A study on forecasting SOFR with a recurrent neural network using long short-term memory cells

University of Twente
Financial Engineering and Management
Master Thesis

Joep Cornelissen

May 2021

Rabobank N.V.

Company supervisor
Philip Marey

University supervisors
Berend Roorda
Wouter van Heeswijk

Management summary

In this research the forecast performance of a neural network on the Secured Overnight Financing Rate (SOFR) is evaluated. The financial market underlying SOFR is studied and suitable exogenous variables are picked to help the neural network forecast SOFR. Through a literature research, a neural network model is chosen which is suitable for forecasting SOFR according to the consulted studies. The recurrent neural network model using long short-term memory cells is chosen and applied to historical data of SOFR and its exogenous variables. The performance of the neural network is evaluated by comparing the performance of the neural network to an autoregressive integrated moving average model with exogenous variables (ARIMAX).

The Secured Overnight Financing Rate (SOFR) was chosen by the Alternative References Rates Committee (ARRC) as the replacement of the US dollar London Interbank Offered Rate (LIBOR). SOFR is fully based on actual transactions in the repurchase agreement (repo) market, making SOFR more volatile and harder to forecast than USD LIBOR. A special feature of SOFR is its end month spikes showing a recurring pattern. The Federal Reserve intervenes in the repo market to regulate liquidity in the financial market, thus influencing SOFR.

For forecasting SOFR seven exogenous variables are selected to be used by the neural network model.

i. The effective federal funds rate is used as a representation of the general trend of SOFR and as the expression of the monetary policy of the federal reserve.
ii. The volume of repurchase agreements executed represents how much liquidity is injected into the financial system by the Federal Reserve to steer rates according to the monetary policy.
iii. The volume of reverse repurchase agreements executed serves as an indication of how much liquidity is drawn out of the financial system by the Federal Reserve.
iv. The Chicago board options exchange volatility index is used as a variable to gauge U.S. market sentiment and the degree of fear in the money market to hold on to money influencing liquidity in the repo market.
v. The total assets on the federal reserve's balance are a representation of the total liquidity that is injected into the financial market by the Federal Reserve.
vi. A date dummy is set on the last day of the month to help the neural network recognize the end of the month.
vii. Another date dummy is set on the dates that the Federal Open Market Committee (FOMC) meetings are scheduled to recognize the dates on which monetary policy might be changed.

In a literature review, the recurrent neural network model using long short-term memory cells is chosen as the type of neural network which should be able to forecast SOFR with the lowest error. The recurrent neural network is a neural network suited for sequence prediction which passes on the information of past observations to make a good forecast. Furthermore, the long short-term memory nodes are used to decide which information gets passed on to make a forecast and which information is forgotten and not passed on for future forecasts. An autoregressive integrated moving average model with exogenous variables is used as a baseline comparison model to compare the performance of the neural network model.

To train a neural network a set of hyperparameters is defined. These hyperparameters can be tuned to improve the training of the neural network resulting in better forecasts of the model. The number of layers and number of neurons per layer can be varied. The root mean squared error is used as the loss function of the neural network model because this loss function punishes the model for making big mistakes which is what is needed when forecasting outliers like the spikes in SOFR. The adaptive moment estimation method is used as an optimizer to train the neural network model, this optimizer is industry standard and the best performing optimizer for complex objectives like neural networks. The learning rate can be varied to adapt the step size the model takes when optimizing the model,

this is important for finding the global optimum and not getting stuck in local optima. The batch size can be adjusted to change the number of data points that are in one sequence that is used to train the model. The number of epochs can be adjusted to change the number of iterations of training the model. The number of epochs is very important for underfitting or overfitting of the model if it is set too low or too high.

The hyperparameters are optimized using a Bayesian optimization algorithm which is an intelligent algorithm for testing combinations of hyperparameters. When picking a new combination of hyperparameters the algorithm considers the performance of already observed values and which unseen values give a high chance of improving the best combination of hyperparameters so far. This is known as the exploitation and exploration trade-off.

To evaluate the performance of the model, experiments are defined to test the recurrent neural network on unseen data. Every experiment has a different period on which the model is trained and a different forecast period. The different forecasting periods are defined to have different characteristics to test if the neural network can handle spikes appearing or disappearing in the data.

The performance of the neural network model is first tested by using all exogenous variables and an average root mean squared error (RMSE) of 0.02720 is found. The neural network model shows superior results to the ARIMAX model on all experiments. It can be concluded that the neural network outperforms the ARIMAX model on minimizing the root mean squared error by 53% if all variables are used. The significance of all variables is assessed based on the P-values attained in the ARIMAX analysis. Based on the P-values we conclude that using the effective federal fed funds rate, end of the month dummy and the volume of reverse repurchase agreements are the best predictors for SOFR. We then use the three significant exogenous variables to train the neural network to further improve the performance of the neural network and we find an average mean squared error of 0.02448, which is 0.00272 lower than the root mean squared error when using all exogenous variables. When the neural network model is trained with the selected exogenous variables it outperforms the ARIMAX model for an average reduction of 45% in root mean squared error. The ARIMAX model fails to recognize when spikes are disappearing and not present anymore in the data. Through the long short-term memory nodes, the recurrent neural network has the ability to recognize the spikes when are apparent and understand when the spikes in SOFR disappear.

Preface

I would like to thank several persons that helped me during the course of this master thesis. Firstly, I would like to thank my supervisor of the University of Twente Berend Roorda for his guidance during my thesis. His support and feedback have always been very useful throughout the whole master of Financial Engineering and Management. Secondly, I would like to thank Wouter van Heeswijk for sharing his knowledge on machine learning and neural networks with me. His critical view helped me to improve my model considerably.

Furthermore, I would like to thank the Rabobank for giving me the opportunity to conduct my master thesis at the RaboResearch Financial Markets department. Especially I would like to thank my supervisor at the Rabobank Philip Marey for all his advice and directions throughout the whole period of my master thesis. I learned a lot from all the knowledge he shared on the financial world, and he gave me a very good understanding of all the players and events in the U.S. money market.

At last, I want to thank my parents for their everlasting support and their sincere advice throughout my entire period at the University of Twente.

Joep Cornelissen

Utrecht, May 2021

Glossary

API	Application Programming Interface
AR	Autoregression
ARIMA	Auto Regressive Integrated Moving Average model
ARIMAX	Auto Regressive Integrated Moving Average model with exogenous variables
ARRC	Alternative References Rates Committee
EFFR	Effective Federal Funds Rate
EOM	End of the month
EOQ	End of the quarter
EOY	End of the year
FCA	Financial Conduct Authority
FOMC	Federal Open Market Committee
FSB	Financial Stability Board
GSIB	Global Systemically Important Banks
IOER	Interest on Excess Reserves
IOSCO	International Organization of Securities Commissions
LIBOR	London Interbank Offered Rate
LSTM	Long Short-Term Memory
MAPE	Mean Absolute Percentage Error
MBS	Mortgage-Backed Security
OLS	Ordinary Least Squares
ONRRP	Overnight Reverse Repo
PMCCF	Primary Market Corporate Credit Facility
QE	Quantitative Easing
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SOFR	Secured Overnight Financing Rate
VAR	Vector Autoregression
VIX	Chicago Board Options Exchange Volatility Index

Table of contents

List of figures

List of tables

1 Introduction

In 2017 the Secured Overnight Financing Rate (SOFR) was selected as an alternative to U.S. Dollar London Interbank Offered Rate (LIBOR). SOFR is a robust, transaction-based and secured rate based on overnight transactions in the U.S. Dollar Treasury repurchase agreement market, or 'repo market' (Alternative Reference Rates Committee, 2020). The repo market is a market for short-term loans collateralized by U.S. treasury. SOFR is fully based on actual transactions in the repo market on a daily basis. This makes SOFR volatile and susceptible to changes in liquidity in the repo market. Changes in liquidity tend to happen at month-end when treasury debt is settled and tax payments are due which results in a shortage of liquidity. This shortage of liquidity causes SOFR to spike on the last day of the month. This phenomenon is a special feature in SOFR which makes it difficult to forecast using conventional linear models. That is where neural networks come into play because neural networks are able to detect complicated non-linear patterns in data. Using a neural network to make forecasts of SOFR is a unique combination that has not been done before. In this thesis, the new reference rate SOFR and neural networks are brought together to explore the new opportunities and difficulties that arise from applying neural networks on time series forecasting.

1.1 Relevance for Rabobank

This research is done for the Rabobank department financial markets research. This department does first-line research to provide timely analysis and strategic thinking on financial markets. That is why the department is interested in the developments of the new alternative reference rate SOFR and how this rate differs from its predecessor LIBOR. Since in the future all contracts will be referencing SOFR and not LIBOR it is useful to have a forecast of SOFR. The forecast is used by corporate clients to mitigate their interest rate exposure and hedge their treasury positions accordingly. Rabobank provides its corporate customers, pension funds and insurers with valuable forecasts of SOFR so these customers enter into a swap contract with the Rabobank to hedge their interest rate risk. The Rabobank does not make a profit on providing the forecast of SOFR, but it does from closing the swap deal.

Different methods can be used to provide a proper forecast of where the market is headed and what the future SOFR will be. The application of classical linear autoregressive methods for analyzing financial time series are known and already thoroughly researched in literature so don't need any further research. Therefore, it is useful for the Rabobank to explore the opportunities of applying neural network models to SOFR for new insights. Neural networks are a special form of artificial intelligence that uses interconnected nodes that work like the neurons in a human brain. Neural networks can learn and model relationships between input and output that are complex and nonlinear to reveal hidden patterns and predictions. Neural networks are suited for chaotic data and predict rare events which is especially the case in SOFR data. Neural networks are a modern-day technique that has gained attention because the required computing power and open-source software libraries like Keras and Tensorflow have become widely available, that is why the Rabobank is eager to exploit this possibility and assess the opportunities of applying neural networks in the future.

1.2 Problem statement

The main goal of this research is to study the forecast performance of neural network models on SOFR. The smallest interval on which SOFR data is available is daily data thus forecasting can be done with a step size of one day at a time. A neural network can make a forecast of SOFR based on prior observations of SOFR. For a more accurate forecast of SOFR, exogenous variables representing the state of the financial market are added to the model. A neural network that suits this type of forecasting needs to be chosen and evaluated. The performance of the neural network is put into perspective by comparing the performance of the neural network to a baseline model. A baseline model which is commonly used is an autoregressive model. Autoregressive models are trained to pick

up patterns in data in a more linear manner where neural networks are able to detect more complex patterns.

The main question of this research is formulated as follows:

"What is the forecast performance of neural network models on SOFR, and how does this compare to an autoregressive model?"

To determine whether a neural network outperforms an autoregressive model, we need to answer a number of sub-questions which will be discussed in section 1.3.

1.3 Research questions and methodology

To answer the main research question, we need to answer the following sub-questions.

- *Research question 1: What factors influence the market that SOFR is based on and can be used as exogenous variables to forecast SOFR?*

First, the transition from LIBOR to SOFR is discussed to get a sense of what SOFR is replacing. The characteristics and behavior of SOFR and LIBOR will be analyzed to identify the remarkable differences. Apparent patterns in the SOFR data are highlighted and the spikes in SOFR are analyzed. After that, the repo market and influence of the Federal Reserve's policy on the repo market underlying SOFR is discussed. Based on the information given on the financial markets some variables can be determined which have a relation with SOFR or the underlying repo market of SOFR and can be used to forecast SOFR. The variables are defined and a dataset for the neural network is created.

- *Research question 2: Which neural network model is best suited for financial time series forecasting?*

To find out what neural network technique is best suited for forecasting SOFR we will perform a literature review. Since there is no specific literature on using neural networks for forecasting SOFR, the research question is generalized to financial time series forecasting. The best-suited variant of a neural network is the neural network with the lowest forecasting error. In this literature review, different neural networks will be discussed and their advantages and disadvantages for forecasting financial time series will be presented. Eventually, a neural network model will be picked to use in this research based on the findings of the literature review. For reference, autoregressive integrated moving average model with exogenous variables is picked as baseline model. This baseline model is used to compare the results of the neural network.

- *Research question 3: How do we apply the chosen neural network models to SOFR?*

Once we know what is important when modeling SOFR it is required to specify how the chosen neural network model is applied to SOFR. The neural network model is discussed in depth to make a neural network less of a black box method. The configurations of input, hidden and output nodes are specified and linked to the data. The specific hyperparameters to optimize the chosen neural network are discussed. Furthermore, the application of the autoregressive model that is used as a baseline model is discussed to be able to compare the neural network performance properly.

- *Research question 4: How do we optimize the hyperparameters of the neural network model to get the best performance of the neural network on the defined experiments?*

To get the best performance out of the neural network model we need to optimize the hyperparameters of the model, to do this an experimental setup needs to be defined. To optimize the hyperparameters an optimization algorithm is used. A number of trials are run with different values for the hyperparameters to find the best combination of hyperparameters possible. To do this, a specific range is defined for every hyperparameter in which the algorithm may find the optimal value. Furthermore, the test setting for experiments needs to be defined namely the division between train and test data, the forecast horizon and the number of trials executed.

- *Research question 5: What is the performance of the neural network model on forecasting SOFR?*

The hyperparameters are optimized and the results of the optimal configuration of hyperparameters is defined. The optimal configuration of hyperparameters is used to make a forecast of SOFR for the designated forecast horizon. The results of the model are compared to the baseline model for the different experiments.

- *Research question 6: What is the variability in outcomes of the neural network model?*

To guarantee that the configuration of hyperparameters found by the optimization algorithm of the neural network model gives consistent results we need to test the robustness of the model. Testing the robustness and variability of the model consists of two parts. The variability of outcomes of the neural network due to the randomness in the training algorithm is evaluated, the model is run multiple times with the same hyperparameter configuration to ensure consistent results.

1.4 Scope, limitations and assumptions

The scope is limited to testing the feasibility of using neural network models to forecast SOFR. SOFR is the only time series that we aim to forecast by making use of a neural network. The assumption is made that if it is possible to make good forecasts for SOFR making use of a neural network then it is also possible to apply a neural network to other interest rate time series. The aim is to explore where the power of a neural network lies compared to the autoregressive baseline model.

Officially SOFR was first published 2nd of April 2018 thus SOFR data is limited to an extent that officially published data is not available for a long period in the past. Synthetically computed data based on repo transactions can be computed so we added this data for the period from 22nd of August 2014 to the 2nd of April 2018.

The assumption is made that adding extra exogenous variables will provide the neural network with more information about the financial markets thus making the neural network smarter. If the exogenous variable does not contribute the neural network should be smart enough to neglect the useless exogenous variable that is given to the model as input. This assumption is tested in this research in chapter 6 after which insignificant variables are deleted.

The neural network model and baseline model are always trained and compared on the same data, this means the same set of exogenous variables and same forecast horizon. The modeler simply gives the model all information that is available after which it is up to the model to decide what information is relevant or not.

The baseline model is defined as an autoregressive integrated moving average model with exogenous variables (ARIMAX). This model is kept basic to remain a baseline model for comparison, the ARIMAX model does get the same information as the neural network to keep the comparison fair. The aim of the study is not to optimize the performance of the ARIMAX model rather to apply the model and use it as a baseline for comparison.

1.5 Thesis outline

In chapter 2 SOFR is discussed in more depth, SOFR is compared to LIBOR, the repo market is explained, and variables are defined for forecasting SOFR. In chapter 3 a literature review is performed to find the best neural network for forecasting SOFR. In chapter 4 the application of the chosen neural network model on SOFR will be discussed in depth. In chapter 5 the experimental setup for tuning the hyperparameters and testing the model is defined. In chapter 6 the performance of the neural network model is presented and compared to the performance of the baseline model. In chapter 7 the variability and robustness of the neural network are tested. In chapter 8 the conclusion is drawn from the results and the main research question is answered. In chapter 8 also the results are discussed, and recommendations are done for further research.

2 Financial markets

In this chapter SOFR is discussed in more depth and the first research question *"What factors influence the market that SOFR is based on and can be used as exogenous variables to forecast SOFR?"* is answered. In section 2.1 the transition from LIBOR to SOFR is explained as a background story on the creation of SOFR. In section 2.2 the remarkable differences in characteristics and behavior between SOFR and LIBOR are analyzed and the specific spike pattern of SOFR is highlighted. In section 2.3 the repo market and repurchase agreements are discussed. In section 2.4 the influence of the Federal Reserve's monetary policy on the repo market underlying SOFR is discussed. In section 2.5 the spikes in SOFR are discussed. Based on the information given on the financial markets a number of variables are determined in section 2.6 which have a relation to SOFR or the underlying repo market of SOFR and which can be used to forecast SOFR. The variables are defined and a dataset for the neural network is being created in section 2.7.

2.1 Transition from LIBOR to SOFR

The London Interbank Offered Rate (LIBOR) and Secured Overnight Financing Rate (SOFR) are both interest rates at which financial institutions lend money to each other. LIBOR can be seen as a benchmark for interest rates at which banks lend money to each other in the international interbank market for short-term loans ranging from overnight to 12 months. SOFR is an overnight interest rate at which financial institutions lend money to each secured with collateral. This means that the collateral in the form of treasury securities can be sold when the borrower is not able to pay back the loan. This type of loan collateralized by treasury securities is called a repurchase agreement and will be explained in depth in section 2.3.

LIBOR is a series of interest rates intended to reflect banks' average costs of short-term, wholesale unsecured borrowing. Each day a panel of banks is consulted on what their interest rates are to borrow funds from other banks in different maturities and currencies. The simple average rate of the middle 50% is then published as LIBOR for the specified currency and tenor. As turned out in the past, this way of computing LIBOR is easily manipulated by the banks in the panel. Also, the market for unsecured wholesale interbank borrowing turned out to be insufficiently active causing the volume of transaction underlying LIBOR to decline considerably. These two reasons caused the Financial Conduct Authority (FCA), the authority regulating LIBOR, to decide that LIBOR is no longer viable and needs replacement. That is why LIBOR will no longer be officially published after December 31, 2021. Banks are advised that new contracts issued should either utilize a new reference rate other than LIBOR or have robust fallback languages which clearly defines an alternative reference rate once LIBOR becomes unavailable (Alternative Reference Rates Committee, 2020).

The Alternative Reference Rates Committee (ARRC) was established in December 2014 by the Federal Reserve Board to identify an alternative reference rate for LIBOR. The ARRC chose SOFR as a replacement of USD LIBOR for the US market (Alternative Reference Rates Committee, 2020). The ARRC considered different term (un)secured rates, overnight (un)secured rates and treasury bill and bond rates and chose SOFR to be best suited. SOFR was chosen because it has some superior characteristics compared to LIBOR and other alternative reference rates which are:

- SOFR is derived from the U.S. Dollar Treasury repo market which is an active market with a diverse set of borrowers and lenders (explained in section 2.3);
- SOFR covers multiple market segment which results in voluminous transaction volumes making it very difficult to manipulate or influence;
- SOFR is determined in a transparent and direct manner based on observable transactions, in contrast to LIBOR which is based on estimates of banks;
- SOFR is produced in compliance with international best practices produced by the International Organization of Securities Commissions (IOSCO);

- SOFR is publicly published on a daily basis by the Federal Reserve Bank of New York.

By transitioning from LIBOR to the new benchmark SOFR some challenges come up due to the inherent differences between the two rates. LIBOR is a forward-looking rate and quoted for a certain maturity, whereas SOFR is a backward-looking overnight rate which is quoted after expiration of the period. Due to LIBOR being a forward-looking rate it has a built-in risk premium component for liquidity and credit risk assumed by the banks. SOFR does not have a risk premium component since SOFR is based on repo transactions which are collateralized by treasury, the treasury can be sold in the treasury market if the counterparty is not able to pay back the loan or defaults. LIBOR is based upon the interbank unsecured funding market measuring the average rate banks can obtain funds. SOFR is based on overnight transactions secured by US treasury securities as collateral. As a result, SOFR is risk-free and secured which generally turns out in a lower rate than LIBOR (Duffy, Ridley, Feng, & Patel, 2019).

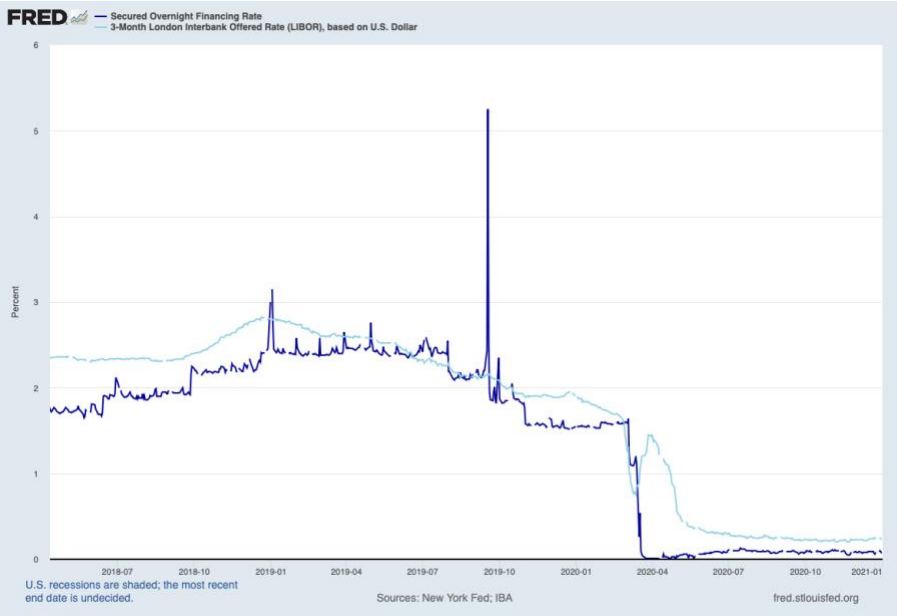


Figure 1:SOFR vs LIBOR(https://fred.stlouisfed.org)

The New York Fed publishes two extensions to the overnight SOFR namely an average SOFR and SOFR index (Federal Reserve Bank of New York, 2021a). These extensions are published to make it easier to reference SOFR in contracts which referenced LIBOR before. The SOFR averages are compounded averages of SOFR a rolling period of 30-, 90- and 180-calender days. The SOFR index measures the cumulative impact of compounding the SOFR on a unit investment over time with an initial value set to 1.000 on April 2, 2018.

For the actual reference of SOFR in future contract different options are given by the ARRC. This can be done through a forward-looking term rate, SOFR compound in advance or SOFR compounded in arrears. These options are shown in Figure 2 and can be broadly summarized as follows:

- A forward-looking rate could be extrapolated from a combination of SOFR futures and overnight index swaps to have a rate which matches the idea of LIBOR.
- SOFR compounded in advance would compound for a given interest period based upon prior observations of SOFR the same length of period as the interest period.
- SOFR compounded in arrears reflect the exact interest rate for the relevant period however this is not known until the interest period is over.

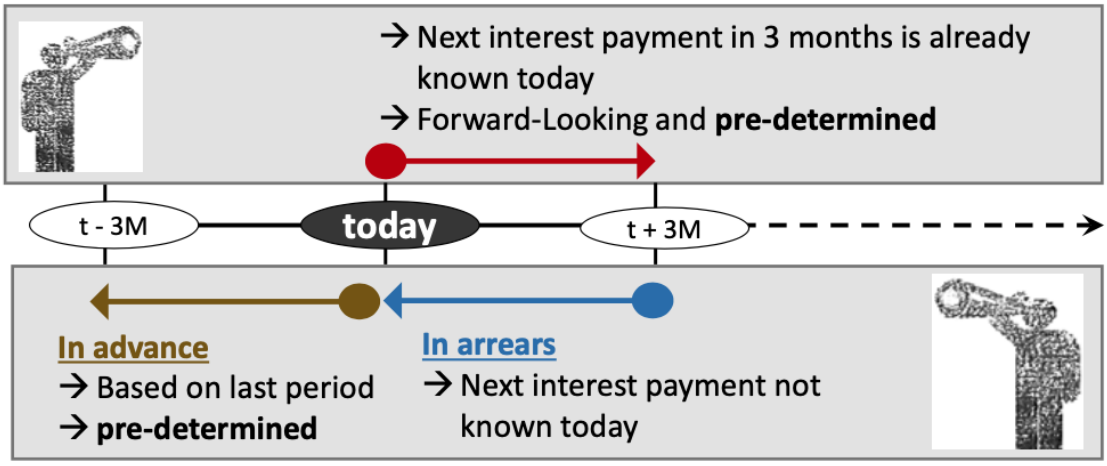


Figure 2: Possible applications of SOFR (Guggenheim & Schrimpf, 2020)

When SOFR is compound in arrears the actual interest rate to be paid is not known until the end of the period. This might give problems to the borrower not knowing the total amount that has to be paid upfront. In order to give borrowers more time to arrange payment different mechanism can be used of which a figure can be found in Appendix 1 and are listed below:

- Simply delaying payment is possible so the payment is due later. This gives the borrower more time to pay however this is not desirable for the lender.
- A 'lag' or 'lookback' mechanism can be applied which uses SOFR of a few days earlier than the interest period in order to end the period a few days earlier so the borrower can arrange payment.
- A 'lockout' mechanism locks out the last few days of an interest periods and repeats the same SOFR rate for these last few days so the exact amount to be paid is known some days upfront.

In all the above-mentioned options the interest rate is compounded so the time value of money is considered. Another option to simplify computations is to use the simple daily SOFR which simply accumulated the daily rates over the interest period without compounding which does not take into account the 'time value' of money which might become apparent when interest rates rise (Duffy, Ridley, Feng, & Patel, 2019).

2.2 SOFR vs Libor

The Secured Overnight Financing Rate (SOFR) was chosen by the Alternative Reference Rates Committee (ARRC) as the replacement of the US dollar LIBOR. There are some notable differences between SOFR and LIBOR (Duffy, Ridley, Feng, & Patel, 2019):

- LIBOR is a forward-looking rate quoted for a certain period of time in the future ranging from overnight to 12 months. SOFR however is a backward-looking overnight rate, quoted the morning after which it relates to;
- LIBOR has a built-in credit risk component. This means LIBOR takes into account liquidity risk and credit risk for the specific term and bank. SOFR is a risk-free rate, so it does not contain any risk premium in its quotation;
- LIBOR is an interbank lending rate which is unsecured. So, no collateral is used in the agreement. LIBOR is measured by the average rate banks are charged in the unsecured funding market. SOFR on the other hand is based on secured overnight transactions with US treasury securities as collateral. This represents the funding cost of the actual transactions in the repo market not only containing banks.

Figure 3 shows a comparison of SOFR, 3-month US dollar Libor and the Effective Federal Funds Rate. As can be seen in Figure 3, SOFR shows more volatile behavior compared to LIBOR this is due to SOFR

being an overnight transaction-based rate fluctuating daily, whfere for LIBOR a longer term generally means that it adapts a little slower to structural changes in the market. However, a longer term does also mean that generally the rate is higher because a higher risk premium is assumed because of the unsecured nature of LIBOR. SOFR being a secured rate we see that SOFR generally produces a lower rate than LIBOR. However, on specific dates spikes occur when SOFR surpasses LIBOR.

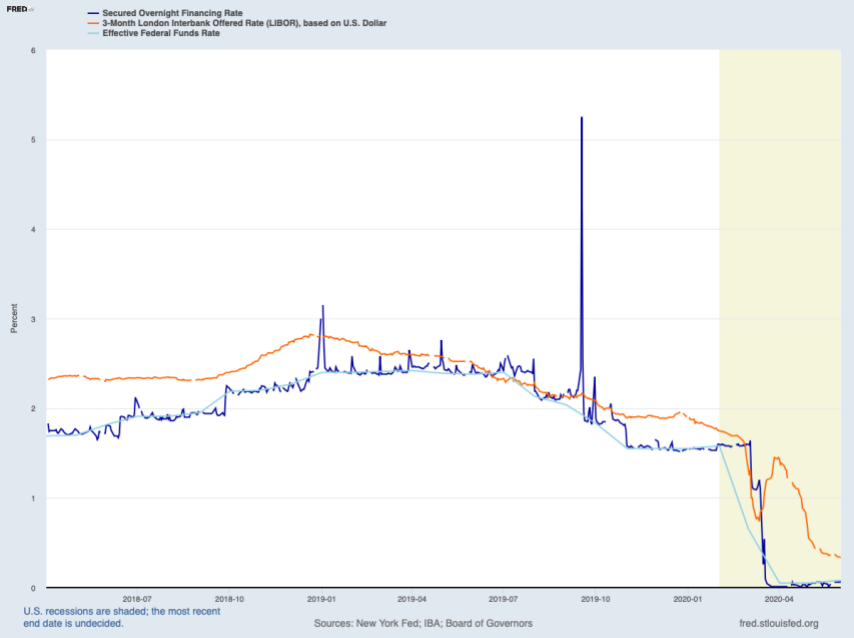


Figure 3: SOFR (blue), 3M USD LIBOR (orange), EFFR (light blue) (https://fred.stlouisfed.org)

2.3 Repo Market

SOFR reflects the overnight interest rate referenced by financial institutions in repurchase agreements. The repo market consists of all repurchase agreement transactions having U.S. treasuries as collateral. In a repurchase agreement, repo, it is stated that one party sells securities to a counterparty and simultaneously agrees to repurchase the same securities from the counterparty at an agreed future date, at maturity, at a repurchase price equal to the original price plus a return on the use of the proceeds during the term of the repo (Alternative Reference Rates Committee, 2020). Simplified a repo is a short-term loan secured with U.S. treasuries as collateral. In Figure 4 a schematic overview is shown of a repurchase agreement. The trade terms are a loan of 1 billion dollars, secured by U.S. treasuries as collateral, with an interest rate of 10 basis points and a margin of 2 percent with overnight maturity. The collateral provider or borrower sells 1.02 billion dollar in U.S. treasuries to a cash investor or lender in exchange for 1 billion dollar in cash at the issue date (date t). At maturity (date t+1) the lender transfers the exact amount of U.S. treasuries back to the borrower and receives its 1 billion dollar in cash back plus an interest premium of 2777.78 dollar which is 10 basis points interest on 1 billion dollar for 1 day.

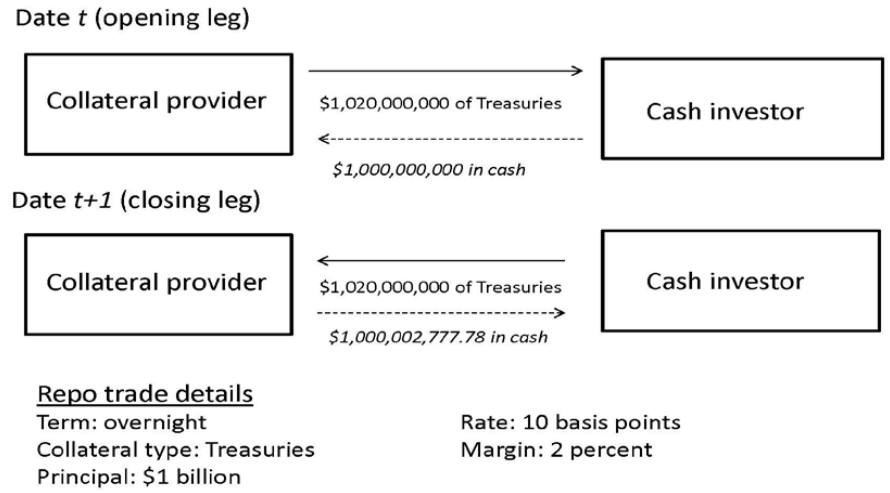


Figure 4: Schematic of a repurchase agreement (Agueci, et al., 2014)

A regular repurchase agreement is initialized by the lender providing the collateral because he is short of cash and needs the cash overnight. A reverse repurchase agreement on the other hand is set up by the borrower or cash investor which is in excess of money and wants return on its excess of cash.

Using the repo structure to "secure" a loan gives the lender the option to sell the collateral if the borrower fails to repay the loan. Because of the collateral, lenders are more willing to make "secured" loans and charging less premium. This makes repos an attractive instrument for funding for different market participants in the Treasury repo market including asset managers, banks, broker-dealers, corporate treasurers, insurance companies, money market funds, pension funds, and securities lending agents. The actual division of members in the repo market is shown in Figure 5, GSIBs are global systemically important banks determined by the Financial Stability Board (FSB). GSIBs are the bigger banks which tend to be too big to fail and are required to have higher capital buffers.

The daily transaction volume in the repo market is roughly $1 trillion which is a factor 2000 more than the average $500 million of transaction volume on which the USD LIBOR is based. This is necessary for SOFR because SOFR is computed purely based on the transactions in the repo market rather than LIBOR which is based on the reported rates of the panel of banks. So, a lower transaction volume for referencing LIBOR has less of an impact on LIBOR since it is based on a broad view of the markets which are also considered by the reporting the banks. For transitioning from USD LIBOR to SOFR the repo market should stay liquid since $200 trillion of financial contracts are daily referencing USD LIBOR which in the future should transfer to referencing SOFR.

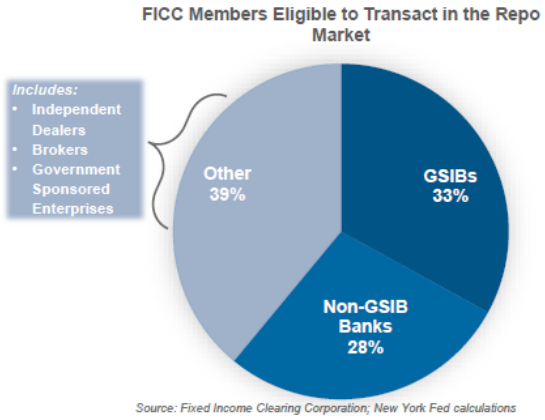


Figure 5: Structure of Repo Market (New York Fed)

2.4 Fed monetary policy

The New York Federal Reserve Bank is using the repo market to conduct its monetary policy. By buying and selling securities in the repo market the New York Fed injects reserves or drains reserves into or from the system.

Prior to the global financial crisis banks wanted to hold only the minimum amount of reserves needed required through legislation and traded their excess reserves in the federal funds market by borrowing and lending. This is the market where banks lend each other money without collateral and interferences of a clearing party like the Fed. However, the Fed could steer the interest rate in this market by adding or draining reserves whenever it wanted to (Cheng & Wessel, 2020).

After the global financial crisis, the Fed uses Quantitative Easing (QE) to stimulate the economy. This meant that there was an overflow of liquidity available for banks and there was less interbank lending and borrowing taking place. This meant that the Fed changed its measures to influence the short-term interest rates these were interest on excess reserves (IOER) and overnight reverse repos (ONRRP), which are both ways for the Federal Reserve to steer the interest rates within the target interest range set by the Federal Open Market Committee (FOMC). The FOMC is the Federal Reserve's monetary policymaking body. The FOMC is responsible for stable prices and economic growth and make policy on this. Eight times a year the FOMC comes together for a scheduled meeting in which the monetary policy is discussed, and the target interest rate is set. This target interest rate is called the target federal funds rate and the actual median of overnight federal funds transactions is called the effective federal funds rate (EFFR). Under normal circumstances, the IOER was assumed to be the lower boundary of the target federal funds rate (Federal Reserve Bank of New York, 2019). However, institutions not eligible to receive the IOER are willing to lend funds at rates below the IOER to institutions which are eligible to IOER. In a market with an overflow of liquidity it turns out that the Federal Reserve acts as the borrower and not the lender by paying the IOER (Gellert & Schlögl, 2019).

In recent years the Fed has continued to buy Treasury securities but not as a form of QE for economic growth but to inject liquidity into the banking system and market stability. By executing daily and long-term repo operations the fed it tries to stabilize the market on the short and long term. This extra involvement of the Fed in the repo market was also needed because of a growing budget deficit for which extra debt was issued through the supply of new Treasuries that had to be absorbed by the repo market.

Since the Covid 19 outbreak the interference of the federal reserve in the repo market has expanded even more. It started with some instant short term repo operations and is now offering a virtually unlimited number of longer-term repos on a weekly basis. This means that the repo market is flooded with liquidity and the amount of money offered is too much for the market to handle. This in turn affects SOFR which is kept 'artificially' low and stable through this monetary policy.

2.5 Spikes in SOFR

Spikes in SOFR have been apparent since it was first published. The spikes in SOFR always happen at the last day of the month. Another moment when spikes occur are near the middle of the month however these are not as big as the end of the month spikes. The end of the month spikes are present due to shortages in liquidity in the repo market caused by a change in the demand and supply of cash and treasuries. For example, if at one day the Fed issues a lot of new treasuries which enter the market at the same day that the corporate tax payments are due this means that demand for cash rises while also the supply of treasuries for repos rises which both have an upward influence on SOFR thus this will create a spike in the SOFR rate. This is an example which especially happens near the middle and the end of the month.

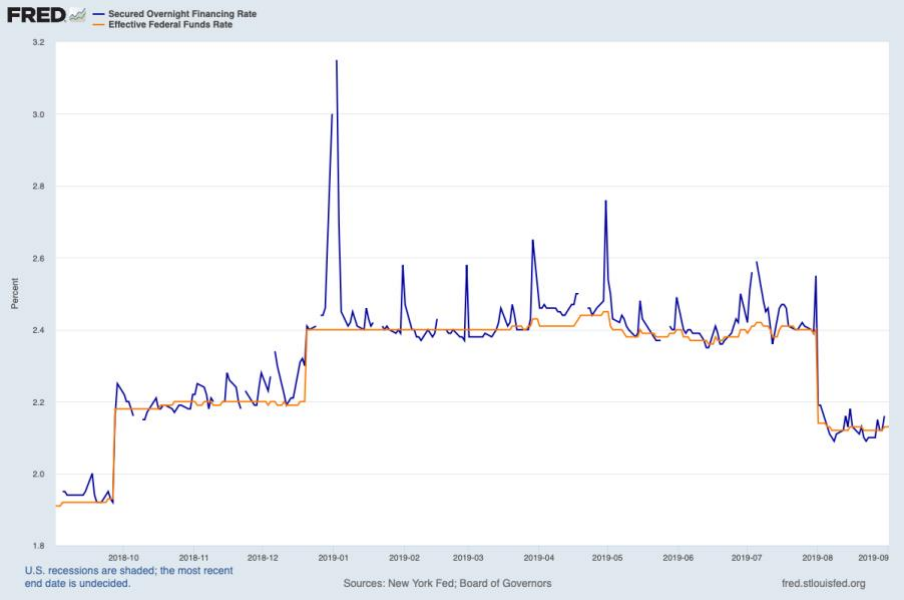


Figure 6: Spikes in SOFR(https://fred.stlouisfed.org)

Since October 15, 2019, the Fed started to purchase treasury bills more actively and do repurchase operations. This smoothens out the end month spikes because of the 'ample' money available in the market regulated by the Fed (Federal Reserve Bank of New York, 2019).

2.6 Variables for forecasting SOFR

In the following section we define a number of exogenous variables which can be used as predictors to forecast SOFR. All variables are published on a daily basis and will be used in the neural network model. The following variables which we expect to be good predictors of SOFR are discussed in the following sections: the Effective Federal Funds Rate(2.6.1), volumes of the Repurchase and Reverse Repurchase Agreements by the New York Fed(2.6.2), the Chicago Board Options Exchange Volatility Index(2.6.3) and the reserve balance of the federal reserve(2.6.4). Two synthetically computed variables will be defined to indicate for the model when it is the end of the month (2.6.5) and at which date the FOMC has meetings (2.6.6).

2.6.1 Effective Federal Funds Rate

Every financial institution holds an account at the Federal Reserve to facilitate regulatory requirements in the form of liquid reserves. Transactions between these accounts take place as institutions borrow and lend overnight reserves to each other to manage their reserve balances and operational cash flows. A volume weighted median of this rate is calculated based on all transactions between Federal Reserve accounts and published every morning as the Effective Federal Funds Rate (EFFR). The EFFR has an impact on very short-term interest rates and is an important gauge of the Federal Open Market Committee (FOMC) for their monetary policy which is determined at eight scheduled meetings per year. Through open market operations the Fed aims to keep the EFFR within the predetermined target range. Another instrument for monetary policy is the Interest on Excess Reserves Rate (IOER). Institutions with an account at the Federal Reserve are given the opportunity to deposit their excess reserves in exchange for the IOER. This makes the IOER and the Fed Funds target range instruments that directly represent the monetary policy of the Federal Reserve without any information of the underlying market. While the Effective Fed Funds Rate is a representation of the monetary policy of the Federal Reserve while also being a representation of the market (Gellert & Schlögl, 2019).

The Effective Federal Funds Rate can be described as a process of jumps with known jump times. The time at which a jump occurs is known because a jump happens when the Federal Reserve announces

a change in monetary policy in their FOMC meetings. Since the EFFR is published daily, the change is reflected directly in the published rate. This makes the EFFR a good predictor for SOFR. If the EFFR and SOFR are compared, SOFR follows the same stepwise function with jumps at known times.

2.6.2 Repo and reverse repo operations

The New York Fed's Open Market Trading Desk is authorized by the FOMC to conduct repo and reverse repo operations in the tri-party repo market. In a repo transaction treasury, agency debt or agency mortgage-backed securities (MBS) are purchased from a counterparty subject to an agreement to resell the same securities at a later date. In an open market repo transaction, the federal reserve functions as the cash investor from Figure 4 and lends money to dealers in the repo market. In doing this the Federal Reserve temporarily increases the quantity of reserve balances in the banking system thus increases liquidity in the repo market. In a reverse repo transaction, the federal reserve acts as the collateral provider of Figure 4. Securities are sold by the fed to a counterparty subject to an agreement to repurchase the securities at a later date at a higher repurchase price. This results in a temporary reduction in the quantity of reserve balances in the banking system resulting in less liquidity in the repo market.

By executing repo and reverse repo operations the Federal Reserve to make sure there is enough supply and demand of 'cash' in the repo market at any given moment to fuel the short-term funding markets with counter parties stabilizing the interest paid for funding and thus SOFR (Federal Reserve Bank of New York, 2021b).

Repurchase agreement are only conducted with primary dealers where reverse repurchase agreements are conducted with primary dealers but also banks, government-sponsored enterprises, and money market funds. The reason why reverse repo operations are open to a wide range of financial firms is to set a lower boundary for short-term interest rates as not all financial institutions have access to deposit money at the Federal Reserve in exchange for the IOER.

Regular repo operations are primarily conducted when there is a sudden need of cash in the repo market which might spike the short-term interest rates. This phenomenon is also called a 'cash crunch', during such a typical event there is an immediate loss of liquidity in the market most of the times due to some event. During a 'cash crunch' all financial institutions are on the hunt for 'cash' while on the same time holding on to their 'cash' or reserves. This results in spiking short-term interest rates and a failing market. As a response to a 'cash crunch' the FOMC then issues repo operations to calm the market with an ample amount of liquidity.

In order to stabilize the market in the longer-term credit is issued by the Primary Market Corporate Credit Facility (PMCCF), stabilized March 23, 2020, to provide extra liquidity directly to the economy by bond and loan issuances for longer terms (Board of Governors of the Federal Reserve System, 2021).

In Figure 7 we see that there were a lot of reverse repo operations during the period before 2018 where the Fed Funds target range was low and had to remain low by issuing reverse repo operations. We see that the reverse repo operations became less as the Fed Funds target range got higher. Furthermore, we see that in September 2019 the regular repo operations start. This is a reaction to the major SOFR surge event of September 17, 2019. Since this date the Federal Reserves has tried to suppress the end of the month spikes through repo operations. In the beginning of the covid pandemic there was a high need of both repo and reverse repo operations after which the PMCCF was established, and the market was flooded with ample liquidity and a Fed Funds target of essentially 0%.

The volume of repo and reverse repo operations are helpful in explaining the behavior of SOFR and the upward or downward pressure that is applied by the federal reserve to SOFR. A change in the volume of repo and reverse repo operations might give an indication when and why spikes are occurring at certain times.

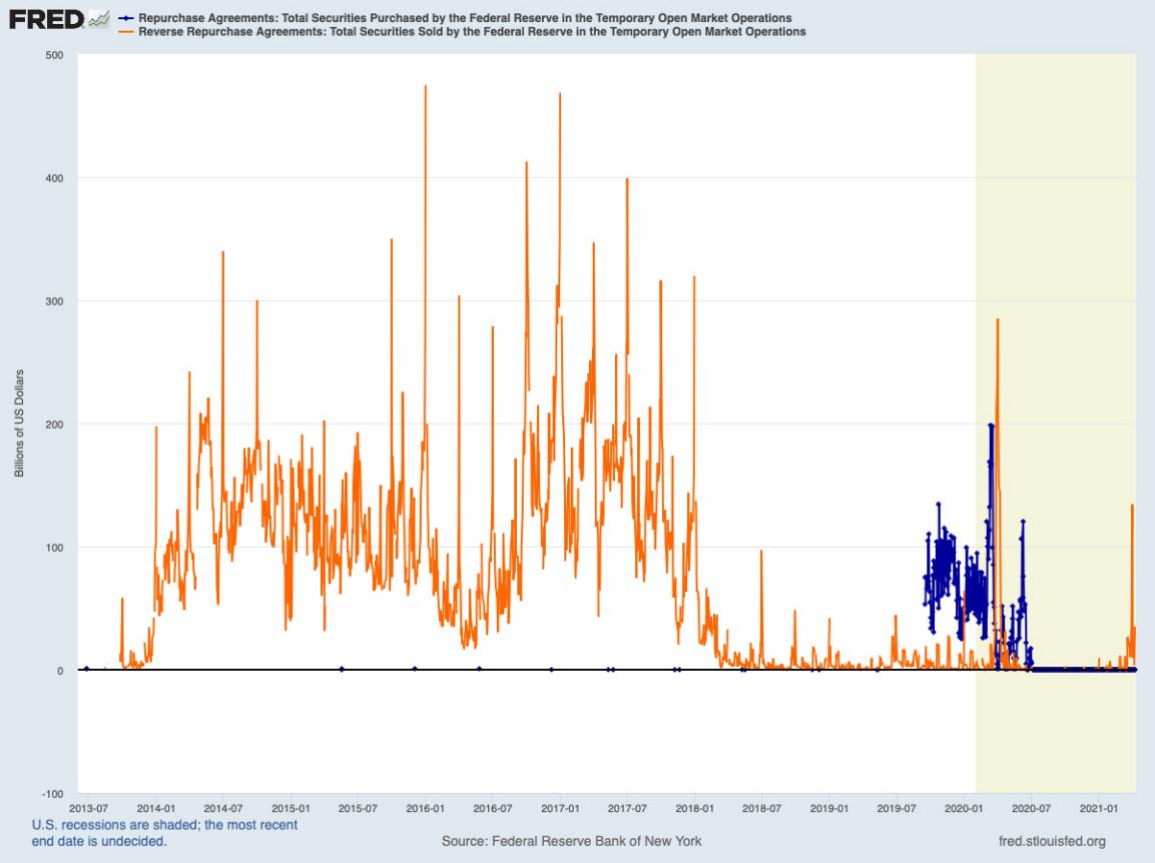


Figure 7: Repo and reverse repo operations

2.6.3 CBOE Volatility index

The Chicago Board Options Exchange Volatility Index (VIX) is a real-time financial market estimate of expected volatility of the S&P 500 Index (Moran & Liu, 2020). The VIX index is derived of the prices of S&P500 index options with near-term expirations dates thus giving an expectation of the near-term price changes generating a 30-day forward projection of volatility. The VIX index is considered as the best way to gauge U.S. market sentiment and the degree of fear among market players hence the alternative name of 'fear index' for the VIX index. A high VIX index correlates with big losses in equity where a low and stable VIX index correlates with stable growth in equity. A high VIX index is almost always a result of an event in the real world which affects price expectations. This makes the VIX index a good indicator of financial turmoil influencing the repo market underlying SOFR. A high 'fear' in the market results in institutions scrambling for cash while holding on to their money. This results in a 'cash crunch' and a dried-up repo market with spiking rates.

2.6.4 Reserve balance

The federal reserve's balance sheet is a reflection of the money supply from the fed within the economy. The federal reserve can increase or decrease the amount of assets and liabilities on its balance sheet to execute the monetary policy set out by the FOMC. The liabilities are the currency in circulation and money in reserve accounts of member banks. The assets on the fed's balance sheet consist of treasury securities, mortgage-backed securities, and loans. Loans are extended through repo operations and through a lending facility which charges the federal discount rate. Through open market operations in the U.S treasury securities market and the repo market the fed regulates the money supply in the U.S. economy (Federal Reserve, 2020). Each week on Thursday the Fed publishes its H.4.1 report which is a statement reporting the situation of the balance sheet of the Federal Reserve system.

The total assets published on the Fed's balance sheet are a reflection of the liquidity in the financial system. Because the Fed will supply liquidity to the market by increasing the size of their balance sheet if there is a liquidity shortage. Conversely the Fed also minimizes excessive surpluses of liquidity through operations by shrinking down the balance sheet. As liquidity shortages result in spikes in short term rates this raises SOFR. So, if the liquidity supplied by the Fed to the financial is high this means that spikes are less likely to occur. The reserve balance can be seen as a sort of stock variable which is the oil of the financial system. If the liquidity is ample then the financial system will run smooth but if liquidity provided by the fed drops then spikes are more likely to occur due to a shortage in liquidity. The level of liquidity is influenced by repo operations in the market which are only needed when liquidity is low.

2.6.5 Date dummy

A special characteristic of SOFR is the end of the month spikes. The end of the month spikes are especially pronounced at the end of a quarter and end of the year. These spikes are not caused by changes in monetary policy of the Federal Reserve but by fluctuations in supply and demand in the repo market underlying SOFR. This fluctuation in supply and demand related to the dealers' balance sheet exposures at month end for regulatory purposes. By examining the spikes it can be concluded that after the spike the SOFR rate returns to the same level as prior to the spike. Figure 8 shows the size of the spikes at the end of the month(EOM), end of the quarter(EOQ) and the end of the year(EOY) for a period between August 2014 and March 2019.

These spikes cannot be explained through the EFFR that is why another variable is needed. A variable is needed which helps the model to indicate when it is the end of the month. The suited modelling technique to do this is to define a dummy variable. A dummy variable is a binary variable of 0 and 1 which is 1 in the case the date is the last trading day of the month and 0 for all other dates. By using this variable, the neural network model will know when it is the end of the month and when to expect a spike. The model needs to be helped by this dummy variable since not every month consists of the same number of trading days and not every last trading day is also the last day of the month.

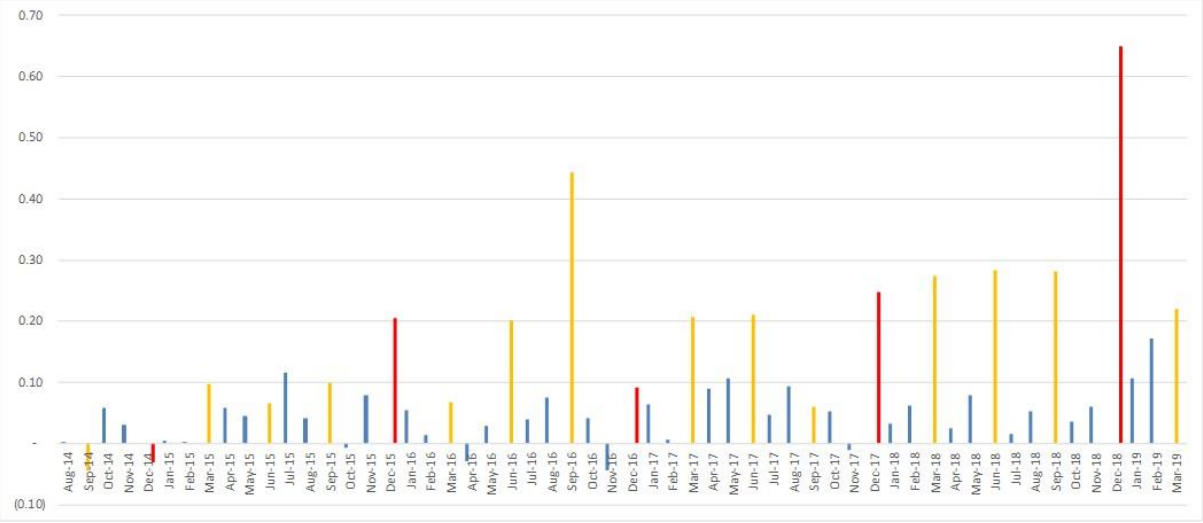


Figure 8: EOM SOFR fixing minus month's average, EOM marked as blue, EOQ marked as yellow, EOY marked as red (Gellert & Schlögl, 2019)

2.6.6 FOMC meeting dates dummy

The Federal Open Market Committee (FOMC) holds eight scheduled meetings per year. During this the monetary policy is discussed and adjusted where necessary. A press conference is being held to present the findings. In response to these meetings the repo market responds to probable changes in monetary policy resulting in a rise or fall of SOFR. This is a result of changes in the Fed Funds Target Range and IOER but also due to anticipations of institutions based on their beliefs of the outcomes of

the meeting. If the Effective Federal Funds Rate and SOFR are analyzed for the FOMC meeting dates, we see that both rates move in the same direction if there is a change in monetary policy however if there is no change in monetary policy than SOFR still responds to some turmoil in the repo market. So, in some cases the Effective Federal Funds Rate alone is not a strong enough indicator of the sentiment in the market that is why another date dummy variable can be added on the FOMC meeting dates.

2.7 Dataset

All variables from section 2.6 can be combined to create one dataset on which the neural network model can be trained. The limiting factor in the size of the dataset is the availability of SOFR data. The first date that SOFR was officially published was April 2, 2018, which makes about 3 years of data. Of this data about 1 year are dates during the covid pandemic where SOFR has been kept artificially low between 0.0 and 0.10. During this time no spikes in SOFR are being observed, this is something the neural network model can be tested on if it understands the change in market circumstances. To extent the dataset two SOFR datasets can be combined, the Federal Reserve Bank of New York has published an indicative SOFR dataset for the period from August 2014 to March 2018 which can be combined with the actual SOFR publications starting April 2018. An extract of the dataset can be seen in Table 1.

DATE	**EFFR**	**Date Dummy**	**FOMC Dummy**	**Total repo (Billions USD)**	**Total reverse repo (Billions USD)**	**Cboe VIX**	**Reserve balance (Millions USD)**	**SOFR**
27/06/2018	1.91	0.0	0.0	0.0	20.68	17.91	4360000	1.90
28/06/2018	1.91	0.0	0.0	0.0	20.42	16.85	4360000	1.93
29/06/2018	1.91	1.0	0.0	0.0	96.97	16.09	4360000	2.12
02/07/2018	1.91	0.0	0.0	0.0	32.14	15.60	4359543	2.04
03/07/2018	1.91	0.0	0.0	0.0	4.85	16.14	4359543	2.00
05/07/2018	1.91	0.0	0.0	0.0	8.21	14.97	4359543	1.97
06/07/2018	1.91	0.0	0.0	0.0	4.20	13.37	4359543	1.93
09/07/2018	1.91	0.0	0.0	0.0	3.60	12.69	4359543	1.89

Table 1:Extract of dataset

2.8 Conclusion of chapter

In this chapter the first research question *''What factors influence the market that SOFR is based on and can be used as exogenous variables to forecast SOFR?''* is answered. To answer this research question SOFR is compared to LIBOR to know what SOFR is replacing. SOFR replaces LIBOR as the interest rate financial institutions charge each other to lend money to each other which is a good indication for all other interest rates in the system. Furthermore, we find that the Federal Reserve intervenes in the repo market through repo and reverse repo operations to steer the effective federal funds rate and SOFR. Based on the information given the following variables are picked to forecast SOFR: the Effective Federal Funds Rate(2.6.1), an end of the month date dummy(2.6.5), a FOMC date dummy(2.6.6), volumes of the Repurchase and Reverse Repurchase Agreements by the New York Fed(2.6.2), the Chicago Board Options Exchange Volatility Index(2.6.3) and the reserve balance of the federal reserve(2.6.4).

3 Neural network selection

In this chapter we aim to search literature the best neural network model suited for forecasting SOFR. A literature review will be used to answer the following question "Which neural network model is best suited for financial timeseries forecasting?" Since there is no specific literature on using neural networks for forecasting SOFR the research question is generalized to finding neural network models for financial timeseries forecasting. More specifically we are searching for a neural network model which is able to process the exogenous variables defined in section 2.6 to forecast SOFR.

Firstly, different neural network models will be reviewed based on theory. The feed forward neural network, recurrent neural network and long short-term memory nodes of recurrent neural networks are discussed in depth respectively in section 3.1, 3.2, 3.3 and 3.4. After that in section 3.5 literature is consulted to pick the neural network model that is able to forecast financial timeseries with the smallest error according to literature. In section 3.6 the autoregressive integrated moving average model with exogenous variables is explained to compare the performance the neural network model to. In section 3.7 neural network models are compared to autoregressive models according to literature.

The goal of time series forecasting is to estimate future data points based upon analysis of past data points. A traditional approach to forecasting financial time series have been linear statistical models. Linear models however have difficulties providing good predictions when the time series shows signs of noise or non-linearity. As noise and non-linearity are present in financial timeseries a non-linear model could prove superior to classic linear statistical models. Non-linearity implies that relation of past interest rates to future interest rates does not have to be linear. A change in interest rate can have multiple different effects on the future interest rate which cannot be captured by a traditional linear model (Thenmozhi, 2006).

Neural networks have proven to be successful in modelling inputs to outputs in patterns that are not linear and too complex for humans to notify. The unique feature of neural networks is the ability to model non-linear relations with very limited prior information on the process to be modeled. A neural network can adapt according to the information that is given to optimize a certain performance metric as objective.

3.1 Basic perceptron

The simplest form of a neural network is a basic perceptron. A basic perceptron is a neural network consisting of a single input layer and an output node. The model architecture of a basic perceptron can be seen in Figure 9. A basic perceptron does not have a time element. In this situation each training instance is of the form $(\bar{X}, y)$, where each $\bar{X} = [x_1, \dots, x_n]$ contains *n* features and an observed value y. For every instance the goal is to make a prediction $\hat{y}$ as close to the observed value y as possible based upon the *n* feature variables (Aggarwal, 2018).

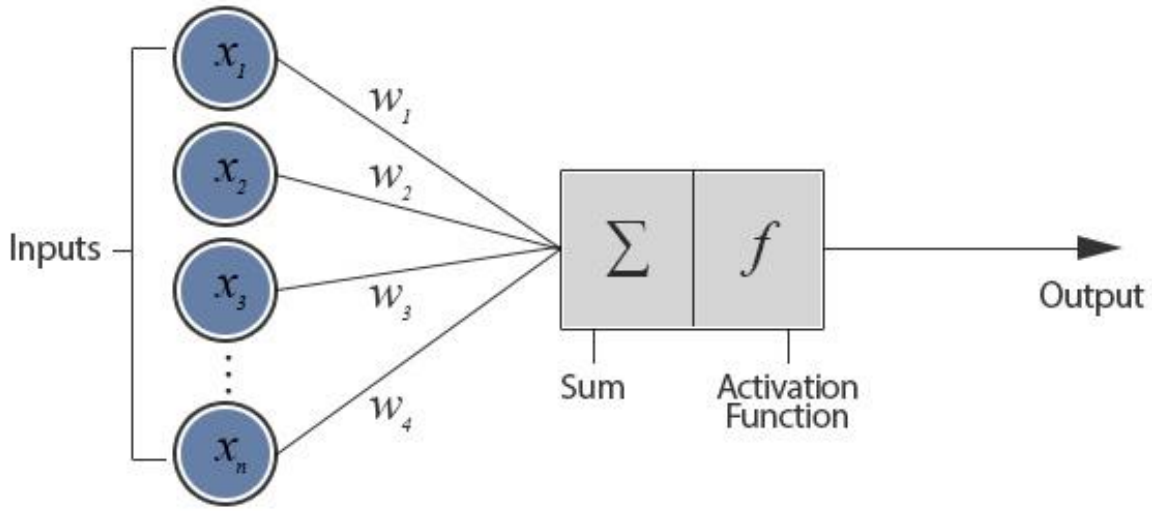


Figure 9: Basic perceptron for the output $\hat{y}$ (Sun, 2017)

The architecture of the basic perceptron of Figure 9 consists of an input layer containing *n* nodes that transmit *n* features $\bar{X} = [x_1, \ldots, x_n]$ with edges of weight $\bar{W} = [w_1, \ldots, w_n]$ through an activation function $\phi()$ and then to an output node. In this basic perceptron the output node is the prediction $\hat{y}$. The prediction $\hat{y}$ can be computed as follows:

$$\hat{y} = \phi(\bar{W} \cdot \bar{X}) = \phi\left(\sum_{j=1}^{n} w_j x_j\right)$$

Equation 1

The goal of the perceptron is to match the prediction of the neural network $\hat{y}$ to the observed value y. The minimization objective function between the prediction and observed value is called the loss function. The ordinary least squares loss function by optimizing the weights $\bar{W}$ for all training instances in a data set $\mathcal{D}$ is expressed as follows:

$$Minimize_{\bar{W}} L = \sum_{(\bar{X},y)\in\mathcal{D}} (y - \hat{y})^2 = \sum_{(\bar{X},y)\in\mathcal{D}} \left(y - \phi(\bar{W} \cdot \bar{X})\right)^2$$

Equation 2

The choice of activation function $\phi()$ is a critical part of a neural network and depends on the nature of the observed value y to be predicted. The most basic activation function $\phi()$ is the linear or identity activation which simply passes on the value: $\phi(v) = v$. If a binary value needs to be predicted, then the sign activation function is suitable since it either outputs -1 or 1. Thus for forecasting a real number the sign activation function is not suitable. When the output value is a real value the activation function should not give a discrete output but give a continuous output. The most used activation functions are the sign, sigmoid and the hyperbolic tangent functions:

$\phi(v) = sign(v)$ (sign activation function)

$\phi(v) = \frac{1}{1+e^{-v}}$ (sigmoid activation function)

$\phi(v) = \frac{e^{2v}-1}{e^{2v}+1}$ (tanh activation function)

3.2 Feed forward neural network

Multilayer neural networks contain multiple computational layers besides the input and output. In the basic perceptron the output layer is the only layer performing a computation by weighting the inputs and applying an activation function to compute the output. In the multilayer neural network intermediate computational layers are added between the input and outputs which are referred to as hidden layers. When there is more than 1 hidden layer in the neural network we speak of a deep neural network. A multilayer neural network like in Figure 10 is called a feed-forward neural network with an input layer containing *m* inputs, one hidden layer containing *l* units and an output layer containing *n* outputs (Aggarwal, 2018).

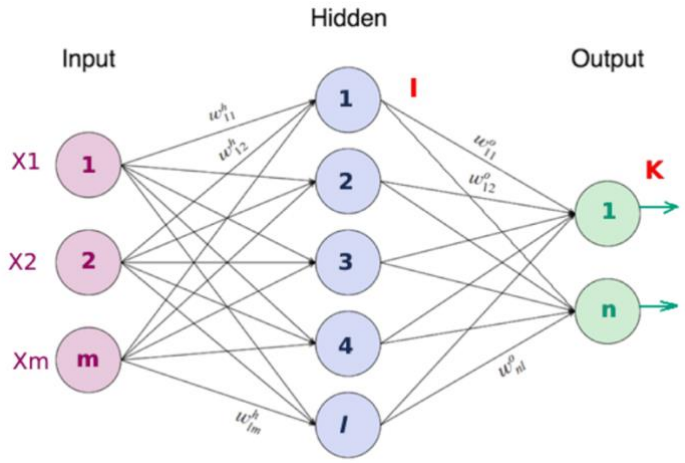


Figure 10: Feed Forward Neural Network (Imam, 2020)

Figure 10 shows a Feed Forward Neural Network containing one hidden layer however it is also possible to have multiple hidden layers before the output is computed. When there is more than one hidden layer the Neural Network is considered 'deep'. The *k* hidden layers of a neural network can be denoted by the column vectors $\bar{h}_1 \ldots \bar{h}_k$ having the dimensionality of $p_1 \ldots p_k$ units in each layer. The weights of the connections between all the layers can be denoted by matrices $W_1 \ldots W_{k+1}$. The first connections between the input layer and the first hidden layer can be contained in matrix W_1 with size $d \times p_1$, where the weights of the following *r*th and *(r+1)*th hidden layer are denoted by a $p_r \times p_{r+1}$ matrix W_r. For the output layer the final matrix $W_{k+1}$is computed with the size $p_k \times o$. Through the above-mentioned matrices the input vector $\bar{x}$ can be transformed into the outputs $\bar{o}$ by the following recursive equations:

$$\bar{h}_1 = \phi(W_1^T \bar{x}) \text{ (Input to hidden layer)}$$

Equation 3

$$\bar{h}_{p+1} = \phi\left(W_{p+1}^T \bar{h}_p\right) \forall p \in \{1 \ldots k-1\} \text{ (Hidden to hidden layer)}$$

Equation 4

$$\bar{o} = \phi\left(W_{k+1}^T \bar{h}_k\right) \text{ (Hidden to output layer)}$$

Equation 5

3.3 Recurrent Neural Networks

In a feed forward neural network all training instances and variables are treated as independent values. However, in a time-series dataset the sequence of values is important since datapoints are closely related to each other. If the single datapoints are treated as independent features, then useful information about the relationship of consecutive values is lost. Two requirements for processing sequences are (i) the ability to receive and process inputs in the same order as they are present in the

data sequence and (ii) the treatment of inputs at each time stamp in a similar manner in relation to previous inputs. The challenge is to construct a neural network with a fixed configuration of parameters but with the ability to process a variable number of inputs (Aggarwal, 2018).

In a recurrent neural network (RNN) the specific position in the sequence, timestamp, is preserved. Instead of a sequence in a single input layer, each layer has a single input corresponding to the timestamp. So, in all the formulas a time-stamp *t* is added to identify what timestamp the value of the input, hidden state or output belongs to. The inputs of a timestamp are allowed to interact with the hidden layers depending on the position in the sequence. For every time stamp the same neural network structure is consulted to ensure similar modeling at each time stamp and this is repeated for every time stamp in the sequence therefore the network is called recurrent. A recurrent neural network is also a feed-forward network with a specific time layering to be able to process a sequence of inputs and produce a sequence of outputs. At each time stamp a single input or multiple inputs can be given corresponding to the time stamp to produce a single or multiple outputs for the same time stamp as the input. In Figure 11 a recurrent neural network is shown with a single hidden layer. For each time the same feed forward neural network is used with the same weight matrices *U*, *V* and *W*. The hidden state vector is computed by combining the input *x* and previous hidden state vector *h* by making use of a $p \times d$ input-hidden matrix U_{xh} and a $p \times p$hidden to hidden matrix W_{hh}. After that the hidden state vector is used to compute the output *o*, a prediction of y_t, by making use of a $d \times p$ hidden-output weight matrix V_{ho}.

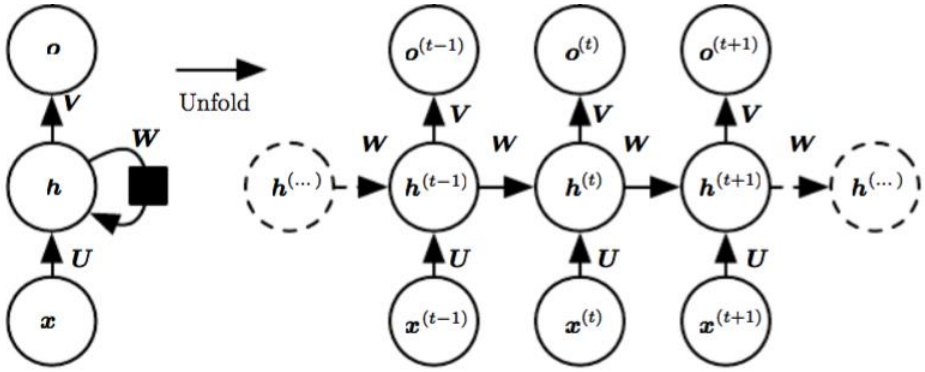


Figure 11:Recurrent Neural Network unfolded (Goodfellow, Bengio, & Courville, 2016)

The recurrent neural network shown in Figure 11 contains of an input vector *x* of *m* input nodes, one hidden layer containing *l* units and an output layer containing *n* outputs. The difference with a feed forward neural network is that every instance has its own time stamp (t). The computation of the recurrent neural network consists of two steps. The first step (Equation 6) is executed for updating the hidden state vector $\bar{h}_t$ based upon the input vector $\bar{x}_t$ and the hidden state vector of the previous time stamp $\bar{h}_{t-1}$. The second step (Equation 7) converts the information of the hidden state vector $\bar{h}_t$ into an output $\bar{o}_t$ which is a prediction of y_t.

$$\bar{h}_t = \phi\left(U_{xh}\bar{x}_t + W_{hh}\bar{h}_{t-1}\right)$$

Equation 6

$$\bar{o}_t = \phi\left(V_{ho}\bar{h}_t\right)$$

Equation 7

Different activation functions can be used in Equation 6 and Equation 7 to the desires of the modeler. The most used activation function for Equation 6 is the hyperbolic tangent function explained before. The activation function for computing the outputs that is generally used is the softmax activation function when the desired output is a probability per output node. The softmax activation function scales all values of the output relative to each other so all outputs sum to 1 and a probability can be

given. The softmax activation function in Equation 8 is applied to the vector $\bar{v} = [v_1, \dots, v_k]$ with k output values.

$$\phi(\bar{v})_i = \frac{\exp(v_i)}{\sum_{j=1}^{k} \exp(v_j)} \quad \forall i \in \{1, \dots, k\}$$

Equation 8

3.3.1 Backpropagation through time

Once the neural network is defined the model parameters, the weight matrices *U, V* and *W*, are trained to minimize the loss *L* of the model. The model parameters of a neural network are trained with backpropagation through time which computes the gradients of the parameters to update the parameters weights resulting in a lower loss function. The backpropagation algorithm consists of two phases: a forward phase and a backward phase. In the forward phase the inputs for a training instance are fed to the neural network and the loss is computed. In the backward phase the gradients of the parameters are computed $\frac{\partial L}{\partial U}, \frac{\partial L}{\partial V}, \frac{\partial L}{\partial W}$. This is done by making use of the chain rule of differential calculus and calculating the gradients back throughout the layers but also back in time through past time stamps back to the input. In depth computation of the gradients will not be discussed in this thesis and can be found in (Aggarwal, 2018, pp. 21-24). Once the gradient of all parameters is known the weights of the parameters can be updated to minimize the loss function and a new iteration of backpropagation is started with a new instance of the training data.

3.3.2 Vanishing and exploding gradients

Increasing the depth of a model or adding the recurrent component to a neural network leads to some practical issues regarding the backpropagation through time. Propagating backwards using the chain rule becomes problematic for networks with a large number of layers or when backpropagating a long period back in time. This can result in updates in earlier layers becoming negligibly small, vanishing gradient, or updates become unproportionally large, exploding gradient (Aggarwal, 2018). One approach to overcome vanishing and exploding gradients is the truncated backpropagation through time which employs a moving window through time on which the model is makes a forward and backward pass. By doing this not the entire sequence is used which disables the neural network to learn long term dependencies in the data. Another approach to this problem is applying a Long Short-Term Memory (LSTM) architecture which employs hidden state nodes which have a 'forget gate' which is able to forget the old state to stop the derivatives from vanishing or exploding. In the next section 3.4 Long Short-Term Memory is further explained.

3.4 Long short-term memory recurrent neural networks

As discussed in section 3.3.2, a recurrent neural network has the problem of vanishing and exploding gradients. The neural network updates get unstable as the weight matrix W is multiplied iteratively which results in a gradient disappearing or blowing up which is both undesirable. This problem is especially present when long sequences are fed to the neural network, and not when short sequences are used. So, to find patterns in long sequences long term memory needs to be added which does not cause vanishing or exploding gradients.

Long Short-Term Memory (LSTM) networks are capable of learning long-term dependencies without the vanishing or exploding gradient problem (Hochreiter & Schmidhuber, 1997). LSTMs are made to remember information for long period of time. A LSTM network changes how the recurrence conditions of the hidden states $\bar{h}_t^k$ of time stamp t and the *k*th layer are passed on. A cell state denoted by $\bar{c}_t^k$ is added to add some sort of long-term memory to the hidden state of the neural network. This cell state is updated by the different 'gates' in the LSTM cell. Three types of gates are present in each LSTM cell to control the cell state:

- Forget gate: outputs a number between 0 and 1 to decide if a feature should be kept or totally ignored for which it respectively returns a 1 or a 0.
- Input gate: decides whether new data is stored in the cell state or not.
- Output gate: decides what part of the information in the cell state and previous hidden sate gets passed out of the cell to the next hidden state. This is based upon the value of the cell state (long term memory) and the newly added data (short term memory).

In Figure 12 a visual representation of a LSTM node is shown. The input of the LSTM node is the input vector x_t or the hidden state from the previous layer $\bar{h}_t^{k-1}$, the hidden state of the previous time stamp $\bar{h}_{t-1}^k$ and the cell state of the previous time stamp $\bar{c}_{t-1}^k$. These inputs are used to compute the outputs which are the hidden state vector of the current time stamp $\bar{h}_t^k$ and the cell state of the current time stamp $\bar{c}_t^k$. The cell state and hidden state are passed on to the next time stamp, to the right in Figure 12, while the hidden state is also used for computation of the next layer in the neural network, going up in Figure 12. To be able to compute the hidden state and cell state following preliminary equations are needed for the forget gate ($\bar{f}$), input gate ($\bar{\imath}$), output gate ($\bar{o}$) and new cell state ($\bar{c}$):

$$\text{Forget gate: } \bar{f} = \sigma\left(W^k \cdot \left[\bar{h}_{t-1}^k, \bar{x}_t\right]^T\right)$$

Equation 9

$$\text{Input gate: } \bar{\imath} = \sigma\left(W^k \cdot \left[\bar{h}_{t-1}^k, \bar{x}_t\right]^T\right)$$

Equation 10

$$\text{Output gate: } \bar{o} = \sigma\left(W^k \cdot \left[\bar{h}_{t-1}^k, \bar{x}_t\right]^T\right)$$

Equation 11

$$\text{New cell state: } \bar{c} = tanh\left(W^k \cdot \left[\bar{h}_{t-1}^k, \bar{x}_t\right]^T\right)$$

Equation 12

Equation 9 to Equation 12 are the computation that are done within the LSTM node to compute the hidden state and cell state. The σ notation in the equations above denotes the sigmoid activation function. Once the preliminary equations are executed inside the LSTM node the hidden state and cell state can be computed through Equation 13 and Equation 14, with the element-wise product of vectors denoted by "$\odot$".

$$\bar{c}_t^k = \bar{f} \odot \bar{c}_{t-1}^k + \bar{\imath} \odot \bar{c} \text{ [Selectively forget and add to long-term memory cell state]}$$

Equation 13

$$\bar{h}_t^k = \bar{o} \odot \tanh\left(\bar{c}_t^k\right) \text{ [Selectively leak long-term memory to hidden state]}$$

Equation 14

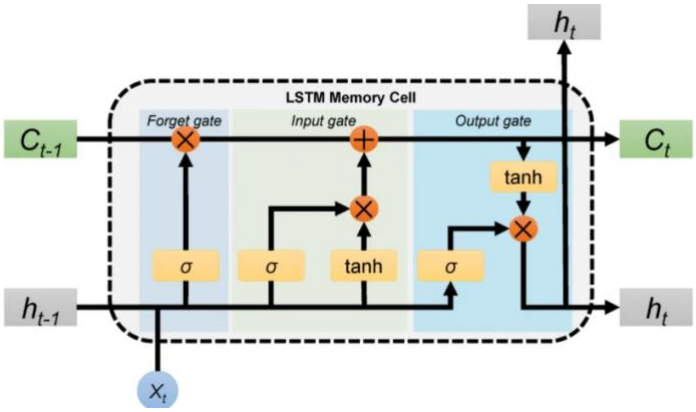


Figure 12: LSTM Cell (Fan, et al., 2020)

3.5 Application to financial timeseries

In section 3.1 to 3.4 the different neural networks that are available were discussed based on theory. To make an educated decision on what neural network to use for forecasting SOFR, literature is consulted to find applications of neural networks to financial timeseries and their performance.

Feed forward neural networks are the simplest and first artificial neural network applied to a univariate financial timeseries containing only one variable (Stock & Watson, 1998). Feed Forward Neural Networks do not show good results for univariate timeseries. For multivariate timeseries Feed Forward Neural Networks might be a better alternative since there is more information available at each timestep.

When comparing the performance of a feed forward neural network to a smooth transition autoregressive model the feed forward model gets outperformed because the feed forward neural network turns out to be an explosive model when applying longer forecasting horizons (Teräsvirta, van Dijk, & Medeiros, 2004)

Feed Forward Neural Networks are the most basic Neural Networks, so therefore in older literature this type of Neural Network is most used. Mainly in older articles, such as (Stock & Watson, 1998), we find that Feed Forward Neural Networks perform quite poorly. This might be because in the past computational power was a lot less than it is today.

In (Verstyuk, 2020) a multivariate time series is modelled using Vector Auto-Regression and Multivariate Recurrent Neural Networks with LSTM nodes. A 2 layered LSTM-RNN has the best performance in forecasting US financial data for example the Fed Funds rate, performing better in terms of forecasting accuracy and interpretability than the benchmark model, Vector Auto-Regression using 3 lags. Especially out of sample performance of a LSTM-RNN is superior to Vector Autoregression (VAR). The Long Short-Term Memory Multivariate Recurrent Neural Network is praised for its capability to discover macroeconomic patterns without any special modeler's intervention, learning causal effects from raw data.

A Long Short-Term Memory Recurrent Neural Network has the following favorable characteristics according to literature:

- Able to discover patterns without modelers intervention. (Bandara, Bergmeir, & Hewamalage, 2020)
- Learns causal effects from raw data. (Verstyuk, 2020)
- Suits the data type of univariate timeseries. (Stock & Watson, 1998)
- Finds the optimal number of lags to hold in its memory. (Verstyuk, 2020)

An extension of LSTM-RNNs is Long Short-Term Memory Echo State Networks. This type of network has the benefit of needing less computational power for optimization because there are less connections between neurons which need to be optimized (Zheng, et al., 2020).

Another useful application might be Long Short-Term Memory Multi-Seasonal Nets, in this application one Neural Network is trained on different timeseries which are related and have the same characteristics (Bandara, Bergmeir, & Hewamalage, 2020). This might be useful for prediction of related financial timeseries.

The previous examples of RNNs all have been 'shallow' in the sense that there is only one hidden layer which is used. Literature suggests that using 'deep' RNNs outperform conventional and LSTM RNNs. In 'deep' RNNs extra computations are done when computing the hidden state or output of the RNN or hidden states are stacked up to create multiple layers of hidden states (Pascanu, Gulcehre, Cho, & Begnio, 2014).

3.6 Autoregressive model

As a baseline model the autoregressive integrated moving average model with exogenous variables (ARIMAX) is chosen. In this section the ARIMAX model is build up step by step to be able to use it to forecast SOFR using exogenous variables.

An autoregressive model regresses a value from a timeseries based on previous values from that same time series (Brooks, 2014). A first-order autoregression, AR(1), uses the value of the previous period to predict the value at the present time. Equation 15 shows the formula of an AR(1) model

$$y_t = \beta_0 + \beta_1 y_{t-1} + \epsilon_t.$$

Equation 15

The first part of the prediction is the β_0 which is a constant factor for each prediction furthermore the value of β_1 decides how heavily we base our prediction on the past value of y. ϵ_t is the error term containing the residual with the assumption of zero mean and constant variance. Applying least squares is optimal when the error term is zero mean. A pth-order autoregression, AR(p), is a multiple linear regression in which the value of the past p timesteps is considered. Equation 16 shows formula of a third-order autoregression model, AR(3).

$$y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \beta_3 y_{t-3} + \epsilon_t.$$

Equation 16

By tuning the betas of the autoregression model we can decide which lag gets most emphasis. To effectively choose the values of the beta's we can use the least squares principle by minimizing the sum of the squared errors. For an AR(1) model we choose a value for β_0 and β_1 that minimizes Equation 17:

$$\sum_{t=1}^{T} \epsilon_t^2 = \sum_{t=1}^{T} (y_t - \beta_0 - \beta_1 y_{t-1})^2$$

Equation 17

An extension of the autoregressive model is the autoregressive integrated moving average model (Brooks, 2014). This combines the moving average model with an autoregressive model. The general MA(q) model has the formula shown in Equation 18.

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + + \theta_q \epsilon_{t-q}$$

Equation 18

We can generalize autoregressive models to AR(p) models as shown in Equation 19.

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t$$

Equation 19

If we combine the general formulas we get the ARMA(p,q) model shown in Equation 20.

$$y_t = c + \epsilon_t + \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

Equation 20

ARIMA modelling has the additional letter 'I' in the abbreviation which stands for 'integrated'. An integrated ARMA model the forecasted variable is differenced before the model is applied. In notation this results in an ARIMA(p,d,q) model in which the variable is differenced d times (Brooks, 2014). In notation for the ARIMA model the backshift notation shown in Equation 21, which shifts the data one period back, can be used for a first order difference.

$$y'_t = y_t - y_{t-1} = y_t - By_t = (1 - B)y_t$$

Equation 21

The ARIMA(p,d,q) model can be formulated with help of the backshift notation and is shown in Equation 22.

$$y_t = c + \epsilon_t + \sum_{i=1}^{p} \phi_i (1 - B)^d y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

Equation 22

To make sure the baseline model is suited for exogenous variables to be added another extension needs to be made to the ARIMA(p,d,q) model. This extension is called a vector autoregression model in which variables other than the forecasted variable can be added. These variables can be added in an endogenous and exogenous manner. Adding them endogenous means that the added variables will also be forecasted. However, in this research our only aim is to forecast SOFR and not the other variables that is why the variables are added exogenous. By adding the variables exogenous the model knows the value of the exogenous variable for the timesteps that should be forecasted. This way of using exogenous variables in an ARIMA model is called ARIMAX with the extra X standing for the exogenous variables that are added. The ARIMAX is expressed in Equation 23 where X_t is a vector of exogenous variables for timestep t and A is a matrix of coefficients (Brooks, 2014).

$$y_t = c + \epsilon_t + \sum_{i=1}^{p} \phi_i (1 - B)^d y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + AX_t$$

Equation 23

3.7 Comparing RNNs and Autoregressive models

A study performed by the Texas Tech University by (Namin & Namin, 2018) compared ARIMA vs LSTM in forecasting economic and financial time series. The study states that ARIMA models have demonstrated good performance in predicting the next lags of time series. The research questions investigates if long short-term memory algorithms are superior to the traditional ARIMA algorithm. The empirical study finds that an average reduction in error rates is obtained by the LSTM is between 84-87 percent when compared to ARIMA indicating superiority of LSTM to ARIMA. This research shows that LSTM is superior to ARIMA for forecasting univariate financial time series like the S&P 500 commodity price index.

If we compare a simple RNN and an AR(p) model they both use the past values to predict the current value of the dependent variable. The AR model uses the same values for the estimators for every prediction. Which means the prediction the AR models gives are very predictable if you know the values of the past p values of the series. This is different for a RNN because of the hidden state which is a vector containing information of the past values. This hidden state vector is constantly updated when we step through the timeseries. The hidden state vector by the two weight matrices W and U decide how the hidden state should be updated based on $\bar{x}^t$ and $\bar{h}^{t-1}$ for the current time step. This means that the hidden state vector contains all information on past timesteps also useless information for the current prediction, but it might be useful for future prediction. That is why we require an extra step to compute the output by parametrizing the hidden state vector with a weight matrix V to decide based on which information the prediction should be made (Namin & Namin, 2018).

The hidden state vector of a RNN allows us to store a lot of information about the past efficiently. The non-linear dynamics allows the RNN to update the hidden states in complicated ways. This makes it hard to visualize for RNNs how a forecast is made since no exact coefficients can be determined on which the forecast is made. This also makes it hard to evaluate if a certain behavior, like the spikes in SOFR, is captured by a neural network. Another drawback is that because a lot of weights are optimized during training of a neural network a lot more computation powers is needed than for training a regression model (Namin & Namin, 2018).

3.8 Conclusion

In this chapter the second research question "*Which neural network model is best suited for financial time series forecasting?"* is answered. When comparing the theory on feed forward neural networks (section 3.2) and on recurrent neural networks (section 3.3) we can conclude that recurrent neural networks are most suitable for forecasting sequences. However, a standard recurrent neural network has the problem of vanishing and exploding gradients which can be solved by applying Long Short-Term Memory nodes (section 3.4) to a recurrent neural network. Long Short-Term Memory nodes will help a recurrent neural network in storing long-term dependencies without vanishing or exploding. We can draw the conclusion that a LSTM-RNN is the best model to apply in our research on predicting multivariate financial timeseries. The further extensions of LSTM-RNNs might be used to improve our results found with regular LSTM-RNNs. The LSTM-RNN models' performance can be compared to an ARIMAX model defined in 3.6 to test its forecasting performance.

4 Application of models

In this chapter we answer the third research question *"How do we apply the chosen neural network models to SOFR?"* The recurrent neural network model with long short-term memory nodes is applied to the dataset specified in section 2.7. The network can be applied on different ways acquiring different outcomes. The different methods for application of a neural network are discussed in section 4.1. For the neural network model itself the input, hidden and output nodes need to be specified before the neural network can be set up. In section 4.2 the algorithm for defining a neural network is discussed and the practical application in python is discussed. To properly train the neural network, the right settings for the hyperparameters need to be chosen. The useful hyperparameters are specified and discussed in section 4.3.

4.1 Application of a Recurrent Neural Network model

The recurrent neural network model is a specific variant of a neural network which is suited for sequencing problems because it makes use of hidden states that store the sequence information up to the time step *t-1*. Basically, the hidden state can be computed at any time based on the previous hidden state h_{t-1} and the current input at time *t*. The simplest application of the recurrent neural network model to SOFR is the univariate method, in this method SOFR is forecasted based on its own past values. Every timestep the current value of SOFR is given as input which is used to update the hidden layers and compute a forecast for *t+1* for SOFR. The univariate RNN method is shown schematically and unfolded in Figure 13.

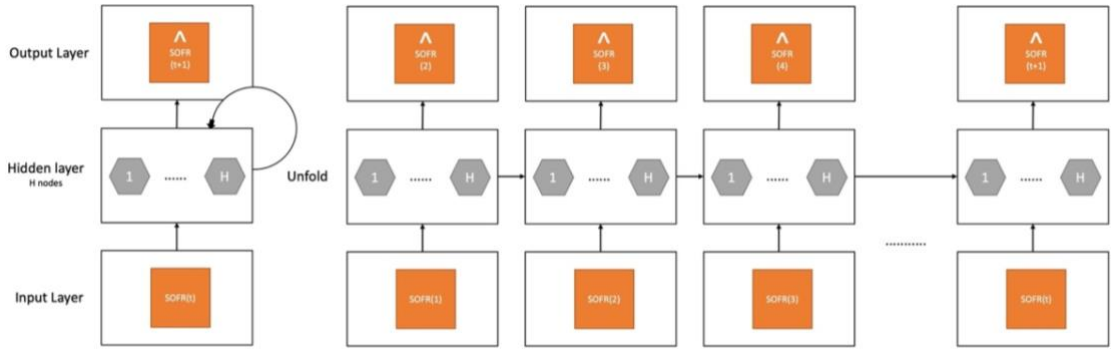


Figure 13: Univariate Recurrent Neural Network

4.1.1 Horizon

The application of the univariate method as shown in Figure 13 has a forecast horizon of one time-step since the output layer contains one node to make a forecast for *t+1*. To increase the forecast horizon, extra nodes should be added in the output layer to be able to forecast more than one time step in the future. The number of nodes in the output determines the forecast horizon of the neural network model. In Figure 14 extra nodes are added to the output layer to have a forecast horizon of n time steps in the future.

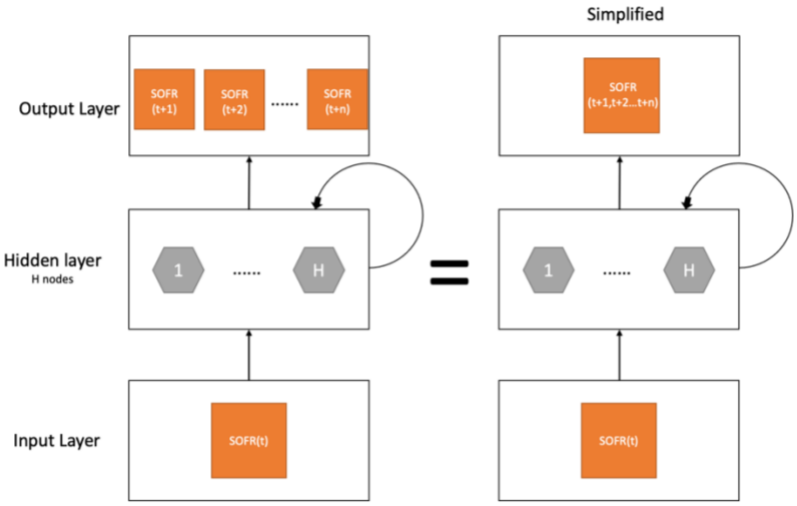


Figure 14: Univariate model longer horizon

4.1.2 Multivariate

To improve the forecast performance of the model, explanatory variables can be added to the input layer. All variables treated in chapter 3 can be added to the input layer or a selection of these variables can be added. In Figure 15 four extra variables are added to the model to give the model more information to base the predictions on. The number of extra variables that are added can be chosen and tuned by the modeler to decide which variables have a positive influence on the predictions made. The method in Figure 15 is called the lagged method since the situation at time step *t* is given to the model after which a prediction is made for *n* time steps, essentially the information of the explanatory variables is lagged so no information after time step *t* is given to the model.

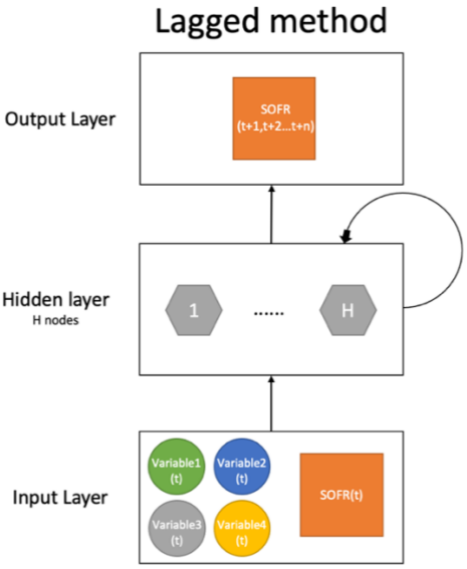


Figure 15: Lagged method

The endogenous multivariate method is shown in Figure 16 which also has the explanatory variables in the output layer. In this model the future values of the explanatory variables are forecasted for the same horizon *n* as we forecast SOFR for. It is the modelers choice to determine which variables are picked or not. The method is called the endogenous method because the future values of all variables are determined inside the model. Using this method, we hope the neural network models the relations between the variables to improve the forecasts of SOFR.

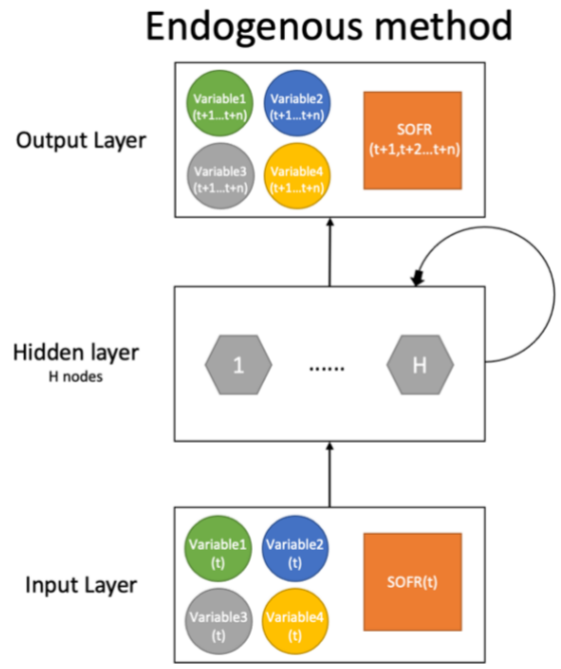


Figure 16: Endogenous method

The exogenous multivariate method is shown in Figure 17. This model is provided with information on the explanatory variables for the full forecast horizon. This allows the model to recognize patterns between the input variables and output variable SOFR at certain timesteps in the forecast horizon. This method is called the exogenous method because all variables are determined exogenous and are hard coded in the model as input.

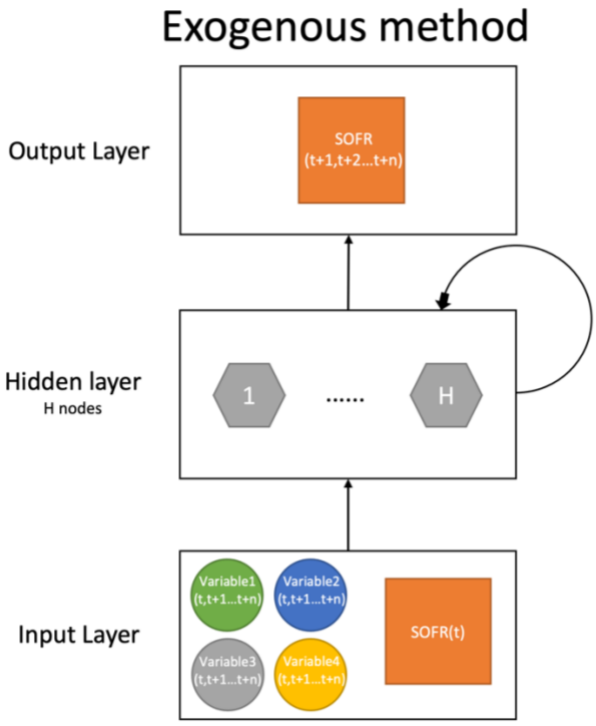


Figure 17: Exogenous method

4.1.3 Hidden layers

As more variables are added to the multivariate model, more relations and patterns in the data can be recognized. This requires the model to have more capacity to store information. This can be done in two ways, by expanding the number of nodes in the hidden layer or adding extra hidden layers. If extra layers are added to the model we speak of a deep recurrent neural network. This model can be seen in Figure 18.

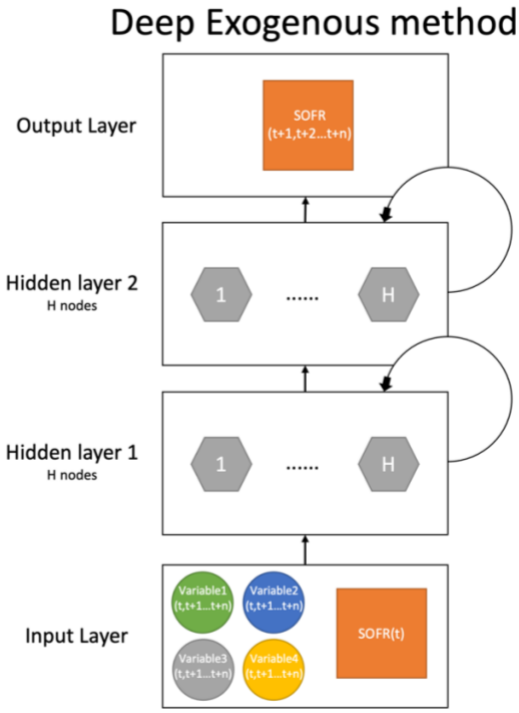


Figure 18: Deep Exogenous method

4.1.4 Overview and conclusion of methods

In Table 2 an overview is given of the different method a recurrent neural network can be applied to the data. For each method the input layers, hidden layers and output layers are specified. For convenience only two variables are added in the multivariate methods, this can be changed to more variables if desired by the modeler. The horizon of the different methods can be changed to meet the modeler's demand of the model. For the lagged, endogenous, exogenous method the explanatory variables defined in section 2.6 are used but in different ways. In the lagged method the variables are only used as a reference at time t while only SOFR gets forecasted for a forecast horizon of n. In the endogenous method the input of variables is the same as the lagged method, but the output of the method also forecasts the variables for a forecast horizon n. In a sense the endogenous method determines the value of variables for future time steps thus converting the explanatory variable to an endogenous variable which value is determined by the model, hence it is called the endogenous method. For the exogenous method this is different since for future time steps the explanatory variables are used as input and are used as exogenous variables determined outside of the model, hence exogenous method. The names of the methods are picked on common sense together with the company supervisor of this research. In chapter 5 and chapter 6 the deep exogenous method will be applied to attain the results and test the forecast performance of the recurrent neural network with LSTM nodes.

Method	Input	Hidden layers	Output	Horizon	Figure
Univariate one-step method	SOFR(t)	1 hidden layer of h hidden LSTM nodes	SOFR(t+1)	1	Figure 13
Univariate method	SOFR(t)	1 hidden layer of h hidden LSTM nodes	SOFR(t+1,…,t+n)	n	Figure 14
Lagged method	SOFR(t) Variable1 (t) Variable2 (t)	1 hidden layer of h hidden LSTM nodes	SOFR(t+1,….,t+n)	n	Figure 15
Endogenous method	SOFR(t) Variable1 (t) Variable2 (t)	1 hidden layer of h hidden LSTM nodes	SOFR(t) Variable1(t+1,…,t+n) Variable2(t+1,…,t+n)	n	Figure 16
Exogenous method	SOFR(t) Variable1(t, t+1,…,t+n) Variable2(t, t+1,…,t+n)	1 hidden layer of h hidden LSTM nodes	SOFR(t+1,….,t+n)	n	Figure 17
Deep exogenous method	SOFR(t) Variable1(t, t+1,…,t+n) Variable2(t, t+1,…,t+n)	2 (or more) hidden layer of h hidden LSTM nodes	SOFR(t+1,….,t+n)	n	Figure 18

Table 2: Overview applied methods

4.2 The algorithm

In section 4.1 we chose the 'deep exogenous method' as the model we are going to apply for forecasting SOFR. An algorithm for the neural network will be built in python to perform all experiments and test the performance of the defined neural network. In section 4.2.1 the python packages for neural network modelling will be discussed. In section 4.2.2 a brief example of the steps of the algorithm is given.

4.2.1 Python packages

To build the neural network in python the Keras deep learning API is used. The API is running on top of the machine learning platform TensorFlow (Keras, 2021). Keras is an interface to build machine learning algorithm and especially neural networks. The sequential model with LSTM nodes is used from the Keras API. In Appendix 2 a short example of the code is shown. To fully define the neural network a number of hyperparameters needs to be defined. These hyperparameters will be discussed in section 4.3.

4.2.2 Algorithm steps

The LSTM RNN algorithm consists of the following general steps shown in the list below. First the data is loaded into python one variable at a time. The data is cleaned, and missing data points are deleted from the dataset. The dataset is then scaled from 0 to 1 for reach variable separately. The dataset is split into a train dataset and test dataset. The test dataset is the dataset on which a forecast is made

after the model is trained on the train dataset. Then the data is reshaped into an understandable format for the neural network. The configuration of the neural network is first defined after which the model is compiled and a loss function is specified. Then the model is 'fitted' which means the neural network is trained on the train dataset. When the model is trained it can be used to make a forecast based on the test dataset. The forecast is compared to the true values of the test dataset and the performance of the neural network can be determined.

For applying a neural network, the algorithm can be divided in the following steps:

1. Loading in the dataset
2. Scaling the data
3. Splitting the dataset into a train dataset and a test dataset
4. Reshape input to (samples, time steps, features)
5. Defining the neural network
6. Compiling the neural network
7. Training the neural network based on the train dataset
8. Making a forecast of the test dataset
9. Evaluating the performance of the network

4.3 Hyperparameters

Once the neural network model is defined, the model can be trained using the specified train dataset. The training of a neural network is a delicate process which greatly influences the performance of the neural network. The trainable parameters are learned by the algorithm during training like the weights of a neural network. The parameters that can be adjusted to customize the training of a neural network are called hyperparameters. We will distinguish hyperparameters in model hyperparameters and algorithm hyperparameters. Model hyperparameters influence the configuration of the model so the number of neurons in a hidden layer and the number of layers the model consists of. Algorithm hyperparameters influence the learning ability of an algorithm. In this section we discuss all hyperparameters however it is not until section 5.2 that we define the actual values of the hyperparameters for the experiments.

4.3.1 Number of layers and Neurons per layer

The number of layers in the neural network and the number of neurons per layer are both model hyperparameters which influence the ability of the neural network to save information of the dataset and recognize patterns in the dataset. A single layer neural network with sufficient neurons is assumed to have enough capability to approximate simple linear function but has difficulties with recognizing more sophisticated patterns due to the limited number of layers (Reed & Marks, 1999). Empirically, more layers in the network results in better generalization for a wide variety of tasks. Adding an extra layer gives the network more memory and capacity to recognize patterns. Generally, the first layers have a higher number of neurons and subsequent layers get fewer neurons as more specific information gets passed on in the network.

4.3.2 Loss function

During the training of a neural network the error between the predicted and actual values is minimized. How the error function is calculated is called the loss function of the model. The loss function is used to estimate the loss of the model in order to update the weights to reduce the loss on the next evaluation. The nature of the problem decides which type of loss function matches the specific needs of the problem. Since our output consists of a real-valued quantity we have a regression problem and we should use a loss function appropriate to this. So, loss function appropriate for classification problems will not be considered.

The most common used regression loss function is the Mean Squared Error, or MSE. The mean squared error is calculated as the average of the squared differences between the actual and predicted values.

This means that the model is punished for making large mistakes since the errors are squared. This also results in values that are always positive regardless of if the predicted values are above or below the actual value. For reporting purposes, the root of the MSE, Root Mean Squared Error or RMSE, is taken since this brings back the squared values to their original magnitude.

The Mean Absolute Error, or MAE, loss is another appropriate loss functions for regression problems. This loss function is more robust to outliers which are predicted values with a large deviation from the actual values. The MAE is calculated as the average absolute difference between the actual and predicted values. This loss function does not punish outliers as hard as the RMSE. To make comparison between models easier the Mean Absolute Percentage Error, or MAPE, can be used. The MAPE is the average absolute percentage error which is calculated by averaging the absolute error per observation divided by the actual value of the observation.

4.3.3 Optimizer

The loss function is essentially a mathematical way of measuring how wrong your predictions are. The optimizer is responsible for minimizing the loss function. During the training process, the optimizer updates the model parameters to make the predictions closer to their actual values. The loss function tells the optimizer when its optimization is moving in the right or the wrong direction. The optimizer is responsible to determine in which direction and with which magnitude changes should be made to the parameters of the model. The learning rate of an optimizer is the number that changes are multiplied by to tune the size of the steps the algorithm takes when optimizing the parameters of the model.

A rather standard optimizer in machine learning is called Gradient Descent. The Gradient Descent algorithm is considered robust and flexible. The algorithm calculates what the effect is of a small change in each individual weight on the loss function and decides in which direction each parameter should be optimized. Then each individual weight is adjusted based on its gradient. This process of calculating the gradient and then adjusting the weight is repeated iteratively to minimize the loss function and improve our model. A downside of applying the Gradient Descent algorithm is that it requires a lot of computation power since the whole dataset is evaluated before an update is done. Another downside is that the algorithm is susceptible to getting stuck in a local optimum or minima. This means that a combination of weights of parameters is found which cannot be further optimized to further reduce the loss function however it might be that a totally different combination of weights of parameters is the true optimum (Algorithmia, 2018).

To speed up the Gradient Descent algorithm it might be beneficial to update on only one training sample or a subset of a training sample rather than your whole training dataset. This method is called the Stochastic Gradient Descent and either uses batches or a random sample of parameters to optimize and speed up the algorithm because of faster convergence (Algorithmia, 2018).

Adaptive Moment Estimation (Adam) is an optimizer which is the most used optimizer and has proven to outperform gradient descent and stochastic gradient descent (Kingma & Lei Ba, 2015). Adam uses momentum and adaptive learning rates to converge faster. Applying momentum means that if the algorithm is moving into one direction for multiple consecutive steps, then the steps will get bigger each time. This helps to converge faster to an optimum. The adaptive learning rate helps with starting off with big steps and make smaller steps as we get closer to an optimum. As the learning rate decays we take smaller steps and we get closer to the optimum (Hansen, 2019).

4.3.4 Learning rate

When specifying the optimizer, a learning rate should also be set. The learning rate of an optimizer influences the convergence of the optimization. The direction and magnitude of a parameter update is given by the learning rate multiplied by the gradient of the loss function. Having a learning rate which is too big or too small results in changing the weights of the parameters too much or too less. If the learning rate is too small, the updates are small, and optimization will take a long time which

results in finding poor local optima or plateau's. If the learning rate is too big, the update will be too large and the optimization will not converge and likely diverge. Another potential problem of a high learning rate that it is more likely to have exploding gradients in your neural network. So, the learning rate should be chosen carefully such that updates are of appropriate size and the optimization should converge to an optimal set of weight of parameters (Katanforoosh, Kunin, & Ma, 2019).

In the adaptive optimization algorithms like Adam the learning rate is decayed during training. This means that the learning rate becomes lower as the optimization is in progress resulting in bigger parameter updates in the beginning and smaller parameter updates as we approach the optimum. The exponential decay rate of the optimizer of the first and second moment can be adapted and is specified by the beta of the Adam optimizer.

4.3.5 Batch size

The batch size is the number of data points that are used to train the model in each iteration before the model parameters are updated. The algorithm iterates over the samples in the batch and makes a prediction for each sample, the predictions are compared to the actual values and the loss is computed. Based on this loss the algorithm updates the weights of the parameter and the algorithm moves on to the next batch in the dataset. A dataset can be divided into one or more batches based on the size of the batches and size of the dataset. Typical batch sizes are 32, 64, 128, 256 and so on depending on the size of the dataset (Katanforoosh, Kunin, & Ma, 2019).

Choosing the right batch size is essential to ensure convergence of the loss function and parameter values but is also important for the generalization of the model. Since batch size determines the frequency of updates, the smaller the batches, the more often updates of the parameters are performed. The larger the batch size the more accurate is the gradient of the loss function.

4.3.6 Epochs

The number of epochs defines the number of times the algorithm will work through the entire training dataset thus completing all batches that the dataset is divided in and updating the parameters of the model. In one epoch the model has had the opportunity to see every sample in the dataset and use it to update the model parameters. The number of epochs is typically rather large into the hundreds or thousands depending on the specific dataset and problem. The number of epochs should be high enough to let the model optimize the parameters until the error of the model is sufficiently minimized. The number of epochs has a great influence on the running time of the algorithm. To stop training the model at a sufficient number of epochs, stopping criteria can be added which stop the algorithm once the loss of the model no longer decreases or another criterion is met (Brownlee, 2017).

4.4 Application of ARIMAX model

The formula of the ARIMAX model that is given in section 3.6 should be applied to the dataset in a same manner as that it is done with the neural network. The ARIMAX model will be applied just like the exogenous method of the neural network application. The exogenous neural network model is required to forecast for a horizon of n observations by making use of the values of the exogenous variables for the same forecast horizon. To determine the p and q value of the ARIMAX(p,d,q) the autocorrelation and partial autocorrelation plots can be used to find the significant number of lags for the model. Once these are known the ordinary least squares (OLS) regression algorithm can be used to find the relevant parameters of the ARIMAX model which can be used to make a forecast.

4.5 Conclusion

In this chapter we answered the third research question *"How do we apply the chosen neural network models to SOFR?"* In section 4.1 the deep exogenous method is chosen to be the suitable method to forecast SOFR based on the dataset. The deep exogenous method uses the exogenous variables purely as input to forecast SOFR. In section 4.2 Keras and Tensorflow are picked as suitable packages to use in python for neural networks. Also, in section 4.2 the steps of the algorithm are defined. In section 4.3 the following hyperparameters are discussed: number of layers, number of neurons per layer, loss function, optimizer, learning rate, batch size and number of epochs. In section 4.4 the application of the ARIMAX model is discussed and explained how to determine the number of lags for the regression.

5 Experimental setup

To find the best model performance out of the recurrent neural network we need to optimize the hyperparameters of the model. Finding the combination of hyperparameter values that have the best performance can be done through hyperparameter tuning. To find the best settings for the neural network model the optimization approach is first specified. After the hyperparameter tuning approach is specified the experimental setup can be defined. To optimize the hyperparameters a number of trials is run with different values for the hyperparameters to find the best combination of hyperparameters possible. To do this, a specific range is defined for every hyperparameter in which the algorithm may find the optimal value. Furthermore, the test setting for experiments needs to be defined namely the division between train and test data, the forecast horizon, and the number of combinations for hyperparameters called trials that are run. After running the experiments, the best configuration can be chosen.

5.1 Hyperparameter tuning approach

Most of the time a general estimate can be made by the modeler on how to best set the hyperparameters of a neural network model. This can be done by making use of rules of thumb for configuring hyperparameters and the experience of the modeler. However, hyperparameters interact with each other so finding the optimal combinations of the numerous hyperparameters is hard to find when using a naïve optimization algorithm. Finding the optimal combination of hyperparameters is a very time-consuming process since for each combination of hyperparameters the whole training process should be completed, validated, and evaluated to test the performance of the hyperparameter combination. There are several approaches to hyperparameter tuning. From very basic approaches to extensive optimization algorithms. In the following paragraphs a few optimization algorithms are explained which are the manual approach, grid search, random search, and Bayesian optimization.

The most basic approach is the manual approach of setting hyperparameters based on the modelers opinion of what is best, training the model with the chosen hyperparameters and scoring the performance of the trained model. This process can be repeated until the modeler is satisfied with the performance of the model. Applying the manual model might help to give the modeler some degree of insights into the model.

Performing a grid search is a more automated approach of trying all hyperparameters in a manually specified search space of hyperparameters. A grid of hyperparameter values is set up and for each combination the model is trained and scored on validation data to find the best performing combination of hyperparameters. A grid search is reliable in low dimensional spaces but when the number of hyperparameters gets higher and the number of options per hyperparameters gets higher this is a very exhausting and time-consuming method. A grid search is especially ineffective when evaluating hyperparameters which turn out to have very little effect on the performance of the model. A grid search can be applied to spot check a certain small area of hyperparameters that is known to have a good performance and to find the final optimal combination of hyperparameters (Bergstra & Bengio, 2012).

A random search defines a search space to randomly select hyperparameters from. The number of combinations that is checked is set on forehand based on the modeler's preferences. A random search generally outperforms a grid search when only a limited number of hyperparameters influences the performance of the model. When time is limited the running time can be easily controlled by setting the number of trials lower. The experiment can be stopped any time while still having a complete experiment. During the experiments extra computing power can be added and run parallel to the prior executions as simply a random set of hyperparameters can be picked and evaluated. Even if an error occurs only the single trial should be repeated or simply abandoned and not executed anymore which does not further affect total experiment (Bergstra & Bengio, 2012).

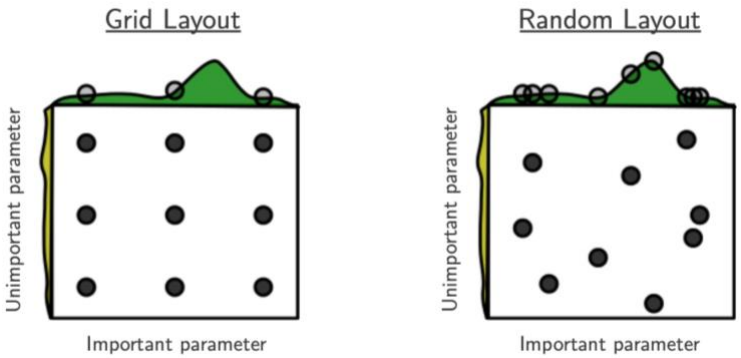


Figure 19: Grid search vs Random Search (Bergstra & Bengio, 2012)

A more advanced way of hyperparameter tuning is Bayesian Optimization. This approach is based on the assumption that hyperparameter tuning is like optimizing a black-box function. The function to be optimized is the loss function of the model which can be done by testing chosen combinations of hyperparameters. Bayesian optimization is based upon two factors, a belief of the loss function of the model called the prior and an acquisition function which decides what combination of hyperparameters is tested next (Wang, Hutter, Zoghi, Matheson, & Freitas, 2016). The prior is the belief of the mean and variance of a loss function which is updated with each new iteration. Based on the belief of the prior the next combination of hyperparameters is chosen. There are two ways to choose the next combination of hyperparameters: a combination near to the observed best values with slightly better performance or explore a totally new combination of hyperparameters of which information of performance is not known. This choice is represented in the acquisition function and known as the exploitation and exploration trade-off. An acquisition function can be interpreted as the utility of unseen solution and the aim is to maximize the acquisition function when proposing a new candidate combination of hyperparameters (Camero, Wang, Alba, & Bäck, 2021). The method of Expected Improvement is a commonly used acquisition function which represents the improvement potential over the hyperparameters state space given the results of prior tested combinations of hyperparameters. In Figure 20 three iterations of a single parameters Bayesian optimization are shown with the belief of the objective function, the prior, shown in purple and the acquisition function in green. The solid line represents the true objective function which is unknown to the modeler. The dotted line represents the believed value of the objective function with the purple shaded area showing the confidence interval of the objective function. The acquisition function is high when uncertainty is high (exploration) and high when the mean of the believed objective function is high (exploitation). The Bayesian optimization iterates the following steps: (i) use the acquisition function to pick a combination of hyperparameters which has the highest value; (ii) evaluate the combination of hyperparameters, (iii) update the prior based on the new data (Wang, Hutter, Zoghi, Matheson, & Freitas, 2016). When the prior is updated with the new data, the acquisition function is updated, a new combination of hyperparameters is picked and evaluated.

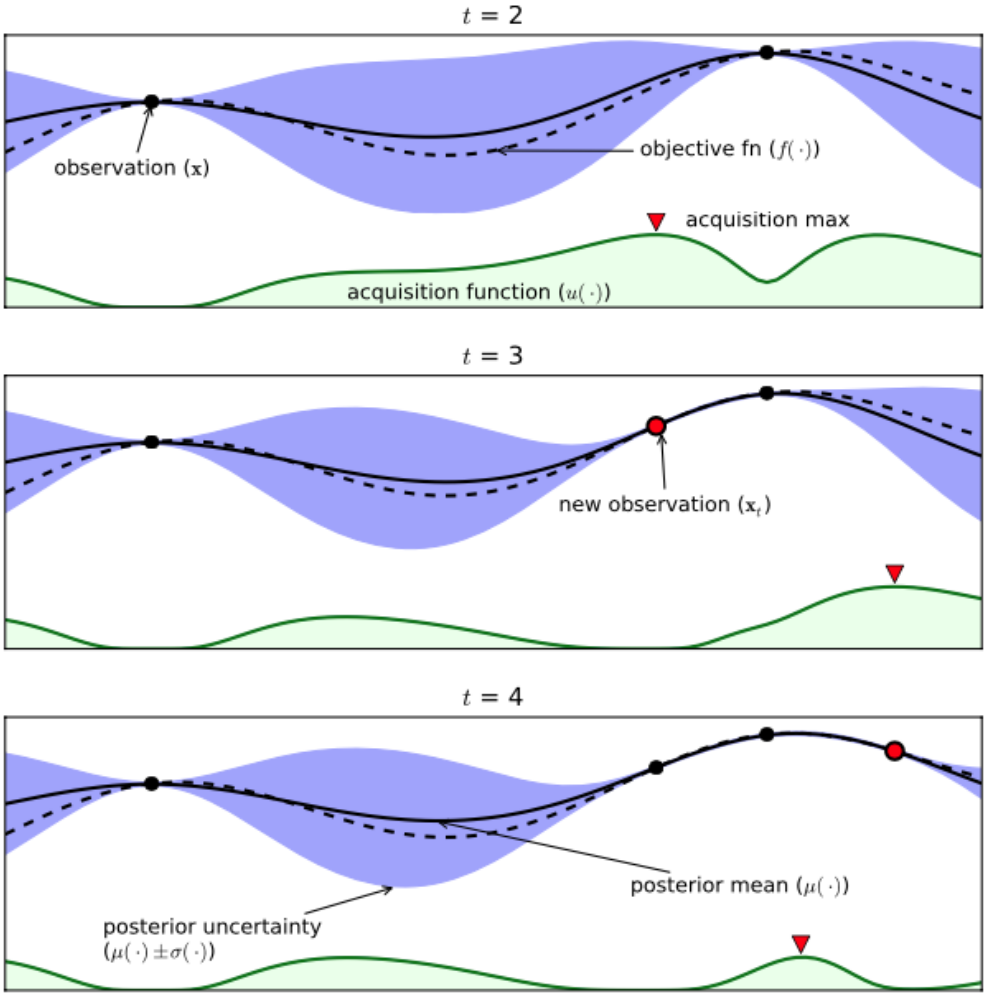


Figure 20: Three iterations of Bayesian optimization (Wang, Hutter, Zoghi, Matheson, & Freitas, 2016)

Bayesian optimization generally requires less trials of combinations of hyperparameters to find an optimal configuration of hyperparameters compared to a grid search or random search. The time in between testing combinations of hyperparameters might be higher due to the sampling process of picking a new combination of hyperparameters based on the acquisition function. This computation time is very low compared to the time it takes to evaluate one combination of hyperparameters, so Bayesian optimization still outperforms the other methods.

For our optimization problem we pick Bayesian optimization as the favorable hyperparameter tuning approach since this approach requires less iterations to find the optimal combination of hyperparameters. Since computation time of the neural network is high we want to minimize the number of combinations of hyperparameters, trials, that is executed to find the optimal combination.

5.2 Hyperparameter set up

For the Bayesian optimization algorithm to be properly executed a search space for the hyperparameters needs to be defined. In section 4.3 all hyperparameters are specified that can be tuned to ensure good performance of the model. The search space is defined as the volume to be searched where each dimension represents a hyperparameter and each point represents one model configuration. The search space gets very big very fast as more model hyperparameters are added that is why not all hyperparameters are made variable in the Bayesian optimization. The loss function and optimizer are not made available to the Bayesian optimization algorithm. The loss function is chosen because of modelers preferences. The optimizer is picked based on best practices.

5.2.1 Number of layers and Neurons per layer

The shape and size of the recurrent neural network model is specified by the number of layers and number of long short-term memory cells or neurons per layer. The model always consists of an input layer and an output layer with a number of LSTM layers in between. The input layer consists of a value for SOFR at time step zero together with all values of the exogenous variables for the forecast horizon. The output layer consists of a dense layer with the same number of neurons as timesteps in the forecast horizon. In between the input and output layer, a number of LSTM layers are situated which can range from 1 to 6. The number of neurons in the individual LSTM layers can differ and range from as low as 32 neurons to 512 neurons in steps of 32 neurons.

5.2.2 Loss function

For the loss function the decision is made to use the Mean Squared Error is used. This loss function punished higher deviations in the forecast. This is useful for training the model to forecast the spikes in SOFR which tend to deviate from the average level that SOFR has during the rest of the month.

5.2.3 Optimizer

For the optimizer of the model the Adam optimizer is chosen. Current literature suggests this is the best and most widely used optimizer for deep neural network models (Kingma & Lei Ba, 2015). A few trials were done for the other suggested optimizer which performed worse than the Adam optimizer as expected.

The loss function and optimizer are not optimized by the Bayesian optimization algorithm since these are choices that can be made based on modelers expertise and experience with the modelled problem.

5.2.4 Learning rate

The learning rate of the Adam optimizer is set to 0.001 by default. The Bayesian optimization algorithm is given a choice between 0.01, 0.001 and 0.0001 as the learning rate of the Adam optimizer. These learning rates are established learning rates and the learning rate can only pick one out of the three options. Once it becomes prevalent that one learning rate is dominant and performing better it might be beneficial to set the learning rate to this value for every iteration of the Bayesian optimization algorithm.

5.2.5 Batch size

The batch size of the model can vary between 96 and 320 with a step size of 32. By default, the batch size is set to 288 since a higher batch size results in a shorter training time of the neural network so the first trial of the Bayesian optimization can be computed quickly to get results to base next combination of hyperparameters upon.

5.2.6 Epochs

The number of epochs of the models has a big influence on the performance of the model due to overfitting and underfitting. The optimal number of epochs is dependent on the number of layers and

neurons in each layer, learning rate and batch size. That is why a broad range is taken for the number of epochs. The number of epochs may vary between a minimum of 200 epochs and a maximum of 1200 epochs. If only a limited number of layers and neurons is used, a high learning rate is picked is used the model tends to converge fast since the model is small and updates are large so overfitting might appear if a too high number of epochs is chosen. The step size of the number of epochs is dependent on the minimum and maximum number of epochs and is specified as one tenth of the difference between the minimum and the maximum number of epochs. So, for the current specified minimum and maximum this results in increments of 100 epochs from 200 to 1200 and a total of 11 options for the number of epochs including the minimum and maximum number of epochs.

5.2.7 Search space hyperparameters

In Table 3 an overview of the search space of the hyperparameters is given. These hyperparameters are the most influential in the performance of the model thus need to be optimized by the Bayesian optimization tuner. The Bayesian optimization technique ensures that the search spaces is searched efficiently and not all feasible configurations have to be evaluated.

Hyperparameter	**Minimum**	**Maximum**	**Step size**
Number of layers	1	6	1
Number of LSTM neurons	32	512	32
Learning rate	0.01	0.0001	Log scale
Batch size	96	320	32
Epochs	200	1200	100

Table 3: Search space hyperparameters

5.3 Experiments

To properly test the capabilities an experimental setup needs to be defined. The aim of the experimental setup is to test the behavior of the model in different situations meaning different forecast periods and different behavior of SOFR. Since the spikes in SOFR existent throughout the whole dataset, different experimental set ups can be used to define a train or forecast period with or without spikes. So, to specify the experimental setup for the model the train and test dataset of the algorithm of section 4.2 can be changed. The division of train and test data depends on what period a forecast is made. Since the model is simply trained with data up until the start of the forecast period. For example, if the forecast period starts at the first of January 2021 then the model is trained with data up until the 31st of December 2020.

In Table 4 5 train/test data splits are defined to test the model on. The train and test data are specified based on the characteristics that are apparent in the data set. The first and second experiment train the model on data with spikes and also forecasts a period including spikes. The third experiment trains the model on data with spikes but tests the model on data without spikes. The spikes in this third test period were suppressed by repos and reverse repos executed by the federal reserve to dampen the spikes. The fourth experiment starts its forecast period just after the covid pandemic hit the financial markets. This experiment should show whether the model can handle a sudden change to almost negative SOFR rates. The fifth experiment also has its forecast period during covid however the model did get a chance to adapt to the change in the behavior of SOFR due to the covid pandemic.

For alle experiments a forecast horizon of 60 business days is used which is right about 3 months in the dataset only considering business days for which data is available. This forecast horizon was

chosen because this would resemble in forecasting 3 spikes. Another reason is that 3 month is the also the tenor of the most referenced USD LIBOR rate namely the 3-month USD LIBOR. While SOFR and LIBOR are two completely different rates with SOFR being overnight and LIBOR being a forward-looking term rate. 3-month USD LIBOR currently gives the best forecast of what the overnight rates might be in 3 months that is why it is relevant to be able to forecast SOFR for 3 months in the future. We want to have forecast horizon as long as possible however a longer horizon than 3 months would result in a less accurate forecast which would not be useful.

Training period characteristic	**Training period start**	**Training period end**	**Test period characteristic**	**Test period start**	**Test period end**	**Forecast horizon (days)**
1. Spikes	22/08/2014	21/06/2018	Spikes	22/06/2018	17/09/2018	60
2. Spikes	22/08/2014	21/04/2019	Spikes	22/04/2019	14/07/2019	60
3. Spikes	22/08/2014	22/11/2019	No spikes	23/11/2019	21/02/2020	60
4. Pre covid	22/08/2014	20/04/2020	Start covid	22/04/2020	19/07/2020	60
5. Spikes + No spikes	22/08/2014	22/11/2020	No spikes	22/11/2020	21/02/2021	60

Table 4: Experimental set up

On purpose no structural breaks like the actual beginning of the covid pandemic in March 2020 are included in the test datasets because these are exceptional cases which cannot be anticipated and ruins any forecast that is made.

5.4 Execution of experiments

Once the training and test data split is defined we search for the optimal by executing the Bayesian optimization algorithm. Before the Bayesian optimization algorithm can be executed the number of trials of the algorithm have to be defined. The number of trials is set to 20 due to time constraints as one trial takes about 5 minutes, however this can be set higher when more time is available. This means that 20 configurations of the LSTM neural network are trained and used to forecast each experiment after which the configuration is chosen which has the lowest root mean squared error (RMSE).

5.5 Conclusion

In this chapter we answered the fourth research question *"How do we optimize the hyperparameters of the neural network model to get the best performance of the neural network on the defined experiments?"* To optimize the hyperparameters of the neural network a number of optimization algorithms are defined in section 5.1. The Bayesian optimization algorithm is chosen since this algorithm requires less iterations to find a good combination of hyperparameters than other algorithms like a grid search or random search. In section 5.2 the search space for all hyperparameters of the neural network are defined to execute the Bayesian optimization algorithm. In section 5.3 the experimental set up is defined based on train and test data splits to test the performance of the neural network on different forecasting period. In section 5.4 a number of 20 trials per experiment is defined for the execution of the Bayesian optimization algorithm.

6 Results

In this chapter the experiments defined in chapter 5 are run to answer research question 5 and 6. To answer the fifth research question "*What is the performance of the neural network model on forecasting SOFR?"* the Bayesian optimization algorithm is run for the 5 different forecasting periods. In section 6.1 the best selection of exogenous variables for the LSTM model and ARIMAX is determined based on significance of individual variables. In section 6.2 the results of the experiments with the best selection of exogenous variables are given. The main performance metric on which performance will be evaluated is the root mean squared error (RMSE) while the mean absolute percentage error (MAPE) is also computed for reference. In section 6.3 the results of the experiments are concluded. In section 6.4 the sixth research question "*What is the variability in outcomes of the neural network model?"* is answered. The variability is tested by repeating one experiment multiple times and check the variability of the results.

6.1 Selection of exogenous variables

The numerical results of the runs with all variables can be found in Table 5 in which the RMSE and MAPE are given for both the LSTM model and the ARIMAX model for all experiments. In the Appendix 3 the visual results of the runs with all variables included can be found. It can be concluded from Table 5 that the LSTM model outperforms the ARIMAX model on every experiment with an average RMSE of 0.02720 for the LSTM model. The neural network LSTM model realizes an average reduction of nearly 53% in RMSE and nearly 55% in MAPE. The reduction in error for experiment 3, 4 and 5 is higher than experiment 1 and 2 because in experiment 3, 4 and 5 SOFR exhibits more irregular behavior which the ARIMAX model finds hard to forecast. The visual results of the experiments containing all variables can be found in Appendix 3. However, if we look at the visual results of the results of the LSTM model we see that the LSTM model does not recognize the spikes occurring in SOFR in experiment 2 and 3.

Experiment	**Start date**	**LSTM RMSE**	**ARIMAX RMSE**	**Reduction in RMSE**	**LSTM MAPE**	**ARIMAX MAPE**	**Reduction in MAPE**
1	22/06/2018	0.03196	0.03926	-18.59%	0.01184	0.01410	-16.03%
2	22/04/2019	0.05404	0.06387	-15.39%	0.01437	0.02321	-38.09%
3	23/11/2019	0.02273	0.16676	-86.37%	0.01076	0.09290	-88.42%
4	22/04/2020	0.01175	0.06718	-82.51%	0.25603	1.00156	-74.44%
5	22/11/2020	0.01555	0.04017	-61.29%	0.25839	0.59621	-56.66%
Average		0.02720	0.07545	-52.83%	0.11028	0.34560	-54.73%

Table 5: Results experiments all variables used

To improve the forecasts, especially on the SOFR spikes, of the LSTM and ARIMAX model we assess the significance of the exogenous variables. Since insignificant variables do not improve the forecast of both models the insignificant variables should be deleted from the dataset on which the models are trained. To determine the significance of the variables the p-value from the ARIMAX model will be used. For each variable a p-value is computed which is between 0 and 1. A p-value below the significance level, typically 0.05, indicates the variable is significant. If the p-value is above the significance level then the variable does not contribute to forecasting SOFR.

In Table 6 the coefficients and p-values of all exogenous variables are shown for the regression done for experiment 2. The coefficient shows the size of the influence of the variable and the p-value shows the significance of the variable. If we take a significance level of 0.05 then the federal funds rate, date dummy and reverse repo variables are significant based on the p-values from Table 6.

EXPERIMENT 2	coef	std err	z	P-value
FedFunds	0.5852	0.0280	21.0220	0.0000
DateDummy	0.0297	0.0010	29.1990	0.0000
FOMCDummy	-0.0019	0.0020	-0.8430	0.3990
Repo	-0.0072	0.0060	-1.1950	0.2320
ReverseRepo	0.0302	0.0060	5.0980	0.0000
VIX	0.0049	0.0080	0.5950	0.5520
Reservebalance	-0.0640	0.0620	-1.0340	0.3010

Table 6: ARIMAX summary experiment 2

In Table 6 the results of the OLS ARIMAX regression for experiment 2 is shown. From the P-values we can conclude that the Fed Funds Rate, Date Dummy and Overnight Reverse Repo transactions are a contribution to the model. To test the significance of the variables we take the setting of experiment 2 and delete the insignificant variables one at a time. For every configuration of variables that are used the model is run and the Bayesian optimization algorithm is executed to find the best model configuration for the LSTM model. In Table 8 the results of the runs are shown when using different variables. It can be concluded that the performance of the model increases as the insignificant variables are deleted. The EFFR, date dummy and reverse repo are picked as significant variables for the best LSTM model for which we will present the results in section 6.2. To verify this decision the same analysis of deleting variables one at a time is done for experiment 1 of which the results are shown in Table 7. We find that the EFFR, date dummy and reverse repo show the best results for the RMSE of the LSTM model.

Variables used for experiment 1	**LSTM RMSE**	**LSTM MAPE**	**ARIMAX RMSE**	**ARIMAX MAPE**
EFFR, date dummy, FOMC dummy, repo, reverse repo, VIX, reservebalance	0.03196	0.01184	0.03926	0.01410
EFFR, date dummy, repo, reverse repo, VIX, reservebalance	0.03324	0.01334	0.03756	0.01360
EFFR, date dummy, repo, reverse repo, reservebalance	0.03181	0.01245	0.03928	0.01437
EFFR, date dummy, repo, reverse repo,	0.03125	0.01342	0.05079	0.02088
EFFR, date dummy, reverse repo,	0.02904	0.01163	0.05047	0.02054
EFFR, date dummy	0.03114	0.01217	0.05678	0.02404

Table 7: Exogenous variables test experiment 1

Variables used for experiment 2	LSTM RMSE	LSTM MAPE	ARIMAX RMSE	ARIMAX MAPE
EFFR, date dummy, FOMC dummy, repo, reverse repo, VIX, reservebalance	0.05404	0.01437	0.06387	0.02321
EFFR, date dummy, repo, reverse repo, VIX, reservebalance	0.05411	0.01516	0.06342	0.02311
EFFR, date dummy, repo, reverse repo, reservebalance	0.04878	0.01476	0.06291	0.02290
EFFR, date dummy, repo, reverse repo,	0.04743	0.01385	0.04833	0.01594
EFFR, date dummy, reverse repo,	0.04663	0.01183	0.04763	0.01556
EFFR, date dummy	0.04631	0.01260	0.04757	0.01564

Table 8: Exogenous variables test experiment 2

6.2 Results experiments

Through the steps in section 6.1 we find that the Effective Federal Funds rate, Date Dummy and Overnight Reverse Repo transactions are the best exogenous variables to use in the LSTM neural network model. In Table 9 the results are shown for the 5 experiments with an average RMSE of 0.02448 and an average MAPE of 0.1252 for the LSTM model compared to 0.04615 and 0.33158 for respectively RMSE and MAPE for the ARIMAX model. As we see in Table 9, the LSTM model outperforms the ARIMAX model on every experiment and on average a reduction of 45% for both the RMSE and MAPE. The visual results of the LSTM and ARIMAX model are shown in Figure 21 to Figure 25. We can conclude that the LSTM model does recognize the spikes in experiment 1 and 2 which is an improvement over the earlier experiments with all variables included (Figure 21 and Figure 22). In experiment 3, Figure 23, the model seems to recognize the first spike after which it also knows that the spikes are not occurring any more at the last day of December 2019 and January 2020 where the ARIMAX model still expects the spikes to occur. For experiment 4 and 5 the LSTM model has learned that the spikes are not occurring anymore while the ARIMAX model is still expecting full spikes at the end of the month (Figure 24 and Figure 25). So, for experiment 4 and 5 the comparison is not exactly fair because the ARIMAX does not adapt to the new circumstances, however for experiment 1,2 and 3 the comparison is fair. If we take the average improvement for experiment 1,2 and 3 we find an average reduction of 32% for the RMSE.

The study from 2018 by (Namin & Namin) mentioned in section 3.7 obtained a reduction of 84 to 87 percent on a univariate comparison between LSTM and ARIMA. If we compare the average reduction of 45% of our model to the study from 2018 we can conclude that a reduction of 45% in RMSE is conservative and realistic. The main difference between the two studies is that in this research we are performing a multivariate analysis by making use of exogenous variables. This helps both the LSTM and ARIMAX model in their forecast and it seems a fairer comparison since the outcomes are closer to each other than in the research from 2018.

Experiment	Start date	LSTM RMSE	ARIMAX RMSE	Reduction in RMSE	LSTM MAPE	ARIMAX MAPE	Reduction in MAPE
1	22/06/2018	0.03178	0.05047	-37.04%	0.01254	0.02054	-38.96%
2	22/04/2019	0.04366	0.04763	-8.33%	0.01160	0.01556	-25.44%
3	23/11/2019	0.01870	0.03806	-50.88%	0.00939	0.01753	-46.43%
4	22/04/2020	0.01331	0.06518	-79.58%	0.38529	1.21552	-68.30%
5	22/11/2020	0.01496	0.02939	-49.10%	0.20718	0.38874	-46.70%
Average		0.02448	0.04615	-44.99%	0.12520	0.33158	-45.17%

Table 9: Results experiments LSTM and ARIMAX, EFFR, date dummy and reverse repos used

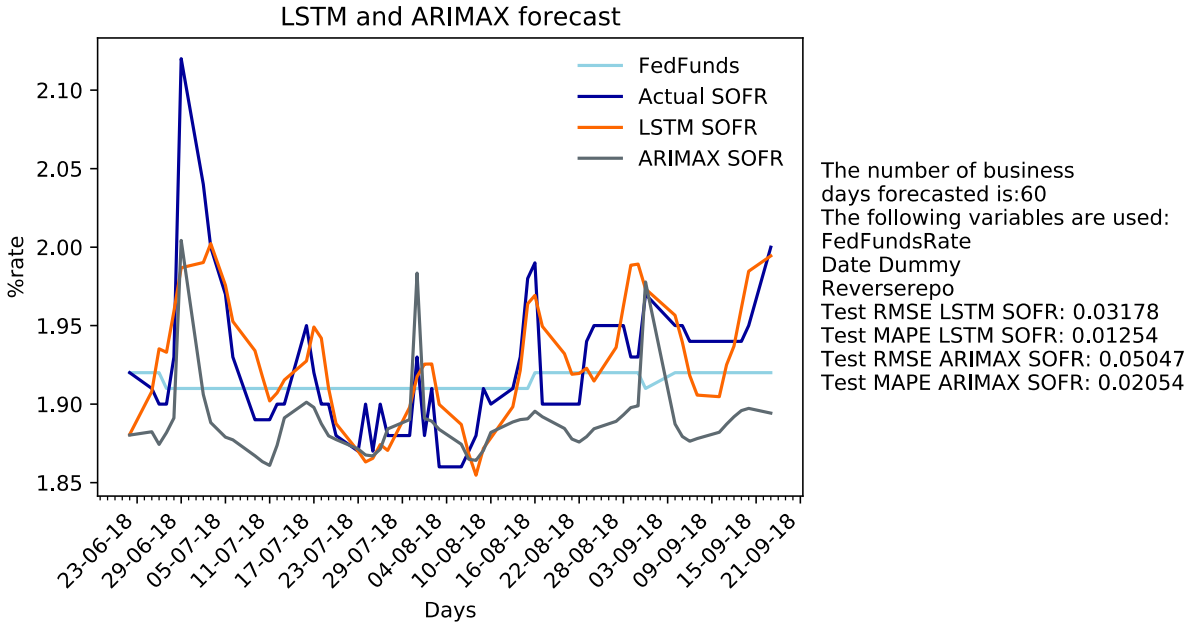
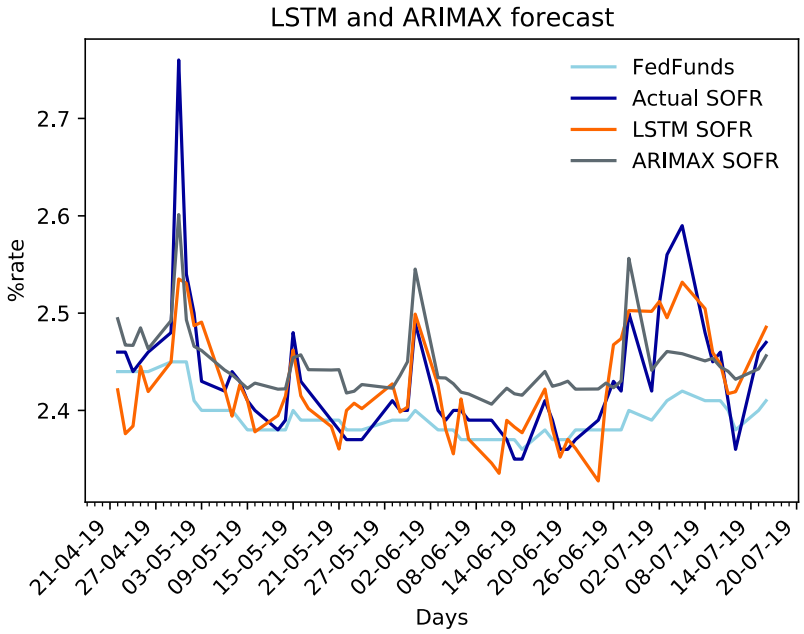
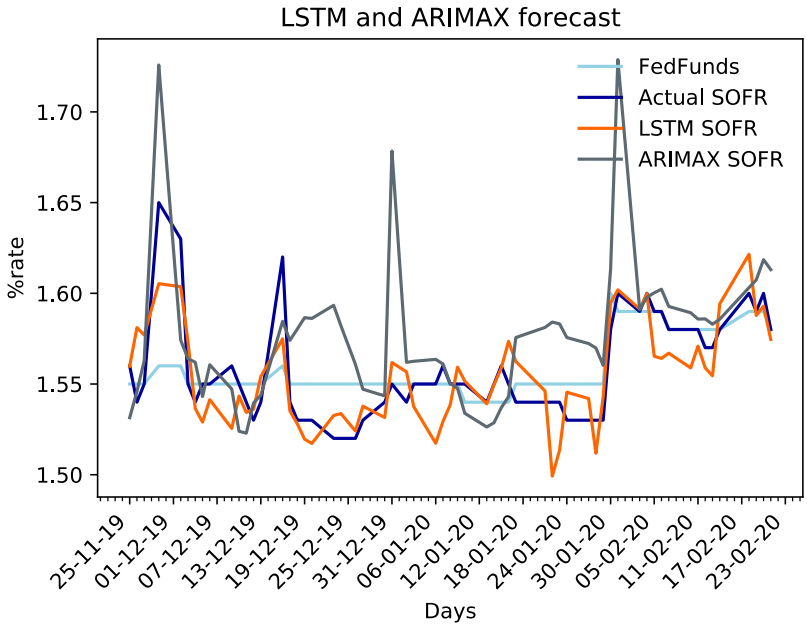


Figure 21: Experiment 1, spikes are recognized by LSTM and ARIMAX



The number of business days forecasted is:60
The following variables are used:
FedFundsRate
Date Dummy
Reverserepo
Test RMSE LSTM SOFR: 0.04366
Test MAPE LSTM SOFR: 0.01160
Test RMSE ARIMAX SOFR: 0.04763
Test MAPE ARIMAX SOFR: 0.01556

Figure 22: Experiment 2, spikes are recognized by LSTM and ARIMAX



The number of business days forecasted is:60
The following variables are used:
FedFundsRate
Date Dummy
Reverserepo
Test RMSE LSTM SOFR: 0.01870
Test MAPE LSTM SOFR: 0.00939
Test RMSE ARIMAX SOFR: 0.03806
Test MAPE ARIMAX SOFR: 0.01753

Figure 23: Experiment 3, spikes are forecasted correct by LSTM but overestimated by ARIMAX

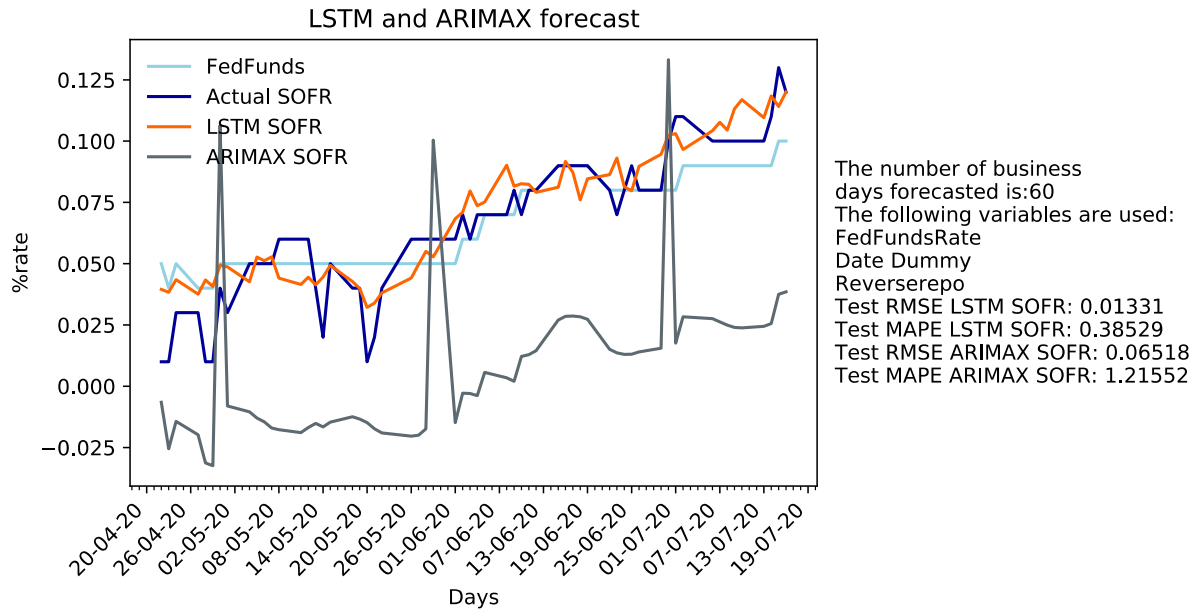


Figure 24: Experiment 4, LSTM model understands that there are no spikes anymore where ARIMAX is still expecting monthly spikes

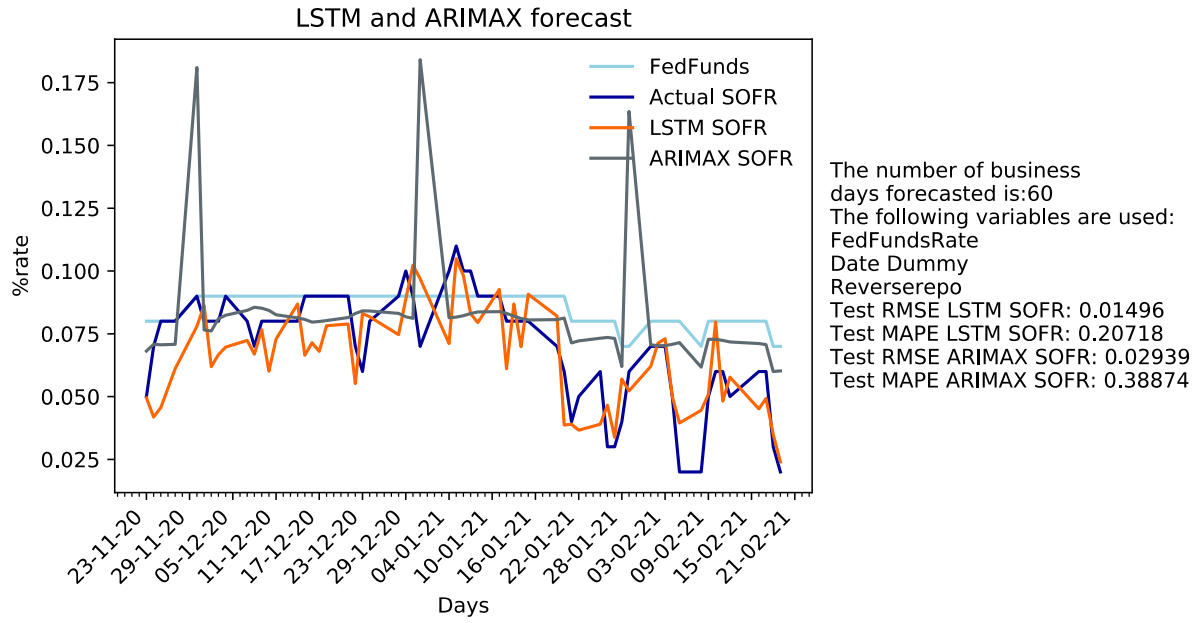


Figure 25: Experiment 5, LSTM model understand that there are no spikes anymore where ARIMAX is still expecting monthly spikes

6.3 Conclusion of results

From the experiments it can be concluded that the LSTM neural network outperforms the ARIMAX model by a reduction of 45% in RMSE. The LSTM neural network shows different results than the ARIMAX model. The LSTM model is good at handling changes in behavior of SOFR. The model is good at adapting to rates stabilizing or spikes disappearing. The LSTM nodes in the neural network seem to be useful in adapting to changes in the financial markets such as interventions from the federal reserve or a covid pandemic crashing the rates down to almost negative rates. The ARIMAX model however cannot handle the changes in the financial markets due to the OLS regression putting as much emphasize on data from long ago as from recent time periods. The long short-term memory nodes do their job in learning the long-term dependencies and having a 'short term' memory of the behavior of the model. This means that the relation the relations between the variables and lags are learned for long term where on short term the actual behavior and patterns of SOFR is stored in the LSTM nodes to always have an accurate forecast based on the financial situations at the moment. This short-term memory is what the ARIMAX model misses since it only models the long-term relation between the variables and assumes that these relations stay the same even if there is a crisis. That is why the LSTM model performs superior in changing environments which SOFR has shown in the past 2 years. In discussion section 7.2.1 a rolling ARIMAX regression is performed to try and help the ARIMAX model in recognizing a change in patterns. However, applying a rolling regression contradicts the assumption of a model picking up patterns in the whole dataset without intervention of the modeler. This is a clear advantage of a LSTM neural network that it does not need any help in determining what information in the dataset is relevant or not. In a rolling regression the modeler needs to determine what information is relevant or not instead of the model doing this.

For every experiment the Bayesian optimization algorithm is run and the best hyperparameter configuration is picked which is best suitable for the experiment. In Table 10 the optimal hyperparameters for each experiment are shown. It is hard to draw a conclusion based on the results of the Bayesian optimization. A number of four layers is most common, only experiment 2 uses significantly lower number of layers. The number of neurons per layer seems to be high in the beginning after which the number of neurons decreases with every layer. The learning rate differs between 0,001 and 0,01 where 0,001 is the standard value of the Adam optimizer defined in section 4.3.3. It would be expected that a higher learning rate results in a lower number of epochs since in each epoch the weights are updated with a bigger step size, however this conclusion cannot be drawn based upon the results of 5 experiments. The batch size of the model also ranges from 128 to 288 for which it can be expected that for lower batch sizes the number of epochs is also lower since in every epoch more batches are handled so more updates takes place however we also cannot conclude this from the results of 5 experiments.

Experiment	**Number of layers**	**Number of LSTM neurons per layer**	**Learning rate**	**Batch size**	**Epochs**
1	4	(1)320 (2)64 (3)64 (4)96	0.001	160	700
2	2	(1)192 (2)32	0.001	288	700
3	4	(1)288 (2)96 (3)160 (4)96	0.01	288	600
4	5	(1)512 (2)32 (3)224 (4)64 (5)64	0.01	224	400
5	4	(1)128 (2)64 (3)160 (4)256 (5)128	0.01	128	900

Table 10: Optimal hyperparameters for each experiment

6.4 Variability of setup

To guarantee that the optimal configuration of the neural network model gives consistent results we need to test the variability of the model. The variability of outcomes of the neural network due to the randomness in the optimization algorithm is evaluated, the model is run multiple times with the same hyperparameter configuration to ensure consistent results.

Experiment 2, starting at 22/04/2019 for 60 business days, is used to test the variability. The configuration of hyperparameters of experiment 2 are shown in Table 10. 2 layers are used with respectively 192 and 32 LSTM neurons per layer. A learning rate of 0,001, batch size of 288 and 700 epochs are used to train the model.

The neural network is trained and used to forecast for 20 iterations. For every iteration the RMSE and MAPE are stored which can be compared to test the variability of the results. In Table 11 the descriptive statistics of the variability tests are shown of the 20 runs of the LSTM model with the same configuration. The results show a mean RMSE of 0,0505 and a standard deviation of 0,00308.

	RMSE	***MAPE***
Mean	0.05051811	0.01407773
Standard Error	0.00068851	0.00021053
Median	0.05154439	0.01422050
Standard Deviation	0.00307909	0.00094150
Sample Variance	0.00000948	0.00000089
Kurtosis	-0.64665941	-0.96242725
Skewness	-0.73212470	-0.19403873
Minimum	0.04406880	0.01227939
Maximum	0.05414503	0.01552394
Count	20	20

Table 11: Descriptive statistics of variability test

To give more insight in the results of the 20 runs a box plot is made for the variability of the RMSE in Figure 26. We can conclude that the results of the model remain quite similar when more iterations are done and that there are only a few outliers. In the Appendix 4 all numerical results of the different repetitions and an extra box plot for the MAPE can be found. If we take a visual inspection of the results all repetitions recognize the spikes in SOFR, and all results look very similar without any big deviations.

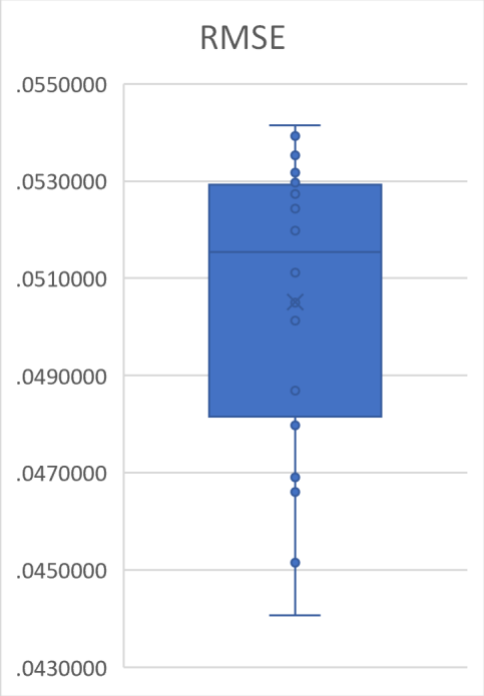


Figure 26: Box plot variability test (RMSE)

6.5 Conclusion

In this chapter we answered the fifth research question "*What is the performance of the neural network model on forecasting SOFR?"* and the sixth research question "*What is the variability in outcomes of the neural network model?"* The fifth research question is answered by running the experiments of chapter 5. We find in section 6.2 that the RMSE performance of the LSTM model is 0.02448 and that the LSTM model outperforms the ARIMAX by a reduction of 45% in RMSE. We can conclude that the LSTM RNN efficiently exploits its long-term memory for learning long term dependencies based on the whole dataset and employs its short-term memory to recognize structural breaks in the data. The LSTM RNN picks up a change in patterns in SOFR data without modelers intervention where the ARIMAX model is not able to recognize this, we can conclude that this is due to the intelligence of machine learning versus traditional statistical approaches like regression. To answer the sixth research question in section 6.4 a variability test is executed in which we find a mean RMSE of 0,0505 and a standard deviation of 0,00308. The box plot in Figure 26 shows low variability of outcomes without any outliers.

7 Conclusion, discussion and recommendations

In this chapter we will discuss the conclusion, discussion, and recommendations of this research. In section 7.1 we will draw the conclusion of this research based on the research questions. In section 7.2 the results of this research are discussed and reflected upon. In section 7.3 recommendations are given for future research.

7.1 Conclusion

In this section we will conclude all findings of this research and answer the research questions stated in the introduction. The main goal of this research is to evaluate the forecast performance of neural networks on SOFR and compare the performance to an autoregressive model. Which is also stated in the main research question of this research:

"What is the forecast performance of neural network models on SOFR, and how does this compare to an autoregressive model?"

More specifically the forecast performance of a long short-term memory recurrent neural network (LSTM RNN) on SOFR is evaluated and compared to an autoregressive integrated moving average model with exogenous variables (ARIMAX) in forecasting SOFR based on the root mean squared error (RMSE). The performance of the LSTM RNN model is expressed in the root mean squared error(RMSE) and has an average value of 0.02448 RMSE over all experiments. This means that the LSTM RNN model outperforms the ARIMAX by an average reduction of 45% in RMSE over all experiments. This result is found by answering the research questions throughout the research.

- *Research question 1: What factors influence the market that SOFR is based on and can be used as exogenous variables to forecast SOFR?*

SOFR is the interest rate financial institutions charge each other when entering a repurchase agreement. A repurchase agreement, or repo, is an agreement in which financial institutions agree to lend money to each other with treasury securities as collateral. SOFR is used as a reference for interest rates throughout the whole financial system. The federal reserve bank intervenes in the repo market through repo and reverse repo open market operations to steer the interest rate in the financial world. The following variables are picked as factors influencing SOFR and can be used as exogenous variables to forecast SOFR:

  - The Effective Federal Funds Rate
  - Volumes of the Repurchase and Reverse Repurchase Agreements by the New York Fed
  - The Chicago Board Options Exchange Volatility Index
  - The reserve balance of the federal reserve.
  - An end of the month date dummy
  - A FOMC date dummy
- *Research question 2: Which neural network model is best suited for financial time series forecasting?*

Based on literature, we find that a recurrent neural network is more suitable for sequence prediction than a feed forward neural network. However, a recurrent neural network has the problem of vanishing and exploding gradients when sequences become longer, to solve this problem long short-term memory nodes are added to the recurrent neural network. Long short-term memory networks are a type of recurrent neural network which uses its 'memory' to selectively store information on previous states of the data. This 'memory' is trained to selectively remember chosen parts of past data, to be used for predictions. Literature confirms that recurrent neural networks with long short-term memory nodes are the best network to forecast financial time series like SOFR.

- *Research question 3: How do we apply the chosen neural network models to SOFR?*

The recurrent neural network with long short-term memory nodes is applied to SOFR while making use of exogenous variables which are treated as given input variables to base the prediction of SOFR upon. The neural network is built in Python through the use of the libraries Keras and Tensorflow. When setting up a neural network hyperparameters need to be set to define the neural network. The following hyperparameters should be set before running the neural network: number of layers, number of neurons per layer, loss function, optimizer, learning rate, batch size and number of epochs.

- *Research question 4: How do we optimize the hyperparameters of the neural network model to get the best performance of the neural network on the defined experiments?*

To optimize the hyperparameters the Bayesian optimization algorithm is picked. Through the Bayesian optimization algorithm, combinations of hyperparameters are selected based on the belief of prior tried combinations. Through the Bayesian optimization algorithm, the search space of hyperparameters gets searched efficiently. A search space of hyperparameters and experimental setup is defined to test the performance of the neural network.

- *Research question 5: What is the performance of the neural network model on forecasting SOFR?*

The long short-term memory recurrent neural network (LSTM RNN) achieves an average root mean squared error (RMSE) of 0.02448. Comparing the performance of the LSTM model to the ARIMAX model a reduction of 45% in RMSE is achieved based on the experiments conducted. We can conclude that the LSTM RNN efficiently exploits its long-term memory for learning long-term dependencies based on the whole dataset and employs its short-term memory to recognize structural breaks in the data.

- *Research question 6: What is the variability in outcomes of the neural network model?*

To test the variability of the neural network one experiment is executed for 20 repetitions with the same hyperparameter configuration. In this variability test, we find that 20 repetitions have a mean RMSE of 0,0505 and a standard deviation of 0,00308.

Based on the results of the experiments we can conclude that the performance of the neural network model is better than the baseline ARIMAX model on all experiments. The LSTM model is good at handling changes in the behavior of SOFR. The LSTM nodes in the neural network seem to be useful in adapting to changes in the financial markets such as interventions from the federal reserve or a covid pandemic crashing the rates down to almost negative rates. The ARIMAX model however cannot handle the changes in the financial markets due to the OLS regression putting as much emphasis on data from long ago as from recent periods. The long short-term memory nodes do their job in learning the long-term dependencies and having a 'short term' memory of the behavior of the model. This means that the relation between the variables and lags are learned for long term where on short term the actual behavior and patterns of SOFR are stored in the LSTM nodes to always have an accurate forecast based on the financial situations at the moment. This smart allocation of long and short-term memory is what the ARIMAX model misses since it only models the long-term relation between the variables for a short period back in time and assumes that these relations stay the same even if there is a crisis. That is why the LSTM model shows a reduction in RMSE of about 45% for forecasting SOFR in changing environments defined in the experiments.

It can be concluded that a recurrent neural network with LSTM nodes is useful for sequence prediction of interest rates time series like SOFR. This is a step in discovering the applications of neural networks in the financial sector. We bring together a new reference rate, SOFR, and a not earlier used machine learning technique, neural networks, for forecasting SOFR and can conclude it is beneficial to use neural networks for forecasting SOFR compared to autoregressive models. Since in recent years more new reference rates like SOFR are introduced, it might be useful to apply neural network on forecasting these reference rates as well.

For forecasting SOFR it can be concluded that the effective federal funds rate, an end of the month date dummy and the volume of overnight reverse repurchase agreements are good predictors to forecast SOFR upon. The effective federal funds rate is a good indicator of the monetary policy of the federal reserve as SOFR moves up and down together with the federal funds rate. To give the model an indication of when to expect the spikes the end of the month date dummy can be used to help the neural network. The volume of the overnight reverse repo operations shows a similar spike pattern like SOFR thus is a good predictor of SOFR, however the causations between SOFR and the overnight reverse repos needs to be further studied to make sure this variable is a proper estimator of SOFR.

7.2 Discussion

Since there was no prior research on applying a neural network to SOFR this thesis had to explore if and how the application of a neural network to SOFR was possible. In this section we reflect and discuss the results of the research. Furthermore, recommendations for further research are given which could not be explored in this thesis since we only scratched the surface of applying neural networks to SOFR.

Inherent of using neural networks is the black-box principle of the algorithm. Neural networks are capable of replicating complex non-linear patterns through optimizing their numerous parameters. After training a neural network the modeler is left with a bunch of weights for the parameters of the neural network. These weights are then used to make a forecast and validate the performance of the neural network. However, it is not transparent how the weights of the neural network influence the forecast that is made. This makes it difficult to determine the significance of the different inputs that are used in a neural network. In a regression, coefficients and p-values are produced on which the modeler can evaluate if the relations that the regression makes are to be expected. This is something that is not possible for neural networks which makes it hard to test if the intuition of the results is correct.

Explainable Artificial Intelligence (XAI) is a research field which is aimed at increasing the transparency and explainability of artificial intelligence. Explainable AI can also be applied to neural networks and in literature two suggestions are given to improve explainability of especially recurrent neural networks (Barredo Arrieta, et al., 2020):

- *Explainability by understanding what a RNN model has learned mainly via feature relevance methods;*
- *Explainability by modifying RNN architectures to provide insights about the decisions they make.*

The first suggestion is aimed at finding what features are relevant meaning which input variables have the most influence on the output. In this suggestion the neural network itself is not changed while in the second suggestion the architecture of the neural network is changed to construct an understandable model. For example, a combination of a Markov Model and RNN is suggested which has the interpretability of a Markov Model but the accuracy of a RNN model. Another solution to enhance explainability might be using artificially created data to see how a neural network reacts to artificially created relations.

When experimenting with the chosen exogenous variables in turns out that the assumption that adding extra variables will improve the forecasting ability of a neural network does not hold. The neural network is not able to determine which variables it should base its forecasts on. This also emerges from the p-values of the exogenous variables in the ARIMAX OLS regression. The p-value of a variables resembles the significance of the exogenous variable to the ability to forecast SOFR. However, this ability is only tested in the linear case of a regression and not in the nonlinear neural network model it does give a good indication of which exogenous variables add something to the model and which don't.

When a neural network is trained multiple times on the same data it will produce different results. Even when all hyperparameters of the neural network are kept the same then still randomness in results will occur (Khandelwal, 2020). This can be due to multiple reasons are:

- Random initialization of weights and biases. Before training a neural network all weights and biases are initialized which affects the training of the model.
- Randomness in regularization methods like dropouts. A random number of nodes is randomly dropped out to prevent the model from overfitting.
- Randomness in the optimizer like Adam or stochastic gradient descent affects the outcome of the neural network.

All above reasons influence the variability of outcomes of a neural network. To reproduce experiments a random seed can be used to initialize the random number generator. Setting a seed value ensure the same random numbers are used for every experiment, resulting in the same results if the same input is used. Variability of a neural network is something that a linear regression model is not bothered by since a linear regression model always finds the same global optimum when fitting the model.

7.2.1 Improving the ARIMAX forecasts

The ARIMAX model is kept basic because as it has the purpose of functioning as a baseline model for reference of the neural network model. The main struggle of ARIMAX is dealing with a structural change in patterns cause by for instance the covid pandemic. Such a structural break in the data is something the ARIMAX model cannot handle. To try and help the ARIMAX model different methods can be applied, for example: a rolling regression or an exponentially weighted regression. Both regression variants are a way of putting more emphasis on recent observations rather than observations from a long time ago. In an exponentially weighted regression, the full dataset of all past observations is used as input for the model but to each observation is certain weight is given which determine how much influence the observation has on the output of the model. The weights being exponentially weighted means that means that more emphasis is given to more recent observations while observations from long ago are still considered but have a low impact on the outcome. A rolling regression does not make use of weights but of a rolling or gliding window. A rolling window means that the window on which the model is fitted, is always set to the same number of data points prior to starting the forecast. This results in a model which is only fitted on the number of most recent data points determined by the size of rolling window.

In an effort to improve the forecasts made by the ARIMAX model a rolling regression was performed for different time windows for all the experiments. For all experiments except experiment 5 the forecasts got worse than the basic ARIMAX forecasts. This is also expected because experiment 5 is the only experiment for which there is more data after the occurrence of a structural break. A rolling regression with a rolling window of 6 months before the first day forecasting showed an improved over the basic ARIMAX forecast. In Figure 27 and Figure 28 the visual results of the two rolling regression is shown with all variables or only with the significant variables used. This ARIMAX regression is trained on data from 23-5-2020 until 23-11-2020. This means that this training dataset only considers data since the covid pandemic, for which SOFR has not been showing any spikes at month. So, no spikes are expected by the ARIMAX model.

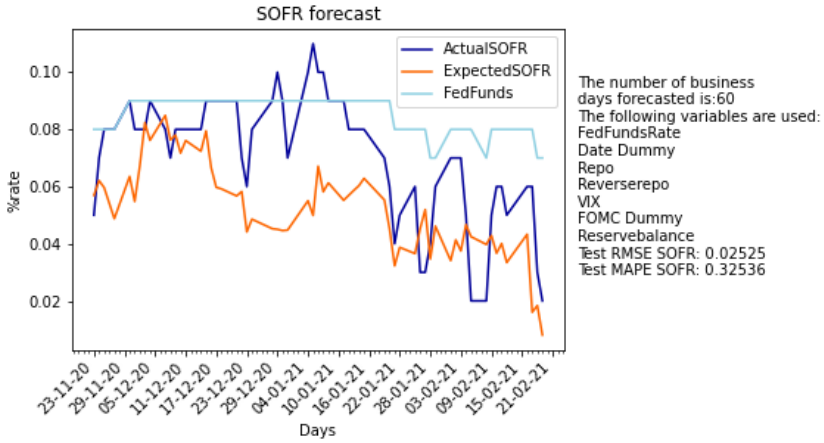


Figure 27: Rolling regression all variables experiment 5 (6 months)

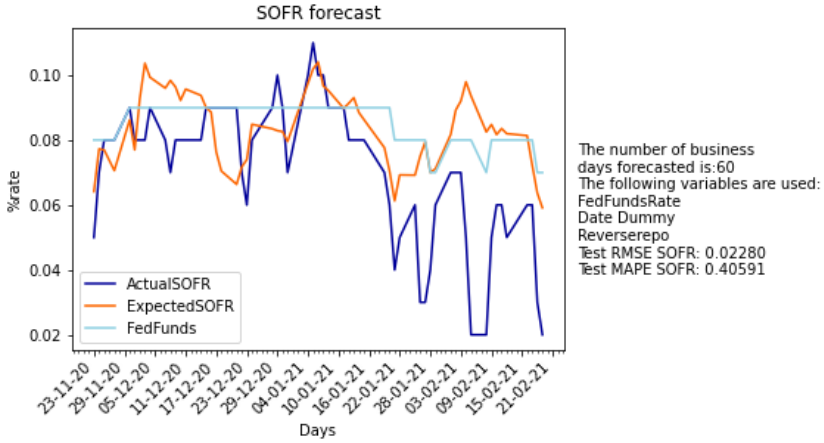


Figure 28: Rolling regression significant variables experiment 5 (6 months)

If we take a closer look at the results of Figure 27 and Figure 28 we see that the RMSE of the experiment with only significant variables is lower than the experiment with all variables, however the MAPE is higher for the experiment with only significant variables. Still both results show an improvement over the basic ARIMAX model which can be concluded from Table 12

Variables used	**LSTM RMSE**	**LSTM MAPE**	**Basic ARIMAX RMSE**	**Basic ARIMAX MAPE**	**Rolling ARIMAX RMSE**	**Rolling ARIMAX MAPE**
EFFR, date dummy, FOMC dummy, repo, reverse repo, VIX, reserve balance	0.01555	0.25839	0.04017	0.59621	0.02525	0.32536
EFFR, date dummy, reverse repo,	0.01496	0.20718	0.02939	0.38874	0.02280	0.40591

Table 12: Rolling regression results compared to LSTM and basic ARIMAX

The results in Table 12 show that an improvement in forecasts can be achieved in applying rolling regressions for the ARIMAX model. However enough data after the structural break should be available to train the model on. It should be noted that the rolling ARIMAX model still does not outperform the LSTM model which is trained on the full dataset so all data that is available is used where in a rolling regression only a part of the data is used.

Applying a rolling regression contradicts the assumption of a model picking up patterns in the whole dataset without intervention of the modeler. A LSTM neural network does not need any help in determining what information in the dataset is relevant or not. In a rolling regression the modeler

needs to actively interact and try different rolling windows to see what information is relevant or not instead of the model doing this for the modeler.

7.2.2 Dynamics behind SOFR

When forecasting SOFR by making use of exogenous variables the intuition of picking the Effective Federal Funds Rate and using a date dummy as exogenous variables in the neural network is quite straightforward however the significance of overnight reverse repurchase agreement (ON RRP) feels a little counterintuitive. For the effective federal funds rate and SOFR it there is a clear relation of moving up and down together based on the monetary policy by the Federal Reserve. However, the federal funds rate typically drops substantially at month-end and quarter-ends, volume in the federal funds and other overnight often falls sharply on these days while the volume at the Federal Reserve's Overnight Reverse Repo Facility often increases sharply (Schulhofer-Wohl & Clouse, 2018). To identify when it is the end of month and the spikes in SOFR occur, the date dummy is used to give the neural network an indication of which day special behavior occurs. As the spikes in SOFR clearly occur at the end of the month the date dummy indicates when the spike in SOFR happens.

The logic behind using the ON RRP is at first sight a little counterintuitive as a predictor of spikes since reverse repo open market operations are in theory executed when a drop in rates is expected and SOFR needs an upward push. This is because during an ON RRP securities are sold in a repurchase agreement to draw liquidity out of the market resulting in a lower supply of cash in de market thus pushing SOFR up. So, the expectation would be that ON RRP are executed in times of too much liquidity in the market and a dip is expected in SOFR. This would result in a negative correlation between SOFR and ON RRP however the opposite turns out to be true. The ON RRP turn out to be a good predictor of the spikes in SOFR. If we look at the behavior of the volume of the ON RRP it should be noted that a proper spikes pattern at month-end can be identified.

The overnight reverse repo operations are executed by a facility based on market demand for reverse repo operations. The facility provides the market with an offer of a rate and maximum amount at which institutions can enter into a reverse repo with the Federal Reserve (Nelson, 2021). So, the financial institutions are free to choose whether they want to make use of the facility offered by the Federal Reserve at a rate slightly lower than the federal funds rate and the IOER. The ON RRP facility is available to a wide range of lenders in the overnight market. Contrary to the IOER which is only available to depository Banks. The facility for overnight reverse repo transactions was established to ensure that overnight repo rates in the repo market would not trade below the lower bound of the federal funds target range since not all financial institutions or are entitled to receive the IOER on their excess money. So, if the Federal Reserve did not create the ON RRP facility these non-eligible IOER institutions get into a contract with other banks who do receive the IOER to get some return on excess reserves. This is also why more overnight reverse repo transactions are executed when the effective federal funds rate is below the IOER since the overnight reverse repo rate offered by the facility is generally some basis points lower than the IOER and federal funds rate. It should be noted that the IOER and federal funds rate are both agreements issued without a collateral where an ON RRP or transaction in the repo market are repurchase agreements which require a collateral. Collateral makes the agreements less 'risky' for the counter party thus a lower rate is asked.

If we take a look at the effective federal funds rate in the period from 2016 to 2018 we see that at the end of the month there is always a little dip in the effective federal funds rate. So, a dip in the federal funds rate implies a rise in SOFR. A possible explanation which of this phenomenon is that ON RRP are executed because there is an arbitrage opportunity in using ON RRP to profit from the drop in the federal funds rate which has the consequence of lowering liquidity in the repo market thus spiking SOFR. However, this need not be the case since causality can also be the other way around that a spike in SOFR causes a spike in the volume of ON RRP. The relation between the federal funds rate, ON RRP and SOFR is a new topic which has not been studied much in literature, only that some relation exists but not which movement causes the others to move as well.

Month-end and quarter-end changes in the federal funds markets and repo markets rates and volumes are short lived. However, they indicate that some regulatory policies are affecting the behavior of the overnight rates in the market. One indication might be the difference in leverage ratio requirements for domestic and foreign banks. As domestic banks leverage ratio requirement is based upon their average total assets maintained throughout a quarter where for foreign banks have a leverage requirement based on month-end or quarter-end assets. On month-ends and quarter-ends the foreign banks try to shrink the size of their balance sheet to reduce their required capital. Which might be an indication of the current behavior in the federal funds and repo market (Keating & Macchiavelli, 2017).

If we take the special case of September 17, 2019, when SOFR surged to a high of 525 basis points. This wasn't even an end-month date but about mid-month of September. This sudden spike in SOFR can be accounted to two developments in the market. First, quarterly tax payments were due for corporations and some individuals. These taxpayers scrambled for money in the money markets and sent their money to the U.S. Treasury. Secondly, the U.S. Treasury department increased its long-term debt by paying off maturing short-term securities while issuing a large quantity of long-term securities. These new securities were bought through money from bank and money market accounts resulting in less money in the system. So, a tax payment date combined with a debt issuance of U.S. treasury reduces the amount of cash in the financial system can cause SOFR to spike (Schulhofer-Wohl, 2019).

7.3 Recommendations for future research

Extra date dummies could be added to help the model in recognizing useful dates even better. These date dummies could be set on the middle of the month, end of quarter or end of year. In the middle of the month a spike can also be recognized in SOFR so the model can be helped by setting a date dummy on this day in the middle of the month. Furthermore, we see that at the end of a quarter and end of the year the spike is generally higher than at the end of the other months. So, it might be useful to help the model distinguish between regular end of the months and end of the quarter or end of the year.

We found a better result by not using all exogenous variables we defined on forehand. This was done by using the P-values of the ARIMAX regression however this could also be done differently. If more computation power would have been available we could have tried every different combination of exogenous variables to test which variables contribute the best to the forecast.

The loss function of the neural network that was used was the root mean squared error. To optimize this loss function for a spikey time series like SOFR a custom loss function could be created to better forecast the spikes of SOFR. When applying a custom loss function the loss at the end of the month could be artificially exaggerated to make the model more prone to successfully forecasting the spikes of SOFR.

Another approach to better forecast the spikes in SOFR could be to use some sort of regime switching model. This means that two separate models are trained, one to forecast the spikes and one to forecast the 'regular' days in between the spikes. This could improve performance of the model by having two models which can be trained differently for their purpose.

The federal reserve balance variable was added as some sort of threshold variable. If this variable is high it means that the liquidity injected by the federal reserve in the system is high. When the system is flooded with liquidity spikes in SOFR are less likely to occur. So once this variable goes over a threshold spikes should not occur anymore, which also makes the opposite true that if the reserve balance shrinks and the variable comes below some sort of threshold than the spikes should reappear. Further research could be done in helping the neural network recognize the threshold values.

The RNN LSTM model that is used only uses LSTM nodes in its architecture. So, between the input and output nodes only hidden layers containing LSTM nodes are used of variable sizes. In future research it might be useful to add dropout layers for regularization. Randomly dropping out nodes during

training is an effective regularization method to reduce the overfitting of a neural network which may improve the performance of the neural network.

Lastly no pattern could be found in the combination of hyperparameters. The combinations of hyperparameters that came out of the Bayesian optimization algorithm were used as optimal combinations of hyperparameters however no conclusion could be drawn from the different optimal combinations of hyperparameters for the different experiment. More experiments need to be run and evaluated to get better insights in the optimal number of layers, neurons in a single layer and epochs for instance. Understanding the relation between these hyperparameters could make it much easier in tuning the hyperparameters by hand and reducing computation time.

Bibliography

The Alternative Reference Rates Committee. (2019). *A User's Guide to SOFR.*

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning.* New York: Springer.

Agueci, P., Alkan, L., Copeland, A., Davis, I., Martin, A., Pingitore, K., . . . Rivas, T. (2014). *A Primer on the GCF Repo Service.* New York: Federal Reserve Bank of New York.

Algorithmia. (2018, May 7). *Introduction to optimizers*. Retrieved from Algorithmia: https://algorithmia.com/blog/introduction-to-optimizers

Alternative Reference Rates Committee. (2020). *SOFR Starter Kit Part I, II & III.*

Bandara, K., Bergmeir, C., & Hewamalage, H. (2020). *LSTM-MSNet: Leveraging Forecasts on Sets of Related Time Series with Multiple Seasonal Patterns.* Melbourna: Faculty of Information Technology, Monash University.

Barredo Arrieta, A., Diaz-Rodriguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., . . . Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion 58* , 82-115.

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 281-305.

Board of Governors of the Federal Reserve System. (2021, April 11). *Primary Market Corporate Credit Facility*. Retrieved from Board of Governors of the Federal Reserve System: https://www.federalreserve.gov/monetarypolicy/pmccf.htm

Brooks, C. (2014). *Introductory Econometrics for Finance.* New York: Cambridge University Press.

Brownlee, J. (2017). *Long Short-Term Memory Network With Python.* Machine Learning Mastery.

Camero, A., Wang, H., Alba, E., & Bäck, T. (2021). Bayesian neural architecture search using a training-free performance metric. *Aplied Soft Computing Journal 106 (2021)*, 1-11.

Cheng, J., & Wessel, D. (2020, January 28). *Brookings.* Retrieved from What is the repo market, and why does it matter?: https://www.brookings.edu/blog/up-front/2020/01/28/what-is-the-repo-market-and-why-does-it-matter/

Duffy, J., Ridley, D., Feng, J., & Patel, A. (2019). *LIBOR and the transition to SOFR: the multiple options to transition.* New York: White & Case.

Fan, H., Jian, M., Xu, L., Zhu, H., Cheng, J., & Jiang, J. (2020). Comparison of Long Short Term Memory Networks and the Hydrological Model in Runoff Simulation. *Water*, 1-15.

Federal Reserve. (2020, July 9). *Credit and Liquidity Programs and the Balance Sheet*. Retrieved from Board of Governors of the Federal Reserve System: https://www.federalreserve.gov/monetarypolicy/bst_fedsbalancesheet.htm

Federal Reserve Bank of New York. (2019, October 11). *Statement Regarding Treasury Bill Purchases and Repurchase Operations*. Retrieved from New York Fed: https://www.newyorkfed.org/markets/opolicy/operating_policy_191011

Federal Reserve Bank of New York. (2021a). *SOFR Averages and Index Data*. Retrieved from Federal Reserve Bank of New York: https://www.newyorkfed.org/markets/reference-rates/sofr-averages-and-index

Federal Reserve Bank of New York. (2021b, April 11). *Repo and Reverse Repo Agreements*. Retrieved from newyorkfed.org: https://www.newyorkfed.org/markets/domestic-market-operations/monetary-policy-implementation/repo-reverse-repo-agreements

Gellert, K., & Schlögl, E. (2019). *Dynamics of SOFR vs Fed Funds.* Sydney: University of Technology Sydney.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning* (Vol. http://www.deeplearningbook.org). MIT Press.

Guggenheim, B., & Schrimpf, A. (2020). *At the crossroads in the transition away from LIBOR: from overnight to term rates.* Bank for Internation Settlements.

Hansen, C. (2019, October 16). *Optimizers Explained - Adam, Momentum and Stochastic Gradient Descent*. Retrieved from Deep Learning: https://mlfromscratch.com/optimizers-explained/#/

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 1735-1780.

Imam, A. (2020, June 16). *Neural Networks (Part 1)*. Retrieved from Medium: https://medium.com/swlh/neural-networks-4b6f719f9d75

Katanforoosh, K., Kunin, D., & Ma, J. (2019). *Parameter optimization in neural networks*. Retrieved from deeplearning.ai: https://www.deeplearning.ai/ai-notes/optimization/

Keating, T., & Macchiavelli, M. (2017). *"Interest on Reserves and Arbitrage in Post-Crisis Money Markets," Finance and Economics Discussion Se- ries 2017-124.* Washington: Board of Governors of the Federal Reserve System.

Keras. (2021). *About Keras*. Retrieved from Keras: https://keras.io/about/

Khandelwal, R. (2020, January 31). *How to solve randomness in an artificial neural network?* Retrieved from towards data science: https://towardsdatascience.com/how-to-solve-randomness-in-an-artificial-neural-network-3befc4f27d45

Kingma, D. P., & Lei Ba, J. (2015). Adam: A Method For Stochastic Optimization. *International Conference for Learning Representations.* San Diego.

Moran, M. T., & Liu, B. (2020, May 15). *Whitepaper: The Vix Index and Volatility-Based Global Indexes and Trading Instruments.* Charlottesville: CFA Institute Research Foundation. Retrieved from Cboe: https://www.cboe.com/tradable_products/vix/faqs/

Namin, S. S., & Namin, A. S. (2018). *Forcasting Economic and Financial Time Series: ARIMA vs. LSTM.* Lubbock: Texas Tech University.

Nelson, B. (2021). *The Overnight Reverse Repurchase Agreement Facility.* Washington: Bank Policy Institute. Retrieved from Bank Policy Institute.

Pascanu, R., Gulcehre, C., Cho, K., & Begnio, Y. (2014). *How to Construct Deep Recurrent Neural Networks.* Montreal: University de Montreal.

Reed, R. D., & Marks, R. J. (1999). *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks.* Cambridge: The MIT Press.

Schulhofer-Wohl, S. (2019). *Chicago Fed Letter Understanding recent fluctuations in short-term interest rates.* Washington: The Federal Reserve Bank of Chicago.

Schulhofer-Wohl, S., & Clouse, J. (2018). *A Sequential Bargaining Model of the Fed Funds Market with Excess Reserves.* Chicago: Federal Reserve Bank of Chicago.

Stock, J., & Watson, M. (1998). *A comparison of linear and nonlinear univariate models for forecasting macroeconomic time series.* Cambridge, MA: National Bureau of Economic Research.

Sun, J. (2017, October 8). *Feedforward Neural Networks*. Retrieved from Sungtae's awesome homepage: https://www.cc.gatech.edu/~san37/post/dlhc-fnn/

Teräsvirta, T., van Dijk, D., & Medeiros, M. C. (2004). *Linear models, smooth transition autoregressions and neural networks for forecasting macroeconomic time series: A reexamination.* Rio de Janeiro: Departamento de Economia, Pontifícia Universidade Católica do Rio de Janeiro.

Thenmozhi, M. (2006). Forecasting Stock Index Returns Using Neural Networks. *Delhi Business Review, 7*(2), 59-69.

Verstyuk, S. (2020). *Modeling Multivariate Time Series in Economics: from Auto-Regressions to Recurrent Neural Networks.* Harvard University.

Wang, Z., Hutter, F., Zoghi, M., Matheson, D., & Freitas, N. d. (2016). *Bayesian Optimization in a Billion Dimensions via Random Embeddings.*

Zheng, K., Qian, B., Li, S., Xiao, Y., Zhuang, W., & Ma, Q. (2020). *Long-Short Term Echo State Network for Time Series Prediction.* Guangzhao: IEEE.

Appendix 1

Models for Using SOFR in arrears

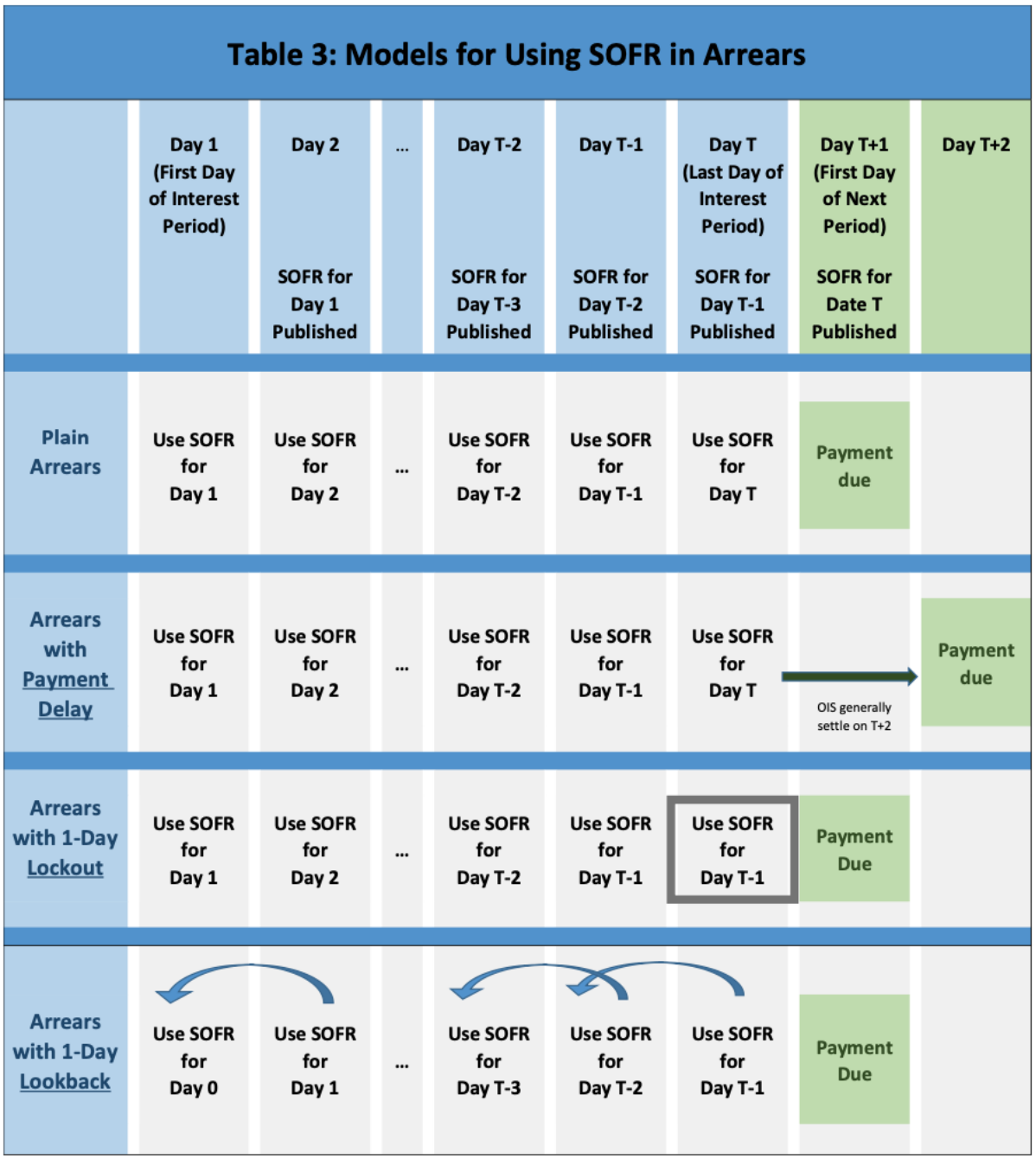


Figure 29: Models for using SOFR in Arrears (The Alternative Reference Rates Committee, 2019)

Appendix 2

Python code

- from keras.models import Sequential
- from keras.layers import Dense, LSTM
- model = Sequential()
- model.add(LSTM(neurons), stateful=True))
- model.compile(loss='mean_squared_error', optimizer='adam')
- model.fit(x_train, y_train, epochs=10, shuffle=False)

- model.predict(x_test, batch_size=128)

Appendix 3

Visual results of the RNN LSTM exogenous model and ARIMAX model containing all variables

Below the results are shown from the different experiments containing all variables.

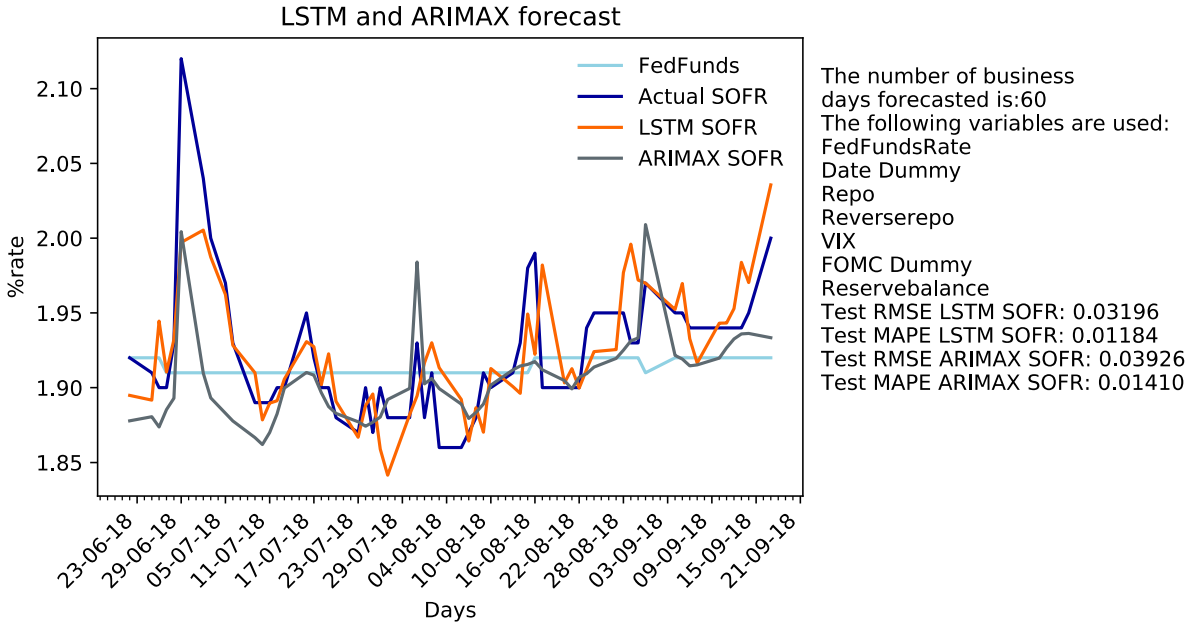


Figure 30: Experiment 1 all variables

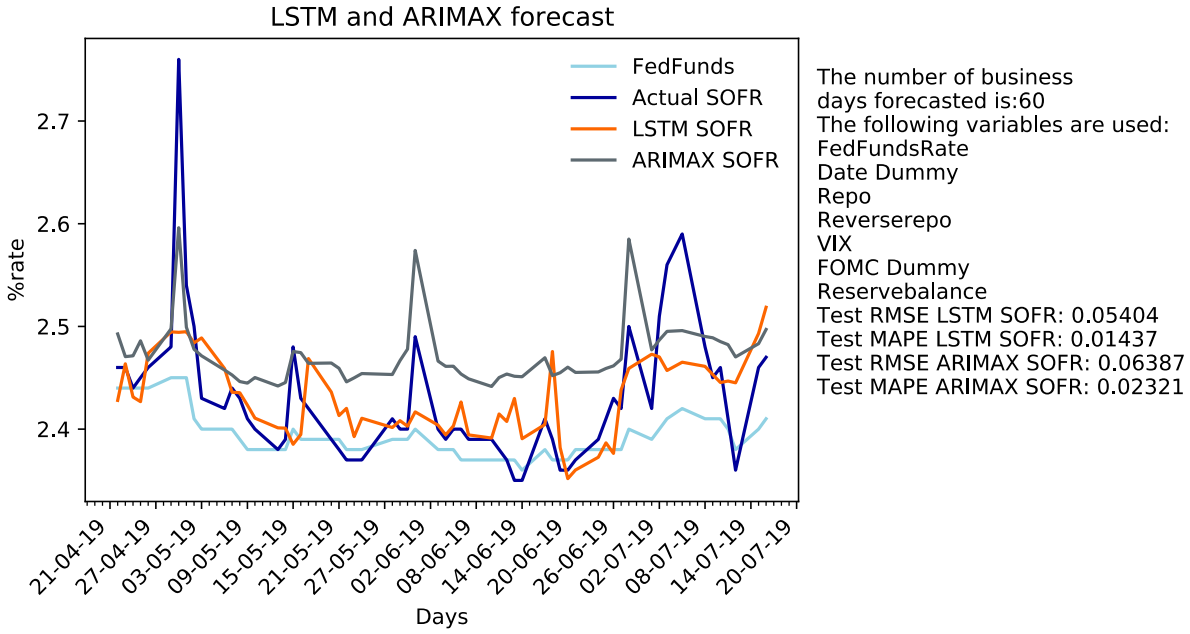
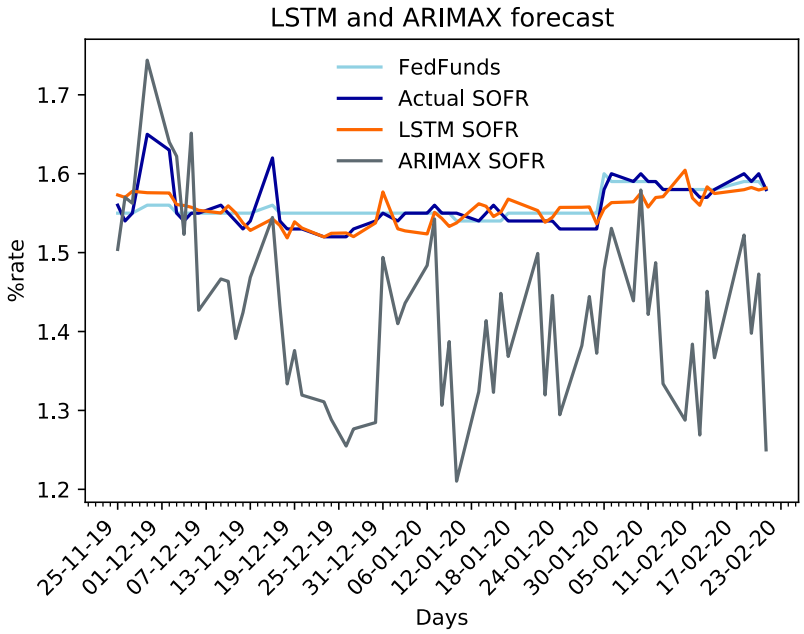
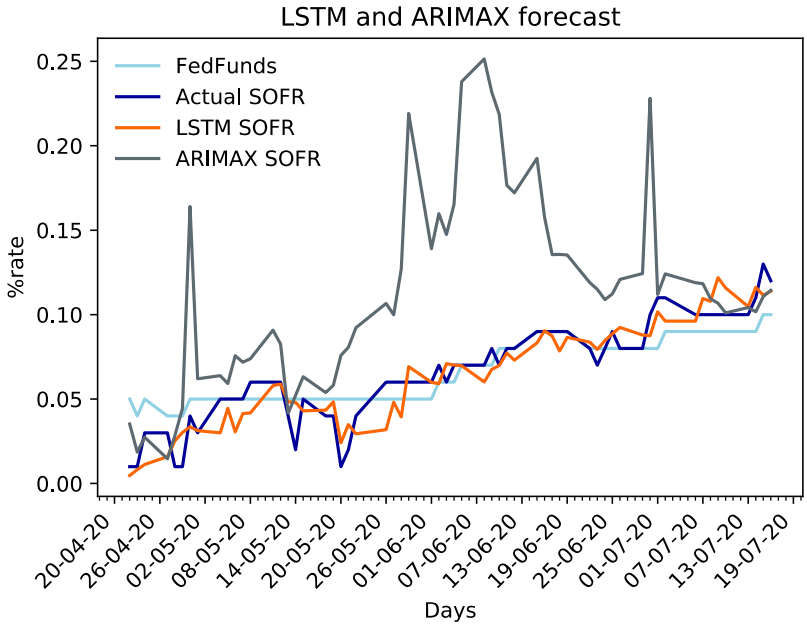


Figure 31: Experiment 2 all variables



The number of business days forecasted is:60
The following variables are used:
FedFundsRate
Date Dummy
Repo
Reverserepo
VIX
FOMC Dummy
Reservebalance
Test RMSE LSTM SOFR: 0.02273
Test MAPE LSTM SOFR: 0.01076
Test RMSE ARIMAX SOFR: 0.16676
Test MAPE ARIMAX SOFR: 0.09290

Figure 32: Experiment 3 all variables



The number of business days forecasted is:60
The following variables are used:
FedFundsRate
Date Dummy
Repo
Reverserepo
VIX
FOMC Dummy
Reservebalance
Test RMSE LSTM SOFR: 0.01175
Test MAPE LSTM SOFR: 0.25603
Test RMSE ARIMAX SOFR: 0.06718
Test MAPE ARIMAX SOFR: 1.00156

Figure 33: Experiment 4 all variables

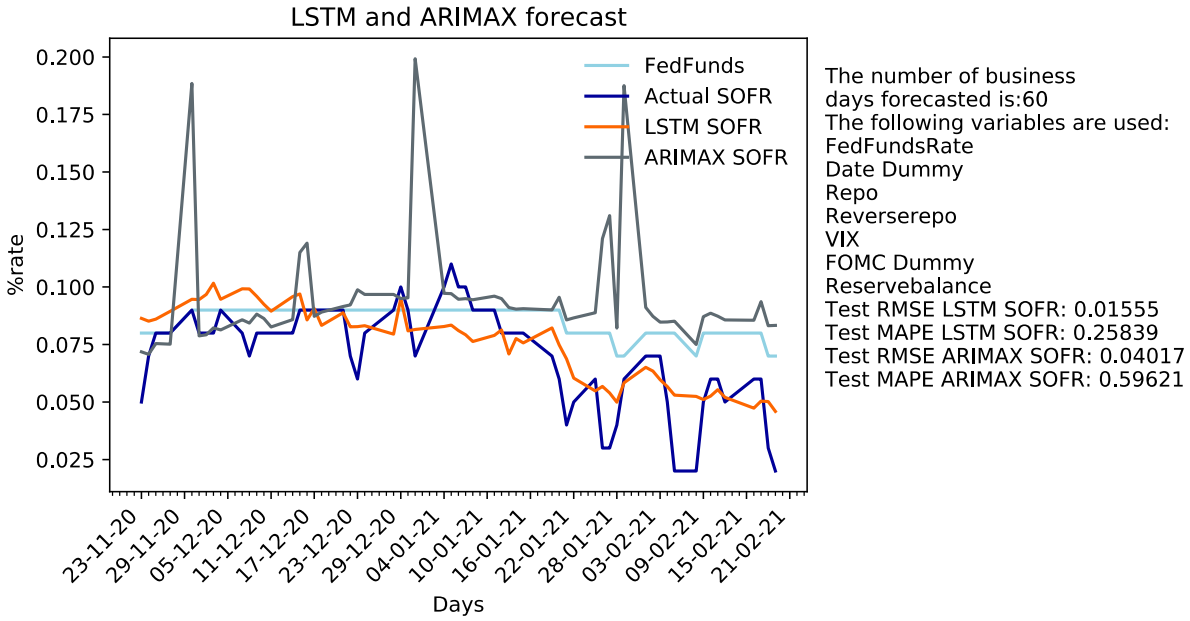
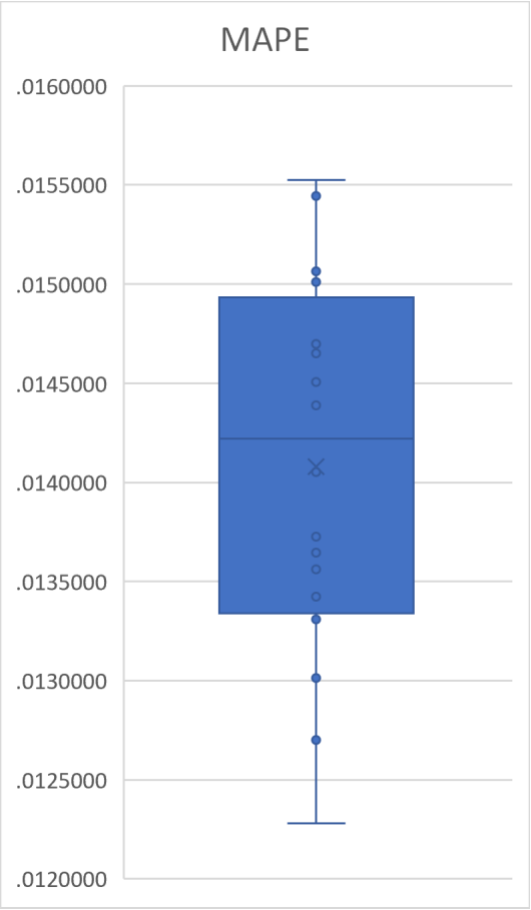


Figure 34: Experiment 5 all variables

Appendix 4

Results for variability

Experiment	**RMSE**	**MAPE**
1	0,05282871	0,01441816
2	0,04868888	0,01372678
3	0,05242677	0,01506575
4	0,05317019	0,01438970
5	0,05392153	0,01501141
6	0,04797048	0,01304863
7	0,04690760	0,01356257
8	0,04875492	0,01405130
9	0,05277183	0,01450732
10	0,05273544	0,01544571
11	0,05111427	0,01342501
12	0,05296460	0,01508070
13	0,04406880	0,01227939
14	0,04514945	0,01270047
15	0,05414503	0,01469628
16	0,05012866	0,01331032
17	0,04660560	0,01364620
18	0,05352933	0,01552394
19	0,05050551	0,01301266
20	0,05197451	0,01465229


MAPE
.0160000
.0155000
.0150000
.0145000
.0140000
.0135000
.0130000
.0125000
.0120000