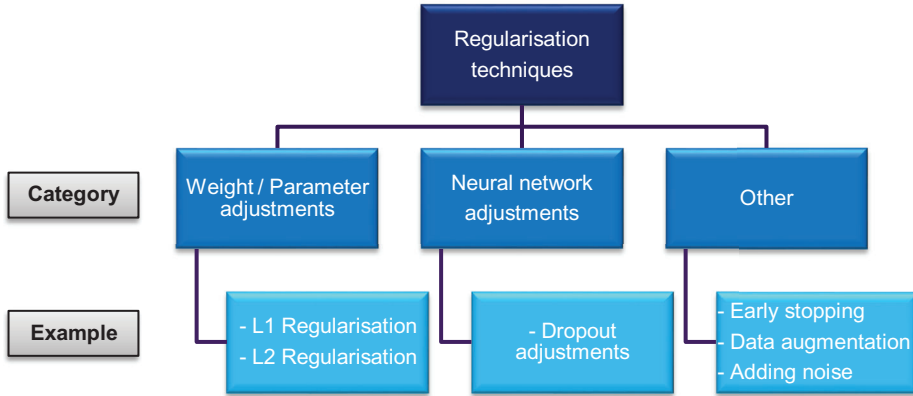


Figure 11. Graphical summary of regularisation techniques (Pramoditha, 2022).

In effect, L1 and L2 regularisation add in some form of penalty when training the model. For the purposes of our analysis, we have used L2 regularisation (also known as *Ridge Regression*) where we have allowed for adjustments to the weights in all non-baseline models via *Keras*'s in-built regularisation methods (Keras, n.d.).

Given that the number of input features is minimal, and we do not have a requirement to minimise the number of input features, we have used L2 regularisation.

With L2 regularisation, a squared penalty term is added to the *loss function*. For example, if we adjust the MAE formula $\frac{1}{n} \sum_{i=1}^n |y - \hat{y}|$ to allow for L2 regularisation, we obtain:

- $\frac{1}{n} \sum_{i=1}^n |y - \hat{y}| + \lambda \cdot \sum_{j=1}^p \beta_j^2$, where y is the actual value and \hat{y} is the predicted value, and β is the penalty term, $\lambda \in [0,1]$ is a regularisation parameter.

5.7. Optuna Hyperparameter Optimisation in More Detail

Hyperparameters, such as the number of neurons per layer, the activation function and learning rate, are set by the modeller and hence influence the overall architecture of a neural network. *Parameters* such as weights and biases are then adjusted by the network during the training process without intervention from the modeller. Various methods to find optimal *hyperparameters* exist including:

1. Manual tuning by the modeller.
2. Grid search and random search approaches, which loop across a search space and try various candidates (combinations of *hyperparameters*), though search results do not feed into future searches.
3. Bayesian optimisation where the aim is to produce a probability distribution of the *objective function* (i.e. loss function). Unlike grid and random searches, Bayesian optimisation techniques keep track of prior evaluations and use these values in future runs.

Hence, Bayesian optimisation techniques should produce solutions that converge more efficiently for more complex problems when compared to manual tuning and grid search approaches.

For our analysis, we used the open-source library *Optuna* to vary (*tune*) the *hyperparameters*. *Optuna* is an *automatic hyperparameter optimisation software framework, particularly designed for machine learning* (Optuna, n.d.). We used the Bayesian optimisation algorithm called Tree-

structure Parzen Estimator, TPE which was introduced in 2011. Please see Bergstra et al. (2011) for more details on this.

Bayesian optimisation techniques involve:

1. Constructing a surrogate probability model of the objective function, which is a simplified version and hence computationally less expensive version of the actual probability distribution of the objective function.
2. Using an acquisition function to choose the next set of candidates to evaluate.

For the purposes of our analysis, we used *Sequential Model-Based Optimisation* which is an iterative process that builds the surrogate probability model of the objective function using an *acquisition function* of TPE (Bergstra et al., 2011).

Introduced in 2013, TPE uses a mixture of 2 Gaussian distributions to set *parameter* values: i. for successful *candidates*, $l(x)$; and ii. unsuccessful *candidates*, $g(x)$: $P(x|y) = l(x)$ if $y < y^*$, or $g(x)$ if $y \geq y^*$, where x is the single *hyperparameter*, y is the loss, y^* is a threshold and $P(x|y)$ is the probability of observing a single *hyperparameter* given a certain loss value.

The aim is to optimise an acquisition function - *Expected Improvement* (EI), which quantifies any potential gain in the *objective function* value from sampling a particular point in the parameter space. The aim of EI is to balance *exploration* (sampling points with uncertain outcomes) and *exploitation* (sampling points that are likely to improve the current best results). Using the same notation as per the original paper (Bergstra et al., 2011):

$$EI_{y^*}(x) = \frac{\gamma y^* l(x) - l(x) \int_{-\infty}^{y^*} p(y) dy}{\gamma l(x) + (1 - \gamma) g(x)}$$

where x , $g(x)$, $l(x)$, y and y^* are as per above, and γ is some quantile of the observed y values.

For an introductory background to *Optuna*, please see Lim (2022) and Akiba et al. (2019). For more background to TPE, please see Watanabe (2023). For more details on algorithms for hyperparameter optimisation, please see Bergstra et al. (2011).

5.8. Activation Functions in More Detail

As mentioned earlier, the overall aim of activation functions is to introduce non-linearity to a neural network in a regression analysis such as a time series prediction model. Table 3 shows the set of activation functions used during hyperparameter optimisation in our analysis.

Table 3. The Set of Activation Functions Used During Hyperparameter Optimisation

ACTIVATION function	Mathematical Formula
relu	0 for $x < 0$ x for $x \geq 0$
elu	$\alpha(e^x - 1)$ for $x < 0$ x for $x \geq 0$
gelu	$x \cdot \Phi(x)$, where $\Phi(x)$ is the standard Gaussian function
swish	$\frac{x}{1 + e^{-\beta x}}$
linear	$a \cdot x$
sigmoid	$\frac{1}{1 + e^{-x}}$
tanh	$\frac{2}{1 + e^{-2x}} - 1$

For more background on activation functions, please see GeeksforGeeks (2018) and Nwankpa et al. (2018).

6. Results

6.1. Summary

In addition to the Baseline model, we ran multiple models from each of the five distinct categories, where we varied the model architecture within the same category e.g. by varying the number of hidden layers. Please see Section 3.3 for more details on the model categories used in our analysis as well as Appendix C.

In this section, we summarise the best-performing model from each model category and use the subscript *best* or suffix of *_best* to represent this.

Figure 12 shows the predicted value against expected value, over the test data set range from the Baseline model.

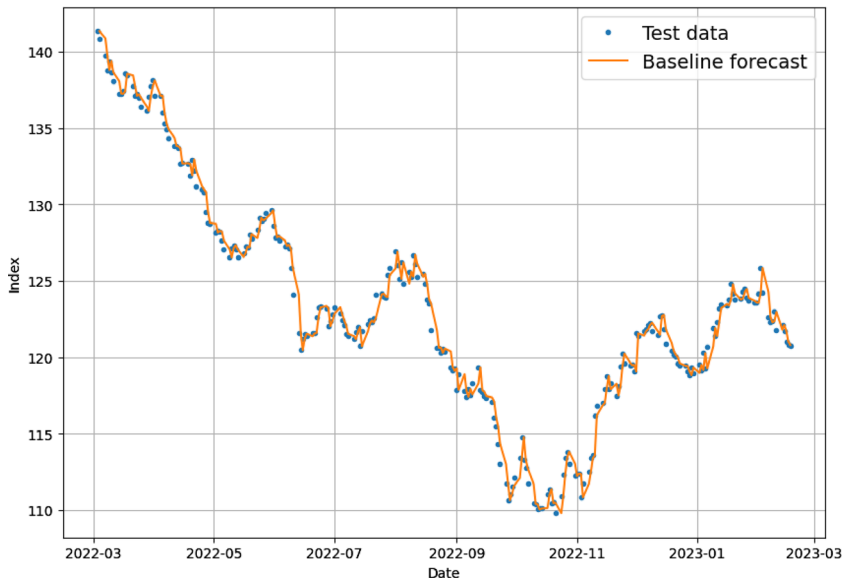


Figure 12. Predictions from the Baseline model over the test range versus actual data.

As can be seen, the overall Baseline model is a fairly good fit to the actual values over the test date range for this index, reflective of the fact that the index has a low daily volatility. Diving deeper, Table 4 summarises the cost functions i.e., MAE and MAPE, for the best-performing model from each category analysed.

Table 4. Comparison of Performance of the Best Performing Model from Each Model Category against the Baseline Model

Model Category	Model	MAE (3 d.p.)	MAE in Relation to the Baseline Model	MAPE (4 d.p.)	MAPE in Relation to the Baseline Model
0	Baseline	0.610	–	0.4969%	–
1	DNN _{best}	0.607	–0.003	0.4947%	–0.0022%
2	CNN _{best}	0.611	+0.001	0.4977%	+0.0008%
3	LSTM _{best}	0.609	–0.001	0.4966%	–0.0003%
4	GRU _{best}	0.615	+0.005	0.5011%	+0.0042%

Please note that we have not fully listed the results from all models in each category for ease of comparison. Also, Table 4 excludes XGBoost which is discussed in Section 6.2.

Figure 13 gives a graphical comparison of the results in order of performance, showing the best performing model from each model category excluding XGBoost.

We have adjusted the y-axis accordingly to highlight the differences in performance.

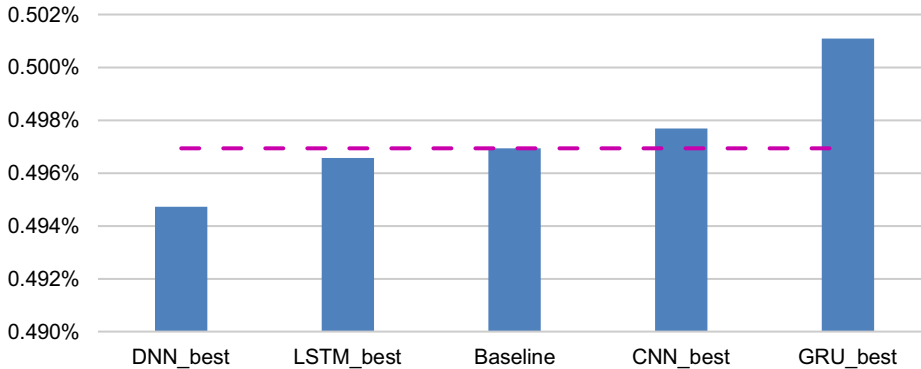


Figure 13. Comparison of best-performing models using MAPE from each model category excluding XGBoost.

On both MAE and MAPE measures, the DNN and LSTM models marginally outperform the Baseline model. The Baseline model in turn marginally outperforms the CNN model, whilst the GRU model performs worst when compared to the other models. Each of the best-performing models from each model category perform within $\pm 0.1\%$ accuracy based on a MAPE measure. Please note that these results are based on the data set examined along with the hyperparameter search ranges and model architecture (as described in Appendix C).

Different results may be obtained should we use, for example, different data ranges. If we exclude data from 2022 and rerun the analysis (assuming a similar training/validation/test split of 70%/20%/10% and hence the date ranges will now differ), we obtain similar conclusions: the models are close to the Baseline model with differences up to approximately $\pm 0.1\%$ (based on a MAPE measure). However, the order of best-performing models and hence those which beat the Baseline model differ to those above with DNN, CNN and GRU now beating the Baseline model and the LSTM model performing worse. Again, given the magnitude of difference in performance between the Baseline model and each other model, the conclusions are not definitive.

6.2. A Note on XGBoost

Although XGBoost is typically used for categorisation problems, it can be extended to time series analysis. For example, please see Brownlee (2021b) and Kaggle (2019).

Using XGBoost on our data set produced unexpected results. As can be seen in Figure 14, the model gives a relatively poor fit compared with other models as it was unable to deal with the “troughs” in the test data set range. The predictions flat-lined and did not go below a level of 123 approximately.

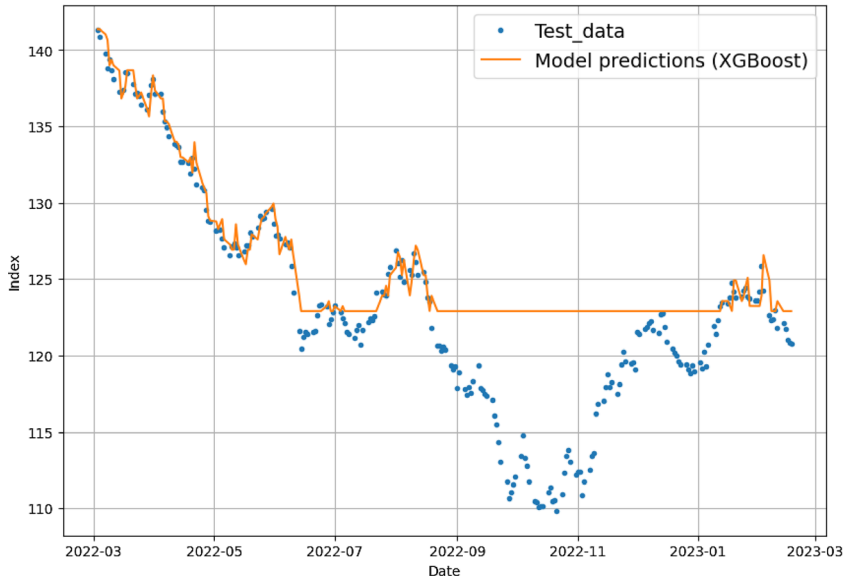


Figure 14. Outputs from XGBoost model runs over the test range versus actual data.

Similarly, if we look at the MAE and MAPE, we see that the XGBoost model is further off the Baseline model predictions when compared with the models earlier (Table 5).

Table 5. Comparison of Performance of XGBoost against the Baseline Model

Model	MAE (3 d.p.)	MAE Relation to Baseline Model	MAPE (4 d.p.)	MAPE Relation to Baseline Model (4 d.p.)
Baseline	0.610	-	0.4969%	-
XGBoost	2.840	+2.23	2.4445%	+1.9476%

It turns out that, although decision tree models such as XGBoost can be applied to time series problems, they struggle to extrapolate values and predict outside of the original training (and validation) data set range (where the minimum in the train data set is approximately 122 in this case). Please see Mavuduru (2020) for further details.

7. Conclusions and Next Steps

7.1. Conclusions

Based on our analysis, given the data range and training/validation/test splits of the *S&P Green Bond Index*, we can draw the following conclusions:

1. The Baseline model is fairly accurate. The forecasts produced by the Baseline model results in an error of approximately 0.5% MAPE over the test data set based on the date ranges used in the body of the paper.
2. The neural network models analysed do not materially (if at all) beat the Baseline model.
3. In saying this, the best-performing models came from the DNN and LSTM categories based on our analysis of this data set. The best-performing models from the CNN and GRU model categories performed worse than the Baseline model.
4. All models differed by up to approximately +/-0.1% from the Baseline model when analysing their MAPE over the test data set range.

5. The XGBoost model was a poor fit compared to the other models analysed, especially over the period July 2022 to February 2023. This seems to be due to decision tree models being unable to predict values outside of the training (and validation) data values.
6. Although the overall performance by the neural networks was not materially an improvement on the Baseline model, we hope that the reader can see that there is potential merit in using neural networks for tackling a time series problem in general i.e., beyond green bond applications. For example, for more complex scenarios, neural networks may provide a viable supplementary/alternative view to traditional techniques. We hope to explore and evidence this further in future papers.

7.2. Future Considerations

The aim of this paper was to introduce the initial stages of our analysis using non-traditional techniques including neural networks. Although we have not considered the points below in this paper, further areas we aim to explore in future papers include:

1. Extending our analysis to a wider data set beyond the *S&P Green Bond Index*, for example to *Bloomberg Barclays MSCI's Green Bond Index* as well as other GSS bonds. By widening the index analysed, it would be good to understand if similar results are produced as per Section 6 across different data sets and indices, and if techniques such as neural networks can successfully be applied to the wider GSS-bond universe.
2. Extending our analysis to a multi-step projection model by increasing the input window and/or output horizon, where we compare results by producing predictions over the next week or month, say. Similarly, we aim to extend our analysis to use alternative windowing techniques such as cross-validation windowing, and pre-processing the data, for example via normalisation or lognormal techniques. In doing so, we can introduce the idea of stationarity into our analysis and build on ideas discussed in Peters et al. (2022) where a hybrid SARIMAX-LSTM model is used. We hope that this will improve the model and model outputs, as we are providing additional data to the model for it to learn from and hopefully interpret underlying patterns, which is not possible if only one day's prior data is fed into the model at a time.
3. Exploring relationships with other indices and hence perform a multivariate analysis. For example, research paper by Gao et al. (2023) suggests that green bond indices are correlated with commodities such as oil. Hence, expanding the above analysis to include this could improve the model outputs. We would expect some form of wider influence and relationship with the general market. By exploring a univariate time series analysis, as we have done in this paper, we are ignoring any such potential relationships.
4. Explore the concept of greenium further and see how this relates to other indices and varies over time for example pre- and post-COVID. For example, Cui et al. (2022) examines how green bonds have reacted to the COVID pandemic.
5. Expand the neural network models used to more exotic models such as a CEEMDAN-LSTM architecture or N-BEATS architecture. For example, the research paper by Wang et al. (2022) suggests that a CEEMDAN-LSTM is more effective for time series analysis on green bonds compared to an LSTM model when analysing a time series on green bonds. The N-BEATS model is explored in Oreshkin et al. (2019) and has been designed with the specific aim of tackling time series problems. For example, it outperformed the Makridakis time series M4 competition model winner by 3% (Oreshkin et al., 2019).

Hence, in exploring the above, we hope that such models will provide improved results compared to the initial models chosen in this report as well as wider insight into GSS bonds and other time series problems in general.

References

- Akiba, T., Sano, S., Yanase, T., Ohta, T. and Koyama, M. (2019). Optuna: a next-generation hyperparameter optimization framework, available at <https://arxiv.org/pdf/1907.10902.pdf> (accessed 30 September 2023).
- Amazon Web Services. (2023). How XGBoost works, available at <https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-HowItWorks.html> (accessed 30 September 2023).
- Analytics Vidhya. (2018). Introduction to XGBoost algorithm in machine learning, available at <https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/> (accessed 30 September 2023).
- ATLAS Collaboration. (2014). Machine learning wins the Higgs challenge, 20 Nov, available at <https://atlas.cern/updates/news/machine-learning-wins-higgs-challenge> (accessed 30 September 2023).
- Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization, available at https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf (accessed 30 September 2023).
- Brownlee, J. (2020). Time series forecasting as supervised learning, Machine Learning Mastery, 15 Aug, available at <https://machinelearningmastery.com/time-series-forecasting-supervised-learning/> (accessed 30 September 2023).
- Brownlee, J. (2021a). Gentle introduction to the Adam optimization algorithm for deep learning, Machine Learning Mastery, 13 January, available at <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/> (accessed 8 January 2024).
- Brownlee, J. (2021b). How to use XGBoost for time series forecasting, Machine learning Mastery, 19 March, available at <https://machinelearningmastery.com/xgboost-for-time-series-forecasting/> (accessed 8 January 2024).
- CFI Team. (n.d.). Green bond, available at <https://corporatefinanceinstitute.com/resources/esp/green-bond/> (accessed 30 September 2023).
- Climate Bonds Initiative. (2020). \$1Trillion mark reached in global cumulative green issuance: climate bonds data intelligence reports: latest figures, 15 Dec, available at <https://www.climatebonds.net/2020/12/1trillion-mark-reached-global-cumulative-green-issuance-climate-bonds-data-intelligence> (accessed 30 September 2023).
- Climate Bonds Initiative. (2021). 2021 Already a record year for green finance with over \$350bn Issued!, 2 Nov, available at <https://www.climatebonds.net/2021/11/2021-already-record-year-green-finance-over-350bn-issued> (accessed 30 September 2023).
- Climate Bonds Initiative. (n.d.). Interactive data platform – climate bonds initiative, available at <https://www.climatebonds.net/market/data/> (accessed 30 September 2023).
- Cui, T., Suleman, M., & Zhang, H. (2022). Do the green bonds overreact to the COVID-19 pandemic? *Finance Research Letters*, 49, 103095, available at <https://www.sciencedirect.com/science/article/pii/S1544612322003208> (accessed 30 September 2023).
- Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization, available at <https://jmlr.org/papers/volume12/duchi11a/duchi11a.pdf> (accessed 10 January 2024).
- European Commission. (2021). Proposal for a regulation of the European parliament and of the council on European green bonds, available at https://eur-lex.europa.eu/resource.html?uri=cellar:e77212e8-df07-11eb-895a-01aa75ed71a1.0001.02/DOC_1&format=PDF (accessed 8 January 2024).
- European Commission. (2023a). Sustainable finance: commission welcomes political agreement on European green bond standard, Daily News, 1 March, available at https://ec.europa.eu/commission/presscorner/detail/en/mex_23_1301 (accessed 8 January 2024).
- European Commission. (2023b). The European green bond standard – supporting the transition, available at https://finance.ec.europa.eu/sustainable-finance/tools-and-standards/european-green-bond-standard-supporting-transition_en (accessed 8 January 2024).
- Gao, L., Guo, K., & Wei, X. (2023). Dynamic relationship between green bonds and major financial asset markets from the perspective of climate change. *Frontiers in Environmental Science*, 10, 1109796, available at <https://www.frontiersin.org/articles/10.3389/fenvs.2022.1109796/full> (accessed 30 September 2023).
- GeeksforGeeks. (2018). Intuition of Adam Optimizer, available at <https://www.geeksforgeeks.org/intuition-of-adam-optimizer/> (accessed 30 September 2023).
- Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks, available at <https://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf> (accessed 30 September 2023).
- Iberdrola. (n.d.). Green bonds, What are green bonds and what are they for?, available at <https://www.iberdrola.com/sustainability/investments-green-bonds> (accessed 30 September 2023).
- IBM. (n.d.). What is a neural network?, available at <https://www.ibm.com/topics/neural-networks> (accessed 8 January 2024).
- JP Morgan Asset Management. (2023). How green, social and sustainability bonds could change the world, Financial Times, available at <https://jpmam.ft.com/how-green-social-and-sustainability-bonds-could-change-the-world> (accessed 30 September 2023).

- Kaggle.** (n.d.). Winning solutions of Kaggle competitions, available at <https://www.kaggle.com/code/sudalairajkumar/winning-solutions-of-kaggle-competitions> (accessed 8 January 2024).
- Kaggle.** (2019). [Tutorial] time series forecasting with XGBoost, available at <https://www.kaggle.com/code/robikscube/tutorial-time-series-forecasting-with-xgboost> (accessed 8 January 2024).
- Keras.** (n.d.). Layer weight regularizers, available at <https://keras.io/api/layers/regularizers/> (accessed 8 January 2024).
- Kingma, D., & Ba, J.** (2014). ADAM: a method for stochastic optimization, available at <https://arxiv.org/pdf/1412.6980.pdf> (accessed 30 September 2023).
- Latham & Watkins.** (2023). The European green bond standard – the new green bond “Gold standard”?, 8 Nov, available at <https://www.lw.com/admin/upload/SiteAttachments/The-European-Green-Bond-Standard-The-New-Green-Bond-Gold-Standard.pdf> (accessed 30 September 2023).
- Lim, Y.** (2022). State-of-the-art machine learning hyperparameter optimization with Optuna, Medium, 31 March, available at <https://towardsdatascience.com/state-of-the-art-machine-learning-hyperparameter-optimization-with-optuna-a315d8564de1> (accessed 8 January 2024).
- Mavuduru, A.** (2020). Why XGBoost can't solve all your problems, Medium, 10 Nov, available at <https://towardsdatascience.com/why-xgboost-cant-solve-all-your-problems-b5003a62d12a> (accessed 8 January 2024).
- Michetti, C., Chouhan, N., Harrison, C., & MacGeoch, M.** (2023). Sustainable debt, global state of the market 2022, available at https://www.climatebonds.net/files/reports/cbi_sotm_2022_03e.pdf (accessed 30 September 2023).
- Mitchell, N.** (2022). Easy being green: are investors persuaded by green bonds?, *The Actuary*, 4 May, available at <https://www.theactuary.com/2022/05/03/easy-being-green-are-investors-persuaded-green-bonds> (accessed 30 September 2023).
- Morde, V.** (2019). XGBoost algorithm: long may she reign!, Medium, 8 April, available at <https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d> (accessed 30 September 2023).
- Musstafa.** (2021). Optimizers in deep learning, Medium, 27 March, available at <https://medium.com/mllearning-ai/optimizers-in-deep-learning-7bf81fed78a0> (accessed 30 September 2023).
- Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S.** (2018). Activation functions: comparison of trends in practice and research for deep learning, available at <https://arxiv.org/pdf/1811.03378.pdf> (accessed 30 September 2023).
- Optuna.** (n.d.). Optuna: a hyperparameter optimization framework, available at <https://github.com/optuna/optuna> (accessed 30 September 2023).
- Oreshkin, B., Carpov, D., Chapados, N., & Bengio, Y.** (2019). N-BEATS: neural basis expansion analysis for interpretable time series forecasting, available at <https://arxiv.org/abs/1905.10437> (accessed 30 September 2023).
- Peters, G., Zhu, R., Tzougas, G., Rabitti, G., & Yusuf, I.** (2022). The role and significance of green bonds in funding transition to a low carbon economy: a case study forecasting portfolios of green bond instrument returns, available at <https://deliverypdf.ssrn.com/delivery.php?ID=898031068069018007094111088014124113024008049068035042027074089093008086104111123102101009029001009024018088074018112095126019098004058073081093065065092124071070073065014043110016004108091127092105118003090105018110087006030077107084090024008027083127> (accessed 30 September 2023).
- Pramoditha, R.** (2022). How to apply L1 and L2 regularization techniques to Keras models, Medium, 6 July, available at <https://medium.com/data-science-365/how-to-apply-l1-and-l2-regularization-techniques-to-keras-models-da6249d8a469> (accessed 30 September 2023).
- Phi, M.** (2018). Illustrated guide to LSTM's and GRU's: a step by step explanation, Medium, 24 Sept, available at <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21> (accessed 30 September 2023).
- Reed, J.** (2023). India raises \$1bn from maiden green bond sale, *Financial Times*, available at <https://www.ft.com/content/381068de-9fe2-425a-a6ca-c59cd8acc522> (accessed 30 September 2023).
- Ryan, T.J.J.** (2020). LSTMs explained: a complete, technically accurate, conceptual guide with Keras, Medium, 2 Sept, available at <https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2> (accessed 30 September 2023).
- Segal, T.** (2022). Green bond: types, how to buy, and FAQs, 21 Sep, available at <https://www.investopedia.com/terms/g/green-bond.asp> (accessed 30 September 2023).
- Shrivastava, S.** (2020). Cross validation in time series, Medium, 14 January, available at <https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4> (accessed 8 January 2024).
- S&P.** (2023). S&P green bond indices: methodology, February, available at <https://www.spglobal.com/spdji/en/documents/methodologies/methodology-sp-green-bond-indices.pdf> (accessed 8 January 2024).
- Statistica Research Department.** (2023). Green bonds worldwide – statistics & facts, available at <https://www.statista.com/topics/9217/green-bonds-market-worldwide> (accessed 30 September 2023).
- Wang, J., Tang, J., & Guo, K.** (2022). Green bond index prediction based on CEEMDAN-LSTM. *Frontiers in Energy Research*, 9, 793413, available at <https://www.frontiersin.org/articles/10.3389/fenrg.2021.793413/full> (accessed 30 September 2023).
- Watanabe, S.** (2023). Tree-structured Parzen estimator: understanding its algorithm components and their roles for better empirical performance, available at <https://arxiv.org/pdf/2304.11127.pdf> (accessed 30 September 2023).

The World Bank. (2023). Green, Social, and Sustainability (GSS) bonds, January, available at <https://thedocs.worldbank.org/en/doc/98c3baab0ea4fc3da4de0e528a5c0bed-0340012023/original/GSS-Quarterly-Newsletter-Issue-No-2.pdf> (accessed 30 September 2023).

Wrobel, S. (2023). Israel raises \$2 billion in its first-ever issuance of 10-year green bonds, The Times of Israel, 11 Jan, available at <https://www.timesofisrael.com/israel-raises-2-billion-in-its-first-ever-issuance-of-10-year-green-bonds/> (accessed 30 September 2023).

Appendix A. Working Party Members and Acknowledgements

Working Party Members

The Working Party is made up of two sub-groups: practitioners and academics.

Below is a list of current Working Party members for the practitioner group, along with details of their position within the Working Party and corresponding LinkedIn link.

Name	Position	LinkedIn
Debashish Dey	Chair	Link
Cem Öztürk	Member	Link
Shubham Mehta	Member	Link

Please note that this Working Party sits within the Lifelong Learning pillar of the IFoA. For further details of the Data Science section of the IFoA, please see here.

Acknowledgements

Please see below acknowledgements for the review of the code and paper:

- Code review by Cem Öztürk [[Link](#)].
- Technical review of paper by:
 - Dr Alexey Mashechkin [[Link](#)].
 - Dr Arjit Das [[Link](#)].
 - Assoc. Prof. George Tzougas [[Link](#)]
- Review of graphs and tables within the paper by Shubham Mehta [[Link](#)].

Appendix B. List of Abbreviations

Below is a list of abbreviations used within this paper.

Abbreviation	Explanation
ARIMA	Auto-Regressive Integrated Moving Average
CEEMDAN	Complete Ensemble Empirical Mode Decomposition with Adaptive Noise
CNN	Convolutional Neural Network
DNN	Deep Neural Network
EIB	European Investment Bank
FTSE	Financial Times Stock Exchange
GRU	Gated Recurrent Unit
GSS bonds	Green, Social and Sustainability bonds
IFoA	Institute and Faculty of Actuaries
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MSCI	Morgan Stanley Capital International
N-BEATS	Neural Basis Expansion Analysis for Interpretable Time Series
RNN	Recurrent Neural Network
S&P	Standard and Poor's
SARIMAX	Seasonal Auto-Regressive Integrated Moving Average with Exogenous factors
TPE	Tree-structure Parzen Estimator

Appendix C. Summary of Models Analysed

Below is a summary of the models analysed in this paper, with a brief description of the underlying model architecture. As mentioned earlier in the paper, the best-performing model per category below based on our analysis and data was highlighted earlier in Section 6 of this paper.

Model	Abbreviated Name	Category	Description of Architecture
0	Baseline	Baseline	Baseline model which assumes today's value is the same as yesterday's value.
1	DNN0	DNN	Feedforward artificial neural network with one hidden dense layer.
2	DNN1		Feedforward artificial neural network with two hidden dense layers.
3	DNN2		Feedforward artificial neural network with three hidden dense layers.
4	CNN0	CNN	Convolutional neural network, with one Conv1D layer and no additional hidden layers.
5	CNN1		Convolutional neural network, with one Conv1D layer and one hidden dense layer.
6	LSTM_0HL_F	LSTM	LSTM neural network, with one LSTM layer, return sequence set to false, and no additional hidden layers.
7	LSTM_0HL_T		LSTM neural network, with one LSTM layer, return sequence set to true, and no additional hidden layers.
8	LSTM_1HL_F		LSTM neural network, with one LSTM layer, return sequence set to false, and one additional hidden dense layer.
9	LSTM_1HL_T		LSTM neural network, with one LSTM layer, return sequence set to true, and one additional hidden dense layer.
10	GRU_0HL_F		GRU
11	GRU_0HL_T	GRU neural network, with one GRU layer, return sequence set to true, and no additional hidden dense layers.	
12	GRU_1HL_F	GRU neural network, with one GRU layer, return sequence set to false, and one additional hidden dense layer.	
13	GRU_1HL_T	GRU neural network, with one GRU layer, return sequence set to true, and one additional hidden dense layer.	
14	XGBoost	XGBoost	XGBoost model. Hyperparameters analysed are described earlier in this paper.

Appendix D. Useful Links

Below is a list of links that we hope that the reader finds useful.

- IBM's introductory series on neural networks: <https://www.ibm.com/topics/neural-networks>
- IFoA's Certificate in Data Science programme: <https://www.actuaries.org.uk/news-and-insights/news/data-science-credential>
- IFoA's Data Science Lifelong Learning page: <https://actuaries.org.uk/about-us/practice-areas/cross-practice-work/data-science/>
- Google *Tensorflow*'s tutorial on time series: https://www.tensorflow.org/tutorials/structured_data/time_series

Cite this article: Dey D. (2024). Time series analysis of GSS bonds. *British Actuarial Journal*. <https://doi.org/10.1017/S1357321724000011>