Bank of England

Deep learning model fragility and implications for financial stability and regulation

Staff Working Paper No. 1,038
September 2023

Rishabh Kumar, Adriano Koshiyama, Kleyton da Costa, Nigel Kingsman, Marvin Tewarrie, Emre Kazim, Arunita Roy, Philip Treleaven and Zac Lovell

Bank of England

Staff Working Paper No. 1,038

Deep learning model fragility and implications for financial stability and regulation

Rishabh Kumar,[1] Adriano Koshiyama,[2] Kleyton da Costa,[3] Nigel Kingsman,[4] Marvin Tewarrie,[5] Emre Kazim,[6] Arunita Roy,[7] Philip Treleaven[8] and Zac Lovell[9]

Abstract

Deep learning models are being utilised increasingly within finance. Given the models are opaque in nature and are now being deployed for internal and consumer facing decisions, there are increasing concerns around the trustworthiness of their results. We test the stability of predictions and explanations of different deep learning models, which differ between each other only via subtle changes to model settings, with each model trained over the same data. Our results show that the models produce similar predictions but different explanations, even when the differences in model architecture are due to arbitrary factors like random seeds. We compare this behaviour with traditional, interpretable, 'glass-box models', which show similar accuracies while maintaining stable explanations and predictions. Finally, we show a methodology based on network analysis to compare deep learning models. Our analysis has implications for the adoption and risk management of future deep learning models by regulated institutions.

Key words: Deep neural networks, fragility, robustness, explainability, regulation.

JEL classification: C450, C520, G180.

---

(1) Bank of England. Email: rishabh.kumar@bankofengland.co.uk
(2) University College London. Email: adriano.koshiyama.15@ucl.ac.uk
(3) University College London. Email: k.costa@cs.ucl.ac.uk
(4) University College London. Email: nigel.kingsman.20@alumni.ucl.ac.uk
(5) Bank of England. Email: marvin.tewarrie@bankofengland.co.uk
(6) University College London. Email: e.kazim@cs.ucl.ac.uk
(7) Reserve Bank of Australia. Email: RoyA@rba.gov.au
(8) University College London. Email: p.treleaven@ucl.ac.uk
(9) Bank of England. Email: zac.lovell@bankofengland.co.uk

The views expressed in this paper are those of the authors, and not necessarily those of the Bank of England or its committees. We are grateful to the participants from the ACM AI in Finance workshop, UCL's Financial Computing and Analytics workshops and CEBRA 2022 conference for their feedback. The authors would also like to thank Philippe Bracke, Alice Parker, Andreas Joseph, Lewis Webber and an anonymous referee.

The Bank's working paper series can be found at www.bankofengland.co.uk/working-paper/staff-working-papers

Bank of England, Threadneedle Street, London, EC2R 8AH
Email: enquiries@bankofengland.co.uk

ISSN 1749-9135 (on-line)

1 Introduction

Deep neural networks or deep learning (DL), a subfield of artificial intelligence (AI), is being rapidly deployed in active financial applications (Bank of England, 2020). The advent of DL builds upon previous technologies and represents a significant leap from prior data analytic techniques used within the financial sector (Gensler and Bailey, 2020).

Surveys by the Bank of England (Bank of England, 2020) show that since the onset of the Covid-19 Pandemic, the utilisation of machine learning (ML) and especially DL has increased rapidly. DL is already being used in finance for fraud detection, regulatory compliance, market surveillance, trading, asset management, risk management, credit underwriting, and insurance underwriting.

However, due to DL models' complexity and their reliance on latent features, the models' explanations are often difficult to observe and explain. To overcome this challenge, a number of techniques (referred to as "post-hoc explanation methods") have been proposed to analyse the models' predictions and provide an interpretation of their decision drivers. In this paper, we focus on one such techniques, SHAP (Lundberg and Lee, 2017), which has proven popular among DL practitioners.

Furthermore, the architecture of DL models presents a paradox. Commentators (Gensler and Bailey, 2020) state that DL models aim to provide state-of-the-art performance. However, compared to commonly used linear methods, DL models are often subject to greater amounts of non-determinism, which can lead to a greater variability of explanations and predictions. These discussions about DL raise the importance of robustness (or stability) (Rudin, 2019) in how they work.

Therefore, the lack of robustness that we uncover in the present paper could arise from both the complex DL model but also the commonly used post-hoc explanations methods.[1]

Model robustness is important because we often aim to understand not just the model, but the underlying phenomenon being modelled (Lipton, 2018). Moreover, understanding the explanations of DL models is of interest for regulators, policy makers and members of the public alike. Explanations that are fragile or non-robust, i.e. there are multiple explanations for a modelled relationship, could produce morally hazardous incentives. The model builder could choose from a set of available explanations the one that seems more likely to satisfy the internal governance process and ignore the explanations that raise questions about the explanations of the model. In addition, this lack of stability may create problems for financial loan applicants who might be re-

[1] For the tasks investigated in this paper, only a single hyperparameter term is passed to DeepLift SHAP (see Section A.3.1), namely the baseline (which contained a subset of test samples) to be used for the calculation of SHAP values. Noting that the setting of the baseline is a point of open discussion at the time of writing (Sundararajan and Najmi, 2020a) and that performing a robust hyperparameter search for the SHAP method is computationally expensive, the specific question of exact fragility of post-hoc explanations is left for future research.

jected for arbitrary reasons. As Selbst and colleagues (Selbst et al., 2019) point out, explanations of models can act as a kind of argumentative support for its decisions.

Model fragility may produce another set of challenges on the macroeconomic front. DL frameworks often use similar datasets and methodologies and are likely to generate pro-cyclical systemic financial risks through herding and concentration of exposures (Gensler and Bailey, 2020). In such a case, having robust explanations would help regulators investigate a model's working and ensure that it is consistent with fundamentals.

In this paper we examine a variety of these models and conduct simulations on two tasks:

a) Predicting credit card default and b) predicting stock returns. Credit cards are one of the most popular sources of finance and the estimation of default risk has an impact on the availability of credit as well as the profitability of credit providers (Stein, 2005). Survey data (Bank of England, 2020) suggests that DL models are increasingly employed in assessing default risk. The latter task was chosen because of the popularity of DL models in assessing stock investment and trading opportunities. Miscalculations of investment opportunities can lead to losses at the firm level. At the aggregate level, the concentration of miscalculations can lead to incorrect asset price discovery and misallocations.

Since multiple DL architectures are best suited for certain types of datasets, we utilise a Deep Neural Network (DNN) architecture for the first task over tabular data, and we utilise long short term memory (LSTM) and gated recurrent unit architectures (GRU) for the second task over time-series data. After changing both the arbitrary (for example initial seed) and fundamental (for example the layers of a network or the neurons) parts of a model, we capture the explanations and predictions. Unlike traditional "glass-box" models, these "black-box" DL models do not reveal which inputs of the model were most influential in reaching their predictions, i.e. their feature importance. This is because DL models make predictions using latent features. To get human understandable explanations, we utilise the DeepLift SHAP ((Shrikumar et al., 2016) (Shrikumar et al., 2017) (Lundberg and Lee, 2017)) algorithm to estimate their feature importance.

Recent papers from the European Banking Authority (European Banking Authority, 2021), FCA and Bank of England (Bank of England, 2022) have highlighted the need to provide "explainability" for governance. However, there is no discussion of the fragility of these explanations itself. In this paper we show that, even if the financial institutions were to provide explanations for their models, there is a risk that, due to the underlying fragility of DL frameworks, these explanations would not show the true explanations of the model, increasing the risks for individual firms and the financial system. Furthermore, more recently financial institutions have been asked to strengthen their operational resilience frameworks (Prudential Regulatory Authority, 2022).

It should come as no surprise that central bankers are also actively looking at understanding

the interepretation of machine learning models and exploring their weaknesses. Bracke et al., propose the Quantitative Input Influence (QII) method for addressing 'black-box' problem when predicting mortgage defaults (Philippe Bracke and Sen, 2019). Joseph (2019) utilises a surrogate parametric regression analysis performed on a model's Shapley value as a solution to the lack of explainability of artificial neural networks (Joseph, 2019). Whilst Buckmann and Joseph (2022) suggest a three step approach using comparative model evaluation, feature importance analysis and statistical inference on Shapley value decompositions to enhance the modelling results and explainability of ML models (Buckmann and Joseph, 2022).

This paper contributes to the nascent ML literature for finance by firstly, assessing the fragility among DL models. The finance industry relies on the trust of end users. If DL models' fragility is pronounced then their deployment might negatively impact people's trust in the system as a whole and act as an impediment to the adoption of these models. Regulators often require financial institutions to demonstrate robust governance structures around models[2]. Ambiguous safety claims of DL models makes it crucial for the finance industry to understand DL models. It is possible that current model governance structures, drafted for simpler models, may not have considered all the risks generated by these new techniques (Gensler and Bailey, 2020). Secondly, in this paper we provide an example tool based in network analysis to monitor these models by looking at observable factors (explanations and predictions). Network analysis comprises a set of techniques to represent the relationships between a group of agents as a graph, with each agent being depicted as a node in the network and each relationship between a pair of agents being depicted as a line between those agents' nodes. The graph presentation offered by network analysis provides a compact, visual means with which to communicate to the reader information as to the occurrence of the relationships amongst a large number of agents at once. This is crucial because regulators conduct "horizon-scanning" to pre-empt and assess fragility in the system. A monitoring tool based on network analysis could provide a "SupTech" solution (Steenis, Huw van, 2019) and enable regulators to observe the system as a whole and stress test scenarios coming from unseen events. Network analysis has been long suggested as tools by central bankers to utilise to understand upcoming risks: According to Haldane (2013) "Network diagnostics ... may displace atomised metrics such as VaR in the armoury of financial policymakers". Network analysis, moreover, does have a number of associated numerical measures or metrics that can prove helpful in assessing how distinct networks differ from each other. This paper avails of such a metric to present network comparisons to the reader.

The use of fragile DL models in crucial business decisions could affect operational resilience and expose institutions to unforeseen risks. In Section 2 we lay out the challenges posed by DL

[2]see (Financial Conduct Authority, 2021; Federal Reserve, 2021), and (Prudential Regulatory Authority, 2018)

frameworks due to fragility; whilst in Section 4 we illustrate our experiments and results, and; finally in Section 5 we highlight their implications for financial stability and regulation.

2 Background and Review of Literature

2.1 The Rashomon Effect and model fragility

In statistics, the Rashomon Effect [3] is a phenomenon where for a given set of data it's possible to construct many equally well-performing models that may differ greatly in their internal structure (and hence in their inner explanations) (Breiman, 2001).

As an example, Dong and Rudin (2019) construct a Rashomon set for the recidivism data set used in the COMPAS algorithm [4]. They find that models in the Rashomon set differ significantly in the importances assigned to certain variables. In particular, the importance of criminal history is lower when the importance of race is higher, and vice versa. In such a case, taking one model out of the Rashomon set to provide "the explanation" would not be an accurate reflection of the patterns in the data - just because criminal history happens to be an unimportant variable in that one explanation does not mean that it is objectively an unimportant variable. Each of those models suggests a different explanation, even though they all have similar input-output mappings. When we lack convergence of multiple methods on the same explanation, we have fragility with the models (even if the input-output mappings are robust).

We demand that DL models are robust because they are often used to reveal and justify underlying relationships. If the models are fragile then explanations generated by any single model are likely to be insufficient.

2.2 Generating explanations for DL

In DL models, the relationship between inputs and predictions cannot be readily understood, even if the structure and the parameters of the model can be observed.

Understanding a model's behaviour is important in explaining the predictions made to support a decision making process, debugging unexpected behaviour, refining modelling and data mining processes, verifying that model behaviour is reasonable, and presenting the model's predictions to stakeholders.

Proposed explainability methods have two main scopes: (i) local methods for individual predictions, where typical users might be individuals targeted by an algorithm, and (ii) global methods

[3] Generally speaking the term "Rashomon effect" comes from the 1950 Japanese film Rashomon, in which a murder is described in four contradictory ways by four witnesses. Thus the event is given contradictory interpretations or descriptions by the individuals involved, thereby providing different perspectives and points of view of the same incident (Davenport, 2009).

[4] The Correctional Offender Management Profiling for Alternative Sanctions, or "COMPAS", algorithm is widely used across the U.S. justice system to assess the likelihood of a defendant becoming a recidivist)

for overall models, where typical users might be researchers and designers of algorithms interested in the general insights that the model produces. These methods can be intrinsic (by definition model-specific), with the model requiring no additional techniques to aid explainability, or post hoc (usually model agnostic Molnar (2021)), with techniques enhancing explainability after model training.

Researchers Ribeiro et al. (2016) defend the use of the model-agnostic approach, arguing that it increases confidence in the use of black-box models. In this study, we focus on a state-of-the-art model-agnostic approach, SHAP (SHapley Additive exPlanations) Lundberg and Lee (2017). SHAP proposes to explain the output of ML models using Shapley values.

Shapley values Shapley (1953b) were initially developed for the Game Theory domain to allocate payouts to players depending on their contribution to a total payoff in a cooperative game. In our context, each feature passed to the model is a player that participates in the game; the prediction is the payoff and the Shapley values are the payout to allocate to each feature depending on its contribution to the prediction (Shapley, 1953a).

DeepLIFT Shrikumar et al. (2017) computes the impact that each of the input features has on the model's output value using backpropagation. For each part of the DL network, it considers how the difference between the output and a reference output value can be explained by the difference between the input and a reference input value, where the reference output value is the output generated by that part of the network when passed the reference input value.

DeepLift SHAP, referred to as 'Deep SHAP' by Lundberg and Lee Lundberg and Lee (2017), combines SHAP values calculated for smaller parts of a network in order to generate SHAP values for the entire network using the same backpropagation methodology as DeepLIFT. Application of DeepLIFT is discussed in detail in the following section.

3 Identifying fragility and its implications in DL models

To understand DL model fragility and its implication for macrosystemic and the micro-consumer issues, we first create a set of DL models for credit default and stock market return predictions respectively. For each dataset, we train a collection of models differentiated by subtle variations in model architectures and model settings.

In all cases, we use the mean squared error loss function,

$$L = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2, \tag{1}$$

for model training, where $\hat{y}_i$ is the model prediction for the i-th training sample, y_i is the true outcome associated with the i-th training sample, and the training set comprises n samples. Then

we estimate the similarity of the models' explanations and predictions.

3.1 Model fragility and Hyperparameters

Understanding the behaviour of DL models poses several challenges. Firstly, DL models make their predictions on hidden variables, which are often uninterpretable as the system learns its own representation of the data which may not align with a human mental model Gensler and Bailey, 2020. Secondly, with different hyperparameters, the paramerters of the model may not be same across different builds of a model even when the input and output values are the same. To overcome these challenges, we decide to focus on the observable elements: the explanations and the predictions of the model. With Shapley values (using SHAP), we create a explanations similarity metric (a measure of how similar the explanations are between two models) and an predictions similarity metric (a measure of how similar the predictions generated by a model are to that of another model).

Shapley values (Shapley, 1953b) originated from a sub field of game theory called "cooperative game theory", which aims to allocate *payouts* to *players* depending on their contribution to the total. In our context, each feature is a *player* that participates in the game; the prediction is the *payout* and the Shapley values communicate how the feature contribution (*payout*) can be diffused across features.

Shapley value is a concept constructed on a solid axiomatic and theoretical foundation (Algaba et al., 2019). In Shapley (1953a), are defined four axioms. First, the efficiency axiom says that the sum of payoffs for all players must be equal to the total worth of the coalition. Second, the null player axiom says if a player's contribution is equal to 0, their payoff is equal to 0. Third, the symmetry axiom says if two players have symmetric contributions, they receive equal payoffs. And, finally, the additivity axiom says the selected payoff vector for the sum of two players must be the sum of the payoff vector for each player.

If we consider a sample with n features $x_1, \ldots, x_n$, $z \in \{0,1\}^n$ where the i-th component of z, z_i, is set to zero to represent the absence of feature i and to one to represent the presence of feature i, and a prediction model $v(z) \to \mathbb{R}$, then we can define the Shapley value of the i-th feature[5] as the weighted average of the marginal prediction value of including feature i across all possible absence/presence combinations of the remaining features:

$$\phi_i(v) = \sum_{F \subseteq \{z_1,\ldots,z_n\} \setminus \{z_i\}} \frac{|F|!\,(n-|F|-1)!}{n!} \left(v\left(F \cup \{z_i\}\right) - v(F)\right) \tag{2}$$

where F is the collection of possible absence/presence combinations.

For our study we utilise the DeepLift - SHAP algorithm to calculate the Shapley values for

[5]See the contributions proposed by Sundararajan and Najmi (2020b), Molnar (2021), Lundberg and Lee (2017).

the neural networks. DeepLIFT (Shrikumar et al. (2017)) uses backpropagation to compute the impact that each of the input terms to the model has on the model's output value. It does this by considering, at each part of a deep learning network, how the difference between the output value for that part of the network and a reference output value for that part of the network can be explained by the difference between the input value for that part of the network and a reference input value for that part of the network (where such reference input value generates the reference output value).

Explicitly, using the nomenclature of DeepLift Shrikumar et al. (2017), if we consider the output value, t, of a specific neuron to be of interest, and we let $x_1, x_2, \ldots, x_n$ represent inputs (where such inputs can either be the predictions of neurons immediately preceding our neuron of interest in the network, or the predictions of neurons earlier in the network, or even the inputs to the overall network) that are sufficient to compute the output value of that neuron of interest, then DeepLIFT gives

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t \tag{3}$$

where, with t^0 representing the reference output value for the neuron, $\Delta t = t - t^0$ is the difference of the output from the reference, and $C_{\Delta x_i \Delta t}$ is the contribution score assigned to Δx_i. For each input x_i, $C_{\Delta x_i \Delta t}$ can be considered that part of Δt which can be "blamed" on the difference of x_i from it's reference value (part of the reference input value above), x_i^0 (and thus, for completeness, $\Delta x_i = x_i - x_i^0$). Note that t^0 is the output value of the neuron, and x^0 the input value to the neuron, which is observed when a reference input is passed to the first layer of the network. That reference input is set by the user relying on domain-specific knowledge. As will be seen below however, DeepLift SHAP by default restricts this by setting the reference input to the first layer of the network to be comprised of the average feature values from the relevant dataset.

We can then proceed by defining a "multiplier", $m_{\Delta x_i \Delta t}$, as

$$m_{\Delta x_i \Delta t} = \frac{C_{\Delta x_i \Delta t}}{\Delta x_i} \tag{4}$$

which gives us finite differences, quantities which are similar to the partial derivative $\frac{\partial t}{\partial x_i}$ (which are finite differences as $\Delta x_i \to 0$). DeepLIFT then is able to combine these multipliers, using a "chain rule for multipliers" Shrikumar et al. (2017) to compute the multipliers for the network's overall predictions with respect to the network's inputs and thus compute the "blame" to assign to each input for the network's overall output value.

Lundberg and Lee Lundberg and Lee (2017) connect the Shapley values discussed above with DeepLIFT. To do so, they first present SHAP (Shapley Additive Explanations), a methodology that extends the use of Shapley values to prediction models that demand the presence of all

features (that is, on any forward pass through the prediction model, no features can be omitted as can be the case when calculating Shapley values). The concept of feature absence is replaced by the concept of reference value - i.e. feature importance considers the change in the prediction model output as compared to that feature taking a reference value. The existence of a unique solution, given by SHAP values, for six additive feature importance methods analyzed is shown. Lin et al. (2019) cite SHAP as a state-of-the-art explainability technique.

DeepLift SHAP, referred to as "Deep SHAP" Lundberg and Lee (2017), combines SHAP values calculated for smaller parts of a network in order to generate SHAP values for an entire network using the same backpropagation as exhibited by DeepLIFT using the "chain rule for multipliers". The use of DeepLIFT in this way relies on the multipliers of equation 4 being re-defined in terms of SHAP values:

$$m_{\Delta x_i \Delta t} = \frac{\phi_i(f^*)}{\Delta x_i} \tag{5}$$

where f^* is the function between the input x and the output value of the neuron of interest, $\phi_i()$ is the Shapley value of the i-th feature (as per Equation 2) and where x_i^0 is set to equal $E[x_i]$ for all i (i.e. the reference input to the neuron is equal to the expected input to the neuron, that is the mean average input when considering all data sample, an input that would yield the neuron output that would be predicted in the absence of knowing any of the input values). By combining importance values computed for smaller parts of the network, which can be more easily solved efficiently, DeepLift SHAP is able to offer a fast approximation of SHAP values for an entire network. In this paper we use the term SHAP, Shapley values and explanations interchangeably.

Explanations and predictions similarity are measured as the pairwise mean absolute deviation of SHAP/prediction values respectively. This value is re-scaled from 0 to 100, with a score of 100 implying duplication and 0 implying uniqueness. For a single pair of models, we calculate the mean absolute deviation of the SHAP/prediction values for model α, with SHAP/prediction values $\{s_{\alpha,1}, s_{\alpha,2}, \ldots, s_{\alpha,n}\}$, and model β, with SHAP/prediction values $\{s_{\beta,1}, s_{\beta,2}, \ldots, s_{\beta,n}\}$, where n is the number of features, as

$$Similarity_{\alpha,\beta}^{MAD} = \frac{1}{n}\sum_{i=1}^{n} |s_{\alpha,i} - s_{\beta,i}| \tag{6}$$

which are then scaled from 0 to 100 to give the relevant similarity figure between models α and β:

$$Similarity_{\alpha,\beta} = 100 \times \frac{Similarity_{\alpha,\beta}^{MAD} - \max(Similarity^{MAD})}{\min(Similarity^{MAD}) - \max(Similarity^{MAD})} \tag{7}$$

After computing the scaled similarity index values for all model pairs, separately, for explanations and predictions, we use ordinary least squares (OLS) regression to solve for the weight vector, $\mathbf{w}$, and intercept, b, giving the solution of best fit for the following set of equations:

$$y_j = \mathbf{w}^\top \mathbf{x}_j + b \quad \forall j \in 1, \ldots, m \tag{8}$$

where y_j is the scaled similarity index for the j-th model pair for explanations or predictions respectively, $\mathbf{x}_j$ is the vector of dummies for differences in hyperparameters for the j-th model pair (the k-th element of the vector of dummies takes a value of one if the k-th hyperparameter differs between the two models, and takes a value of zero otherwise) and m is the total number of model pairs. If all models with subtle changes to their hyperparameters were providing similar explanations and predictions respectively then we would find that the respective intercept term, b, tends towards 100 and all the terms $\{w_i\}$ of the respective weight vector $\mathbf{w}$ tends towards zero. An illustrative example is provided in the appendix under section A.5.

3.2 Comparing with traditional models

We compare the explanations provided by DL models with logistic regression for the credit scoring dataset and auto-regressive models for the stock returns (time-series) dataset respecitvely. This was done to provide a benchmark to compare model fragility.

3.2.1 Logistic Regression

In its simplest form, logistic regression uses the specific sigmoid function, $\sigma(x) = 1/(1+exp(-x))$, to model a binary variable. With a feature vector $\mathbf{x} \in \mathbb{R}^n$, a weight vector $\mathbf{w} \in \mathbb{R}^n$ and an intercept parameter $b \in \mathbb{R}$, we compute the probability $P(\hat{y} = 1)$ for the binary variable $\hat{y} \in \{-1, 1\}$ as

$$f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b) \ \in (0, 1) \tag{9}$$

For training, we use the logistic loss function $L(\mathbf{w}, b)$:

$$\ell(\mathbf{w}^\top \mathbf{x_i} + b, y_i) = log(1 + exp(-y_i(\mathbf{w}^\top \mathbf{x_i} + b))) \tag{10}$$

$$L(\mathbf{w}, b) = \sum_{i=1}^{m} \ell(\mathbf{w}^\top \mathbf{x_i} + b, y_i) \tag{11}$$

where there are m samples in the training set. Standard convex solvers can be used to find the $(\mathbf{w}, b)$ pair that minimise $L(\mathbf{w}, b)$. We loop through various solvers ('newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga') and their recommended penalties.

3.2.2 Auto Regression

To predict stock returns we construct an autoregression model with lagged variables of order one as input variables. We utilise Python's statsmodels library Seabold and Perktold (2010). We seek the solution for the weight vector, $\mathbf{w}$, and intercept, b, of best fit, through minimisation of the loss function as per Equation 1, for the following set of equations:

$$y_t = \mathbf{w}^\top \mathbf{x}_{t-1} + b \quad \forall t \in 1, \ldots, m \tag{12}$$

where y_t is the stock return of the dependent stock for the current time period, $\mathbf{x}_{t-1}$ is the vector containing the performances of the independent stocks for the preceding time period, and m is the number of time periods.

3.3 Comparing DL models using network analysis

Financial policy makers and regulators often need to understand the interactions and workings of financial participants as a whole. To quantify the behaviours of simulated models in the market we utilise network analysis based representations. Network analysis comprises a set of techniques to represent the relationships between a group of agents as a graph, with each agent being depicted as a node in the network and each relationship between a pair of agents being depicted as a line between those agents' nodes. The graph presentation offered by network analysis provides a compact, visual means with which to communicate to the reader information as to the occurrence of the relationships amongst a large number of agents at once. This could allow policy makers and regulators to visually observe the effects of model output, and in particular how similar market participants' models' outputs are, in the market setting. In particular, policy makers would be able to observe how market participants' model outputs might converge or diverge during periods of market stress. As stated earlier, network analysis has long been proposed as tool policy makers could rely upon (Haldane, 2013). Network analysis, moreover, does have a number of associated numerical measures or metrics that can prove helpful in assessing how distinct networks differ from each other. This paper avails of such a metric to present network comparisons to the reader, and is described below.

We model network graphs on two specific data - explanations and predictions. Metrics from network analysis methods such as degree centrality (DC) could enable regulators to monitor and supervise the model predictions. With this method, one could interpret the models as agents in the market and their behaviours could be mapped using network graphs.

We use the Pearson correlation coefficient over explanations and predictions pairs to construct links between the models or agents. The coefficient is a statistical measure that quantifies the

relationship between two variables. Revisiting our nomenclature from Subsection 3.1, we can compute the value of the Pearson correlation coefficient by $\rho_{\alpha\beta}$ for a pair of models, α and β, where model α has SHAP/prediction values $\{s_{\alpha,1}, s_{\alpha,2}, \ldots, s_{\alpha,n}\}$, and model β has SHAP/prediction values $\{s_{\beta,1}, s_{\beta,2}, \ldots, s_{\beta,n}\}$, where n is the number of features, as

$$\rho_{\alpha\beta} = \frac{\sum_{i=1}^{n}(s_{\alpha,i} - \bar{s}_\alpha)(s_{\beta,i} - \bar{s}_\beta)}{\sqrt{\sum_{i=1}^{n}(s_{\alpha,i} - \bar{s}_\alpha)^2}\sqrt{\sum_{i=1}^{n}(s_{\beta,i} - \bar{s}_\beta)^2}} \tag{13}$$

where $\bar{s}_\alpha$ is the mean average of the values $\{s_{\alpha,1}, s_{\alpha,2}, \ldots, s_{\alpha,n}\}$, and $\bar{s}_\beta$ is the mean average of the values $\{s_{\beta,1}, s_{\beta,2}, \ldots, s_{\beta,n}\}$.

The coefficient takes a value between -1 and 1, where a value of -1 indicates a perfect negative correlation (where, for example, it would be observed that as model α predicts higher values, model β predicts lower values, and vice versa), a value of 0 indicates no correlation, and a value of 1 indicates a perfect positive correlation (where it would be observed that as model α predicts higher values, model β would also predict higher values, and vice versa). The correlation coefficient is interesting to look at in various contexts because it allows us to quantify the strength and direction of the relationship between two variables. For this study, correlation analysis is important because it provides a metric with which to measure the strength of relationship between sets of predictions, with each set of predictions generated by a distinct model (agent), and the strength of relationship between sets of explanations, with each set of explanations generated by a distinct model (agent).

The correlation matrices of agents' explanations or predictions can be interpreted as weighted networks where all node pairs (i.e. agents) are connected. Specifically, agents i and j are connected by a weight of $\rho_{ij} \in [-1, 1]$, which is the Pearson correlation coefficient between agents' explanations or predictions. The Pearson correlation coefficient has been used extensively in the network analysis of time series of stock prices (Mantegna (1999)). Using a 'winner takes all' (Chi et al., 2010) method we set a minimum correlation threshold[6] to remove some edges from the diagram, leaving only the edges whose Pearson correlation coefficient are in the top quartile.This gives us with a filtered $n \times n$ correlation matrix C whose (i, j)-th element can be written as

$$c_{ij} = [\![\rho_{ij} \geq x_k + f \cdot (x_{k+1} - x_k)]\!]\rho_{ij} \tag{14}$$

where $[\![P]\!]$ is equal to 1 if P is true, and equal to 0 otherwise. x_k is the kth observation in ascending order (rounded up to one decimal place), f is the fractional part of $(n + 1) \cdot 0.75$, and x_{k+1} is the (k+1)th observation (also rounded up to one decimal place).

[6]By setting the 75th percentile as a threshold, we limit our analysis to the top 25% of the strongest relationships in the data set. By utilising a distributional cut-off we develop a comparable set, of strong correlations, among the predictions and explanations.

The adjacency matrix represents a graph as a matrix, with entries indicating the connections between vertices. An entry of 1 indicates a connection (link), while an entry of 0 indicates no connection. We arrive at the adjacency matrix by performing a further operation on the filtered correlation matrix, C, so that each element a_{ij} of the adjacency matrix, A, can be computed as

$$\begin{aligned} a_{ij} &= [\![c_{ij} \geq 0]\!] \quad \text{for } i \neq j, \\ a_{ii} &= 0. \end{aligned} \tag{15}$$

The adjacency matrix can be obtained from an edge list or an adjacency list representation of the graph. A node with several edges has a high similarity on explanations and predictions with different models. Degree Centrality (DC) captures the number of edges that one specific node has. If a node has a large number of edges it is reasonable to admit that this node is relevant for the network. In our experiment, a node represents a model with unique architecture. Considering DC as shown in Boccaletti et al. (2006), a graph G can be described by an adjacency matrix $\mathbf{A}$ (the diagonal contains zeros), an $n \times n$ square matrix whose entries a_{ij} $(i, j = 1, 2, \ldots, N)$ are equal to 1 if a link between nodes i and j exists, and zero otherwise. Considering an undirected graph, we use the degree centrality definition whereby, for a given node i, the degree centrality (k_i) is the fraction of nodes that node i is connected to. The degree centrality is thus calculated as

$$k_i = \frac{1}{n} \sum_{j=1}^{n} a_{ij}. \tag{16}$$

where i and j are nodes, and a_{ij} is an entry in the adjacency matrix $\mathbf{A}$. It should be noted that our DC calculation is invariant to the size of the network and thus presents itself as a measure that can be used to compare how the connectedness of one network differs from that of another network (without requiring the networks to be equally sized).

A highly correlated dense network shows that there are strong and numerous relationships between the nodes. This indicates that the agents are highly interconnected and that predictions/explanations generated by one agent are shared by other agents, which can be a weakness if the agents have the same biases and errors.

On the other hand, a sparse network shows that there are fewer and weaker relationships between the nodes. This may indicate that the agents are less interconnected and that the predictions/explanations of one agent will not necessarily be shared with the other agents, making the network more resilient to biases and errors that can cause systemic weaknesses.

3.4 Simulations

We simulate different variations of neural network models and then capture their predictions and explanations. Code is provided in our GitHub repository[7]. Algorithm 1 describes our methodology.

Algorithm 1 Function for simulations of explanations and predictions

Require: Data randomly split [a] into train data and test data with the same seed parameter

for `Enumerate product of setup parameters: seed count, number of epochs, learning rate, number of nodes in hidden layer, optimizer` **do**

$class.model \leftarrow$ `number of nodes in hidden layer`
$Class.Optimizer \leftarrow model,$ `optimizer, learning rate`
$Class.Optimizer.train \leftarrow$ `train data, number of epochs`
$Class.Optimizer.evaluate \leftarrow$ `test data`
return $modelPredictions$
$predictionsList \leftarrow modelPredictions$

$func.getSHAPvalues \leftarrow model,$ `test data`
return $SHAPvalues$
$SHAPlist \leftarrow SHAPlist + SHAPvalues$

end for
return $modelPredictions, SHAPList, modelHyperParameters$

[a] For the credit data models the train test split is a randomly shuffled dataset with 80:20 train-test split. For stock market analysis we trained the models on two time periods. In the first period models are trained and test prior to the financial crisis (with 80:20 train-test split). In the second scenario the models were trained before the crisis but tested during the crisis period. Please refer to section 4.2 for exact dates.

4 Experiments

4.1 Experiment 1: Credit Default Dataset

The dataset by Yeh and Lien Yeh and Lien (2009) contains data about customer defaults payment in Taiwan. This is the larger dataset used in this study. The two target classes have 23364 cases (77.88%) of 'good credit' and 6636 cases (22.12%) of 'bad credit'. This dataset has been utilised by various academic studies that studied credit rating Jadhav et al. (2018) and credit defaults Chishti and Awan (2019). More details are found in the appendix section under B.

4.1.1 Model Fragility

We captured model predictions and explanations per model and then compared each models' predictions and explanations with all other models. The dataset also contained dummies for hyperparameters. Through our calculations we generated a dataset that had dummy values for

[7] https://github.com/bank-of-england/deep-learning-herding

hyperparameter changes and an index which represents the similarity of predictions and explanations.

Table 1: explanations and predictive similarity regression for credit default. Standard errors in parentheses. (see section 4.1.1) [Generalisation loss mean 0.13]

	Explanations	**Predictive**
Intercept	52.8*** (0.9)	73.9*** (0.8)
Seed	-3.9*** (0.7)	-8.3*** (0.6)
Epochs	-0.0*** (0.0)	-0.0*** (0.0)
Learning Rate	-3.7*** (0.6)	-3.9*** (0.5)
Hidden Layer Size	-3.7*** (0.7)	-6.2*** (0.6)
Optimiser: Adam	-1.2*** (0.3)	-1.9*** (0.3)
Optimiser: SGD	-1.2*** (0.3)	-1.9*** (0.3)

*p<.1, ** p<.05, ***p<.01

In the above table we regress the index for pairwise similarities of explanations and predictions on the dummy variable that reflect hyper-parameter changes. The similarity indexes are scaled from 0 to 100 where 100 reflects duplication among a pair of model. The intercept stipulates the average similarity when controlling for changes to hyper-parameters among two models. We note that predictive similarity intercept is about 1.4 times higher than explanation similarity intercept that indicates models are more likely to have similar predictions but more dissimilar explanations. The coefficients on the dummy variable changes indicate that changes to hyper-parameter affects two models predictions and explanations. Even arbitrary changes such as model seed can affect models diverging in predictions and explanations.

Table 1 shows the coefficients for linear regression (see Section 3.1) generated from our results for explanations and predictive similarity, where the dummies for hyperparameter changes are features and the similarities indexes are targets. The "Intercept" term in Table 1 is the constant value from the regression, which reflects the average explanations and predictive similarity. The other regressors show the average impact of changes to the hyperparameters on the target. This table highlights that models are on average more likely to have similar predictions but their explanations will vary substantially. It also shows that factors like the seed of a model can significantly affect explanations and predictions.

We expected that since all models optimise on the same data - generating the same approximation function - all models would converge and become identical in their explanations and predictions. The performance for these models were also very similar if not the same (test accuracy mean $\sim$ 78%-80%). These results show that even with very subtle differences we get a Rashomon set, multiple models with similar performance but different in their inner explanations. The implications of this finding are discussed in a later section.

4.1.2 Comparing fragility of traditional methods with DL

In the previous subsection, we noted a multiplicity of DL models that perform similarly but have different inner explanations. Motivated by this, we explore if a comparable dissimilarity exists in explanations when using logistic regression, a technique commonly used for credit scoring Centre for Data Ethics and Innovation (2020). In the charts below we only consider DNN models with

high levels of performance based on test accuracy (above 78%). Figures 1a and 1b compare the mean error chart for explanations of logistic regression (log-odds) and DNN (SHAP).

The graphs highlight that when it comes to DNN models the explanations have high variability, where often an explanation can even switch directions from being positive to negative. On the other side, logistic regression provides consistent global explanations that do not change over their hyperparameters.

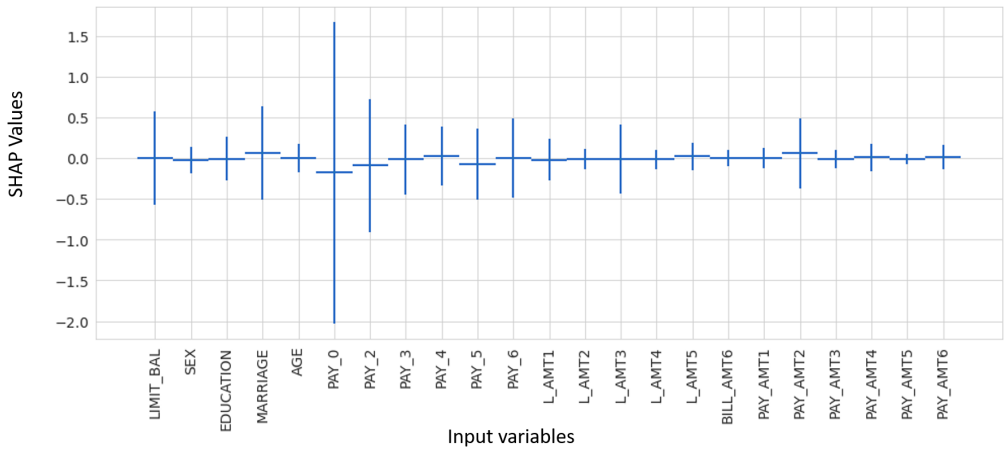
To understand the variability among deep learning models we visualise the most and least important variables from the set of equally performing models. If the DL models were robust then there must be one most important feature and one least important feature in terms of its effect size. Starting from the left in the Figure 2 we note that Pay_0 (the repayment status in the last month before default) is the most important variable for 34% of the models in the set. Other times, for example, in 9% of the models, Sex was the most important variable. Looking at the graph on the right side, we note that Pay_0 was the least important variable among 34% of the models. Having such variability will suggests the presence of the Rashomon set of equally well performing models with differing explanations. Such variability furthermore will make assessment of actual relationships very challenging.

4.1.3 Understanding model behaviour through networks

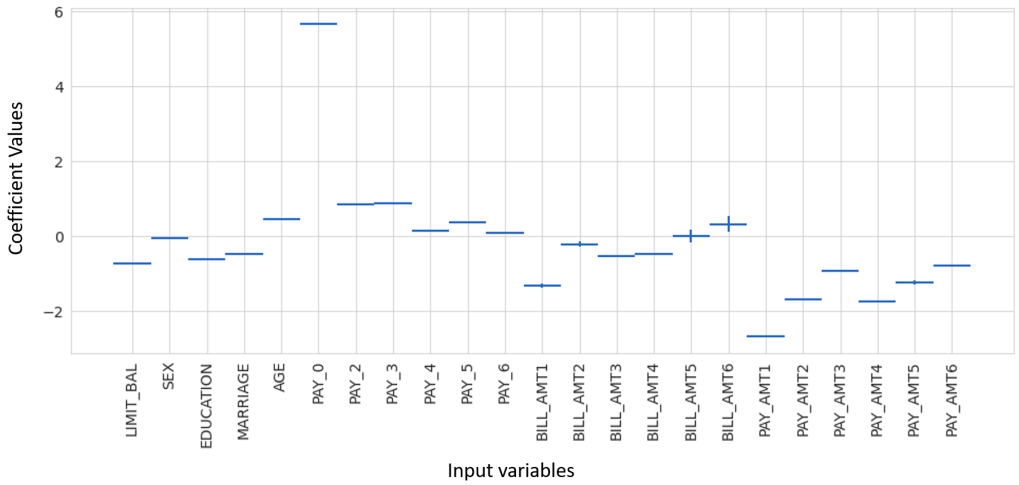
To further understand and visualise model predictions and explanations we employ network analysis. Network analysis takes into account the overall complexity of model behaviour and then charts it in a manner that non-machine learning experts can understand. Our analysis rests on a few assumptions. First, we assume that model explanations are captured through SHAP values. Second, we assume that models are working without access to information from each other. Finally we assume that pairwise correlation reflects the commonality in model explanations and predictions.

Degree centrality (DC) generated from our network analysis can provide some evidence on how strongly models are related in terms of their explanations and predictions. Ideally robust models would showcase similar levels of centrality of explanations and predictions. In the presence of a Rashomon set we would find a higher DC for predictions (models predicting similar outcomes) but lower DC for explanations (models suggesting differing explanations). This is the behaviour that is indeed revealed by our analysis. Figure 3 reveals that DC for predictions is significantly higher than for explanations, and thus that there is a lot more variation of model explanations being exhibited than there is variation of predictions. The reader should note that, where degree centrality values have been provided (in Figure 3 and elsewhere), they have already been subject to normalisation by dividing the raw degree centrality figure by the maximum possible degree in

Figure 1: Mean error bar for predicting credit default variables



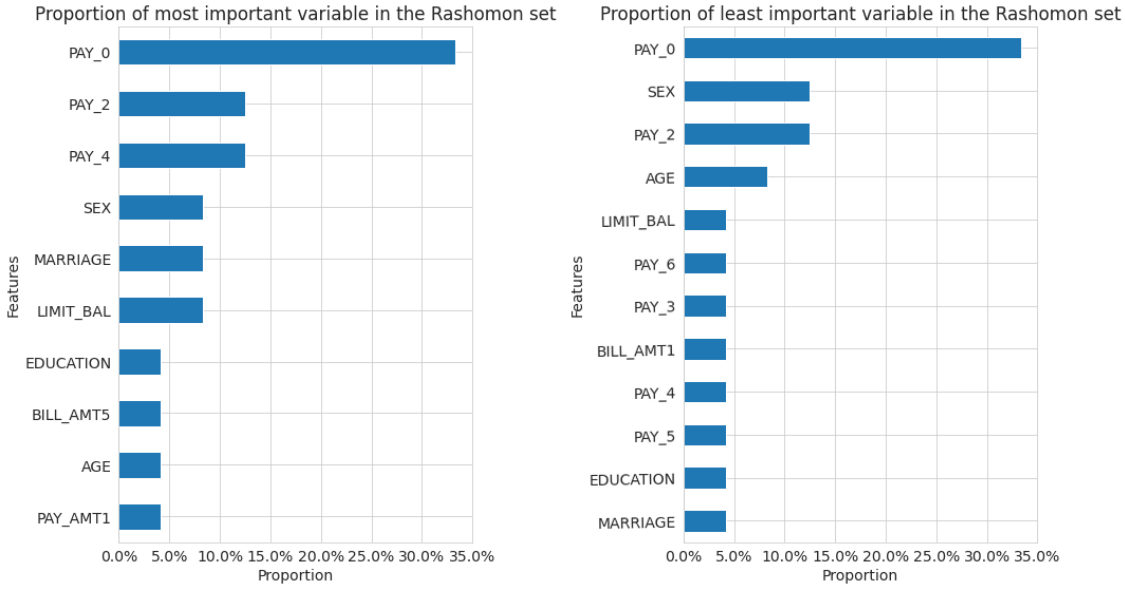
(a) Mean error bar for credit default variables SHAP values (see section 4.1.2)



(b) Mean error bar for credit default variables under Logistic Regression (see section 4.1.2)

Y axis's denotes the shapely values and the coefficient effect sizes. The x axis are the input variables in the models. The vertical lines reflect the 95% confidence interval for the average effect sizes of the models the Rashoman set of equally well performing models. We note that the confidence intervals for SHAP values are very wide compared to coefficients reflecting inconsistency of explanations among deep netural network based models.

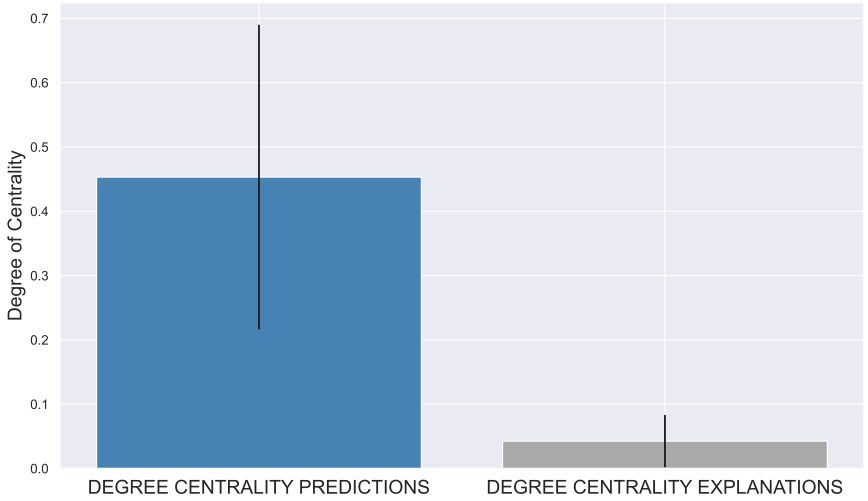
Figure 2: Graphs showing most and least important features (see section 4.1.2)



(a) (b)

a simple graph, $n-1$, where n is the number of nodes in $\mathcal{G}$.

Figure 3: Mean degree centrality for predictions (left) and explanations (right) (see section 4.1.3)



4.1.4 Discussion:

When looking at the results for credit default using a multilayered DNN architecture, our results from pairwise comparisons show low concentration on explanations and high similarities on predictions. This result exemplifies the Rashomon effect where models produce similar outcomes but dissimilar inner explanations. When explanations are compared to traditional methods such as logistic regression, we found explanations for DNN to be unstable whereas logistic regression to be more stable. Subtle differences in model development can affect its feature importance, which can affect how model developers will be able to explain the explanations. The model's feature importance changes even when the model has subtle arbitrary differences (for example the seed initialisation). In Figures 1a and 1b, we can see that the model explanations for DNN is variable compared to logistic regression. Similarly, the results generated using our network analysis approach, as given in Figure 3 and showing a reduced Degree Centrality figure for explanations as compared to the Degree Centrality figure for predictions, communicate the reduced level of relationships between our DNN models when considering explanations as compared to model predictions.

A reason for the variability among DL frameworks could be due to its reliance on random processes [8]. Another reason could be that a larger number of parameters which often implies that a model is unable to admit a unique solution. Thus a training method could select a solution out of many that are possible. This is less likely with low parameters model where the unique solution typically exists as was the case in our analysis.

[8]Completely reproducible results are not guaranteed across PyTorch releases, individual commits, or different platforms. See more here https://pytorch.org/docs/stable/notes/randomness.html

Professionals developing monitoring and supervisory technology could use our framework. The graphs presented in Figure 10 in the appendix highlights how models higher agreement between models for predictions as compared to agreement between the same models for explanations.

There could be various reasons for the divergence in explanations and predictions. First our method of understanding DL explanations is based on SHAP values, which are an approximation and thus subject to noise introduced by the approximation process (noting that approximation errors can propagate within the DeepLift SHAP procedure). Second, it could be that changes in hyperparameters also affect any randomisation processes inside the PyTorch Paszke et al. (2019) framework, even in the presence of a fixed seed.

4.2 Experiment 2: Stock Market Dataset

The data used corresponds to the daily return for 75 stocks that are members of the FTSE 100 Index (UKX). The period of analysis comprises July 20, 2001 to March 15, 2019. In total, 331,500 observations were considered, with 4,420 observations for each stock. To measure the results in periods of greater and lesser economic stability, the results were compared at two different times: *Before Crisis* and *During Crisis*. The test period in BC is January 04, 2005 to December 29, 2006. And the test period DC is July 19, 2007 to December 21, 2008. The train period in DC is the period before the test period. We follow similar simulation loops as stated in Section 3.More dataset details are found in the appendix section under B.

4.2.1 Fragility and Hyperparameters

Akin to the previous section, we train multiple DL models with stock data but different hyper-parameters. As before, our initial hypothesis is that these subtle changes would not affect the model's predictions and feature importance. Since all models optimise on the same values - using the same loss function - all models should converge and become identical.

In a detour from the previous experiment on credit card data, we now create two sets of simulations. In the first one, we train and test the data before the crisis; in the second, we train the data before the crisis but test it during crisis. This enables us to observe relationships among DNN models from a stable time period to a more turbulent time. We also utilise time series specific DL frameworks, Long short-term memory (LSTM) and Gated recurrent units (GRU). As stated previously, these architectures are better suited to capture temporal dependencies and are widely used in finance.

As with Experiment 1, we create a dataset of the explanations and predictions of DL models. We then calculate the similarity indices for features and predictions. Finally, we perform linear regression, as per Experiment 1 and as per Section 3.1, across the different dummies for pairwise

hyperparameters' changes as features, and the feature and prediction similarities indexes as targets. As seen in Table 2 and Table 3, we find a substantial and statistically significant effect of hyperparameters on the pairwise feature similarity index but a small effect of hyperparameters on the output similarity indexes. This implies that two model developers having different hyperparameters, even if they are arbitrary (for example the initial seed), could get different explanations but very similar results. In other words, each model would show different reasoning for their behaviour but still display very similar results overall. This shows the presence of a Rashomon set of models, even in the presence of small or even arbitrary changes to the models, which could display substantial differences in their explanations.

Table 2: Predictive and explanations similarities regression for stock market return before crisis. Standard errors in parentheses. (see section 4.2.1) [Generalisation loss: 0.066]

	LSTM		GRU	
	Explanations	**Predictive**	**Explanations**	**Predictive**
Intercept	59.6***(1.2)	86.1***(2.1)	52.0***(1.2)	85.0***(2.4)
Seed	-6.4***(1.0)	-3.9**(1.8)	-8.0***(1.0)	-1.2 (2.1)
Epochs	0.0 (0.0)	-0.0***(0.0)	0.0 (0.0)	-0.0***(0.0)
LR	-3.1***(0.9)	-17.3***(1.5)	-3.9***(0.9)	-20.5***(1.8)
Hidden	-3.75***(0.9)	-2.6 (1.6)	-3.7***(0.9)	-2.3 (1.8)
Adam	-7.3***(0.4)	-2.2***(0.8)	-3.2***(0.4)	-5.1***(0.9)
SGD	-7.3***(0.4)	-2.2***(0.8)	-3.2***(0.4)	-5.1***(0.9)
* p<.1,	** p<.05,	***p<.01		

Table 3: Predictive and explanations similarities regression for stock market return during crisis. Standard errors in parentheses. (see section 4.2.1) [Generalisation loss: 0.11]

	LSTM		GRU	
	Explanations	**Predictive**	**Explanations**	**Predictive**
Intercept	52.9***(1.3)	88.9***(2.1)	55.3***(1.3)	88.8***(2.0)
Seed	-6.4***(1.1)	-2.5 (1.9)	-7.4*** (1.1)	-1.1 (1.7)
Epochs	0.0 (0.0)	-0.0***(0.0)	0.0 (0.0)	-0.0***(0.0)
LR	-3.5***(0.9)	-29.3***(1.6)	-3.6***(1.0)	-12.5***(1.5)
Hidden	-3.7***(1.0)	-1.0 (1.6)	-3.2***(1.0)	-1.0 (1.5)
Adam	-6.8***(0.5)	-2.2***(0.8)	-3.6***(0.5)	-3.6***(0.8)
SGD	-6.8***(0.5)	-2.2***(0.8)	-3.6***(0.5)	-3.6***(0.8)
* p<.1,	** p<.05,	***p<.01		

Akin to table table 1, in the above tables we regress the index for pairwise similarities of explanations and predictions on the dummy variable that reflect hyper-parameter changes. The similarity indexes are scaled from 0 to 100 where 100 reflects duplication among a pair of model. The intercept stipulates the average similarity when controlling for changes to hyper-parameters among two models. We note that predictive similarity's intercept is higher than explanations similarity's intercept indicating models are more likely to have similar predictions but more dissimilar explanations. The coefficients on the dummy variable changes indicate that changes to hyper-parameter affects two models predictions and explanations. Even arbitrary changes such as model seed can affect models diverging in predictions and explanations.

4.2.2 Comparing fragility of traditional methods with DL

In the previous subsection, we found a multiplicity of models that perform similarly but have different inner explanations. As with Experiment 1, we now explore if dissimilarity in feature importance exists with other commonly used techniques such as autoregressive models. In the

tables below we restrict our attention to models with the highest levels of performance based on accuracy. DL explanations (see Figure 4a) behave similarly to what we saw in the previous experiment. Often an explanation can even switch directions from being positive to negative. On the other side, linear based regression (see Figure 4b) provides consistent global explanations that do not change over their hyperparameters settings. Please note that, in this experiment, solutions for the linear regression models were well defined given the low number of features when compared to the number of data points, ensuring a unique solution.

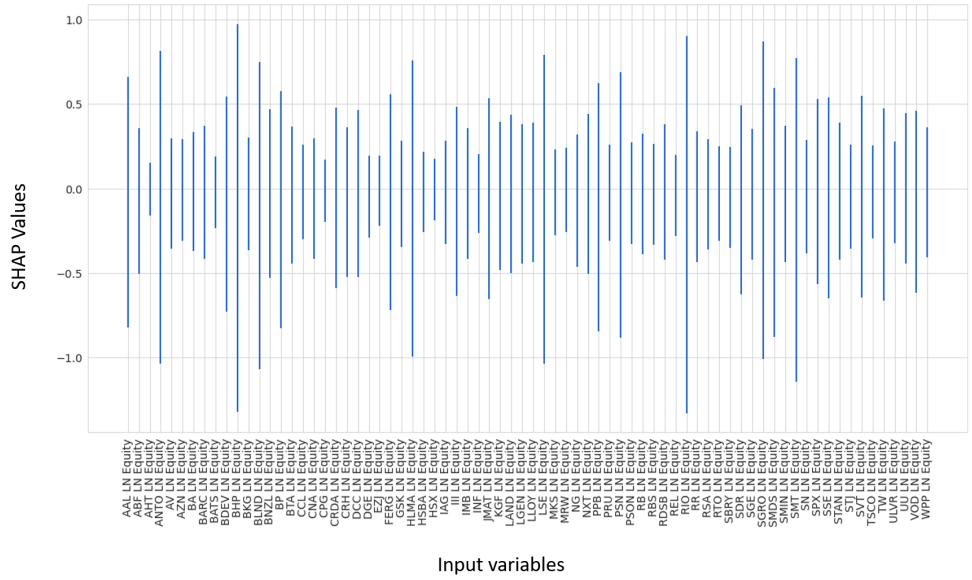
Exploring the variability among deep learning models, we visualise the most and least important variables from the set of equally performing models. If the set of DL models were robust or not fragile then there must be one most important feature and one least important feature in terms of its effect size. Firstly we note that there are several most/least important features in the set of models. Moreover starting from the left graph in Figure 5 we note that one feature (BDEV LN Equity) is presented as the most important feature 12% of the time but least important about 31% of the time. Having such variability suggests the presence of the Rashomon set of equally well performing models with differing explanations. Such variability furthermore will make assessment of actual relationships very challenging.

4.2.3 Understanding model behaviour through networks

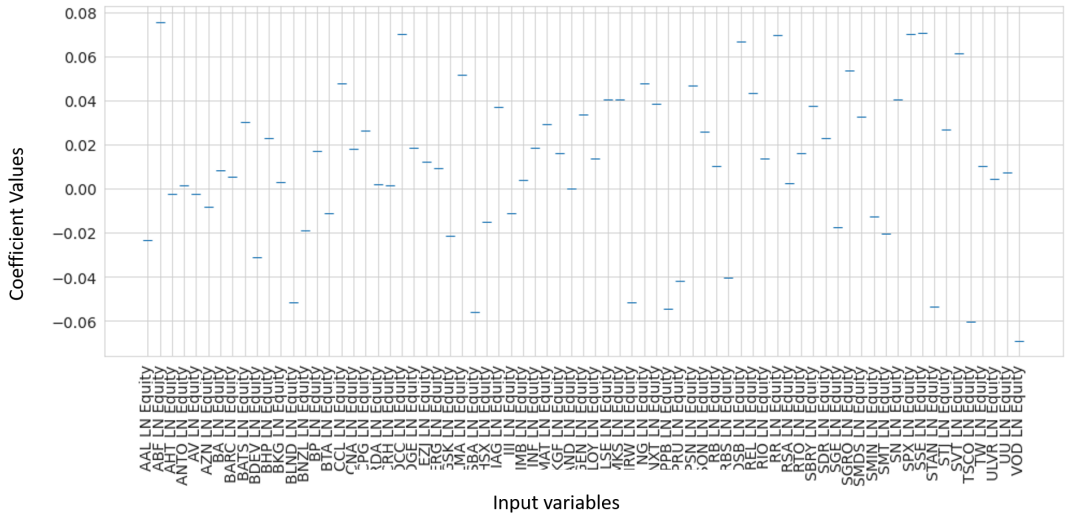
To understand model behaviour, we employ network analysis and its measures of centrality. Our analysis of model predictions and explanations rests on a few salient assumptions that were mentioned in Section 3.

As stated before, degree centrality generated from the network analysis gives evidence on how strongly related models are in-terms of their explanations or predictions. Ideally robust models would showcase similar levels of centrality of explanations and predictions. In the presence of a Rashomon set we would find a higher DC for predictions (models predicting similar outcomes) but lower DC for explanations (models suggesting differing explanations). Looking at Figure 6 we note that DC for predictions is lower during the pre crisis period suggesting an interesting result that even with similar explanations model outputs are differing. Looking at the during crisis period we note that models are predicting more similarly but providing more dissimilar results. Again, as per Subsection 4.1.3, is important to note that the degree centrality values are normalised by dividing the raw degree centrality figure by the maximum possible degree in a simple graph, $n-1$, where n is the number of nodes in $\mathcal{G}$. This normalization was performed to facilitate comparison between different networks, since degree centrality may not be directly comparable depending on the size and structure of the network. Therefore, normalization is important to make comparison between networks possible.

Figure 4: Mean error bar for predicting stock return variables



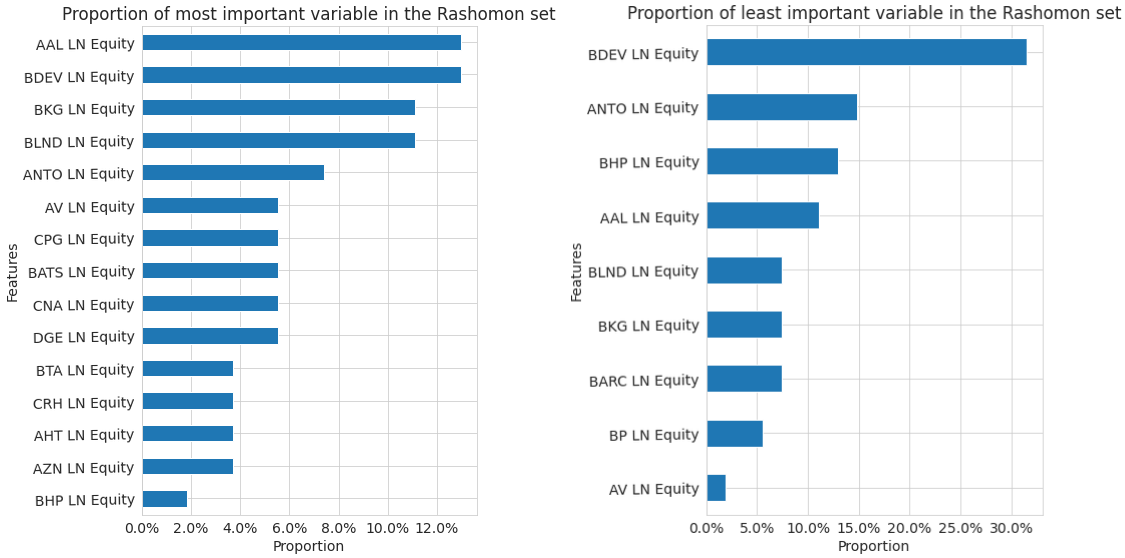
(a) Mean error bar for predicting stock return variables SHAP values (see section 4.2.2)



(b) Mean error bar for predicting stock return variables under Linear Regression (see section 4.2.2)

Y axis's denotes the shapely values and the coefficient effect sizes. The x axis are the input variables in the models. The vertical lines reflect the 95% confidence interval for the average effect sizes of the models the Rashoman set of equally well performing models. We note that the confidence intervals for SHAP values are very wide compared to coefficients reflecting inconsistency of explanations among deep netural network based models.

Figure 5: Graphs showing most and least important features (See section 4.2.2)



(a) (b)

Table 4: t-Test for difference of means with Degree of Centrality (DC) of model predictions. (see section 4.2.3) Note: * reflects difference is significant at atleast p<.05

Models	Before Crisis	During Crisis
LSTM-Predictions DC*	0.2459	0.3588
GRU-Predictions DC*	0.2661	0.4072
LSTM+GRU-Predictions DC*	0.2425	0.5947

Table 4 shows results for t-test of means of different centrality metrics during the two time periods. Essentially we see statistically significant increases in concentration in the crisis period. Combining different architectures together we observe similar behaviours where in the pre-crisis period heterogeneity in behaviour is transformed to more homogeneity during the crisis period.

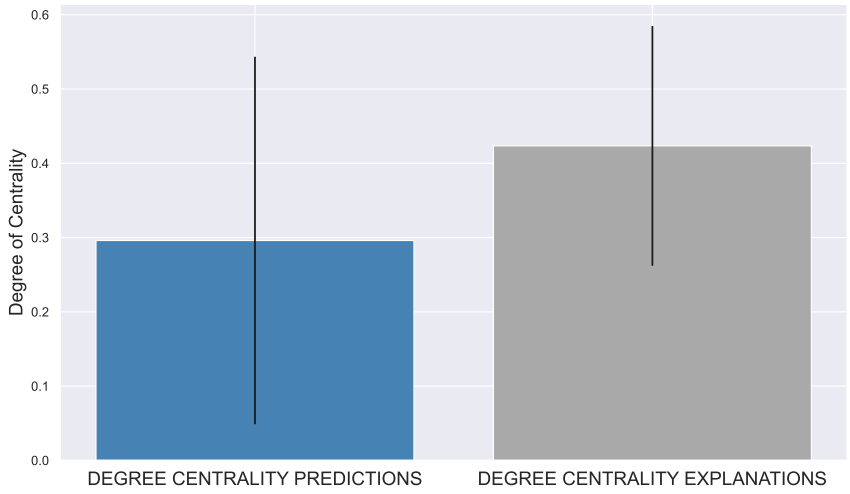
Figure 12 in appendix depicts the network graphs of predictions from the DL models during two time periods. This graph is a visual representation for degree of centrality which clearly shows the formation of one cluster during a more turbulent time.

4.2.4 Discussion

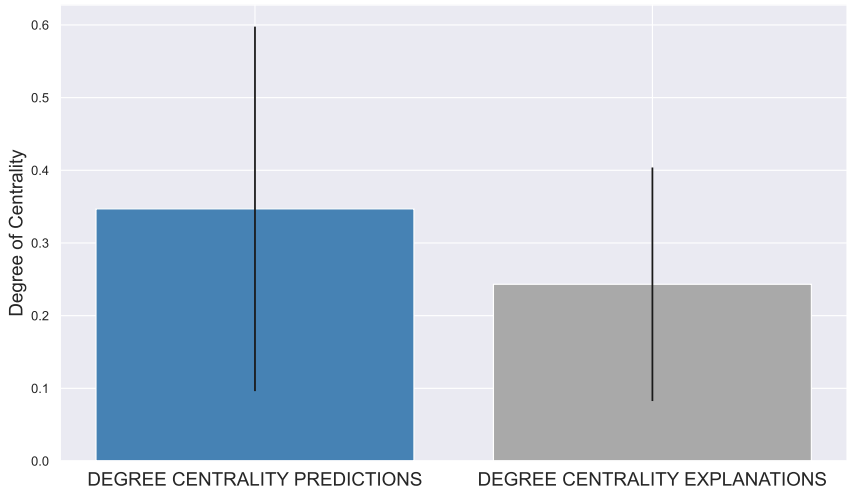
Our results are similar to those of the first experiment. In addition, our temporal analysis implies two things.

First, with respect to concentration risk, we can see more consistency of predictions among DL models in turbulent periods. In particular, our network analysis approach reveals that the prediction sets generated by different agents (models) become more concentrated during the crisis period, as shown by the degree centrality measure increasing for both the LSTM and GRU models during the crisis period (see Table 4 and Figure 6. Note that Figure 12 in the Appendix also

Figure 6: Mean degree centrality for predictions and explanations before and during crisis (see section 4.2.3)



(a) Before Crisis



(b) During Crisis

communicates the same via a graph presentation).

Second, whilst robust model explanations are desirable during a crisis period to understand the cause of the crisis, we see that, due to the Rashomon effect, we might find that the explanations arrived at may be inconsistent and possibly contradictory. Considering the network analysis approach, it can be seen how during a crisis period, even though explanations become more similar (Figure 11 in the Appendix), they nevertheless still show more dissimilarity than predictions at such time (Figure 12b in the Appendix).

We reiterate that our network analysis framework could be used by professionals developing supervisory technology to enhance their methods and observe concentration at a macro scale.

The experimentation relies on SHAP to assess model reasonings. This method has its own limitations. We find that the current state-of-the-art explanability method of SHAP values can be affected by arbitrary hyperparameter changes. Since SHAP is an approximation for Shapley values which again is an approximation for feature importance it's possible that these approximations are not robust to subtle hyperparameter changes. Better development of explanability metrics is required to monitor and assess neural network behaviours.

5 Implications for Regulation and Financial Stability

In this study, we focused on understanding the robustness of DL models in terms of their predictions or outputs and their explanations. We found that DL models with different hyper-parameter settings can yield similar performance in terms of overall accuracy and individual predictions . However, our results generally indicate non robustness or fragility when it comes to explanations. These findings posses specific challenges to the application of these models in the finance sector as they can have regulatory and financial stability implications. These implications are explored in the following points:

5.1 Consumer Trust

Finance is an industry that relies on trust. Our results suggesting model fragility and multiple explanations can have several implications downstream.

Lakkaraju and Bastani (2020) found that users that were shown multiple explanations, for the same input-output pairs, for a given black-box model, largely did not trust the model. Results from our experiments show the presence of models that work similarly but produce different explanations. Thus having variability in model explanations might lead to lack of trust by consumers (or consumer bodies) and decrease the use of automated tools thus forgoing efficiency gains that come with automatic decision making. Under UK GDPR and EU GDPR there are rights to "ex-

planation" provision through which individuals could ask for explanations of models (Information Commissioner's Office (2021) Kaminski (2019)). Users could find that contradictory explanations lead to dwindling faith in financial institutions.

5.2 Moral Hazard

The Rashomon effect shown in our simulations creates moral hazard in the following way: knowing that some explanations are more acceptable to end-users than others, developers may select the model that provides the most acceptable explanation to end users.

Research by (Lakkaraju and Bastani, 2020) demonstrates that "fairwashing" succeeds in improving perceptions. The authors use a black-box model that predicts whether to release defendants on bail. They discover from surveys that their experimental subjects considered model predictions positively when models were fair-washed i.e. when their explanation depended less on sensitive features such as race and gender. Our results from Experiment 1 depicted that 34% of the models in the set of equally performing models displayed Pay_0 as the most important feature. Nevertheless in the same set about 12% Sex was also the most important feature.

Moral hazard incentives have long been shown to change people's behaviour, for example by redirecting investments towards riskier projects (Boyd and De Nicolo, 2005). Researchers argue that moral hazard "played a central role in the events leading up to the [great financial] crisis" of 2007-08 (Dowd, 2009). Current regulation by The Prudential Regulation Authority's (Prudential Regulatory Authory, 2022) approach to banking supervision (2018) highlights how prudential regulation is aimed at addressing "moral hazard" challenges and allow firms to fail smoothly.

A framework that is not robust produces outcomes that may not reflect true reality, i.e., may not follow the logic of financial mathematics. If financial institutions are aware that there are equally well-performing models with different explanations they are less likely to accept the presented model as the ground truth, thereby remaining vigilant. Current policy issued by the PRA on algorithmic trading (Prudential Regulatory Authority, 2018) or discussion papers by the Bank of England (Bank of England, 2022) does not yet explicitly consider moral hazard. This research highlights the importance of moral hazard in terms of model development for financial institutions.

5.3 Inability to debug evolution of concentration risk

Most computing software has flaws ("bugs") or develop flaws over time. Thus it's likely that automated systems will similarly exhibit bugs. However, due to model fragility, model developers might themselves find it challenging to pinpoint problems from tools that rely on DL frameworks. A framework that does not provide consistent reasoning of its actions will be very hard to debug

as we will not have full confidence that our efforts to rectify any flaws will in fact lead to desired changes.

Concentration risk is one of the main possible causes of major losses in a credit institution. The events during the 2008-2009 financial crisis have several examples of concentrations risks within institutions (CEBS, 2010). One of the key findings from our simulations is that models even with different architectures are likely to behave in the same way but provide a different explanation to their inner explanations. This result might lead developers of models to becoming over-confident and believing that their models are able to capture a unique relationship in the data. However as the network in Figure 12 shows, in the macro sense, we encounter a system in which most agents behave very similarly to each other and thus are concentrating towards similar outcomes. This concentration could be due to spurious herding where most autonomous agents react similarly.

As Table 4 shows, measures of concentration become significantly stronger in times of distress. This could prove dangerous for the financial system which is increasingly becoming automated and relying more and more on complex machine learning models such as neural nets. Developers of AI should also invest more time in understanding how their models would work during turbulent times and have fail-safes to disengage the models under certain untested conditions. This could be especially crucial in mitigating flash crash events.

6 Conclusion and recommendations

This paper looks for evidence of fragility among DL based models and its implications on financial stability. We assess if there is agreement among models' explanations as well as their predictions. We find that models display high degrees of commonality when it comes to their predictions but less in their feature importance sets. Using the state-of-the-art SHAP values for feature importance, we find a curious result where model creators would get similar predictions but the explanations for arriving at the predictions would be very different. The results suggests the presence of a Rashomon effect. For areas where high degree of consistency and explainability is required it might be more prudent to use simpler explainable models.

The presence of a high degree of prediction commonality does not imply that these models are duplicates of each other. From our regressions we conclude that even with subtle changes to model architecture, we can get variation in model explanations. One would assume that repeated iteration over the dataset would lead to a point where all different architectures' predictions and explanations become duplicated. Inductive biases[9] can significantly shift DL models' behaviours

[9]In machine learning, the term inductive bias refers to a set of (explicit or implicit) assumptions made by a learning algorithm in order to perform induction, that is, to generalize a finite set of observation (training data) into a general model of the domain. See more here: https://link.springer.com/referenceworkentry/10.1007/978-1-4419-9863-7$_9$27

and reasoning.

Our results have several implications for regulation and financial stability. Due to the Rashomon effect and getting different explanations we could see consumer trust being affected. The presence of different explanations presents a morally hazardous scenario where model developers could choose DL models with explainations that might be appealing to their risk and governance mechanisms. If the governance mechanisms are unaware of DL model fragility they might underweight the risk of adverse outcomes under different scenarios.

For instance, when we look at different time periods we find further interesting insights. Using network analysis we are able to map out how different models would perform before and during the financial crisis of 2007-2008. The figures and analysis show that during crisis the models start becoming more similar in predictions. This brings up more challenges for the monitoring of active DL models in the financial markets. Developers of DL models should invest more time in understanding how their models would work during turbulent times and include fail-safes to stop the models under certain untested conditions.

Finally, our results highlight that a model with variable explanations could be hard to debug, as there is no certainty that efforts to rectify flaws will in fact lead to desired changes.

Reflecting on our results and implications, we suggest the following recommendations for model developers:

1. Conduct fragility testing on explanations and predictions: Model developers before deploying a deep learning model must conduct simulations similar to ours to understand their models robustness in-terms of explanations and predictions.A risk mitigating approach would be to create a wide array of competing models with a view to adopting only those models that display acceptably low amounts of variability.

2. Use simpler models for high stakes decision making: Under the current state of knowledge, certain models will exhibit randomness in their processes. In this case, organisations should strongly reconsider the usage of models that are too opaque. If the Rashomon Effect holds true then it is possible that there exists an interpretable model that performs as well as opaque models (Rudin, 2019).

3. Enhance transparency by sharing models and code: Often solving constrained problems is generally harder than solving unconstrained problems. With complete and clean data it is easier to use a black box machine learning method than to troubleshoot and solve computationally hard problems. In such a case, we must ensure that these models are as transparent as possible. Academics argue that transparency would create a positive feedback loop with any deficiencies being able to be corrected (by members of the public, agencies or consultants) creating a system which is predictable, robust and less prone to unethical manipulation (Chang et al., 2012). Reverse-

engineered credit rating models contributed to the Great Financial Crisis of 2007-2008 (Morgenson and Story, 2010). Thus transparency, in our case, could lead to models that ensure improvements in one's financial factors resulting in an improvement in one's creditworthiness.

4. Creating fail-safes: Research by (Lin et al., 2019) highlights that "wider adoption of financial artificial intelligence can amplify certain systemic risks for the financial system relating to size, speed, and linkage". We also noted that it's hard to monitor and prepare for DL model misbehaviour in different scenarios. Thus it is crucial that model developers provide fail-safes and human in the loop settings to ensure safety.

Reflecting on our results and implications we suggest the following recommendations for regulators to consider while engaging financial institutions:

1. Fragility of models: Regulators could enquire if financial institutions have included fragility of the deep learning models in the internal governance and risk management processes. Furthermore, regulators could enquire about the trade-offs of different machine learning frameworks, variability due to hyperparameter changes and impact during times of untested conditions.

2. Performance trade-off of opaque vs. transparent models: Regulators could enquire whether the financial institution in question had tried to utilise rule-based methods or simpler, more stable models in their development of tools.

3. Supervising Automation using Automation: More broadly, regulators could think about regulating deep learning based tools and also create tools and technologies that supervise these technologies. No single private financial institution can monitor the system. Regulators stand at a unique point where they can gather all the data and look for vulnerabilities. In our paper we provide an approach using network analysis to monitor autonomous agents and find vulnerabilities. Something like this can be expanded for future work.

Regardless of the interesting results and implications of this paper, we note several weaknesses. The research presented in this paper considered a single credit dataset and a single stock market dataset, whilst considering a single model-agnostic explanation methodology (SHAP). With respect to future work, understanding whether alternative model-agnostic explanation methodologies yield comparable results would be valuable. Finally, extending the research herein to actual models already being utilised in the financial industry would offer strong insights with regards to these issues.

Existing financial sector model governance and monitoring regimes, built in an earlier era of data analytic technology, are likely to fall short in addressing the new risks posed by deep learning. Sufficiently mitigating and understanding such risks will require additional research and discussion. We hope that the risks we have identified and frameworks we have developed might help contribute to furthering the dialogue with respect to model safety in the time of AI.

References

E. Algaba, V. Fragnelli, and J. Sánchez-Soriano. *Handbook of the Shapley value*. CRC Press, 2019.

Bank of England. The impact of covid on machine learning and data science in uk banking, 2020. URL https://www.bankofengland.co.uk/quarterly-bulletin/2020/2020-q4/the-impact-of-covid-on-machine-learning-and-data-science-in-uk-banking.

Bank of England. Artifcial intelligence public-private forum, 2022. URL https://www.bankofengland.co.uk/-/media/boe/files/fintech/ai-public-private-forum-final-report.pdf?la=en.

S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4):175–308, 2006. ISSN 0370-1573. doi: https://doi.org/10.1016/j.physrep.2005.10.009. URL https://www.sciencedirect.com/science/article/pii/S037015730500462X.

J. H. Boyd and G. De Nicolo. The theory of bank risk taking and competition revisited. *The Journal of finance*, 60(3):1329–1343, 2005.

L. Breiman. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231, 2001.

M. Buckmann and A. Joseph. An interpretable machine learning workflow with an application to economic forecasting. 2022.

CEBS. Cebs guidelines on the management of concentration risk under the supervisory review process. *European Banking Authority*, 2010.

Centre for Data Ethics and Innovation. Review into bias in algortihmic decision-making. *Independent Report*, 2020.

A. Chang, C. Rudin, M. Cavaretta, R. Thomas, and G. Chou. How to reverse-engineer quality rankings. *Machine learning*, 88(3):369–398, 2012.

K. T. Chi, J. Liu, and F. C. Lau. A network perspective of the stock market. *Journal of Empirical Finance*, 17(4):659–667, 2010.

W. A. Chishti and S. M. Awan. Deep neural network a step by step approach to classify credit card default customer. In *2019 International Conference on Innovative Computing (ICIC)*, pages 1–8, 2019. doi: 10.1109/ICIC48496.2019.8966723.

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. URL https://arxiv.org/abs/1406.1078.

F. Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

C. Davenport. *Media bias, perspective, and state repression: The Black Panther Party*. Cambridge University Press, 2009.

J. Dong and C. Rudin. Variable importance clouds: A way to explore variable importance for the set of good models. *arXiv preprint arXiv:1901.03209*, 2019.

K. Dowd. Moral hazard and the financial crisis. *Cato J.*, 29:141, 2009.

European Banking Authority. Eba discussion paper on machine learning for irb models, 2021.

Federal Reserve. Supervisory guidance for assessing risk management at supervised institutions with total consolidated assets less than $100 billion, 2021. URL https://www.federalreserve.gov/supervisionreg/srletters/sr1611.htm.

Financial Conduct Authority. Implementation of investment firms prudential regime, 2021. URL https://www.fca.org.uk/publication/policy/ps21-9.pdf.

G. Gensler and L. Bailey. Deep learning and financial stability. 32, November. 2020. URL https://ssrn.com/abstract=3723132.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

A. Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012.

A. G. Haldane. *Rethinking the financial network*. Springer, 2013.

Information Commissioner's Office. Guide to the general data protection regulation (gdpr), 2021. URL https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/individual-rights/rights-related-to-automated-decision-making-including-profiling/.

S. Jadhav, H. He, and K. Jenkins. Information gain directed genetic algorithm wrapper feature selection for credit rating. *Applied Soft Computing*, 69:541–553, 2018. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2018.04.033. URL https://www.sciencedirect.com/science/article/pii/S1568494618302242.

A. Joseph. Parametric inference with universal function approximators. 2019.

M. E. Kaminski. The right to explanation, explained. *Berkeley Tech. LJ*, 34:189, 2019.

H. Lakkaraju and O. Bastani. " how do i fool you?" manipulating user trust via misleading black box explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 79–85, 2020.

Z. Q. Lin, M. J. Shafiee, S. Bochkarev, M. S. Jules, X. Y. Wang, and A. Wong. Do explanations reflect decisions? a machine-centric strategy to quantify the performance of explainability algorithms. *arXiv preprint arXiv:1910.07387*, 2019.

Z. C. Lipton. The mythos of model interpretability, 2017. URL https://arxiv.org/abs/1606.03490.

Z. C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

S. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions, 2017. URL https://arxiv.org/abs/1705.07874.

R. N. Mantegna. Hierarchical structure in financial markets. *The European Physical Journal B-Condensed Matter and Complex Systems*, 11(1):193–197, 1999.

C. Molnar. *Interpretable Machine Learning*. 2021. URL https://christophm.github.io/interpretable-ml-book/.

G. Morgenson and L. Story. Rating agency data aided wall street in deals. *New York Times*, (April 24), 2010.

A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, 2015.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

C. J. Philippe Bracke, Anupam Datta and S. Sen. Machine learning explainability in finance: an application to default risk analysis. 2019. URL https://www.bankofengland.co.uk/-/media/boe/files/working-paper/2019/machine-learning-explainability-in-finance-an-application-to-default-risk-analysis.pdf.

Prudential Regulatory Authority. Supervisory statement ss5/18 algorithmic trading. 2018. URL https://www.bankofengland.co.uk/-/media/boe/files/prudential-regulation/supervisory-statement/2018/ss518.

Prudential Regulatory Authority. Operational resilience: Impact tolerances for important business services. 2022. URL https://www.bankofengland.co.uk/-/media/boe/files/prudential-regulation/supervisory-statement/2021/ss121-march-22.pdf?la=en&hash=ED32FF8608D88C585FD47B82F0C5FF0A3751E4EE.

Prudential Regulatory Authory. The prudential regulation authority's approach to banking supervision, 2022. URL https://www.bankofengland.co.uk/-/media/boe/files/prudential-regulation/approach/banking-approach-2018.pdf.

M. T. Ribeiro, S. Singh, and C. Guestrin. Model-agnostic interpretability of machine learning, 2016. URL https://arxiv.org/abs/1606.05386.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

H. Sak, A. Senior, and F. Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014. URL https://arxiv.org/abs/1402.1128.

S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

A. D. Selbst, D. Boyd, S. A. Friedler, S. Venkatasubramanian, and J. Vertesi. Fairness and abstraction in sociotechnical systems. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 59–68, 2019.

L. S. Shapley. *17. A Value for n-Person Games:*, pages 307–318. Princeton University Press, 1953a. doi: doi:10.1515/9781400881970-018. URL https://doi.org/10.1515/9781400881970-018.

L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953b. ISSN 0027-8424. doi: 10.1073/pnas.39.10.1095. URL https://www.pnas.org/content/39/10/1095.

A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016.

A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

Steenis, Huw van. The future of finance report, 2019. URL https://www.bankofengland.co.uk/-/media/boe/files/report/2019/future-of-finance-report.pdf?la=en&hash=59CEFAEF01C71AA551E7182262E933A699E952FC.

R. M. Stein. The relationship between default prediction and lending profits: Integrating roc analysis and loan pricing. *Journal of Banking & Finance*, 29(5):1213–1236, 2005.

M. Sundararajan and A. Najmi. The many shapley values for model explanation. In *International conference on machine learning*, pages 9269–9278. PMLR, 2020a.

M. Sundararajan and A. Najmi. The many shapley values for model explanation, 2020b.

I.-C. Yeh and C.-h. Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2):2473–2480, 2009.

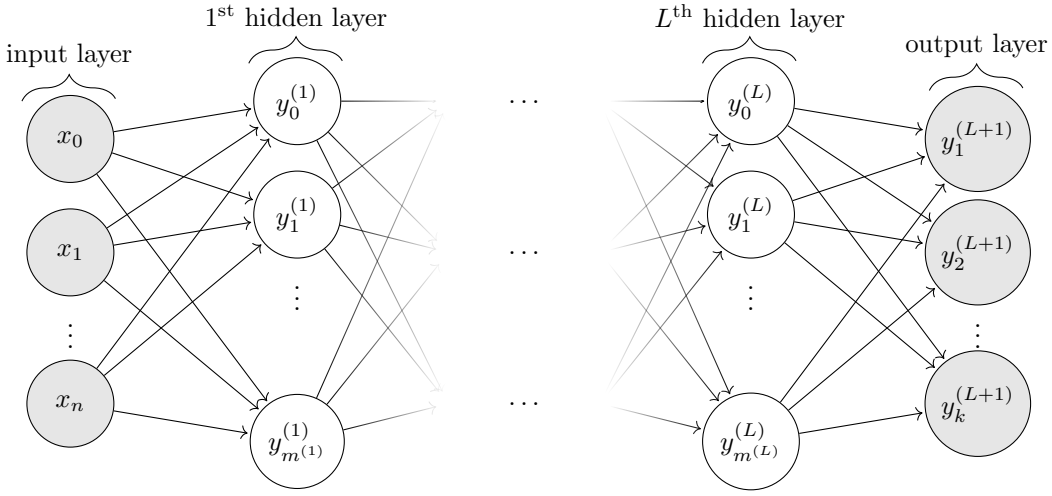
A Appendix

A.1 Deep Learning

Deep learning relies on neural networks conceptually inspired by the structure of the brain. Figure 7 is a simple example of a neural network. Each node is depicted by a circle, each edge connecting nodes is represented by a blue line, and each layer labelled. Data is fed from the first layer called the input layer. Each node contains a function that operates over a vector of weights (which form a subset of the model parameters) and comprises a non-linear transformation. As the network gets dense (with many hidden layers) the parameter calculation becomes complex. The parameter space of deep learning is exponentially larger than that of previous data analytics. For instance linear regression model parameters scale in number linearly with the number of variables included in the model. In neural networks, as the network grows in width (number of nodes in each layer) and depth (number of layers), the number of parameters grows exponentially.

Deep learning algorithms attempt to model high-level abstractions in data by using multiple processing layers. During the model training process, a model's mapping function is attempting to learn how these abstractions can be captured across several compositional operations $f(x) = f_1 f_2 f_3 \dots f_n(x)$. Various deep learning architectures such as deep neural networks, convolutional neural networks, deep belief networks and recurrent neural networks have been applied to fields like computer vision, automatic speech recognition, natural language processing, audio recognition and bioinformatics where they have been shown to produce state-of-the-art results on various tasks (Goodfellow et al., 2016; Chollet, 2021).

Figure 7: Neural network representation of a $(L+1)$-layer perceptron with n input units and k output units. The l^{th} hidden layer contains $m^{(l)}$ hidden units.



In spite of the automated nature of neural networks, there is still much human involvement in the modelling process. Creators are able to set "hyperparameters" which alters the number

of layers, the number of nodes in each layer, the nodes' activation functions, data normalization techniques, regularization techniques, and other customisations. These hyperparameters are adjusted based on the problem class and computational resource trade-offs. Setting hyperparameters creates inductive bias, priming models before seeing data. Moreover, this ability of neural networks to extract latent features from datasets is both a source of incredible predictive power and a potential source of weakness (Nguyen et al., 2015). These latent features are often unobservable, and even when they can be extracted they may not be meaningful to humans.

We use certain architectures of neural networks. Formally, we express their distinctions by the mathematical steps used to produce an output:

1. Feed-forward neural network Goodfellow et al. (2016)

$$y = \sigma(W_n(\dots\sigma(W_2(\sigma(W_1 x))\dots)) \tag{17}$$

where x and y are the input and output respectively, n is the number of network layers, W_i is the weight matrix for layer i , and σ is the non-linear activation function.

2. Long Short-term Memory (LTSM) is a special kind of recurrent neural network (RNN). RNNs are useful for time series data with each neuron able to "remember" information pertaining to the predictions of earlier neurons. The fundamental issue in a simple RNN is the *vanishing gradient problem*, an incapability to capture the long-term dependencies in a series, whereby the recurrent connections struggle to "remember" the neuron output information for those neurons much earlier in the network Graves (2012). This problem is solved using a LSTM network that considers long-term dependencies in a time series.

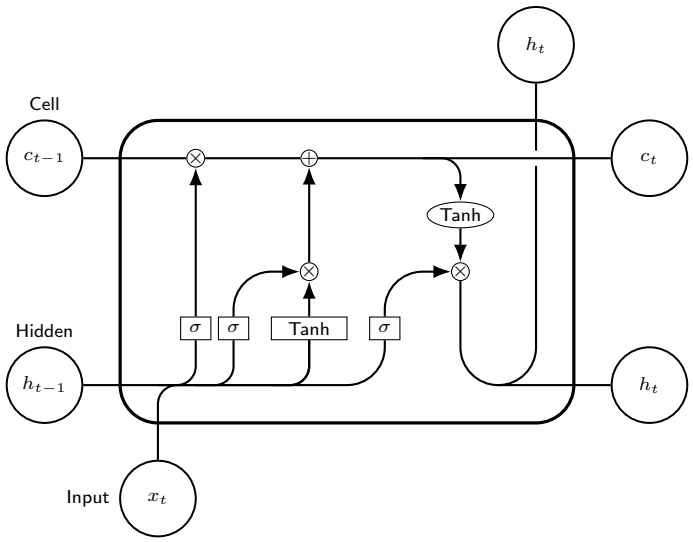
   As described by Sak et al. (2014), for each element in the input sequence, each layer computes the equations

$$\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{(t-1)} + i_t \odot g_t \\
h_t &= o_t \odot tanh(c_t) \\
\hat{y}_t &= W_y h_t + b_y
\end{aligned} \tag{18}$$

   with i_t, f_t, g_t, o_t denoting the input, forget, cell and output gates, respectively; c_t and h_t the cell activation vector and the cell output activation vector respectively; $\hat{y}_t$ the output; σ

the sigmoid function, *tanh* the hyperbolic tangent function, and $\odot$ the Hadamard product. The input, forget, and output gates are responsible for the transfer of information across the architecture, whilst the cell c_t accumulates the information processed across these gates. As their names imply, the input gate decides how much the time t input and previous hidden state still matters for the current moment; the forget gate acts as a "reset", zeroing the accumulated information stored in the cell; the output gate modulates what part of the current cell state makes it to the final hidden state.

Figure 8: Memory block for an LSTM model with one cell.



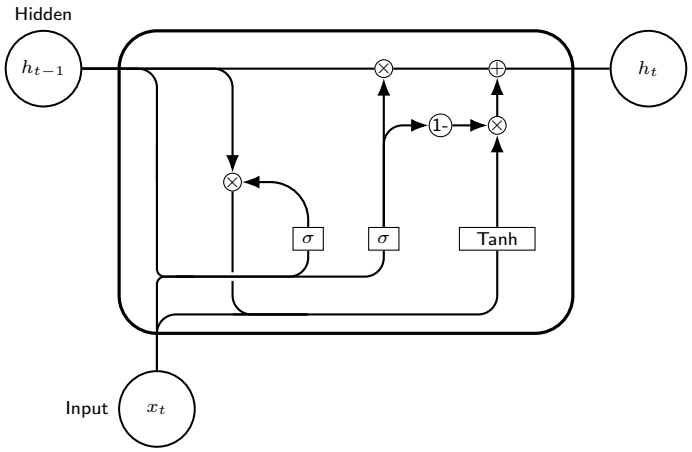
3. Proposed by Cho et al. (2014), the GRU model has only two *gates*: a *reset gate* and an *update gate*. Just like an LSTM model, a GRU seeks to solve the *vanishing gradient problem*. This occurs through two vectors (*gates*) that decide which information will be taken to the output and which will not. In this way, the structure of the model allows the adaptive capturing of the dependence of a considerable volume of data without the model's previous information being discarded. For each element of the input sequence, each layer will compute the equations Paszke et al. (2019)

$$
\begin{aligned}
r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}) \\
z_t &= \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}) \\
n_t &= \tanh\big(W_{in}x_t + b_{in} + (W_{hn}(r_t \odot h_{(t-1)}) + b_{hn})\big) \\
h_t &= (1 - z_t) \odot n_t + z_t \odot h_{(t-1)}
\end{aligned}
\tag{19}
$$

where h_t is the hidden state vector at time t, x_t is the input at time t, $h_{(t-1)}$ is the hidden state layer at time $t-1$ or the initial hidden state at time 0, and r_t, z_t, n_t are the reset, update, and new gates. σ is the activation function (sigmoid) and $\odot$ is the Hadamard product. In a multilayer GRU, the input $x_t^{(l)}$ of the l-th layer ($l \geq 2$) is the hidden state

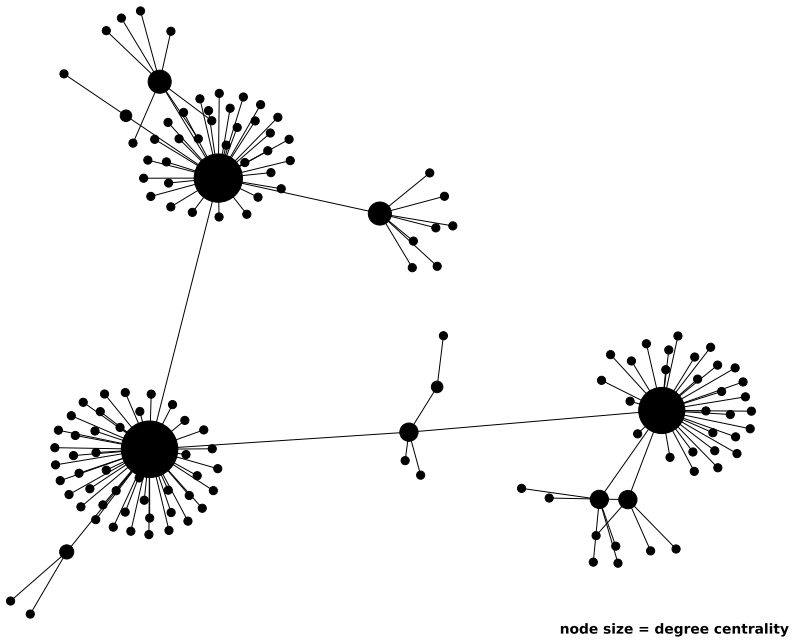
$h_t^{(l-1)}$ of the previous layer times the *dropout* $\delta_t^{(l-1)}$ where $\delta_t^{(l-1)}$ is a Bernoulli random variable where the user sets the probability of 0 being returned.

Figure 9: Memory block for a GRU model with one cell.

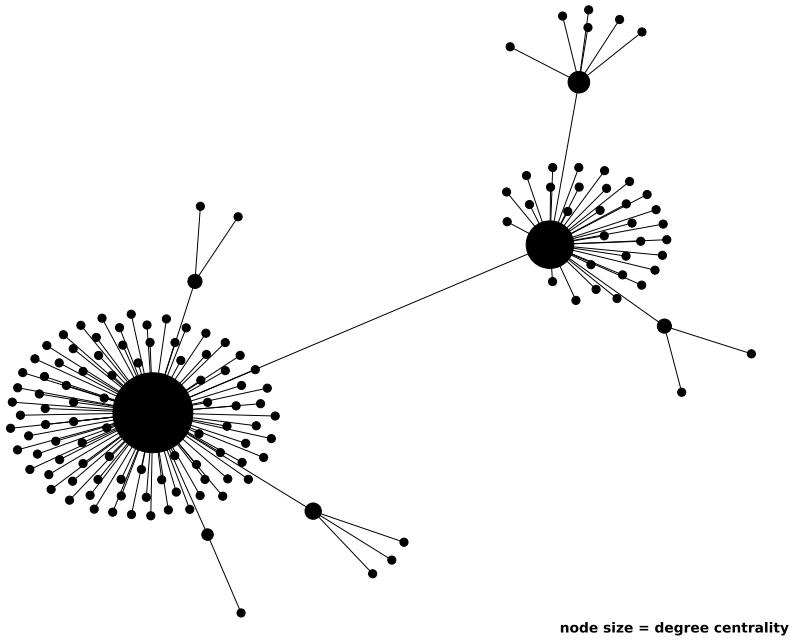


A.2 Network Analysis

Figure 10: Graphs showing degree of centrality - Credit Explanations and Predictions



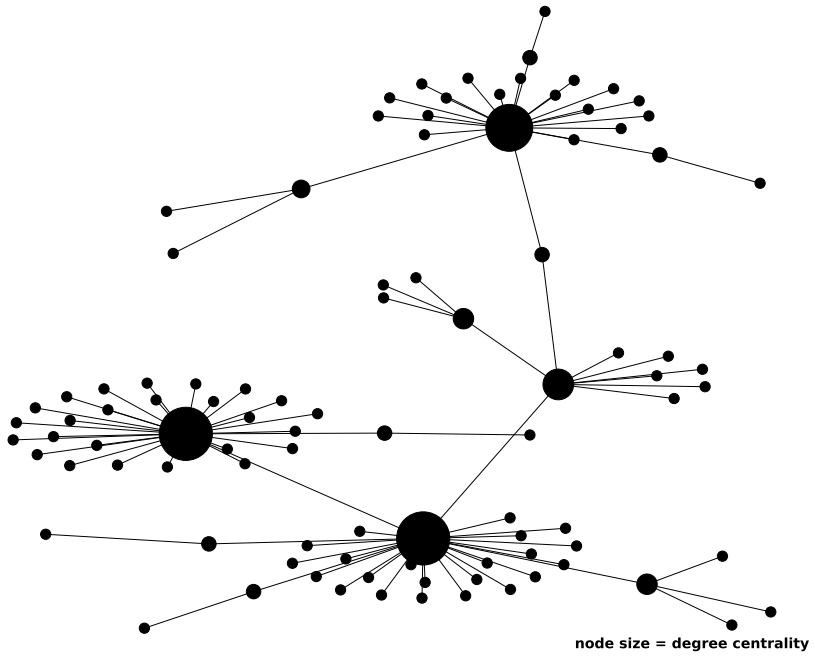
(a) Explanations



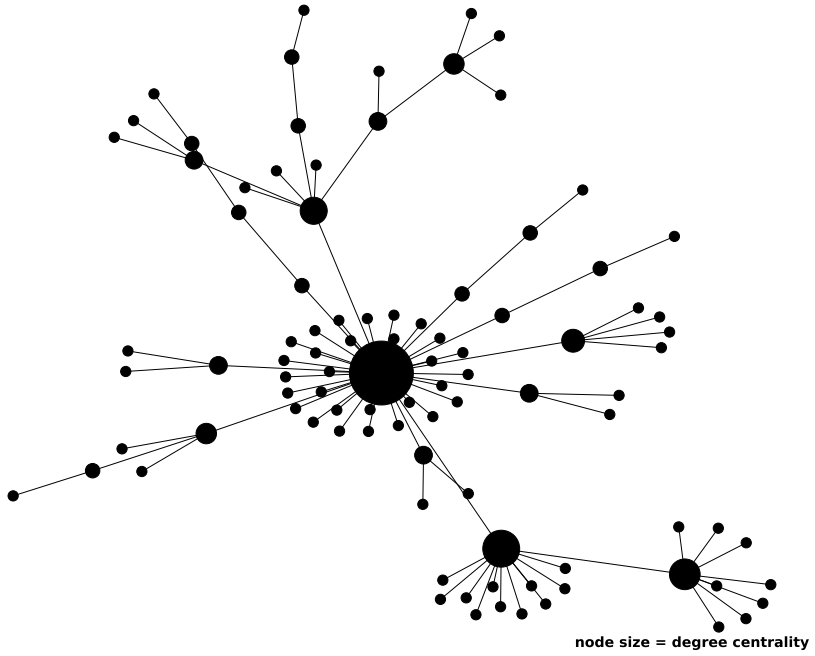
(b) Predictions

The degree centrality for a node is the fraction of nodes it is connected to. The degree centrality values are normalized by dividing by the maximum possible degree in a simple graph n-1 where n is the number of nodes in G. Color-filled circles represent the nodes community structure, and the black lines connecting them represent the edges. Node size reflects a degree centrality. By default, the layout of the nodes and edges is automatically determined by the Fruchterman-Reingold force-directed algorithm. It can be observed that graphs (a) and (b) exhibit a denser behavior and indicate that the agents are highly interconnected.

Figure 11: Graphs showing degree of centrality - Stock Market Explanations



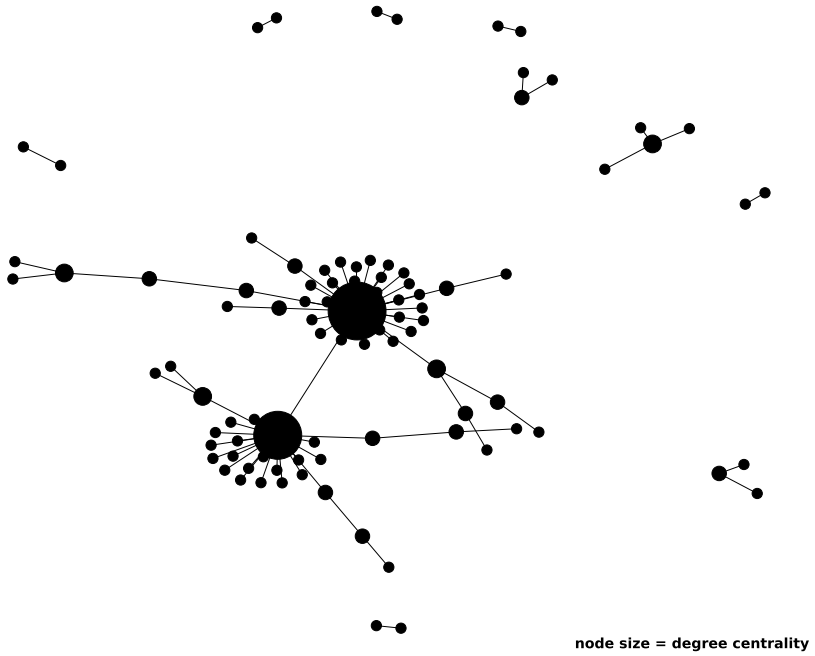
(a) Before Crisis



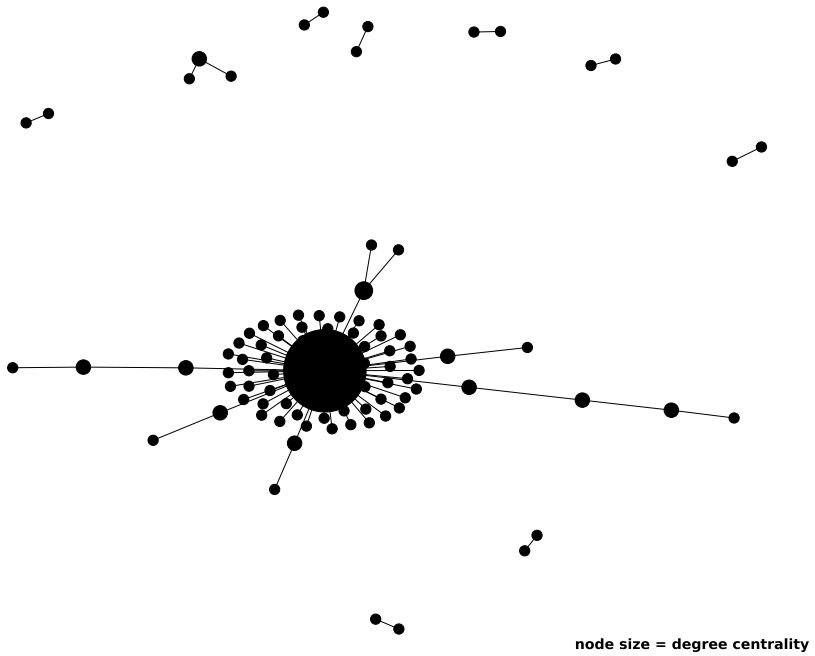
(b) During Crisis

The degree centrality for a node is the fraction of nodes it is connected to. The degree centrality values are normalized by dividing by the maximum possible degree in a simple graph n-1 where n is the number of nodes in G. Color-filled circles represent the nodes community structure, and the black lines connecting them represent the edges. Node size reflects a degree centrality. By default, the layout of the nodes and edges is automatically determined by the Fruchterman-Reingold force-directed algorithm. It can be observed that graphs (a) and (b) exhibit a denser behavior and indicate that the agents are highly interconnected.

Figure 12: Graphs showing degree of centrality - Stock Market Predictions



(a) Before Crisis



(b) During Crisis

The degree centrality for a node is the fraction of nodes it is connected to. The degree centrality values are normalized by dividing by the maximum possible degree in a simple graph n-1 where n is the number of nodes in G. Color-filled circles represent the nodes community structure, and the black lines connecting them represent the edges. Node size reflects a degree centrality. By default, the layout of the nodes and edges is automatically determined by the Fruchterman-Reingold force-directed algorithm. In the above graph we aimed to assess the behaviours of model predictions in turbulent times highlighting that model behaviours had become more concentrated in the crisis period. However as seen in the previous subsection we found that model explanations remained inconsistent.

A.3 Explainable AI

A black-box, in the machine learning context, describes models for which the relationship between inputs and predictions can't be readily understood, even if the structure and the parameters of the model can be observed. In general terms, an explainable or interpretable algorithm is one where the reasons for a decision can be questioned and explained in a way that makes sense to humans. Understanding a model's behaviour is important in explaining predictions made to support a decision making process, debugging unexpected behaviour for a model (contributing to improve the model accuracy), refining modelling and data mining processes, verifying that model behaviour is reasonable, and presenting the model's predictions to stakeholders.

Deep learning based artificial neural network models have high accuracy but a poor capacity for explainability. This characteristic makes deep learning models a challenge for institutions who need transparency, robustness, and regulatory compliance from the tools used to support decision-making.

Explainability methods have two main scopes: (i) local methods for individual prediction, and (ii) global methods for overall models. These methods can be intrinsic (by definition model-specific) or post hoc (usually model agnostic).

- Local: this scope focuses on understanding the algorithm's behaviour over only a small part of the feature space (that is, only over a narrow range of values for the inputs to the algorithm). The typical users of local explanations are individuals being targeted by an algorithm, as well as members of the judiciary and regulators trying to make a case about potential discrimination.

- Global: this scope focuses on understanding the algorithm's behaviour over a large part, and possibly all, of the feature space. The typical users are researchers and designers of algorithms, since they tend to be more interested in the general insights and knowledge discovery that the model produces rather than specific individual cases.

- Model-specific: a model is designed and developed in such a way that it is fully transparent and explainable by design. That is, an additional explainability technique is not required to be overlaid on the model in order to be able to fully explain its working and predictions.

- Model-agnostic: a mathematical technique is applied to the predictions of any algorithm including very complex and opaque models, in order to provide an interpretation of the decision drivers for those models. According to Molnar (2021), model-agnostic means that these methods can be applied to any machine learning model and are applied after the model has been trained. In this way, the independence of the model makes model-agnostic methods flexible and powerful.

Table 5: Models local (specific and agnostic) and global (specific and agnostic).

	Model-specific	**Model-agnostic**
Local	(1) Linear Model (2) Decision Tree (3) Rule-based System	(4) LIME (5) Counterfactual Explanations
Global	(1) Linear Model (2) Decision Tree (3) Rule-based System	(1) Feature Importance (2) Partial Dependence (3) SHAP

Table 5 provides interpretability model examples across the four quadrants implied above. In this study, the focus is directed to the SHAP (SHapley Additive exPlanations) model (Lundberg and Lee (2017)). It is a game theoretic approach proposed to explain the output of machine learning models. The method connects optimal credit allocation with local explanations using Shapley values.

Shapley values Shapley (1953b) originated from a sub field of game theory called "cooperative game theory", which aims to allocate *payouts* to *players* depending on their contribution to the total. In our context, each feature is a *player* that participates in the game; the prediction is the *payout* and the Shapley values communicate how the feature contribution (*payout*) can be diffused across features.

Shapley value is a concept constructed on a solid axiomatic and theoretical foundation Algaba et al. (2019). In Shapley (1953a), are defined four axioms. First, the efficiency axiom says that the sum of payoffs for all players must be equal to the total worth of the coalition. Second, the null player axiom says if a player's contribution is equal to 0, their payoff is equal to 0. Third, the symmetry axiom says if two players have symmetric contributions, they receive equal payoffs. And, finally, the additivity axiom says the selected payoff vector for the sum of two players must be the sum of the payoff vector for each player.

If we consider a sample with n features $x_1, \ldots, x_n$, $z \in \{0,1\}^n$ where the i-th component of z, z_i, is set to zero to represent the absence of feature i and to one to represent the presence of feature i, and a prediction model $v(z) \to \mathbb{R}$, then we can define the Shapley value of the i-th feature[10] as the weighted average of the marginal prediction value of including feature i across all possible absence/presence combinations of the remaining features:

$$\phi_i(v) = \sum_{F \subseteq \{z_1,\ldots,z_n\} \setminus \{z_i\}} \frac{|F|!\,(n - |F| - 1)!}{n!} \left(v\left(F \cup \{z_i\}\right) - v(F)\right) \tag{20}$$

where F is the collection of possible absence/presence combinations.

Ribeiro et al. (2016) defends the use of the model-agnostic approach to explaining the predictions of a machine learning model. The article's considerations follow a line in which the use of

[10] See the contributions proposed by Sundararajan and Najmi (2020b), Molnar (2021), Lundberg and Lee (2017).

the model-agnostic approach (applied after training the models) increases the agents' confidence regarding the use of black-box models.

Lipton (2017) brings a critical view to the concept of *interpretability*, since different ideas for the same concept are identified in the literature. Considering a set of publications, the text proposes that critical literature is absent in the machine learning community. Thus, the formulation of the problem ends up being a methodological flaw in the works, generating problems that are not reasonably solved through algorithms.

A.3.1 DeepLift SHAP Global Approximation

DeepLIFT (Shrikumar et al. (2017)) uses backpropagation to compute the impact that each of the input terms to the model has on the model's output value. It does this by considering, at each part of a deep learning network, how the difference between the output value for that part of the network and a reference output value for that part of the network can be explained by the difference between the input value for that part of the network and a reference input value for that part of the network (where such reference input value generates the reference output value).

Explicitly, using the nomenclature of DeepLift Shrikumar et al. (2017), if we consider the output value, t, of a specific neuron to be of interest, and we let $x_1, x_2, \ldots, x_n$ represent inputs (where such inputs can either be the predictions of neurons immediately preceding our neuron of interest in the network, or the predictions of neurons earlier in the network, or even the inputs to the overall network) that are sufficient to compute the output value of that neuron of interest, then DeepLIFT gives

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t \tag{21}$$

where, with t^0 representing the reference output value for the neuron, $\Delta t = t - t^0$ is the difference of the output from the reference, and $C_{\Delta x_i \Delta t}$ is the contribution score assigned to Δx_i. For each input x_i, $C_{\Delta x_i \Delta t}$ can be considered that part of Δt which can be "blamed" on the difference of x_i from it's reference value (part of the reference input value above), x_i^0 (and thus, for completeness, $\Delta x_i = x_i - x_i^0$). Note that t^0 is the output value of the neuron, and x^0 the input value to the neuron, which is observed when a reference input is passed to the first layer of the network. That reference input is set by the user relying on domain-specific knowledge. As will be seen below however, DeepLift SHAP by default restricts this by setting the reference input to the first layer of the network to be comprised of the average feature values from the relevant dataset.

We can then proceed by defining a "multiplier", $m_{\Delta x_i \Delta t}$, as

$$m_{\Delta x_i \Delta t} = \frac{C_{\Delta x_i \Delta t}}{\Delta x_i} \tag{22}$$

which gives us finite differences, quantities which are similar to the partial derivative $\frac{\partial t}{\partial x_i}$ (which are finite differences as $\Delta x_i \to 0$). DeepLIFT then is able to combine these multipliers, using a "chain rule for multipliers" Shrikumar et al. (2017) to compute the multipliers for the network's overall predictions with respect to the network's inputs and thus compute the "blame" to assign to each input for the network's overall output value.

Lundberg and Lee (2017) connect the Shapley values discussed above with DeepLIFT. To do so, they first present SHAP (Shapley Additive Explanations), a methodology that extends the use of Shapley values to prediction models that demand the presence of all features (that is, on any forward pass through the prediction model, no features can be omitted as can be the case when calculating Shapley values). The concept of feature absence is replaced by the concept of reference value - i.e. feature importance considers the change in the prediction model output as compared to that feature taking a reference value. The existence of a unique solution, given by SHAP values, for six additive feature importance methods analyzed is shown. Lin et al. (2019) cite SHAP as a state-of-the-art explainability technique.

DeepLift SHAP, referred to as "Deep SHAP" Lundberg and Lee (2017), combines SHAP values calculated for smaller parts of a network in order to generate SHAP values for an entire network using the same backpropagation methodology as exhibited by DeepLIFT using the "chain rule for multipliers". The use of DeepLIFT in this way relies on the multipliers of Equation 22 being re-defined in terms of SHAP values:

$$m_{\Delta x_i \Delta t} = \frac{\phi_i(f^*)}{\Delta x_i} \tag{23}$$

where f^* is the function between the input x and the output value of the neuron of interest, $\phi_i()$ is the Shapley value of the i-th feature (as per Equation 20) and where x_i^0 is set to equal $E[x_i]$ for all i (i.e. the reference input to the neuron is equal to the expected input to the neuron, that is the mean average input when considering all data sample, an input that would yield the neuron output that would be predicted in the absence of knowing any of the input values). By combining importance values computed for smaller parts of the network, which can be more easily solved efficiently, DeepLift SHAP is able to offer a fast approximation of SHAP values for an entire network.

A.4 Methodology

Credit Default:

Seeds = [10,20,30,40]

Epochs = [60, 100]

Learning Rate = [1E-3,1E-4, 1E-5]

Hidden Dimensions = [4,10, 20]

Optimisers = ['SGD', 'ADAM']

Stock Market:

Models = ['LSTM', 'GRU']

Seeds = [10,20,30,40]

Epochs = [60]

Learning Rate = [1E-3,1E-4, 1E-5]

Hidden Dimensions = [4,10, 20]

Optimisers = ['SGD', 'ADAM']

A.5 Methodology generating an illustrative dataset

Table 6: Illustrative example of a dataset post model simulations

Pairwise Comparison	Feature 1	Feature 2	Feature n	Hyperparameter 1	Hyperparameter 2	Prediction 1	Prediction 2
Model 1	0.1	0.2	0.6	3	10	1	1
Model 2	0.2	0.3	0.4	1	20	0	1
Model 3	0.1	0.2	0.6	6	10	1	1

Table 7: Capturing Absolute deviation

Pairwise Comparison	Feature 1	Feature 2	Feature n	Hyperparameter 1	Hyperparameter 2	Prediction 1	Prediction 2
$\lvert Model1 - Model2\rvert$	0.1	0.1	0.2	2	10	1	0
$\lvert Model2 - Model3\rvert$	0.1	0.1	0.2	5	10	1	0
$\lvert Model3 - Model1\rvert$	0	0	0	3	0	0	0

Table 8: Capturing Mean Absolute deviation with model hyperparameter dummies

Model Pair	Pairwise Explanation Mean Deviation	Pairwise Prediction Mean Deviation	Hyperparameter 1 Dummy	Hyperparameter 2 Dummy
$\lvert Model1 - Model2\rvert$	0.13	0.5	1	1
$\lvert Model2 - Model3\rvert$	0.13	0.5	1	1
$\lvert Model3 - Model1\rvert$	0	0	1	0

Table 9: Re-scaling Mean absolute deviation of explanations and predictions into 0 to 100 index

Model Pair	Pairwise Explanation Mean Deviation	Pairwise Prediction Mean Deviation	Hyperparameter 1 Dummy	Hyperparameter 2 Dummy
$\lvert Model1 - Model2\rvert$	0	0	1	0
$\lvert Model2 - Model3\rvert$	0	0	0	0
$\lvert Model3 - Model1\rvert$	100	100	0	0

Model Pair	Similarity Index: Explanation	Hyperparameter 1 Dummy	Hyperparameter 2 Dummy	Similarity Index: Predictions
$\lvert Model1 - Model2\rvert$	0	1	1	0
$\lvert Model2 - Model3\rvert$	0	1	1	0
$\lvert Model3 - Model1\rvert$	100	1	0	100

B Summary of Datasets

Table 10: Summary for Credit default data set

Variable	**count**	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
LIMIT_BAL	30000.0	167484.322667	129747.661567	10000.0	50000.00	140000.0	240000.00	1000000.0
SEX	30000.0	1.603733	0.489129	1.0	1.00	2.0	2.00	2.0
EDUCATION	30000.0	1.853133	0.790349	0.0	1.00	2.0	2.00	6.0
MARRIAGE	30000.0	1.551867	0.521970	0.0	1.00	2.0	2.00	3.0
AGE	30000.0	35.485500	9.217904	21.0	28.00	34.0	41.00	79.0
PAY_0	30000.0	-0.016700	1.123802	-2.0	-1.00	0.0	0.00	8.0
PAY_2	30000.0	-0.133767	1.197186	-2.0	-1.00	0.0	0.00	8.0
PAY_3	30000.0	-0.166200	1.196868	-2.0	-1.00	0.0	0.00	8.0
PAY_4	30000.0	-0.220667	1.169139	-2.0	-1.00	0.0	0.00	8.0
PAY_5	30000.0	-0.266200	1.133187	-2.0	-1.00	0.0	0.00	8.0
PAY_6	30000.0	-0.291100	1.149988	-2.0	-1.00	0.0	0.00	8.0
BILL_AMT1	30000.0	51223.330900	73635.860576	-165580.0	3558.75	22381.5	67091.00	964511.0
BILL_AMT2	30000.0	49179.075167	71173.768783	-69777.0	2984.75	21200.0	64006.25	983931.0
BILL_AMT3	30000.0	47013.154800	69349.387427	-157264.0	2666.25	20088.5	60164.75	1664089.0
BILL_AMT4	30000.0	43262.948967	64332.856134	-170000.0	2326.75	19052.0	54506.00	891586.0
BILL_AMT5	30000.0	40311.400967	60797.155770	-81334.0	1763.00	18104.5	50190.50	927171.0
BILL_AMT6	30000.0	38871.760400	59554.107537	-339603.0	1256.00	17071.0	49198.25	961664.0
PAY_AMT1	30000.0	5663.580500	16563.280354	0.0	1000.00	2100.0	5006.00	873552.0
PAY_AMT2	30000.0	5921.163500	23040.870402	0.0	833.00	2009.0	5000.00	1684259.0
PAY_AMT3	30000.0	5225.681500	17606.961470	0.0	390.00	1800.0	4505.00	896040.0
PAY_AMT4	30000.0	4826.076867	15666.159744	0.0	296.00	1500.0	4013.25	621000.0
PAY_AMT5	30000.0	4799.387633	15278.305679	0.0	252.50	1500.0	4031.50	426529.0
PAY_AMT6	30000.0	5215.502567	17777.465775	0.0	117.75	1500.0	4000.00	528666.0
default payment next month	30000.0	0.221200	0.415062	0.0	0.00	0.0	0.00	1.0

Table 11: Summary for Stock return data set

Stock Name	**count**	**mean**	std	min	25%	50%	75%	max
AAL LN Equity	4420.0	0.000084	0.028530	-0.224831	-0.014312	0.000309	0.015013	0.205232
ABF LN Equity	4420.0	0.000298	0.014191	-0.166260	-0.006737	0.000338	0.007476	0.085155
AHT LN Equity	4420.0	0.000608	0.040553	-1.131693	-0.013630	-0.000046	0.014970	0.642104
ANTO LN Equity	4420.0	0.000468	0.026647	-0.192403	-0.012487	-0.000047	0.014026	0.200597
AV LN Equity	4420.0	-0.000252	0.024953	-0.406043	-0.010365	-0.000084	0.010193	0.223780
AZN LN Equity	4420.0	0.000065	0.016184	-0.167426	-0.007823	-0.000011	0.008123	0.134303
BA LN Equity	4420.0	-0.000001	0.018680	-0.230862	-0.008983	0.000305	0.009358	0.117658
BARC LN Equity	4420.0	-0.000302	0.028975	-0.285682	-0.011851	-0.000288	0.011239	0.549466
BATS LN Equity	4420.0	0.000323	0.014209	-0.112377	-0.007208	0.000275	0.008101	0.120149
BDEV LN Equity	4420.0	0.000152	0.030080	-0.343632	-0.013116	0.000524	0.013477	0.325307
BHP LN Equity	4420.0	0.000333	0.023698	-0.162613	-0.012352	0.000239	0.013526	0.205986
BKG LN Equity	4420.0	0.000317	0.021875	-0.491136	-0.009480	-0.000010	0.010537	0.252326
BLND LN Equity	4420.0	0.000029	0.018203	-0.217447	-0.008577	0.000421	0.008841	0.112024
BNZL LN Equity	4420.0	0.000322	0.013832	-0.105301	-0.006628	0.000147	0.007503	0.079156
BP LN Equity	4420.0	-0.000072	0.016719	-0.140390	-0.008522	-0.000056	0.008392	0.105637
BTA LN Equity	4420.0	-0.000169	0.019756	-0.233167	-0.009599	-0.000125	0.009590	0.118651

Table 11 continued from previous page

Stock Name	**count**	**mean**	std	min	25%	50%	75%	max
CCL LN Equity	4420.0	0.000219	0.020317	-0.179856	-0.008854	0.000130	0.009687	0.154048
CNA LN Equity	4420.0	-0.000198	0.016661	-0.168393	-0.008722	-0.000045	0.008311	0.139243
CPG LN Equity	4420.0	0.000174	0.017633	-0.290106	-0.007881	-0.000010	0.008745	0.096856
CRDA LN Equity	4420.0	0.000591	0.017378	-0.114549	-0.007919	-0.000044	0.009166	0.122865
CRH LN Equity	4420.0	0.000111	0.021354	-0.179270	-0.010543	-0.000201	0.011213	0.126814
DCC LN Equity	4420.0	0.000436	0.015952	-0.102399	-0.007303	-0.000053	0.008091	0.120855
DGE LN Equity	4420.0	0.000256	0.012833	-0.097975	-0.006525	-0.000010	0.007091	0.094664
EZJ LN Equity	4420.0	0.000180	0.026557	-0.288585	-0.012337	-0.000009	0.012999	0.209509
FERG LN Equity	4420.0	0.000118	0.022621	-0.352734	-0.009665	-0.000063	0.010384	0.142959
GSK LN Equity	4420.0	-0.000132	0.013958	-0.090988	-0.007611	-0.000046	0.007312	0.101538
HLMA LN Equity	4420.0	0.000462	0.016058	-0.082873	-0.007757	-0.000046	0.008802	0.107025
HSBA LN Equity	4420.0	-0.000088	0.016346	-0.208042	-0.007341	-0.000137	0.007215	0.144103
HSX LN Equity	4420.0	0.000372	0.017769	-0.323090	-0.007081	-0.000052	0.008329	0.173459
IAG LN Equity	4420.0	0.000039	0.026927	-0.255406	-0.013841	0.000313	0.014271	0.156759
III LN Equity	4420.0	-0.000028	0.021793	-0.179388	-0.009676	0.000350	0.010025	0.191812
IMB LN Equity	4420.0	0.000269	0.014148	-0.098653	-0.007262	0.000147	0.007857	0.096686
INF LN Equity	4420.0	0.000232	0.021464	-0.287788	-0.008670	-0.000053	0.008824	0.193880
JMAT LN Equity	4420.0	0.000160	0.018826	-0.115693	-0.009476	0.000388	0.009766	0.141028
KGF LN Equity	4420.0	-0.000126	0.019619	-0.116559	-0.009880	-0.000125	0.009751	0.097670
LAND LN Equity	4420.0	-0.000079	0.016938	-0.168990	-0.007813	0.000078	0.008371	0.120777
LGEN LN Equity	4420.0	0.000080	0.023982	-0.340810	-0.009473	-0.000021	0.009953	0.242952
LLOY LN Equity	4420.0	-0.000432	0.029147	-0.414679	-0.010332	-0.000394	0.010083	0.407919
LSE LN Equity	4420.0	0.000494	0.021747	-0.180680	-0.008614	-0.000043	0.009636	0.266531
MKS LN Equity	4420.0	-0.000049	0.018866	-0.281523	-0.009249	-0.000026	0.009440	0.171889
MRW LN Equity	4420.0	-0.000053	0.016211	-0.155395	-0.008004	-0.000044	0.007843	0.136821
NG LN Equity	4420.0	0.000028	0.013388	-0.138431	-0.006237	0.000482	0.006906	0.153062
NXT LN Equity	4420.0	0.000317	0.018670	-0.163604	-0.007923	0.000260	0.008859	0.122724
PPB LN Equity	4420.0	0.000638	0.017960	-0.246481	-0.007456	-0.000054	0.008719	0.179776
PRU LN Equity	4420.0	0.000097	0.025441	-0.223194	-0.011182	-0.000012	0.011122	0.210594
PSN LN Equity	4420.0	0.000378	0.025167	-0.322299	-0.011414	0.000408	0.012450	0.162678
PSON LN Equity	4420.0	-0.000112	0.018408	-0.343717	-0.008776	-0.000011	0.008863	0.160515
RB LN Equity	4420.0	0.000348	0.013821	-0.105945	-0.007189	-0.000025	0.007884	0.099657
RBS LN Equity	4420.0	-0.000694	0.033826	-1.095780	-0.011905	-0.000427	0.011235	0.305000
RDSB LN Equity	4420.0	-0.000022	0.016421	-0.098338	-0.008340	0.000224	0.008519	0.132053
REL LN Equity	4420.0	0.000132	0.015394	-0.165405	-0.007427	-0.000013	0.007936	0.106524
RIO LN Equity	4420.0	0.000260	0.026115	-0.457920	-0.011941	0.000497	0.013163	0.196582
RR LN Equity	4420.0	0.000241	0.021282	-0.217738	-0.010134	-0.000011	0.010154	0.142284
RSA LN Equity	4420.0	-0.000338	0.022152	-0.242665	-0.009024	-0.000083	0.008750	0.169166
RTO LN Equity	4420.0	0.000017	0.019632	-0.361017	-0.007943	-0.000050	0.008471	0.160697
SBRY LN Equity	4420.0	-0.000214	0.017658	-0.232387	-0.008419	-0.000016	0.008224	0.135567
SDR LN Equity	4420.0	0.000204	0.022668	-0.290444	-0.009969	0.000492	0.010576	0.280113
SGE LN Equity	4420.0	0.000186	0.019644	-0.129508	-0.008996	0.000502	0.009524	0.132378

Table 11 continued from previous page

Stock Name	count	mean	std	min	25%	50%	75%	max
SGRO LN Equity	4420.0	-0.000065	0.019526	-0.274610	-0.007954	-0.000012	0.008398	0.167944
SMDS LN Equity	4420.0	0.000239	0.022222	-0.153866	-0.010082	-0.000064	0.010241	0.178172
SMIN LN Equity	4420.0	-0.000023	0.017175	-0.398869	-0.008161	-0.000009	0.008888	0.109360
SMT LN Equity	4420.0	0.000371	0.014624	-0.119761	-0.006466	0.000709	0.007970	0.100606
SN LN Equity	4420.0	0.000254	0.016167	-0.139109	-0.008015	-0.000034	0.008118	0.104608
SPX LN Equity	4420.0	0.000542	0.015078	-0.113103	-0.006893	-0.000046	0.007833	0.100892
SSE LN Equity	4420.0	0.000060	0.013311	-0.129945	-0.006437	-0.000009	0.007029	0.134520
STAN LN Equity	4420.0	-0.000084	0.023143	-0.179487	-0.010964	-0.000198	0.010404	0.262235
STJ LN Equity	4420.0	0.000144	0.023521	-0.232908	-0.010434	-0.000044	0.010737	0.239275
SVT LN Equity	4420.0	0.000105	0.013878	-0.160706	-0.006978	-0.000011	0.007372	0.151524
TSCO LN Equity	4420.0	-0.000081	0.016495	-0.174221	-0.008448	-0.000112	0.008324	0.139513
TW LN Equity	4420.0	-0.000008	0.034488	-0.539120	-0.012651	0.000262	0.013128	0.548230
ULVR LN Equity	4420.0	0.000199	0.013996	-0.116927	-0.006890	0.000358	0.007367	0.125956
UU LN Equity	4420.0	-0.000031	0.013803	-0.283347	-0.006836	-0.000010	0.007221	0.107908
VOD LN Equity	4420.0	-0.000143	0.018368	-0.213956	-0.009088	-0.000086	0.008832	0.119414
WPP LN Equity	4420.0	-0.000020	0.019303	-0.148020	-0.009622	-0.000042	0.009841	0.098015
WTB LN Equity	4420.0	0.000296	0.017305	-0.163896	-0.007711	-0.000010	0.008612	0.165766

Figure 13: UKX stock returns and histogram from July 20, 2001 to March 15, 2019.

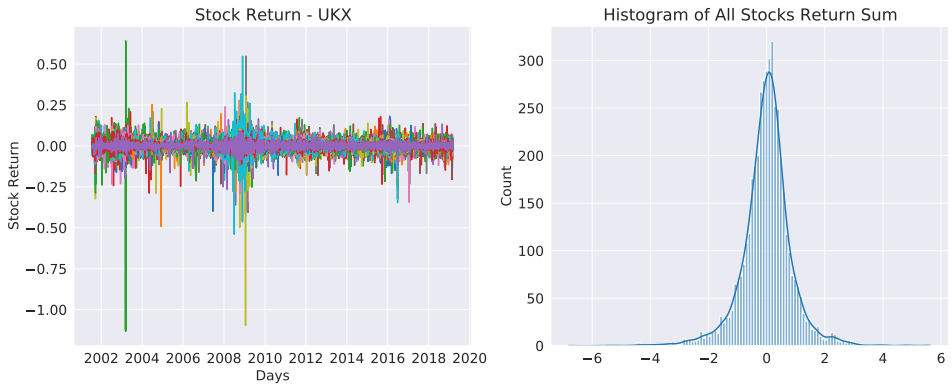


Figure 14: Target stock from July 20, 2001 to March 15, 2019.