RESEARCH REPORT

Investigating explainablity methods in recurrent neural network architectures for financial time-series data

Author:
Warren FREEBOROUGH
Student number:
723388
Orchid:
0000-0003-3825-3401

Supervisor:
Prof. Terence VAN ZYL

UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

A research report proposal submitted in partial fulfillment of the requirements for the degree of Master of Science in the field of e-Science

in the

School of Computer Science and applied Mathematics
University of Witwatersrand

June 9, 2022

Declaration of Authorship

I, Warren FREEBOROUGH student number 723388, declare that this research report titled, "Investigating explainablity methods in recurrent neural network architectures for financial time-series data" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this research report has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this research report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the research report is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

[illegible]

Warren FREEBOROUGH
June 9, 2022

UNIVERSITY OF WITWATERSRAND

Abstract

Faculty of Science
School of Computer Science and applied Mathematics

Master of Science in the field of e-Science

Investigating explainablity methods in recurrent neural network architectures for financial time-series data

by Warren FREEBOROUGH

Statistical methods were traditionally used for time series forecasting. However, new hybrid methods demonstrate competitive accuracy, leading to increased machine learning-based methodologies in the financial sector. However, very little development has been seen in explainable AI (XAI) for financial time series prediction, with a growing mandate for explainable systems. This study aims to determine if the existing XAI methodology is transferable to the context of financial time series prediction. Four popular methods, namely: ablation, permutation, added noise, integrated gradients, were applied to an RNN, LSTM, and a GRU network trained S&P 500 stocks data to determine the importance of features, individual data points and specific cells in each architecture. The explainability analysis reveals that GRU displayed the most significant ability to retain long-term information, while the LSTM disregarded most of the given input and instead showed the most notable granularity to the considered inputs. Lastly, the RNN displayed features indicative of no long-term memory retention. The applied XAI methods produced complementary results, reinforcing paradigms on significant differences in how different architectures predict. The results show that these methods are transferable in the financial forecasting sector, but a more sophisticated hybrid prediction system requires further confirmation.

Acknowledgements

I would like to acknowledge the efforts of my supervisor Prof. Terence van Zyl whom has provided considerable technical advice throughout the project and time to review the content. Furthermore, I would like to thank my family for the support whilst undertaking this degree. Lastly, I would like to thank the National e-Science Postgraduate Teaching and Training Platform for funding and enabling myself to undertake this research.

Contents

List of Figures

List of Tables

List of Abbreviations

RNN	**R**ecurrent-**N**eural **N**etwork
ES	**E**xponential **S**moothing
ARIMA	**A**uto**R**egressive **I**ntergrated
MA	**M**oving **A**verage
LSTM	**L**ong **S**hort **Term** **M**emory
GRU	**G**ated **R**ecurent **U**nit
ML	**M**achine **L**earning
STL	**S**easonal and **T**rend using **L**oess
NLP	**N**atural **L**anguage **P**rocessing
AOPC	**A**rea **O**ver **P**erturbation **C**urve
AI	**A**rtificial **I**ntelligence
XAI	e**X**plainable **A**rtificial **I**ntelligence
SMAPE	**S**ymmetric **M**ean **A**bsolute **P**ercentage **E**rror

List of Symbols

$T_{l,f}$	Time series of length l and f features
w	Sequence length of input
h	Prediction window
A	Ablation area

List of Publications

1. W. Freeborough and T. van Zyl, T. "Investigating Explainability Methods in Recurrent Neural Network Architectures for Financial Time Series Data". *Applied Sciences*, 12(3), p.1427, 2022 [1]

Chapter 1

Introduction

The financial sector comprises a diverse group of businesses, such as banks, insurance, real estate and investment companies. Their essential service is to provide financial services to customers in the form of loans, investments and payouts. Subsequently, the financial sector deals with large amounts of money, bringing inherent risks.

1.1 Background

In order to optimise profits, businesses in the financial sector are keenly interested in various mathematical techniques that aim to quantify risk and predict future trends. Understanding the probability of risk and its associated costs allows businesses to make decisions that minimise losses. Equally, predicting future events based on past information allows businesses to identify opportunities that have yet presented themselves. Time series forecasting is one such means to achieve this.

1.1.1 Time series Forecasting

Time series data changes with respect to time, and generally, these temporal changes and patterns are of interest. Identification and understanding of these patterns allow for forecasting, whereby future predictions are made based on past knowledge. Most time series comprises three features: trend, seasonality, and cyclics. Trend describes the long-term changes within the data, namely: increasing, decreasing or both (changing trend). Seasons, or the seasonality of the data, describe repetitive and predictable frequency patterns that usually occur due to time factors, such as days of the week or time of year. Lastly, cyclic patterns are similar to seasonal patterns; however, they do not demonstrate a fixed and predictable frequency [2].

The finance industry shows particular interest when it comes to time series forecasting. Such a high-stakes environment coupled with the use of automatic electronic trading systems has driven traders to search for increasing more accurate tools to predict future outcomes. For most of this time statistical methods proved to be the most simple and successful for financial time series forecasting [3].

1.1.2 Statistical Methods

Statistical methods refer to method that leverage statistical principles in its functioning. Statistical methods rely on assumptions regarding the properties of the data such as normality, linearity and equality of variance. It is through understanding the properties present in the data that allow for extrapolation given that future data points follow the same distributions and properties.

Holt-Winters Smoothing

Initially, Holt proposed using exponentially decaying weighted averages of previous observations to make future predictions [4]. This proposition reasons that recent time-points contribute more than distant time-points to the forecast. In its simplest form it follows,

$$\begin{aligned} l_t &= \alpha y_t + (1-\alpha) l_{t-1} \\ \hat{y}_{t+1|T} &= \sum_{j=0}^{T-1} \alpha(1-\alpha)^j y_{T-j} + (1-\alpha)^T l_0, \end{aligned} \tag{1.1}$$

where $0 < \alpha < 1$ is the smoothing parameter at time point t. However, both frequency and trend is not considered, and while Holt went on to incorporate trend into the model, it was Winters that included seasonality [5]. Winters proposed equations for level (l_t), trend (b_t) and seasonality (s_t):

$$\begin{aligned} \hat{y}_{t+h|t} &= hb_t + s_{t+h-m(k+1)} \\ l_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \\ b_t &= \beta^*(l_t - l_{t-1}) + (1-\beta^*) b_{t-1} \\ s_t &= \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma) s_{t-m}, \end{aligned} \tag{1.2}$$

with three corresponding smoothing parameters: $0 < \alpha, \beta^*$ and $\gamma < 1$, and m representing the length of seasonality. Whilst exponential smoothing (ES) is considered a standard prediction algorithm as it is found to be quite accurate in forecasting, despite its simplicity and susceptibility to outliers [6].

ARIMA

The auto-regressive integrated moving average (ARIMA) model requires stationary data: a constant mean and variance regardless of time. To make data stationary requires the removal of the trend and seasonality components. There are numerous ways to achieve stationarity, such as differencing, transformations and seasonal differencing. The ARIMA model integrates an auto-regression and a moving average model. The auto-regressive model uses lagged y_t in combination with θ_p parameter which is based on the correlation between y_{t-n} and y_t in:

$$y_t = c + \theta_1 y_{t-1} + \theta_2 y_{t-2} + \cdots + \theta_p y_{t-p}, \tag{1.3}$$

where p is the number of lagged forecast errors in the prediction. The moving average makes predictions that centre around an average c by considering the error ϵ_t between the previous prediction $\hat{y}_{t-1}$ and actual value y_{t-1} by:

$$y_t = c + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + \cdots + \phi_q \epsilon_{t-q}, \tag{1.4}$$

where ϕ is the gradient coefficient and q is the order of the moving average model. The gradient parameter provides insight regarding if the next prediction falls above or below the preceding period mean, which allows for future predictions. The ARIMA method combines both approaches incorporating a differencing parameter d, which finds the difference y'_{t-1} between two series at time t [7]. Taken together ARIMA is calculated using:

$$y'_t = c + \theta_1 y'_{t-1} + \cdots + \theta_p y'_{t-p} + \phi_1 \epsilon_{t-1} + \cdots + \phi_q \epsilon_{t-q} + \epsilon_t \tag{1.5}$$

Compared to ES, the ARIMA method shows improved fit to training data, while the ES methods demonstrate greater accuracy in predictions [8]. Otherwise, these two methods' performance is comparable and has formed the benchmark in forecasting. Commonly, benchmarking machine-learning-based forecasting involves comparing model performance against ARIMA and ES.

1.1.3 Machine-Learning Methods

Artificial intelligence (AI) is increasingly an integral part of society, with it applied in fields ranging from finance to healthcare and computer vision [9], [10]. The premise behind machine-learning models is that given enough data, a model can learn properties in the data without explicit programming to do so. Learning occurs through an accuracy metric that undergoes iterative improvement through gradient descent [11]. Several architectures have since been developed to excel at specific tasks. One such architecture is the recurrent neural network (RNN) which excels at sequential problems.

RNN

The RNN derive their name from the recursive nature of the architecture whereby sequence data is fed into recurrent neural network cells (Figure 1.1). This design is well suited for sequential data as RNNs are able to learn short patterns in data, making them desirable in natural language processing (NLP), computer vision and time series forecasting. However, RNNs are unable to learn long-term patterns in data since they suffer from either the vanishing gradient or exploding gradient problem [12]. To combat the inability to learn long-term patterns there has been research into alternative RNN architectures, giving rise to the long short term memory (LTSM) and gated recurrent unit (GRU) neural networks.

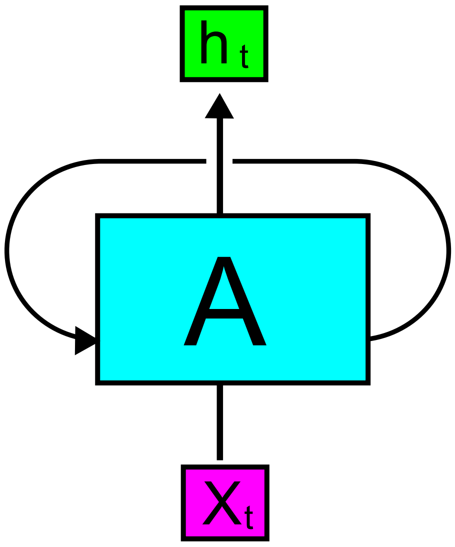

FIGURE 1.1: General overview of a typical RNN architecture. A RNN takes a sequential input (X_t) into a recurrent activation cell (A) that outputs a predictions (h_t) that is also fed into subsequent cells.

LTSM

The LTSM is architecturally distinct compared to traditional RNNs, housing several additional gates: the input, forget and output gates (Figure 1.2). Each cell of a LTSM receives a different input from the sequence data (X_{t-1}), in addition to receiving information from the previous cell in the form of the cell state (C_{t-1}) and hidden state (h_t). Given the input the forget gate controls how much information is written to the current cell state (C_t) using a sigmoidal (σ) function [13]. The function compresses the information $\mathbb{R} \rightarrow [0,1]$ and through a dot multiplication with C_{t-1} "forgets" selective information pertaining to the previous cell state. Next, the input gate is used to manipulate the information that will be written to the cell state [13]. Lastly, the output gate modulates how much information to reveal to the next cell as the hidden state and produce an output if the architecture allows for it. To combat the vanishing gradient problem, during back propagation, the error is propagated backwards along the cell state to the individual gates where the weights are updated [14].

GRUS

The GRU architecture is a simplified version of the LTSM (Figure 1.3). This simplification is achieved by combining the input and forget gate to create the update gate, which exposes the entire memory to operations, thereby reducing the level of control in updating the cell state. Additionally, GRUs are further simplified by merging the hidden and cell states when passing the information through the cell. The reset gate controls the uptake of information

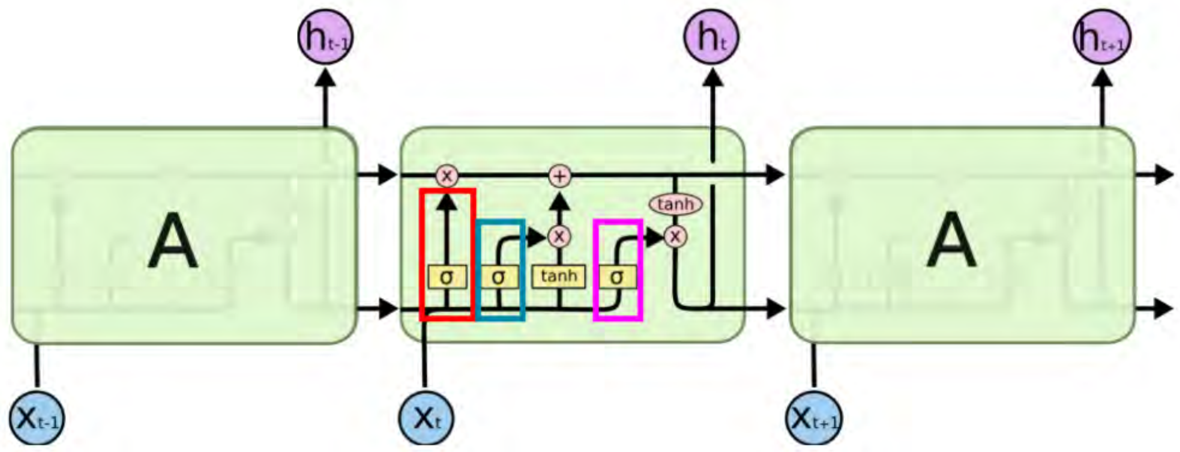

FIGURE 1.2: Structural overview of a LSTM neural network showing how the input (X_t) leads to the outputting the hidden state of the cell(h_t). Coloured boxes designate the various gates, namely the forget gate (red), input gate (blue) and output gate (purple). Adapted from [15]

.

from the previous cell, acting as a selective information filter. Due to these changes, GRUs train faster than LTSM whilst still producing comparable performance [16]. However, there is an inverse relationship between explain-

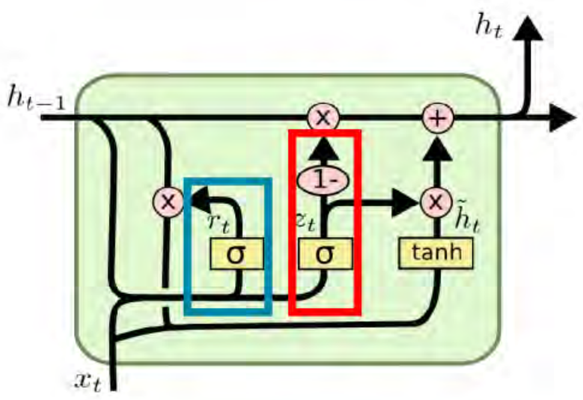

FIGURE 1.3: Structural overview of a GRU neural network showing how the input (X_t) leads to the output(h_t). Coloured boxes designate the two gates, namely the update (red) and the reset gate (blue). Adapted from [17]

.

ability in a system and its predictive accuracy [18]. Consequently, neural networks and deep learning, which offer the highest accuracy for tasks such as natural language processing and computer vision, are also the least interpretable systems [19]. Specifically, their inherent complexity and non-linear nature that enables such models to learn abstract patterns whilst leading to difficulties in explaining their predictions.

1.1.4 Explainable AI

Explainable AI describes methods used to improve understanding behind the predictions made by algorithms. The aim is to improve accountability, reduce perpetuating bias and allow for greater understanding behind mechanisms that govern prediction for a given algorithm [20]. Machine learning models which lack explainable elements are classed as being a "black box" and run the risk of perpetuating computer-based discrimination and bias [21]. Furthermore, in the worst-case scenario, failing to understand how an AI system may function could lead to physical harm as AI becomes further integrated into society [22]. Recognising the necessity of explainable AI (XAI) has seen a concerted effort by governments to implement laws concerning the implementation of explainable prediction.

Broadly speaking, there are two means in which one can improve explainability, implementing inherently explainable models or post hoc analysis. An example of explainable models includes decision trees, general linear regression, or the K-nearest neighbours approach, whose design provides an intuitive understanding of the prediction. However, as previously stated, these methods are not as accurate due to the inability to learn complex non-linear relationships. In contrast, post hoc analyses are methodologies applied to already learnt ML systems and prove more popular since they attempt to make a previously poorly understood but accurate algorithm explainable.

Post hoc analyses are further divided into deriving global or local understanding depending on whether they provide understanding behind all predictions given a model or why a specific prediction was reached, respectively [23]. While other ML fields such as natural language processing and computer vision have made strides in creating explainable prediction systems, time-series RNN architectures have not experienced as much progress. Currently, the SHAP and local interpretable model-agnostic explanations (LIME) values are foremost choices when it comes to agnostic XAI methods [24], [25].

SHAP values derive explainability by determining how much a feature contributes to the prediction by measuring the average change in prediction for all possible coalitions. In contrast, LIME functions as a local explainability predictor that perturbs the underlying data of a given model before calculating the distance between permuted and original data. Subsequently, the model attempts to fit a linear model on the n most informative features to infer the contribution of features in the model. Practitioners commonly use these methods to generate counterfactual explanations that describe the necessary changes required in input to change a classification [26]. While LIME and SHAP are desirable for their model-agnostic properties, researchers have also observed success among model-specific XAI methods.

Recent advances in time series XAI methods focus on convolutional neural networks (CNN) applications. The XCM algorithm developed by Fauvel *et al.* provides insights behind feature importance through time and feature attribution maps in various health data [27]. This algorithm boasts granularity in both global and local explainability and is based on gradient-weighted class activation mapping (Grad-CAM) [28]. Additionally, Viton *et al.* similarly incorporated the use of a CNN to generate heatmaps to describe feature and time importance surrounding the decline of patients in health dataets [29]. Both methods demonstrate the practicality of CNN in determining feature importance in deep classification networks. However, these methods prove ineffective for financial time series forecasting. The methods are either not applicable to time series data or are directed towards classification algorithms, further highlighting the need for XAI methods for financial forecasting models. In devising such an algorithm, insight can be gained from examining XAI methods used in other fields.

1.2 Problem Statement

Explainable time series forecasting tools would enable insights behind stock trading and allow for transparency and understanding regarding decisions made from predictions. However, the historical use of statistical forecasting methods has led to stagnation in XAI research in this field. As the use of deep "black box" machine-learning methods increase in the financial sector, so does the risk of perpetuating algorithmic bias. Subsequently, this study investigates the applicability of commonly used XAI techniques on RNN architectures for financial time series forecasting.

1.3 Significance and Motivation

The M5 competition demonstrated superior accuracy among deep and ensemble machine-learning-based financial forecasting methods. Furthermore, as machine-learning models become further integrated into society, there is an increase in legislation requiring transparency from models to protect individuals. Subsequently, models are facing pressure to comply with explainability criteria, and failure to do so may fail to deploy the model [30]. This expectation to comply with legislation necessitates developing and benchmarking a basis of explainability to progress this field. There has been some progress in time series XAI methods [31], [32]. However, these XAI methods were developed for classification tasks in the healthcare field and not for financial regression tasks. While applying these XAI methods to a financial setting is feasible, it is equally important to consider the differences behind the motivation of XAI methods between fields. Finance seeks XAI not only to comply with legislation but also to use XAI evaluative tool to determine if the model is learning non-existent patterns or is overfitting as forecasting models have a high risk of backtest overfitting[33]. In contrast, healthcare may focus on the mechanisms responsible for producing the data rather than the model's behaviour. For example, Nguyen *et al.* developed their XAI method to understand the reason behind the patient decline and not understand how the model was making predictions. Consequently, XAI methods designed for models used in healthcare may not be appropriate in a financial setting.

The methods presented in this study are not novel as they are used extensively in multiple fields. However, the application of these existing XAI methods in financial time series forecasting will be novel. Furthermore, the alterations made to the ablation will change the XAI method classification from local to a global XAI method. Collectively, these results seek to establish a baseline for explainability within the financial time series forecasting sector. If successful, this study would demonstrate that machine-learning-based forecasting systems are able to be held to the same explainable criteria as others, such as health and computer vision.

1.4 Research Question

Despite the existence of numerous post hoc explainablity methods, it is still not known how well these function within a time series context. This study seeks to answer the question of how effective existing explainability methods are on various RNN architectures within the context of financial time series forecasting.

1.4.1 Research Aims

The overall aim of this research was to implement post hoc analyses in various RNN architectures in order to explore explainability in time series forecasting. The aim was achieved by first training various open-sourced RNN (RNN, LTSM, GRU) models on financial time series data before implementing four post hoc analyses (added noise, ablation, block-bootstrap permutation and inter-grated-gradient). This research attempted to better quantify how predictions were made using these architectures and provided comment on which of these methods has the best performance in time series forecasting.

1.4.2 Objectives

In order to complete the research aim, the following objectives were achieved:

1. Find and construct a multidimensional financial time series dataset spanning 2-10 years.
2. Find, implement and train open-source code of the three RNN architectures (RNN, LTSM, GRU) of interest
3. Implement and modify, if necessary, the post hoc analyses (added noise, ablation, permutation and integrated-gradient) on each of the RNN networks.
4. Develop a visualization strategy to derive meaning from post hoc analyses in the context of the developed RNN networks.

1.5 Outline

The subsequent chapter will go into further detail regarding the methods used in this study. Chapter 3 will expand upon details surrounding the nature of data, models' architecture, and the implementation of the four XAI methods. Subsequently, Chapter 4 focuses on the results for each of the XAI methods and the interpretations of each figure. Lastly, the conclusion will summarise and bring context to this study whilst making future research recommendations.

Chapter 2

Research Methodology

Before the subsequent methods can be explicitly detailed, it is essential to describe the overarching research design. Understanding the research design is useful in focusing on what the research is investigating and how it was achieved. Furthermore, the research can only be considered successful if there is a clear defined goal as what the research was trying to achieve.

2.1 Research Design

The undertaken research was considered confirmatory applied research. Confirmatory research refers to research that tests *a priori* hypotheses, whereas applied research aims in addressing a practical problem. Specifically, our research aimed to confirm whether established post hoc analysis are applicable in time series forecasting. Furthermore, this study addressed a practical problem: given historical stock prices of the S&P 500, can an accurate RNN-based algorithm be developed and produce explainable reasons behind its predictions. The study provided evidence in support of the hypothesis for the explored XAI methods.

2.2 Methodology

Many consider the results to be the defining feature regarding whether a study successfully achieved the outlined aim. However, if a study is not reproducible, the results will fail to have any tangible impact in solving its described problem. To ensure reproducibility, requires an extensive methodology section that describes what was performed in the research simply and effectively. Furthermore, it is equally essential to describe how the data was acquired and processed.

2.2.1 Dataset and Processing

A multivariate time series $\mathbf{T}_{l;f}$ was constructed from the daily S&P 500 between 02 December 1984 and 28 May 2021, excluding weekends, spanning $l = 9197$ days [1]. The stock price index value of the S&P 500 is determined by calculating the market cap of the 500 largest companies in the United States. This calculation takes into consideration the number of held shares in the market and the market price for a single share. Thereby satisfying the first objective. Each day contains $f = 5$ features representing the opening, closing, adjusted closing, maximum price, minimum price and volume traded for the day for each stock. Subsequently, the trend was removed through seasonal and trend decomposition using LOESS. The augmented Dickey-Fuller test confirmed if the time series was stationarity before it underwent global normalization [34].

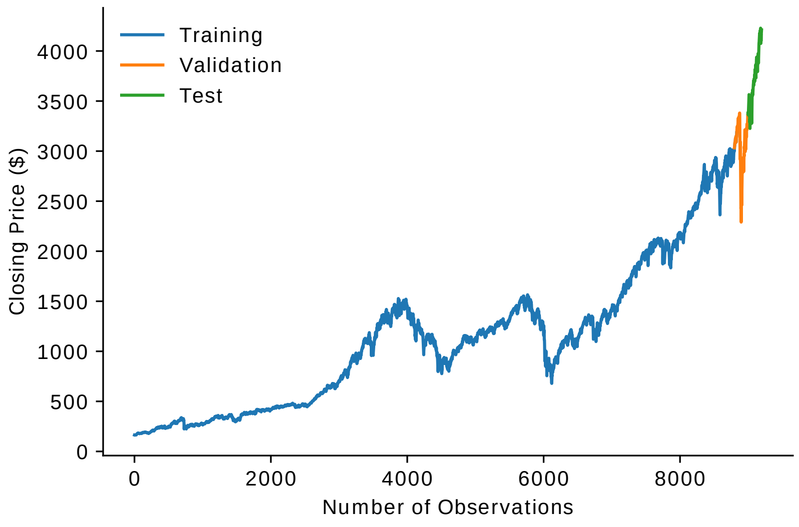

FIGURE 2.1: The closing price of S&P 500 stocks divided into training, validation and test datasets [1]

Additionally, as shown in Figure 2.1, the data was split into training, validation and test dataset, where the validation and test datasets constitute the last 400 data points, split evenly between them. Furthermore, the last 99 values of the training dataset were prepended to the validation set to allow for prediction of the first validation value. The inclusion of the preceding 99 values of the validation set was repeated in the test dataset to allow for the same predictions [1].

2.2.2 Models

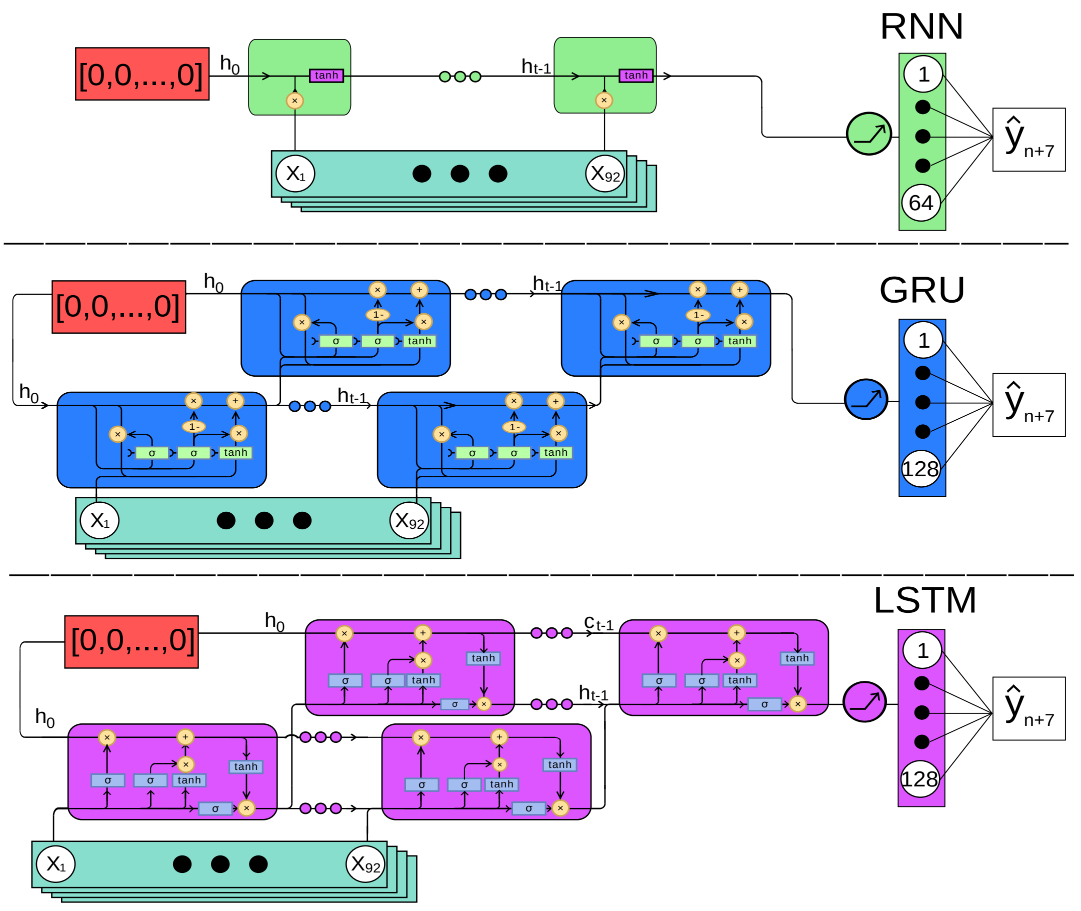

FIGURE 2.2: Models were initialized with a zero vector (red region) and provided a series of 92 days of financial data. Non-linearity was introduced in each model using the ReLU function prior to the fully-connected layer where dropout was applied to the LSTM and GRU. Dropout was applied to the 8,46,61,67,83,92 and 123rd value in the fully connect layer. The same dropout was applied to the GRU with the addition of the 64 and 125th value

TABLE 2.1: Hyper-parameters and test accuracy for the various RNN architectures [1]

Model	Hidden States	Layers (# Cells)	Dropout	Alpha	Test Accuracy (SMAPE)
RNN	64	1	0.000	0.005	1.83
GRU	128	2	0.065	0.010	1.81
LSTM	128	2	0.050	0.008	1.81

The study used three different recurrent neural network architectures as models to investigate explainability methods: a standard RNN, a GRU, and an

LSTM. Hyper-parameters [# hidden states, layers (# Cells), dropout and alpha] were optimised using Adam, minimising mean-squared error on the validation set [35], [36]. As shown in Figure 2.2, the networks follow the same general structure. The models used $w = 92$ time steps as an input window, representing a financial quarter. The models forecast the closing price $\hat{y}_t$ at a horizon $h = 7$ days in the future.

We trained the models for 30 epochs with mini-batches of size 3000. After that, we evaluated model performance using the symmetric mean absolute percentage error (SMAPE):

$$\frac{2}{n}\sum_{i=1}^{n}\frac{|Y_i - \hat{Y}_i|}{|Y_i| + |\hat{Y}_i|} \tag{2.1}$$

where Y_i is the actual value and $\hat{Y}_i$ is the predicted value over n predictions. Despite the differences in the parameters among the models, the models were comparable in accuracy, as shown in Table 2.1. Subsequent training and optimisation of RNN model lead to the completion of objective 2.

2.2.3 Explainability Methods

The following explainability methods all use the same principle to test different aspects of RNNs. The principle is that significance of a particular input, node or feature is determined by measuring the change in prediction following its perturbation. The ablation and integrated gradients methods focus on input importance, whereas the noise and permutation reveal node and feature importance.

Integrated Gradients

The integrated gradient method is popular in both NLP and computer vision field as an alteration to traditional feature gradient methods. Integrated gradients is an approach that assigns importance to features as attributions [37]. It achieves this by considering the gradients respect to its input whilst desaturating the data to predefined baseline [38]. A baseline represents a state of no information for the model prediction. By integrating between the baseline and original dataset, the change in gradient versus the model's predictive accuracy allows for the determination of nodal importance. The change in gradients are scaled against the models' inputs, creating the attributions, whose sign and magnitude provide insights behind the importance of each input in the model.

In order to adapt this method for time-series data, the baseline needs to provide no information conferred by the previous inputs allowing the determination of nodal level importance without noise from previous time points. Subsequently, the baseline used in the study replaced each value with the average of the previous 91 entries. In doing so, the model only receives information regarding the average of the previous time-step, thereby receiving no new information other than the trend [1].

Ablation

Ablation is a technique used in the computer vision to identify which pixels are most crucial for a given classification. The method traditionally functions by removing information from a grid of pixels (block or line) by zeroing the entry. This ablation region then moves over the image resulting in a change in the prediction, the magnitude in the prediction error is used to infer importance in the picture [39]. This information can then be overlaid onto the original image, giving insight to the original image.

Adapting this method to the time series data, feature ablation blocks can be introduced whereby the mean of previous features are forward fed into the sliding window of the RNN. In doing so, as the RNN passes over the ablation block and the model receives no new information, allowing inference of feature importance based on the magnitude in the change in prediction. This is reminiscent of the baseline in the integrated gradients method, with the distinction that the ablation area does not change with respect to the models input while the baseline does.

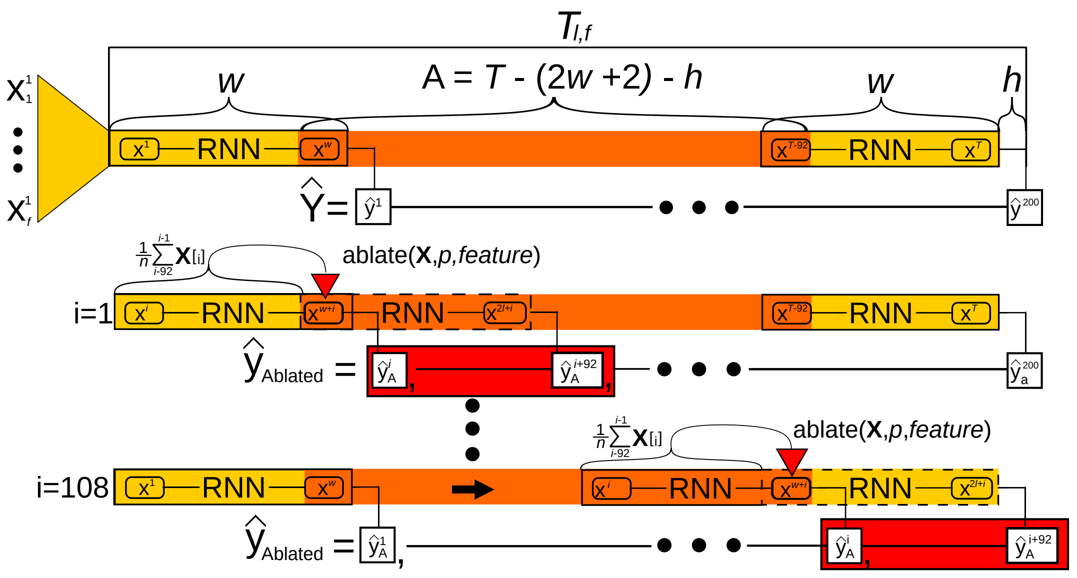

FIGURE 2.3: .
Overview of the ablation method on the Test dataset. The time series T comprises of an ablation area (orange), a prediction horizon (h) and two flanking regions (yellow) of size w where l is the sequence length of the RNN models. As the models slides over T predictions that differ from the $\hat{Y}$ due to inclusion of the ablated input are noted (red) [1].

Ablation studies use regions of non-information to determine significant data points in an input. In RNNs, zeroing of input can be achieved through forward filling with the average of prior inputs. The rationale for using the average is that an RNN exploits prior information in the time-series. Therefore, replacing a single input value with the average of the previous cells removes any information supplied by those cells as seen in Figure 2.3. A single data point would be ablated and fed into the model, whereby predictions would be made by sliding over the single ablated feature value (Algorithm: 1

and Figure 2.3). For a given multivariate time series $\mathbf{T}_l$ of length l, there exists a region $\mathbf{A}$ which can be ablated at point p, given that there are $w-1$ values that precede the value and an additional $w-1$ values that follow it. It follows that there must be a sufficient number of entries preceding the ablated value, whilst still allowing the RNN to slide over the time-series, generating the errors. Specifically, in this study, using models that take in $w = 92$ values and have 7 day forecast horizon ($h = 7$), $\mathbf{A}$ spans the central 108 values. The method produces w pairwise errors $\mathbf{e}$, calculated by taking the absolute value of the differences between the ablated and non-ablated predictions as the RNN slides over the ablated data. The algorithm returns the average percentage pairwise error for all the inputs into the RNN [1].

```
input  : A time series $\mathbf{T}_{l,f}$                    // length $l$ and features $f$
       : A RNN model $(\cdot)$ $l-(w+h)$predictions
output: An Error matrix $\mathbf{E}_{Avg}$ of size $w \times f$
$Error_{Avg}$ =[ ]
$\hat{\mathbf{y}} \leftarrow$ model($\mathbf{X}$)
for $feature$ in $f$ do
    $E_{Feature}$ = [ ]
    /* Iterate over region A                                     */
    for $i$ in 1:len ($\mathbf{A}$) do
        $\mathbf{X}_{Ablated} \leftarrow$ ablate($\mathbf{X}, i+91, feature$)
        $\hat{\mathbf{y}}_{Ablated} \leftarrow$ model($\mathbf{X}_{Ablated}$)
        $\mathbf{e} \leftarrow \frac{|\hat{\mathbf{y}}_{Ablated} - \hat{\mathbf{y}}|}{\hat{\mathbf{y}}}$
        /* Concatenate non-zero errors from RNN sliding over ablated
           value                                                 */
        $\mathbf{E}_{Feature} \leftarrow$ concat($\mathbf{E}_{Feature}$, $\mathbf{e}$ [$i$:$i+91$])
    end
    $\mathbf{E}_{Avg}[feature] \leftarrow \frac{1}{\texttt{len}(\mathbf{A})} \sum_{\texttt{len}(\mathbf{A})}^{j=1} \mathbf{E}_{feature}[j]$
end
return $\mathbf{E}_{Avg}$
```

Algorithm 1: Ablation Algorithm

Added Noise

Drawing inspiration from the ablation method, specific noise is introduced to the input instead of removing information parsed into the RNN (through forward feeding). The reasoning behind this method is that if an input is uninformative, then the introduction of random noise in the input would not cause a large change in the prediction. Conversely, if the feature input were necessary, a randomised value would lead to a more considerable change in the prediction. However, the larger the difference between the original and the altered input, the greater the prediction error. This ratio between original and altered values must be noted and factored into the results, or we must standardise the amount of noise added. We have opted to observe the change in prediction when 1% of noise is added [1].

While the ablation methods seek to remove information from the algorithm, the added noise technique is more nuanced because it still provides informative knowledge. Furthermore, isolating the noise to a specific cell in the unrolled RNN allows for observing information flow through the network.

A variant of random noise was implemented on the trained networks to test which cells in the network contributed most to predictions. The noisy time series, $\mathbf{X}_{Noise}$ is constructed by adding 1% noise to all features f iteratively in $T_{l,f}$ so that following unrolling, the same cell in the RNN model receives additional noise for each prediction (Algorithm 2). This approach ensures that the added noises are localised to a single position in the RNN model, whereas the ablation method leveraged the model sliding over the ablation to infer importance. The calculated SMAPE between the resulting prediction, $\hat{\mathbf{y}}_{Noise}$, and original prediction, $\hat{\mathbf{y}}$, inferred the degree that each cell contributes to the prediction.

```
input  : A time series $\mathbf{X}_{l;f}$                    // length $l$ and features $f$
       : a RNN model $(\cdot)$ $l-(w+h)$ predictions
output: SMAPE error array $\mathbf{e}_{Day}$ of size $w$
$\mathbf{e}_{cell}$ = [ ]
$\hat{\mathbf{y}} \leftarrow$ model ($\mathbf{X}$)
/* Iterate over each cell in the unrolled RNN model $(\cdot)$          */
for each cell in 1:w do
    $\mathbf{X}_{Noise}$ = [ ]
    for i in 1:l-w-7 do
        $\mathbf{X}_{temp}$ = $\mathbf{X}$[i:i+w]
        /* Adds 1% noise to all features received by cell          */
        $\mathbf{X}_{Noise}[i] = \mathbf{X}_{temp}[cell;] + \frac{\mathbf{X}_{temp}[cell;]}{100}$
    end
    $\hat{\mathbf{y}}_{\text{noise}} \leftarrow$ model ($\mathbf{X}_{noise}$)
    $\mathbf{e}_{SMAPE} \leftarrow$ SMAPE ($\hat{\mathbf{y}}_{Noise}$, $\hat{\mathbf{y}}$)
    $\mathbf{e}_{cell} \leftarrow$ concat ($\mathbf{e}_{cell}$,$\mathbf{e}_{SMAPE}$)
end
return $\mathbf{e}_{cell}$
```

Algorithm 2: Noise Algorithm

Permutation

The permutation method is a global explainability method. It is global in that it aims to explain why all predictions are made with a given model, as opposed to a single prediction as with local explainability models. The permutation method infers feature importance by quantifying the predictive error following permuting all values from a single feature [40]. Thereby determining feature importance for all predictions and not in a single-use case.

This method entails permuting each feature by randomly replacing each value with another value that preceded it in the time series. Thereby ensuring that the model is not exposed to any future information when making a prediction. Following the permutation of a single feature, the SMAPE was calculated between the permuted and original predictions (Algorithm 3). Each feature was permuted 300 times to account for the stochastic nature of permutation. Feature importance was inferred through fitting a simple linear regression to fit one-hot-encoded (OHE) permuted features to the SMAPE error. Specifically, the magnitudes of the regression coefficient determined the feature importance [1].

```
input  : A time series $X_{l;f}$                         // length $l$ and features $f$
       : a RNN model $(\cdot)$ $l-(w+h)$ predictions
output: Importance of each feature to prediction
$\mathbf{X}_{OHE} \leftarrow [\,]$
$\hat{\mathbf{y}}_{Error} \leftarrow [\,]$
$\hat{\mathbf{y}} \leftarrow$ model $(\mathbf{X})$
for feature in $f$ do
    /* Repeated to address stochastic nature of permutation        */
    for i in 1:300 do
        $\mathbf{X}_{Permuted} \leftarrow$ permute($\mathbf{X}$,$feature$)     // Permute all of $\mathbf{X}_{f=Feature}$
        $\hat{\mathbf{y}}_{Permuted} \leftarrow$ model $(\mathbf{X}_{Permuted})$
        $\mathbf{e}_{SMAPE} \leftarrow$ SMAPE $(\hat{\mathbf{y}},\hat{\mathbf{y}}_{Permuted})$
        $\mathbf{X}_{OHE} \leftarrow$ concat $(\mathbf{X}_{OHE}$, OHE for Feature $F)$
        $\hat{\mathbf{y}}_{Error} \leftarrow$ concat $(\hat{\mathbf{y}}_{Error}, \mathbf{e}_{SMAPE})$
    end
end
OLS $\leftarrow$ Fit $\mathbf{X}_{OHE}$ to $\hat{\mathbf{y}}_{Error}$ using a simple linear regression
Importance $\leftarrow \frac{1}{n}\frac{OLS}{\sum OLS}$      // OLS refers to the feature coefficients
return Importance
```

Algorithm 3: Permutation Algorithm

2.2.4 Visualisation

Visualisation is a critical tool in focusing information and explaining results. Visualisation is even more critical in XAI when considering the function of XAI methods for explaining complicated models. Failure to adequately visualise results from XAI methods may result in an inability to explain model prediction, invalidating the methodology. Therefore, result visualisation is equally essential, justifying why it is an objective. Taking inspiration from existing XAI methods, a heatmap is a common and powerful tool to visualise large amounts of information. These have been extensively used in different XAI methods to present various explainable metrics [31], [32], [39]. An option is to generate results for each prediction and visualise the results as a video, thereby allowing the identification of patterns as it moves through time. However, visualising the local explainability proved too noisy and deviated from the aim of the study: to understand model behaviour and not the reasoning behind a specific prediction. Instead, a global visualisation approach is a better fit: global in the sense that it visualised patterns existing in multiple predictions. This approach also allowed for the comparison between all methods since they were all reporting different aspects of a singular mechanism: the behaviour of RNN in time series forecasting.

2.2.5 Hardware and Software

All three models were implemented using the Pytorch(v1.8.1) library within a Python(v3.8.8) environment [41]. Data was scraped and pre-processed using the Pandas-Datareader (v0.80), Scipy (v1.6.2) and Statsmodel (v0.12.2) libraries respectively [42], [43]. The Captum (0.3.1) library was implemented with the integrated gradients method [44]. Custom scripts that incorporated patsy (v0.5.1) and Statsmodels generated results for the ablation, permutation and noise methods. Visualisation was performed using a combination of Matplotlib (v3.3.4) and Seaborn (v0.11.1) [45], [46]. A 64-bit system incorporating an Intel Core i7-7700HQ CPU with 16GB RAM and a NVIDIA Geforce GTX 1050 GPU with 4GB internal RAM performed the model training and analysis. Due to restrictions in GPU memory requirements, the GRU and LSTM analysis could only run on the CPU.

2.3 Limitations

There are a few limitations in the methodologies used in this study. A notable limitation of this study is that the methods applied here may not apply to deep-learning methods. This limitation follows the understanding that the presented methods leverage visualisation of results to understand the model prediction, requiring human input to provide insights behind models. Consequently, the presented methods may not function well in deep-learning models that require longer time-series inputs or high dimensional data. Furthermore, whilst the different error metrics provided understanding for different aspects of the network, they may not prove intuitive to average persons seeking to benefit from explainable AI. Additionally, consistent use of a single more-suitable metric may allow better comparisons between models.

2.4 Conclusion

The methods provided above have undergone minor alterations to be applicable in a time series forecasting context. The integrated gradients and the ablation methods leverage the same paradigm surrounding providing the model with no new information by incorporating a feed-forward average approach. The results will reveal where each network places importance on its inputs. In contrast, the added noise and permutation methods rely on similar principles: inferring importance through the magnitude of change following altering a specific time-interval or feature, respectively. The added noise method identifies which time-intervals contribute most to prediction, whilst the permutation method quantifies an overall feature's importance. These methods will provide global insights behind the three models at varying resolutions and different focuses if successful.

Chapter 3

Results and Discussion

It needs to be noted that many of the results and conclusions presented in this dissertation have been published in a previous paper [1]. The results below provide information behind the mechanisms that culminate in prediction in various RNN architectures, demonstrating success as post hoc XAI methods [1]. Additionally, the results complement each other, providing supporting evidence for the well-established paradigms. Although there may be overlap between information produced by the methods, it is crucial to consider that different methods provide different granularity and focus. The needs of the model developers will determine which combination of methods are most applicable.

3.1 Ablation

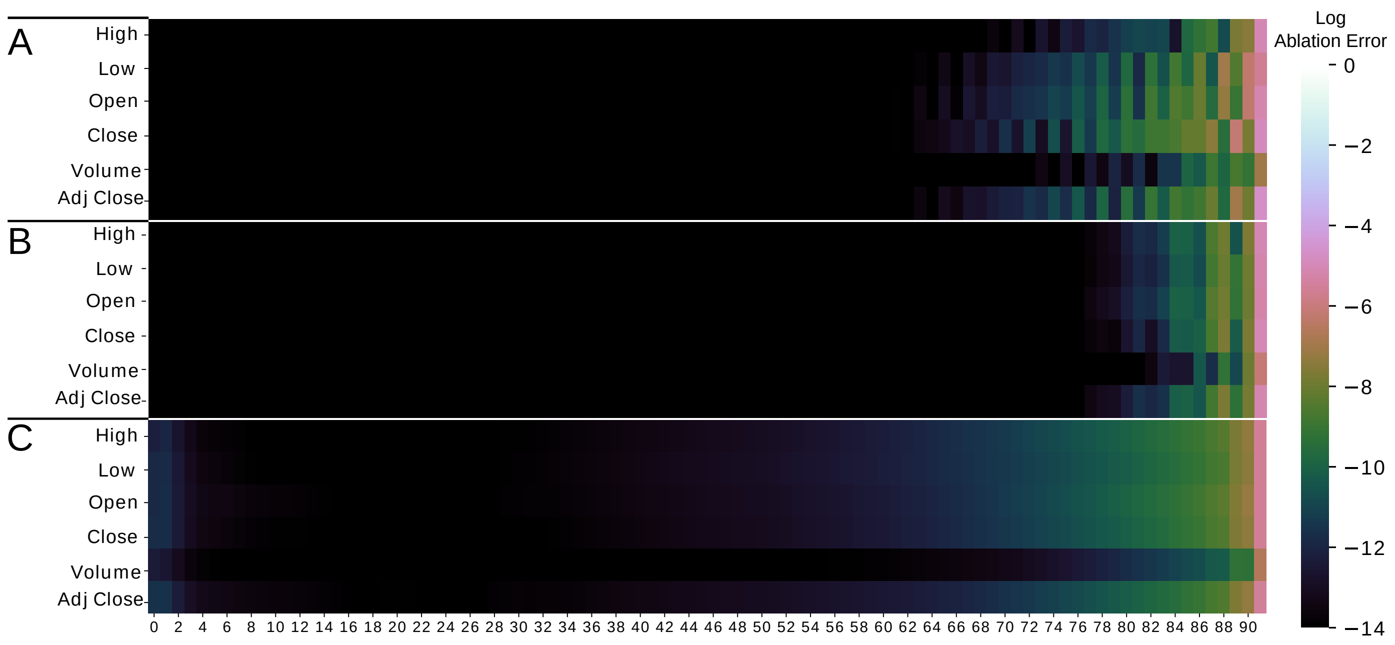

FIGURE 3.1: The log percentage SMAPE difference between the ablated prediction and reference prediction.
A) RNN, B) LSTM and C) GRU neural networks [1].

Traditionally, ablation uses an ablated pattern to determine the importance of inputs within the region. Consequently, a common criticism for ablation is that it leads to false-positive/negatives as informative and non-informative inputs may be grouped in a single pattern. Furthermore, it is relatively slow as numerous iterations are required to test many patterns. However, a consequence specific to time series forecasting is that ablating one or more points will lead to numerous prediction errors as the model slides over the ablated area. Furthermore, the prediction error equals the sum of all errors that result from the ablation region falling within the model, making it difficult to determine a single input's importance when the error pertains to numerous input ablations. To account for this, a single ablated point was fed into the model, producing w predictive errors: one for each position the ablation sits in the network. The reasoning behind this choice is that any ablation region's predictive error can be derived by knowing all the individual predictive errors that would make up the region. Furthermore, limiting the ablation area to a single input prevents noise from other inputs from influencing the predictive error, conflating importance [1].

Ablation can be considered a more straightforward form of integrated gradients: the technique removes information from a feature, and the change in prediction is measured. We can then use the magnitude of the predictive error to infer the feature's importance. If an input is non-informative to prediction, ablating it would fail to produce a large change in prediction, leading to a near-zero error. A unifying feature between models is that the most informative cells lie at the last few inputs (cells 90-92). This result follows the logic that the most recent inputs provide a stronger basis for prediction than more distant inputs and is expected in time series forecasting. This reasoning forms the basis of ES and has stood as the statistical benchmark for many years [4], [5].

Feature importance decays, albeit at different rates, moving backwards through the model which is evident in the spectrum gradient which is evident in Figure 3.1. A rapid change in colour represents a smaller window in which information is retained from past information. The GRU model shows the greatest ability to retain past information, whilst the LSTM appears to be least capable of retaining past information. However, when considering additional information from the integrated gradients and noise methods, it later becomes apparent that this is not the case and will be discussed in sections 3.2 and 3.3. Specifically, the GRU network considers the most features in making predictions, with most of the uninformative inputs situated between cells 4-38 as seen in Figure 3.1. In contrast, the LSTM shows the least number of informative features, which most are present from the 78th cell. Interestingly, the RNN presents with an alternating banding pattern of features importance from the 60th cell onward. This pattern perpetuates in both the attributions and cell importance as seen in Figure 3.2 and 3.3.
Notably, volume was the only feature that showed consistently lower importance following ablation in all three models at any given point. This result suggests that the models are purposely disregarding volume when making predictions.

3.2 Added Noise

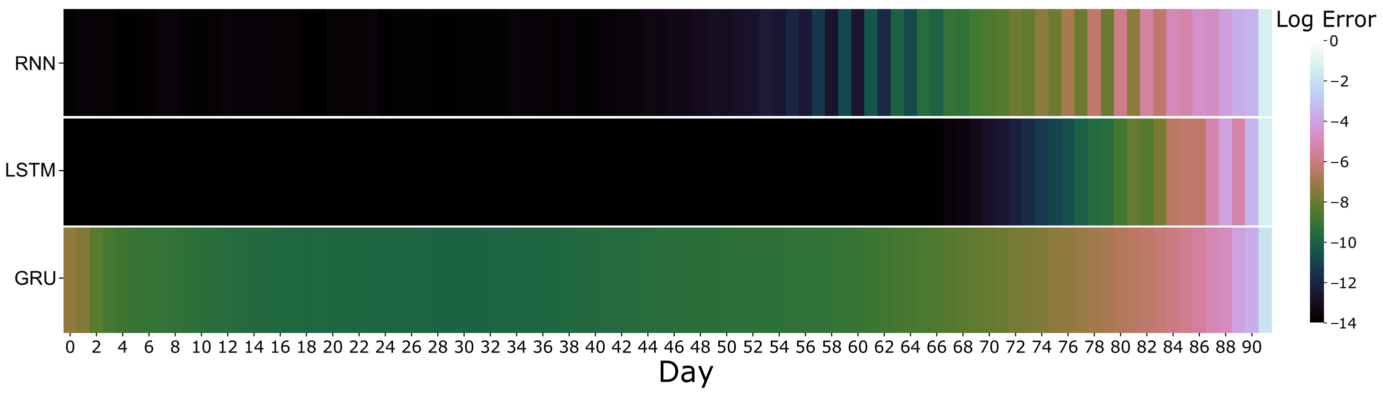

FIGURE 3.2: The log error introduced after adding 1% of noise at each day [1]

Adding a defined noise level (1%) to all inputs in a particular cell determined which cells, and by extension, which days were most important to the prediction. Localising the noise to a particular cell ensures that the RNN model does not slide over the augmented input like the ablation method. Consequently, the added noise method sacrifices granularity at the input level in exchange for sensitivity on the importance of a single day, providing improved visualisation in the changes to importance between time intervals [1].

Complementing the ablation method, the results provides further evidence that almost all the predictive power lies in the last three (90-92) cells, as seen in Figure 3.2. Furthermore, the decay of importance between models becomes more apparent and shows more sensitivity compared to the ablation results (Figure 3.1). Notably, this improved sensitivity shows that the GRU dismisses most of the information between the 23 and the 35th nodes. Furthermore, the LSTM displays a characteristic gradient spectrum suggesting memory retention as it perpetuates through the cell state leading to prediction. However, this gradient sits between the 69-81st nodes, whereby the nodes situated towards the front (82-92) show a more irregular but subtle banding pattern more indicative of the RNN.

The banding between the RNN and LSTM does not match, suggesting that the mechanism behind its presentation differs between models. A clear spectrum pattern is predominantly absent from the RNN, which primarily presents as a banding pattern alternating between time intervals of higher and lower importance. This lack of decay in importance provides further evidence for the models' inability to retain distant information as one would not expect such regular dramatic changes in model attention. Specifically, if a cell holds information from the previous n cells, one would not expect a significant change in importance in the following cell, given that it would hold approximately the same past information.

Interestingly, the GRU is the only one of three models to demonstrate a recovery in the magnitude of the importance at the 1-3rd cells. This recovery would entail that the GRU is selectively applying focus to information entering the first three cells. Given that the window size represents a financial quarter (w =92), this observation would suggest that the strength of the market at the start of a financial quarter is a greater indicator than the market 28 days preceding prediction for future stock prices.

The RNN demonstrated the most irregular of the results regarding input importance compared to the LSTM and GRU (Figure 3.1 and 3.3A). Notably, these irregularities present as alternating bands of low and high importance. This banding pattern is prevalent in both the magnitudes and signs of the attributions. By considering both the model properties and understanding behind attributions, these results suggest that the function of this pattern is to fine-tune the prediction. The RNN derives most of its meaning from the last three days of input (cells 90-92), whereby all subsequent inputs alternate between adjusting the prediction up and down from the moving average, whilst these adjusts decaying in magnitude. This banding pattern is also present in the permuted results (Figure 3.2). This behaviour suggests that the RNN is randomly adjusting predictions rather than applying focus to specific areas in the data, thus supporting the paradigm that RNNs are unable to learn long term information. Given this information, the results suggested that the RNN model could provide comparable performance with smaller input size.

3.3 Integrated Gradients

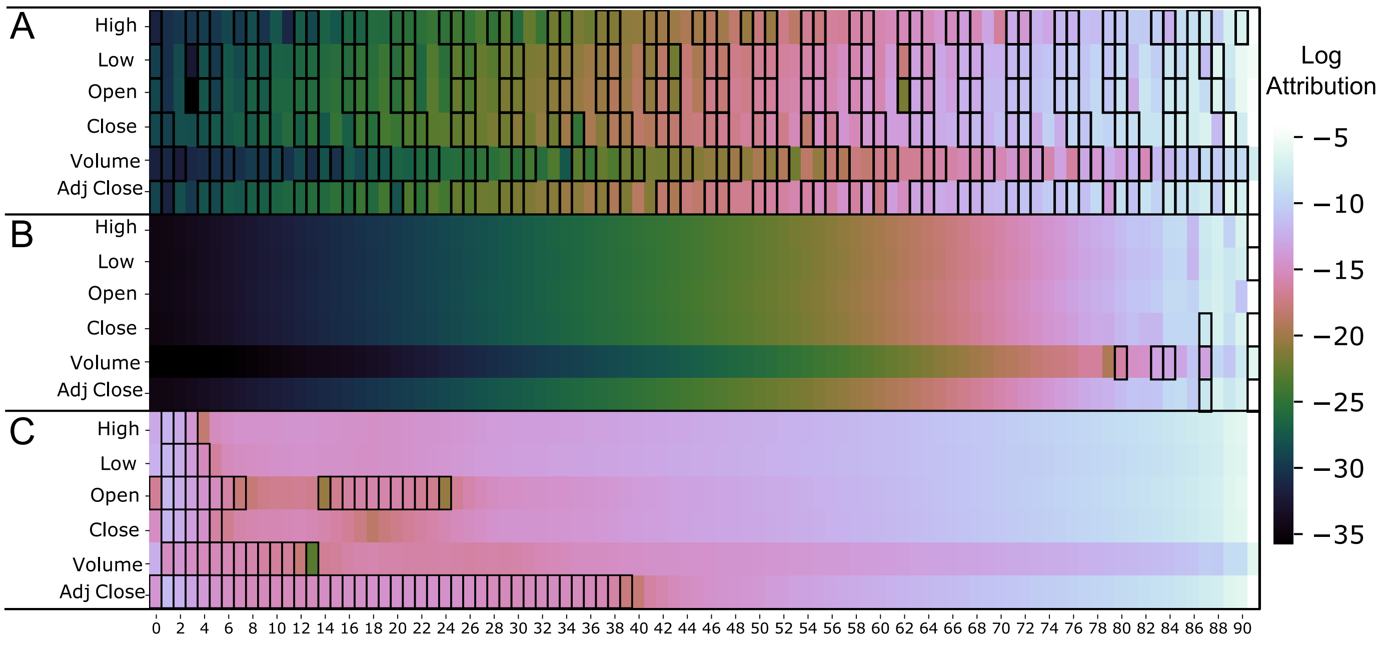

FIGURE 3.3: .

The absolute log attributions for A) RNN, B) LSTM and C) GRU neural networks. Bordered areas represent negative attributions [1]

Integrated gradients have become a popular and intuitive method to measure the correlations between the input features and correct prediction. The attribution magnitude pertains to how strongly the input contributes to prediction, whereas the sign relates to if the correlation is positive or negatively associated with a given prediction. Traditionally, an input with zero attribution represents an input with no influence on prediction, whereas a high positive value would suggest that the input feature is positively associated with the correct prediction. However, in regression, the signs of the attributions represent regions within the neural networks responsible for increasing (+) or decreasing (-) the prediction, whilst the magnitude refers to the scale of change. These attributions are determined relative to the baseline, the mean of the last 91 cells, following similar logic behind the ablation method. The logic behind this baseline is that if an input to a time series consists of only noise, there would be a given mean and variance; therefore, supplying only the mean from previous cells contributes no new information from the past. Such a baseline would determine input attributions in the absence of past information in the time series, enabling the determination of individual input importance without the noise from previous cells. Consequently, it is unsurprising that integrated gradients show greater sensitivity than the ablation method as the model introduces noise as it slides over the ablated input because it still receives information from inputs surrounding the ablation.

There are both commonalities between the models as well as model-specific features [1]. Notably, the magnitude of the attributions for volume are consistently lower than the attributions for any other feature in a given day, shown in Figure 3.3. This provides additional support that the models largely disregard the volume when making predictions, which was first noted in the ablation results (Figure 3.1). All three models demonstrated their largest magnitudes clustered towards the last inputs (87-92).

Interestingly, the patterns of negative attributions present differently in each model, revealing the intricacies behind the model prediction. The RNN model demonstrates an alternating pattern between positive and negative attributions, reminiscent of the banding pattern seen in Figures 3.1 and 3.2. The pattern extends into the magnitudes of the attribution. These results suggest that the function of this banding pattern is to fine-tune the prediction. The RNN predominately uses the last three days of input (cells 90-92) to form the predictions, whereby all subsequent inputs fine-tune the prediction. Specifically, the model alternates between adjusting the prediction up and down from the moving average, whilst these adjustments become smaller as they move backwards in time. This behaviour suggests that the RNN is randomly adjusting predictions rather than applying focus to specific areas in the data, thus supporting the paradigm that RNNs cannot learn long-term information. Given this information, the results suggested that the RNN model could provide comparable performance with a smaller input size.

Provided with greater sensitivity, the characteristic spectrum associated with long-term learning is clearly visible in the LSTM attributions. Furthermore, the LSTM had all its negative attributions localised at the model's front (82-92) and showed significantly lower numbers than both the GRU and

RNN. Despite this, the LSTM network in this study assigns very little importance to data preceding the 78th cell (Figure 3.3). One possibility is that the LSTM demonstrates a lack of memory retention. However, when provided with increased sensitivity from the integrated gradients, the LSTM demonstrates the spectrum decay of importance present in the GRU network. This result provides evidence for remembering past information, demonstrating the flow and decay of past information through the cell state, supporting the paradigm that LSTMs can learn long-term information [47].
If the LSTM can learn distant information, it does not explain why it assigns such low importance to inputs preceding the 78th cell. This conflict between the ablation and attribution results suggests that the model deliberately forgets long-term information to improve the prediction. If true, then long term memory retention appears to be less critical in prediction than the GRU for this instance.

A possible reason for the deliberate exclusion of distant information is that the sequence length (92) may be insufficient for the LSTM to learn long-term patterns adequately. Given that LSTM performs better with longer and more complex sequences compared to the GRU, the LSTM may rely more on its granularity in memory gates when provided a smaller input size [48]. Unlike the GRU, the LSTM can fine-tune the contents of its memory through the output gate, meaning that the predictive strength of the LSTM, in this context, is derived from how it applies importance explicitly to each input rather than long-term patterns. This idea is further supported when considering the attributions of the LSTM (Figure 3.3). The LSTM demonstrates the least number of negative attributions, and they are all localised towards the end of the network meaning these inputs are critical in reducing the magnitude of the prediction whilst considering relatively few inputs. Consequently, the LSTM considers fewer data points of relatively high importance to adjust the prediction down to the correct level. Interestingly, while the 92nd cell confers most of the information to the prediction, it also is the cell that contains most of the negative attributions, excluding only the open feature. This result implies that the LSTM functions by primarily considering the opening price, at the 92nd, to raise the prediction to an appropriately high level whilst considering the remaining features to create an upper ceiling in the prediction.
Equally in the GRU, the higher sensitivity reveals further evidence regarding the model's long memory retention and attention placement. This evidence is most apparent in the larger overall magnitudes observed in the attributions: any input in the GRU confers more information to prediction than any input preceding the 60th day in the LSTM (Figure 3.3). Given that GRUs combine the input and forget gate, creating an update gate, GRUs are only able to act on the entire contents of their memory when remembering past information [16]. Subsequently, GRUs demonstrate less precision in forgetting past information compared to LSTM. Therefore, the results suggest the GRU model relies on remembering more past information while the LSTM leverages its precision in forgetting past information to achieve comparable levels of accuracy.

When considering the details regarding the attention placed in the model, the role surrounding the negative attributions becomes clearer. Specifically, most of these negative attributions are situated where the GRU places attention, which is indicated by the increase seen in the attributions around the 1-4th cells. While most inputs function to increase the prediction, the model focuses on the first 12 days to lower the prediction. The magnitudes of negative attributions suggest that the attention in early inputs lowers the prediction as a form of fine-tuning rather than creating an upper threshold of prediction like that seen in the LSTM. Interestingly, the magnitudes of the first and last inputs in the region of negative attributions are smaller than those in the centre. This observation suggests that the GRU specifies the regions that reduce prediction when the magnitude of the attribution drops below a specific threshold. If accurate, this would provide an intuitive explanation for how the GRU model selected specific inputs to fine-tune prediction rather than contribute to it.

Lastly, it is worth noting that ablation provides less sensitivity compared to integrated-gradients when comparing results (Figures 3.1 and 3.3). However, ablation provides a less abstract understanding of the input importance. Indeed, the black regions in the ablation results are absent in the integrated gradients method, showing practically where the regions of non-importance lie regarding prediction. Consequently, ablation is arguably more informative to users who seek to understand what information is vital to prediction rather than the particulars underlying the trained model, in which case attributions becomes more informative to users.

3.4 Permutation

Initially, feature importance was determined by dividing the error of permuting a single feature by the sum of permuting each feature. This mode of determining feature importance relied on the assumption that the change in SMAPE error following permutation displayed linear properties: specifically, the error needed to be additive. This assumption proved to be correct for RNN; however, the GRU and LSTM showed that permuting all features produced an error greater than the sum of its parts as shown in Figure 3.4. This error discrepancy implies that the model has learned some non-linear relationships between features when making the prediction. The error discrepancy suggests that the GRU, and to a lesser extent the LSTM, uses information from one feature to modulate the information of another feature when making predictions. Specifically, this non-linear relationship results in a 22% and 2% discrepancy between the sum of all feature errors and permuting all features in the GRU and LSTM, respectively. Consequently, using the above-proposed means would underestimate feature importance in the GRU and LSTM networks. An alternative strategy would be required to ensure that individual feature importance sums to 100% .

The focus of the permutation methods was to determine which feature contributes the most to prediction for each RNN architecture. Interestingly, all three models assigned the lowest priority to the volume of traded stock, whilst there was no single feature with the highest priority between the three

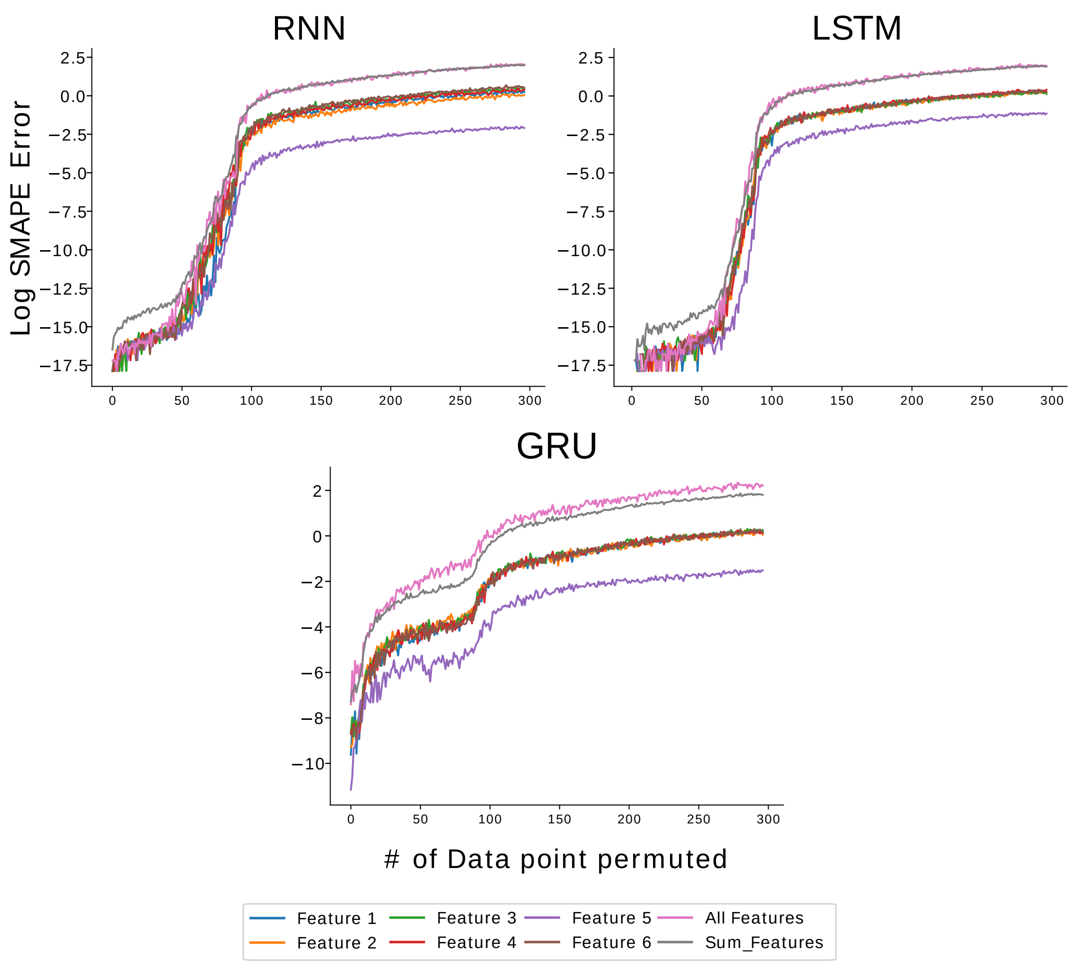

FIGURE 3.4: The SMAPE error produced from permuting a varying number of data points in individual features. Additionally, the error of permuting all the features and the sum of individually permuted feature errors are shown

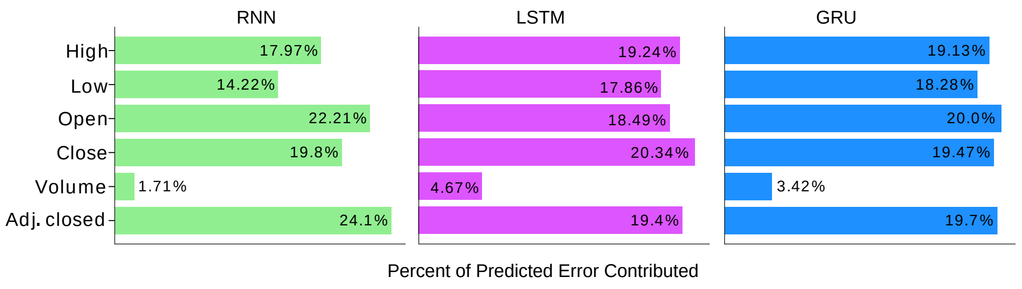

FIGURE 3.5: Feature importance derived from permutation method for RNN, LSTM and GRU [1]

models, as shown in Figure 3.5 [1]. Consequently, the volume of traded stocks had little to no effect on predictions in all three models. The potential reason is that volume only refers to the number of stocks traded within a given day, not necessarily the price. The volume for a given stock could increase following a spike in stock price, indicating investors' interest in buying the stock. Equally, the volume may also increase following the decrease of a stock which indicates that investors are selling a given stock following the reduction in stock price. Consequently, volume taken in isolation does not provide a strong correlation to the price of a stock at closing. Notably, all three models are cognisant of the lack of correlation between volume and closing stock price and have primarily disregarded volume when making predictions.

Notably, the GRU and the LSTM place higher importance on volume compared to the RNN, and by considering the attributions, the reason becomes clear (Figure 3.3). In the LSTM, volume importance comes from the negative attributions, whereby 45% of negative attributions fall in the volume feature. In comparison, the GRU focuses on the first few cells (0-4), which contains a region of negative attributions present in the volume feature. While the LSTM uses the most recent inputs to create the upper predictive ceiling, the GRU uses the most distant inputs, where it places attention, for its predictive ceiling. This observation around attributions and attention may explain why the RNN largely disregards the volume data: it cannot apply attention in long-term patterns or precisely regulate information retention.

Overall, there are similarities between the feature importance for GRU and LSTM compared to the RNN. This similarity is present despite the attributions displaying considerably different modes of prediction (Figure 3.3). It is unlikely that the ability to retain long-term information, as seen with LSTM and GRU neural networks, is responsible given that the RNN shows greater retention than the LSTM (Figures 3.1 and 3.2).

In contrast, the RNN shows the most differences in where it places importance in its features. The RNN appears to emphasise the adjusted closing price and the opening rather than the high and low price for stocks. This result suggests that both the opening and the adjusted closed is more critical to prediction in the absence of long-term information. This logic makes sense as it is unsurprising that one should consider the most recent opening and closing price, adjusted against stocks traded, to predict the closing price. Consequently, long-term memory enables a more nuanced prediction by considering the change in patterns in more features with respect to closing price when making a prediction, as seen in the GRU and LSTM.

Chapter 4

Conclusion

This study aimed to demonstrate the successful application of existing XAI post-hoc methods in financial time series forecasting in various 'black box' RNN architectures. The four presented methods provided complementary evidence, at varying granularity, behind the learnt strategies that different RNN models use to make predictions by examining the importance of features, individual inputs, and time intervals. Notably, the RNN showed limited ability to retain past information, opting to randomly fine-tune prediction using past information. Whereas both the GRU and LSTM showed evidence of retention of past information, they distinctly demonstrated different strategies exploiting the nature of their gates to make comparable predictions. These results support existing paradigms pertaining to the mechanisms governing the investigated RNNs.

The understandings derived from the presented results indicate the successful integration of long-standing XAI methods to time series forecasting to provide a global understanding of model prediction strategies. Consequently, these findings suggest that the financial time series forecasting field is not as far behind in XAI advancement as previously thought and should be held to the same XAI standards seen in other fields using machine learning methods. We acknowledge that this study is not a comprehensive exploration of existing XAI methods, nor does it aim to provide the standard for XAI methods in time series forecasting.

Instead, subsequent research should explore alternative metrics consistent between methods and provide more intuition behind the error they report. Additionally, the performance of these methods may not apply to all deep-learning strategies, as it relies on visualisation to derive meaning. In such cases, the use of CNN may provide useful means to quantify deep model behavior and prove promising for the finance sector [49], [50]. Furthermore, whilst the different error metrics provided understanding for different aspects of the network, they may not prove intuitive to average persons seeking to benefit from explainable AI. In conclusion, this study provides a strong basis for further research into XAI methodologies for time series forecasting by demonstrating the success of various existing methods in unveiling mechanisms behind predictions in various RNN architectures.

Bibliography

[1] W. Freeborough and T. van Zyl, “Investigating explainability methods in recurrent neural network architectures for financial time series data,” *Applied Sciences*, vol. 12, no. 3, p. 1427, 2022.

[2] C. Chatfield, *Time-series forecasting*. CRC press, 2000.

[3] T. Mathonsi and T. L. van Zyl, “A statistics and deep learning hybrid method for multivariate time series forecasting and mortality modeling,” *Forecasting*, vol. 4, no. 1, pp. 1–25, 2022.

[4] C. Holt, “Forecasting seasonals and trends by exponentially weighted averages (onr memorandum no. 52),” *Carnegie Institute of Technology, Pittsburgh USA*, vol. 10, 1957.

[5] P. R. Winters, “Forecasting sales by exponentially weighted moving averages,” *Management science*, vol. 6, no. 3, pp. 324–342, 1960.

[6] A. Kotsialos, M. Papageorgiou, and A. Poulimenos, “Long-term sales forecasting using holt–winters and neural network methods,” *Journal of Forecasting*, vol. 24, no. 5, pp. 353–368, 2005.

[7] G Box and G Jenkins, “Time series analysis-forecasting and control. san francisco: Holden day. 553 p.,” 1970.

[8] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[9] E. Tjoa and C. Guan, “A survey on explainable artificial intelligence (xai): Toward medical xai,” *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[10] C. Rudin, “Algorithms for interpretable machine learning,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1519–1519.

[11] M. Andrychowicz, M. Denil, S. Gomez, *et al.*, “Learning to learn by gradient descent by gradient descent,” in *Advances in neural information processing systems*, 2016, pp. 3981–3989.

[12] S. Squartini, A. Hussain, and F. Piazza, “Preprocessing based solution for the vanishing gradient problem in recurrent neural networks,” in *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS'03.*, IEEE, vol. 5, 2003, pp. V–V.

[13] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, “Lstm: A search space odyssey,” *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[14] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[15] C. Olah, "Understanding lstm networks," *colah's blog*, p. 1, 2015, [Accessed: 2021-03-31]. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[16] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[17] G. Drakos, "What is a recurrent neural networks (rnns) and gated recurrent unit (grus)," *GDCoder*, p. 1, 2019, Accessed: 2021-04-10. [Online]. Available: https://gdcoder.com/what-is-a-recurrent-neural-networks-rnns-and-gated-recurrent-unit-grus/.

[18] P. P. Angelov, E. A. Soares, R. Jiang, N. I. Arnold, and P. M. Atkinson, "Explainable artificial intelligence: An analytical review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 11, no. 5, e1424, 2021.

[19] B. Goodman and S. Flaxman, "European union regulations on algorithmic decision-making and a "right to explanation"," *AI magazine*, vol. 38, no. 3, pp. 50–57, 2017.

[20] A. Holzinger, "From machine learning to explainable ai," in *2018 world symposium on digital intelligence for systems and machines (DISA)*, IEEE, 2018, pp. 55–66.

[21] R. B. Parikh, S. Teeple, and A. S. Navathe, "Addressing bias in artificial intelligence in health care," *Jama*, vol. 322, no. 24, pp. 2377–2378, 2019.

[22] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1721–1730.

[23] S. M. Lundberg, G. Erion, H. Chen, *et al.*, "Explainable ai for trees: From local explanations to global understanding," *arXiv preprint arXiv:1905.04610*, 2019.

[24] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st international conference on neural information processing systems*, 2017, pp. 4768–4777.

[25] M. T. Riberio, S. Singh, and C. Guestrin, "Why should i trust you?" *Explaining the Predictions of Any Classifier. In KDD*, 2016.

[26] R. Confalonieri, L. Coba, B. Wagner, and T. R. Besold, "A historical perspective of explainable artificial intelligence," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 11, no. 1, e1391, 2021.

[27] K. Fauvel, T. Lin, V. Masson, É. Fromont, and A. Termier, "Xcm: An explainable convolutional neural network for multivariate time series classification," *Mathematics*, vol. 9, no. 23, p. 3137, 2021.

[28] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[29] F. Viton, M. Elbattah, J.-L. Guérin, and G. Dequen, "Heatmaps for visual explainability of cnn-based predictions for multivariate time series with application to healthcare," in *2020 IEEE International Conference on Healthcare Informatics (ICHI)*, IEEE, 2020, pp. 1–8.

[30] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[31] T. T. Nguyen, T. Le Nguyen, and G. Ifrim, "A model-agnostic approach to quantifying the informativeness of explanation methods for time series classification," in *International Workshop on Advanced Analytics and Learning on Temporal Data*, Springer, 2020, pp. 77–94.

[32] E. Delaney, D. Greene, and M. T. Keane, "Instance-based counterfactual explanations for time series classification," in *International Conference on Case-Based Reasoning*, Springer, 2021, pp. 32–47.

[33] D. H. Bailey, J. Borwein, M. Lopez de Prado, A. Salehipour, and Q. J. Zhu, "Backtest overfitting in financial markets," *Automated Trader*, 2016.

[34] S. E. Said and D. A. Dickey, "Testing for unit roots in autoregressive-moving average models of unknown order," *Biometrika*, vol. 71, no. 3, pp. 599–607, 1984.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[37] G. S. Goh, S. Lapuschkin, L. Weber, W. Samek, and A. Binder, "Understanding integrated gradients with smoothtaylor for deep neural network attribution," in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, 2021, pp. 4949–4956.

[38] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International Conference on Machine Learning*, PMLR, 2017, pp. 3319–3328.

[39] R Meyes, M Lu, C. W. de Puiseau, and T Meisen, "Ablation studies to uncover structure of learned representations in artificial neural networks," in *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, The Steering Committee of The World Congress in Computer Science, Computer . . ., 2019, pp. 185–191.

[40] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: A corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010.

[41] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[42] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with python," in *Proceedings of the 9th Python in Science Conference*, Austin, TX, vol. 57, 2010, p. 61.

[43] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "Scipy 1.0: Fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[44] N. Kokhlikyan, V. Miglani, M. Martin, *et al.*, "Captum: A unified and generic model interpretability library for pytorch," *arXiv preprint arXiv:2009.07896*, 2020.

[45] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 03, pp. 90–95, 2007.

[46] M. L. Waskom, "Seaborn: Statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.

[47] F. A. Gers and E Schmidhuber, "Lstm recurrent networks learn simple context-free and context-sensitive languages," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1333–1340, 2001.

[48] T. Trinh, A. Dai, T. Luong, and Q. Le, "Learning longer-term dependencies in rnns with auxiliary losses," in *International Conference on Machine Learning*, PMLR, 2018, pp. 4965–4974.

[49] E. Hoseinzade and S. Haratizadeh, "Cnnpred: Cnn-based stock market prediction using a diverse set of variables," *Expert Systems with Applications*, vol. 129, pp. 273–285, 2019.

[50] I. E. Livieris, E. Pintelas, and P. Pintelas, "A cnn–lstm model for gold price time-series forecasting," *Neural computing and applications*, vol. 32, no. 23, pp. 17 351–17 360, 2020.