UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE CIÊNCIAS SOCIAIS APLICADAS
DEPARTAMENTO DE ECONOMIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ECONOMIA

RONY ERIK MIKAEL DA FRANCA LEPPÄNEN

PREDICTING PAYMENT FLOWS WITH DEEP LEARNING - Case Study of BNDES Credit Card

Recife

2019

RONY ERIK MIKAEL DA FRANCA LEPPÄNEN

PREDICTING PAYMENT FLOWS WITH DEEP LEARNING - Case Study of BNDES Credit Card

Dissertação apresentada ao Programa de Pós-Graduação em Economia da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Economia.

Área de concentração: Teoria Econômica.

Orientador: Profº. Dr. Francisco de Sousa Ramos.
Coorientador: Profº. Dr. José Lamartine Távora Junior.

Recife

2019

Catalogação na Fonte
Bibliotecária Ângela de Fátima Correia Simões, CRB4-773

L592p Leppanen, Rony Erik Mikael da Franca
Predicting payment flows with deep learning: case study of BNDES credit card / Rony Erik Mikael da Franca Leppanen. - 2019.
101 folhas: il. 30 cm.

Orientador: Prof. Dr. Francisco de Sousa Ramos e Coorientador Prof. Dr. José Lamartine Távora Junior.
Dissertação (Mestrado em Economia) – Universidade Federal de Pernambuco, CCSA, 2019.
Inclui referências e apêndices.

1. Séries Temporais financeiras. 2. Macroeconomia. 3. Econometria. I. Ramos, Francisco de Sousa (Orientador). II. Távora Júnior, José Lamartine (Coorientador). III. Título.

330 CDD (22. ed.) UFPE (CSA 2019 – 095)

RONY ERIK MIKAEL DA FRANCA LEPPÄNEN

PREDICTING PAYMENT FLOWS WITH DEEP LEARNING - Case Study of BNDES Credit Card

Dissertação apresentada ao Programa de Pós-Graduação em Economia da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Economia.

Aprovada em: 30/08/2019.

ACKNOWLEDGEMENTS

Here I would like to express my gratitude to Alessandra Sleman Cardoso (Gerente da ADIG/DEPOD/GEROP – Gerência Operacional, do DEPOD – Departamento de Plataformas Digitais, da ADIG – Área de Operações e Canais Digitais, do BNDES – Banco Nacional de Desenvolvimento Econômico e Social – Brazilian Development Bank) for providing the dataset for the empirical study.

RESUMO

A previsão de eventos futuros pode ajudar organizações e empresas a tomar decisões mais informadas, levando a resultados mais desejáveis em termos de alinhamento estratégico. Entretanto, a previsão de séries temporais financeiras e econômicas é uma tarefa desafiadora devido a dependências não lineares e propriedades não estacionárias que os dados do mundo real geralmente exibem, entre outras questões. Com dados transacionais, como pagamentos, esses problemas se tornam mais aparentes e relevantes. Esta tese busca encontrar evidências de que a modelagem linear é suficiente para entender as complexidades que os dados financeiros do mundo real oferecem, comparando seu desempenho a modelos mais complexos por meio de um estudo de caso de previsão. Como os dados empíricos consistem em pagamentos provenientes do uso do cartão de crédito do BNDES, sendo de natureza transacional, as interdependências estruturais e a dinâmica do mercado regional no Brasil também são discutidas por meio de estatísticas e ilustrações descritivas. As principais contribuições desta tese incluem o desenvolvimento e a implementação de uma metodologia de previsão adequada para métodos baseados em aprendizagem profunda com entradas uni e multivariadas derivadas de dados de pagamento. A competitividade dos métodos de aprendizagem profunda é verificada através de várias métricas e testes estatísticos. Além disso, a literatura sobre a avaliação de impacto regional do BNDES no Brasil é ampliada, e a aplicação empírica dos dados do cartão de crédito do BNDES é a primeira de seu tipo.

Palavras-chaves: Séries temporais financeiras, macroeconomia, econometria, aprendizagem de máquina, aprendizagem profunda, computação inteligente, pagamentos

ABSTRACT

The prediction of upcoming events can aid organisations and enterprises in making more informed decisions resulting in more desirable outcomes in terms of strategic alignment. However, financial and economic time series forecasting is a challenging task due to nonlinear dependencies and nonstationary property that real-world data commonly exhibits, among other issues. With transactional data such as payments, these issues become more apparent and relevant. This thesis seeks to find evidence whether linear modelling is sufficient for understanding the complexities that real-world financial data offers by comparing its performance to more complex models via a forecasting case study. Since the empirical data consists of payments from the BNDES Credit Card usage thus being transactional in nature, underlying interdependencies and regional market dynamics in Brazil are also discussed through descriptive statistics and illustrations. The main contributions of this thesis include developing and implementing a forecasting methodology that suits for deep learning-based methods with uni- and multivariate inputs derived from payment data. The competitiveness of deep learning methods is verified via various metrics and statistical testing. Furthermore, the literature on evaluating the regional influence of BNDES in Brazil is expanded, and the empirical application to the BNDES Credit Card data is the first of its kind.

Keywords: Time series forecasting, macroeconomics, econometrics, machine learning, deep learning, intelligent computing, payments

LIST OF ILLUSTRATIONS

LIST OF TABLES

GLOSSARY OF TERMS AND ABBREVIATIONS

AdaBoost	Adaptive Boosting
ADF	Augmented Dickey-Fuller
AE	Autoencoder
ANN	Artificial Neural Network
AR	Autoregressive
ARCH-LM	ARCH-Lagrange Multiplier
ARIMA	Autoregressive Integrated Moving Average
BaH	Buy & Hold -strategy
BCB	Banco Central do Brasil (Central Bank of Brazil)
BNDES	Banco Nacional de Desenvolvimento Econômico e Social (Brazilian Development Bank)
BPTT	Backpropagation Through Time
BVAR	Bayesian VAR
CCP	Central Counterparty
CNN	Convolutional Neural Network
COMPASS	Central Organising Model for Projection Analysis & Scenario Simulation
CSD	Central Securities Depository
CSV	Common Stochastic Volatility
DM	Diebold & Mariano
DNN	Deep Neural Network
DNS	Deferred Net Settlement
DS	Directional Symmetry
DSGE	Dynamic Stochastic General Equilibrium
DTW	Dynamic Time Warping
EGARCH	Exponential GARCH
ELM	Extreme Learning Machine
ETF	Exchange-Traded Fund
EWMA	Exponential Weighted Moving Average
FMI	Financial Market Infrastructure
FMSE	Forecast Mean Squared Error
GARCH	Generalized Autoregressive Conditional Heteroskedasticity

GAS	Generalized Autoregressive Score
GCN	Graph Convolutional Neural Network
GNN	Graph Neural Network
GOP	Generalized Operational Perceptron
GRU	Gated Recurrent Unit
HMAE	Heteroskedasticity-adjusted MAE
HMSE	Heteroskedasticity-adjusted MSE
ISE	Indicador de Seguimiento a la Economia (Colombian Economic Monitoring Index)
KOSPI	Korea Composite Stock Price Index
KPSS	Kwiatkowski–Phillips–Schmidt–Shin
LOB	Limit Order Book
LOG	Logistic Regression Classifier
LSTM	Long Short-Term Memory
LVPS	Large-Value Payment System
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MCMC	Markov-Chain Monte-Carlo
MDA	Mean Directional Accuracy
MI	Mutual Information
MIDAS	Mixed Data Sampling
MLP/FFNN	Multilayer Perceptron/Feed-Forward Neural Network
MSE	Mean Squared Error
MSFE	Mean Squared Forecast Error
NARX-ANN	Non-Linear-Autoregressive-Exogenous-ANN
NG	Normal-Gamma
NMSE	Normalized Mean Squared Error
PCA	Principal Component Analysis
PP	Phillips-Perron
RAF	Random Forest
RBM	Restricted Boltzmann Machine
RCLSTM	Random Connectivity LSTM
ReLU	Rectified Linear Unit
RGB	Red-Green-Blue

RMSE	Root Mean Squared Error
RMSFE	Root Mean Squared Forecast Error
RNN	Recurrent Neural Network
RSI	Relative Strength Index
RTGS	Real-Time Gross Settlement
RV	Realised Volatility
SMA	Simple Moving Average
SSS	Securities Settlement System
STR	Sistema de Transferência de Reservas (Reserves Transfer System)
SVR	Support Vector Regression
TR	Trade Repository
VAR	Vector Autoregressive
WS	Wilcoxon Signed Rank

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Presentation

Forecasting financial and economical time-series' is not a straightforward task due to the various properties that real-world data exhibits. The data belonging to different markets are rarely independent from each other and often the underlying distribution of data generating processes cannot be anticipated or assumed as static. By nature, many of the real-world data streams progressing on time possess noisy, erratic, dynamic and chaotic characteristics leading to nonlinearities and unstable complexity (TRAN et al. (2019) and SUN et al. (2018)). In particular, real-world financial and economic systems may experience unexpected events related to political issues and government interventions, movements of macroeconomic indices and changing trends in high-value commodities, financial news and reports as well as changing sentiments of market participants, irregular activities in critical infrastructures, societies and nature, to name a few (LÄNGKVIST et al. (2014) and NASSIRTOUSSI et al. (2014)). As such, a relatively small event or change in one data source can have a significant impact to other ongoing processes due to multiplicative effects making it difficult to describe the output of these systems proportional to its inputs. However, because of their relative ease on solvability and interpretability, research across a range of disciplines, including finance and economics, has been dedicated to finding linear approximations of nonlinear phenomena[1]. These factors raise the question, which is currently under debate in the scientific community, of how one should approach these systems when managing them (CAVALCANTE et al. (2016)).

1.2 Objective

The main motivation of this work is to empirically shed light to the capabilities of current state-of-the-art in intelligent computing when evaluated against traditionally used models from econometrics and macroeconomic forecasting, via a preliminary financial time series prediction case study. The purpose of the study is to gather evidence whether linear models are sufficient for understanding, incorporating and leveraging as much of the

[1] Autoregressive Integrated Moving Average (ARIMA) for example assumes linearity thus excluding more complex joint distributions and focuses on fixed temporal dependencies having the need of specifying the number of lags.

complexities that real-world financial data offers, and also if the use of more complex models offers any further benefits. In addition to the forecasting exercise, descriptive statistics are also provided with brief insights on participant dynamics since payment data is used in the case study, or in other words, the data is transactional in nature.

Contributions of the thesis include a literature review of recent works in macroeconomic and deep learning-based forecasting, and a successful development and implementation of methodology for making predictions with uni- and multivariate inputs derived from payment data, which can be applied to other transactional scenarios. Evidence supporting the competitiveness of deep learning methods are provided through various metrics and statistical testing. This thesis expands the literature on evaluating the regional influence of BNDES in Brazil by inspecting data from BNDES Credit Card usage[2], and also provides a first attempt in forecasting aggregated payment flows using deep learning.

1.3 Structure of the thesis

From now on, the thesis unfolds as follows. FMIs (Financial Market Infrastructure) are introduced and their relationship to the Brazilian Development Bank (BNDES) which provided the data used in the experiments is established. Then a literature review based on recent works in the field of macroeconomic forecasting, deep learning and graph-based deep learning is provided. Next, the deep learning methods used in the experiments are briefly discussed following a detailed description of the methodology carried out. As the forecasting exercise is realised, an assessment on the results is provided, before which descriptive statistics and interdependencies in payment data are explored. Finally, some limitations of the study and suggestions for future works are outlined. Concluding remarks wrap up the thesis.

[2] See e.g. CORSEUIL (2019) and BNDES (2018a), as well as the BNDES digital repository of research https://web.bndes.gov.br/bib/jspui/?locale=pt_BR.

2 FINANCIAL MARKET INFRASTRUCTURES AND BNDES

FMI is a multilateral system among participating institutions, including the operator of the system. It is used for clearing, settling, or recording monetary transactions such as transfer of funds, securities, derivatives, or other financial transactions. FMIs are generally sophisticated systems that handle substantial amounts of transactions and sizable monetary values. They can differ significantly in organisation, function and design - FMIs can be legally organised in a variety of forms, including associations of financial institutions, non-bank clearing corporations, and specialised banking organisations. They may be owned and operated by a central bank or by the private sector, operate as for-profit or non-profit entities. Bank and non-bank FMIs can be subject to different licensing and regulatory schemes depending on organisational form. Nevertheless, FMIs typically establish a set of common rules and procedures for all participants, a technical infrastructure, and a specialised risk-management framework managing the risks they incur. Also, through the centralisation of specific activities, FMIs allow participants to manage their risks more efficiently and effectively, and in some instances, reduce or eliminate certain risks. FMIs can also promote increased transparency in particular markets and some of them are critical to helping central banks conduct monetary policy and maintain financial stability (CPSS-TCIOSC (2012)).

The term FMI refers to payment systems, central securities depositories (CSDs), securities settlement systems (SSSs), central counterparties (CCPs), and trade repositories (TRs)[3]. These infrastructures facilitate the clearing, settlement and recording of monetary transactions. There can be significant variation in design among FMIs with the same function: some FMIs settle in real-time while others may use deferred settlements, or settle individual transactions while others settle batches of transactions. The presumption is that all CSDs, SSSs, CCPs and TRs are systematically important. In general, a payment system is systemically important if it has the potential to trigger or transmit systemic disruptions. These may be systems that are the sole payment system in a country or the principal system in terms of the aggregate value of payments. They can also be, among other things, systems that mainly handle time-critical, high-value payments or settle payments used to effect settlement in other systemically important FMIs (CPSS-TCIOSC (2012)).

A payment system is a set of instruments, procedures and rules for transferring funds between or among participants. Payment systems are typically based on an agreement

[3] See CPSS (2016) for the definition of each FMI. In some cases, exchanges or other market infrastructures may own or operate entities or functions that perform centralised clearing and settlement processes.

between or among participants and the operator of the arrangement, and the transfer of funds is executed using an agreed-upon operational infrastructure. A payment system is generally categorised as either a retail payment system or a LVPS (Large-Value Payment System). A retail payment system is a funds transfer system that typically handles a large volume of relatively low-value payments in forms such as cheques, credit transfers, direct debits, and card payment transactions. Retail payment systems may be operated either by the private sector or the public sector, using multilateral DNS (Deferred Net Settlement) or RTGS (Real-Time Gross Settlement) mechanisms. LVPS is a funds transfer system that typically handles large-value and high priority payments. In contrast to retail systems, many LVPSs are operated by central banks, using an RTGS or equivalent mechanism (CPSS-TCIOSC (2012)). While the processing of large-value payments is not a sufficient condition for a system to be considered systemically important, systems handling primarily large-value payments are usually considered as systemically important since they are vulnerable to events that may threaten the stability of a financial system as a whole (CPSS (2005)).

Before the mid-1990s, when Brazil was battling with chronic inflation of up to 2% per month, the changes in the Brazilian payment, clearing and settlement systems aimed at increasing the speed of processing of financial transactions. Further on, the focus shifted to risk management in the reform carried out by the Central Bank of Brazil (BCB) in 2001 and 2002: the BCB implemented a system called the STR (Reserves Transfer System) that uses RTGS for funds transfers (CPSS (2011)). Today, the central-bank controlled STR serves as the backbone of the Brazilian financial system. STR is the Brazilian LVPS which settles transactions in the monetary, foreign exchange and capital markets among institutions that hold accounts at BCB. The clearing and settlement systems' operations are also settled through the STR as well as funds transfers related to the collection of income taxes and payments for the federal government (BCB (2011)).

There are three different types of accounts in STR: i) Bank Reserves Accounts held by banking institutions; ii) Settlement Accounts held by non-banking institutions, payment institutions and FMIs such as clearing and settlement providers; and iii) Government Single/Unique Account held by the National Treasury. When considering Bank Reserves Accounts, holding is mandatory for commercial and universal banks with commercial bank activities and savings banks, and optional for development, investment, foreign exchange and universal banks without commercial bank activities. Regarding Settlement Accounts, holding is mandatory for systemically important FMIs such as clearing and settlement providers and optional for non-systemically important FMIs, non-banking institutions authorised to operate by BCB and payment institutions (BCB (2011)). Participants in STR can execute funds transfers of unlimited value on their own or a customer's behalf for credit to the account of another participant or its customer. As of 23/8/2019, there are 233 participants in total in STR and one of them is BNDES, holding a reserves account in BCB since 30/6/2014 (BNDES (2019) and BCB (2019)). BNDES[4] is the main financing institution for development in Brazil.

Micro and small sized companies play a significant role in the Brazilian economy and employment. In 2016, there were 6.8 million establishments corresponding 99% of all firms. These accounted for 16.9 million (54.5%) of all formal non-agricultural jobs while being responsible for 44% of all salaries paid. Between 2006 and 2016, micro and small sized companies generated in total of 5 million jobs as the number of establishments increased 21.9%. More than one fourth of the Brazilian GDP was generated by micro and small sized enterprises in 2011 (SEBRAE (2014)). Moreover, when compared to medium and large sized companies, the micro and small companies are more resilient to shocks such as crises (SEBRAE (2016)).

Despite their importance for economic growth, micro and small sized companies have had difficulties in accessing financial support in the form of credit which may contribute to the high rate of defaults among these companies (MACHADO et al. (2011)). To finance their business, entrepreneurs often seek other sources than banks due to the cost of money, bureaucracy and the excessive requirement of guarantees. As an alternative to banks, most entrepreneurs negotiate payments with suppliers or use post-dated checks, overdrafts or corporate credit cards (BNDES (2018b)). To this end, BNDES has made efforts in overcoming the challenges faced by micro and small sized companies regarding financial support thus contributing to the economic and social growth in Brazil.

BNDES offers various elements of financing through which enterprises of different sizes can access financial support. Some of the principal elements include products[5] which define general funding rules according to each purpose with specific clients, objectives and conditions. For example, Finame is designed to help acquiring, producing and modernising machines and equipment, and micro-sized entrepreneurs, who may or may not have access to the traditional financial system, can opt for small loans through Microcredit.

[4] BNDES is the Brazilian development bank whose mission is to foster sustainable and competitive development in the Brazilian economy as well as generate jobs while reducing social and regional inequalities.
[5] More information about these products can be found from BNDES' website (in Portuguese), https://www.bndes.gov.br/wps/portal/site/home/financiamento.

The BNDES Credit Card is a product launched in 2002 focusing on acquisition of goods and services targeted for micro, small and medium sized enterprises (MSEs). More specifically, it is a line of pre-approved credit based on the idea of providing a credit card to companies similar to what individuals have, which initially represented a paradigm shift in offering financial support for smaller enterprises. Instead of acting with MSEs through a network of commercial banks spread over the country, the Credit Card automates transaction procedures electronically removing the need of case-by-case analysis and the products (for investing) are conveniently made available in one marketplace[6] (CORSEUIL et al. (2019)). In essence, the Credit Card brings together suppliers and buyers from various industrial sectors by facilitating investments in products such as equipment and machines used in production, commerce and services. Here, BNDES provides support offering long-term payment plans, lower credit tax rates, guarantees and security, among others[7].

The data for the experiments in this thesis was provided by the BNDES and contains operations from the BNDES Credit Card[8] usage. The data is reviewed in detail in Section 5.1 and Chapter 6.

[6] The portal, or e-catalog, where investments can be made, is available in BNDES Credit Card's website for users with proper credentials (https://www.cartaobndes.gov.br/cartaobndes/).

[7] The tax rate is fixed and updated monthly by BNDES. In August 2019, the tax rate was 1.21%. For more information, see https://www.cartaobndes.gov.br/cartaobndes/PaginasCartao/Taxa.asp?Acao=L. Payment plans are available from 3 to 48 months. For each client, pre-approved credit can be acquired as far as 2 million reals.

[8] I would like to thank Alessandra Sleman Cardoso (Gerente da ADIG/DEPOD/GEROP – Gerência Operacional, do DEPOD – Departamento de Plataformas Digitais, da ADIG – Área de Operações e Canais Digitais, do BNDES – Banco Nacional de Desenvolvimento Econômico e Social – Brazilian Development Bank) for providing the dataset.

3 LITERATURE REVIEW AND RECENT WORKS

3.1 Macroeconomic forecasting

Before reviewing machine and deep learning literature, a summary of recent works in the field of macroeconomic forecasting is provided. Lately, studies in macroeconomic forecasting have been focusing on investigating whether including information from different markets and incorporating innovations leads to enhanced predictive ability. Especially the development of reduced-form models has been accelerating and produced many novel approaches. Moreover, a strand of literature focuses on improving Bayesian VAR (BVAR) methods and their variants, as they are standard tools in macroeconomic forecasting and policy analysis for practitioners such as central bankers and scientific community.

Carriero et al. (2019) provided a comprehensive evaluation on the current state-of-the-art in both reduced-form and full structural modelling. The authors computed forecasts for output growth and inflation for seven different economies using three classes of latest reduced-form forecasting models and the performance of a medium-sized structural DSGE[9] (Dynamic Stochastic General Equilibrium) was compared to the reduced-form models in the US, the UK and the euro area. The RMSE (Root Mean Squared Error) and logscores[10] were used for measuring the models' relative forecasting performances and DM-test (Diebold & Mariano) was used for verifying the statistical significance of forecasts. The study used datasets up to 155 variables[11] including a mix of measures of economic activity such as survey data, prices and financial variables of different frequencies. In another work, Domit et al. (2019) delivered a comparison of a medium-scale BVAR and the DSGE model COMPASS (Central Organising Model for Projection Analysis & Scenario Simulation) developed by the Bank of England, outlined in Burgess et al. (2013). Both models were estimated with UK economy data and quarterly forecasts for horizons of up to three years were produced. The forecasting performance of the models was evaluated with RMSFE (Root Mean Squared Forecast Error) and the statistical difference of RMSFEs was analysed with DM-test.

[9] The employed DSGE followed the Smets-Wouters' approach (SMETS and WOUTERS (2007)).
[10] The authors argued that the maximization of the logscore is equivalent to the minimisation of the Kullback-Leibler distance between the model and the true density.
[11] Time-varying VARs and VARs with stochastic volatility were excluded from the study due to their adaptability to large datasets.

Due to the availability of more granular financial data, Knotek II and Zaman (2019) made forecasts of macroeconomic variables with multivariate quarterly BVARs[12] of differing dimensions conditioned to nowcasts generated from high-frequency financial variables. The authors also explored which set of financial variables produced the most accurate predictions. Chen and Ranciere (2019) examined the forecasting power of financial variables by predicting several macroeconomic variables using a set of financial variables via relatively simple panel model. In addition to the whole and highly imbalanced sample (62 countries), the model was also estimated twice for advanced economies, once for emerging markets and once for low-income markets. Country-specific and pooled international data were also used. Another swing at mixing low- and high-frequency variables for macroeconomic forecasting was taken by Gorgi et al. (2019) by the means of a hybrid[13] reduced-form model. The approach can be specified to deal with conditional heteroskedastic errors and parameter updates that are robust to outliers. In other words, stochastic volatility and fat-tailed distributions can be incorporated in the model. The authors used the proposed framework with dynamic factor model specifications for forecasting.

In a recent work, Louzis (2019) argued that there exist many available priors in the BVAR literature, but they are not informative enough in many cases when considering the trend of the process in stationary and nonstationary variables, or the steady state. To that end, the author proposed incorporating prior beliefs of the steady state in a VAR model and investigated whether the information from steady-state priors further enhances forecasting quality, among other things. More specifically, for steady-states the author proposed an adaptive hierarchical NG-prior and a time-varying parameter model, and the proposed settings were generalised to account for fat-tailed and heteroskedastic innovations by augmenting the models with CSV (Common Stochastic Volatility) and Student-t errors. The alternative specifications were estimated using Gibbs and Metropolis-within-Gibbs algorithms. Korobilis and Pettenuzzo (2019), on the other hand, proposed a relief on the computational load in estimating BVAR models while maintaining forecast accuracies. They argued that the majority of existing applications featuring hierarchical priors are limited because they rely on heavy-duty simulation-based MCMC (Markov-Chain Monte-Carlo) methods which may become infeasible when dealing with high-dimensional data. Other

[12] Techniques such as using Normal-inverse Wishart conjugate and “sum of coefficients” priors were employed for making the model less susceptible to overfitting.
[13] GAS (Generalised Autoregressive Score) was used with MIDAS (Mixed Data Sampling) model for enabling a data-driven approach for updating time-varying parameters. The authors argued GAS models are flexible, easy to implement and lightweight to use.

approaches have been introduced in the literature, but they have their own restrictions. That said, the authors proposed a new simulation-free estimation algorithm for (high-dimensional) VARs employing independent hierarchical shrinkage priors[14].

While there were some discrepancies in the findings of works mentioned above, many of them concluded that including information from different markets with varying frequencies and taking nonlinearities into account indeed improves prediction performance. One of interesting rationales, brought up by Chen and Ranciere (2019), was that high-frequency variables such as financial ones have a forward-looking nature since they capture information about the future of the economy that is not yet reflected in current macroeconomic outcomes. Moreover, leveraging different types of innovations may result in seizing information from unknown factors, and their interrelations acting in the complex real-world environment, which can be beneficial in terms of forecasting.

In the following section, the literature on machine and deep learning related to forecasting is treated more in depth.

3.2 Machine and deep learning

In their paper Cavalcante et al. (2016) presented a comprehensive review through reputable papers published between 2009 and 2015 on how methods from computational intelligence have been utilised in several financial applications. The authors survey articles focusing on pre-processing and clustering of financial data, forecasting future market movements and mining financial information. They also discuss the main challenges, open problems and opportunities as well as future directions in the applications of solving financial problems using computational intelligence. Interestingly, the authors found for example that the application of deep learning methods to the field of financial forecasting remained still largely unexplored and they acknowledged deep learning as an important contribution for future works. Längkvist et al. (2014) reviewed recent developments at that time in deep learning and unsupervised feature learning for time-series problems including models and techniques that were used for modelling temporal relations. While the paper dealt with various time-series tasks such as motion capture, speech and music recognition, the authors recognised the potential of deep learning for predicting stock markets with multiple data sources, which had not yet been realised in the scientific community. Online-text-mining for

[14] The basis for the proposed algorithm comes from van den Boom et al. (2015), which is further generalised and applied to hierarchical priors of Normal-Jeffreys, Spike-and-Slab and NG.

market prediction was reviewed by Nassirtoussi et al. (2014) with the objective of clarifying the theoretical and technical framework of approaching the problem. They argued that correctly interpreting the sentiments of people in social media and financial news can increase the predictability of financial markets by presenting successful example studies. The authors reviewed past literature with an emphasis on the state-of-the-art at the time and offered a comparative analysis of the available systems including main differentiating factors among the works. Observations on the lack of research and suggestions for future works were also made where one of the identified aspects was machine learning in the context of market-predictive text-mining.

One of the unifying components of the mentioned reviews from a couple of years ago is the pursuit of understanding real-world time series' complex nature through more advanced machine learning methods instead of traditional approaches which either estimate parameters from an assumed time-series model or do not have the capacity to accurately model such data by containing only a small number of nonlinear operations. One of the main interests of exploration in the study of forecasting financial time series' is deep learning-based techniques as mentioned above: they are able to learn representations and derive complex, nonlinear and hierarchical features with little or no prior knowledge from high-dimensional input data and their capabilities have been demonstrated through breakthroughs in various applications such as computer vision, speech recognition, natural language processing and recommender systems (LIU et al. (2017)). Next, some recent studies in applying machine and deep learning to forecasting financial and economical time series are reviewed.

In a recent work Siami-Namini et al. (2018) provided empirical evidence on using a LSTM (Long Short-Term Memory) model in forecasting univariate economical and financial time series data by comparing its performance to the traditional ARIMA model. The authors used monthly data from several commonly known stock market indices with a time frame of January 1985 to August 2018. Various monthly economic time series were also collected for different time periods spanning from 50 to 70 years. The predictive performance of the models was evaluated by the means of RMSE. Rolling forecasts, or walk-forward model validation, was used, where the model was rebuilt and trained each time with all available data before making a new prediction. The LSTM model outperformed ARIMA in all five stock indices and financial time series predicted with average error reductions of 87 and 84 percent, respectively. The stateful model was constructed with one hidden layer consisting four memory cells and optimised using Adam and MSE as the loss function. However, further specifications such as data transformation were deliberately left out and other performance measures for validating the models could have been added. Nevertheless, the results demonstrated the superiority of the LSTM model. Finally, the authors recognize the potential for extending the use of deep learning-based algorithms to other prediction problems in economics and finance under multivariate setting.

In another recent work by Sun et al. (2018) a novel approach to univariate financial time series forecasting was presented. The authors hypothesised that the composition of multiple learning algorithms, or ensemble learning, can obtain better predictive performance in comparison to a single algorithm. In their approach, the AdaBoost-algorithm (Adaptive Boosting) was used for enhancing and integrating the forecasting results of all predictors from a set of LSTM models. In other words, multiple LSTM models were created and trained according to the resampled training data by AdaBoost, and forecasting results were combined. The proposed approach was empirically tested using daily data from two stock indices and exchange rates, and forecasting accuracies were evaluated with MAPE (Mean Absolute Percentage Error) and DS (Directional Symmetry). The model was compared against predictions made by a set of single models, namely LSTM, ELM (Extreme Learning Machine), SVR (Support Vector Regression), MLP (Multilayer Perceptron) and ARIMA as well as AdaBoost-equipped ensemble models MLP, SVR, ELM and LSTM. ARIMA was found to be consistently outperformed by the other individual models and all ensemble learning approaches were superior to single models. The proposed AdaBoost-LSTM outperformed all other benchmark models and produced 19 to 22 percent improvements on directional forecasts when compared to ARIMA. While the results show the efficacy of AdaBoost-LSTM, one could argue that the datasets, which had more or less 600 samples each and spanned from January 2015 to May 2017, were relatively small. Thus, it would be interesting to verify the results with larger samples. The specifics of chosen hyperparameters for the LSTM predictors were also not disclosed.

While not an application to financial data, Hua et al. (2019) presented a novel approach to time series prediction using the LSTM architecture. Inspired by the work of Hill et al. (2012) on events before synaptic connections are formed in neural microcircuits, the authors investigate whether, instead of applying the conventional fully connected LSTM, randomly created neural connections resulting in a sparse LSTM, or RCLSTM (Random Connectivity LSTM), could yield potential benefits to LSTM’s performance and efficiency. In essence, an assigned probability value that obeys an arbitrary statistical distribution indicated the tendency of corresponding neuron pairs to be connected[15]. Given its computational efficiency, the uniform distribution was chosen by the authors and a hand-crafted threshold value determined the percentage of connected neurons. In this way, the total number of involved parameters to be trained in the RCLSTM can be substantially decreased reducing also the computational load. The proposed model's performance was empirically tested with real-world network traffic data and location history data from mobile users. The model was compared against SVR, MLP and ARIMA, and validated using RMSE for traffic data and prediction accuracy[16] for human mobility prediction. In traffic prediction, and with only 1 percent neural connections, the RCLSTM outperformed ARIMA, SVR and FFNN. While having a three-layer stack with a memory cell size of 300 per layer, it performed also better than a conventional LSTM with a memory size of 30. The prediction accuracy of RCLSTM in user-mobility task was also close to the conventional LSTM even with 30 percent neural connectivity. Overall, the proposed model was capable of traffic prediction and user-location forecasting with less than half the neural connections. The obtained results suggest that in latency-stringent and resource-constrained applications, RCLSTM is more suitable than the conventional LSTM. It would be interesting to see how this type of architecture would perform in financial applications, especially when there is not much data available in terms of frequency. In this case, the RCLSTM could be a viable option to prevent overfitting and able to capture the essential features of the underlying data generating process while modelling long-term dependencies.

Another novel approach was proposed by Maggiolo and Spanakis (2019) where a hybrid model is built. While ensemble models commonly merge the outputs of multiple similar individual models, different approaches are combined in the hybrid setting. This is also the direction the authors take: they propose a combination of CNN (Convolutional Neural Network) and RNN (Recurrent Neural Network). They argue that RNN, and its variants LSTM and GRU (Gated Recurrent Unit), are able to model long-term and infrequent dependencies over time whereas CNN can model short-term highly frequent patterns in time series. Conceptually, the proposed model extracts feature from the time-dependent input via CNN and then the extracted sequence is encoded with three recurrent GRU units followed by a linear transformation to obtain the output. In addition to modelling

[15] Note that this is different from dropping out individual neurons from a model architecture, as single connections are being discussed. A neuron can have multiple connections and when a neuron is dropped, all connections are lost.
[16] Defined as the ratio of the number of correct predictions to the total number of predictions.

nonlinearities as explained, the authors also generated predictions using linear regression so that the final prediction for the whole model is the sum of linear and nonlinear predictions. Initially, the model also takes three versions of the original input time series creating two additional down sampled versions by a factor of ½ and ¼, averaging the values accordingly. The causal convolution in the model's convolutional part takes inspiration from van den Oord et al. (2016). The proposed model was empirically tested in two uni- and two multivariate settings, which were nonfinancial, and three metrics, namely MSE, MAE (Mean Absolute Error) and DTW (Dynamic Time Warping), were used to compare the model's prediction accuracy against LSTM- and GRU-equipped RNN models, SVM, ARIMA and LSTNet (LAI et al. (2018)) which is similar to the proposed model. In the multivariate setting, SVM was dropped and ARIMA was changed to ridge regression model. The results demonstrated that the proposed model outperformed the baseline models under multivariate setting. However, possibly because the datasets were simple enough for simpler models, the proposed architecture did not excel in the univariate setting. It is worth mentioning that for all datasets, normalisation and gaussian filters were applied for denoising. The inclusion of attention mechanism was suggested for future works from the authors. A financial or economical multivariate setup would also be an interesting use-case for the proposed model.

Shah et al. (2018) compared the performance of LSTM and DNN (Deep Neural Network) when applied to predicting stock prices. The authors argue that there were only a few studies that addressed doubts such as when and why LSTM networks perform well, and in which cases LSTM should be chosen over a DNN. A daily dataset with closing prices spanning from 1997 to 2017 was extracted from the Bombay Stock Exchange index for empirical testing. The data were normalised to relative change over the specified time window at hand: the authors justify the conversion by explaining that predicting percent change instead of actual value is more suitable, especially for guaranteeing convergence for the optimisation process. Furthermore, for the DNN, the converted data was mapped to [0,1]-scale with min-max procedure. The authors also explain thoroughly the processes for choosing the models' architectural structures, hyperparameters and optimisation methods. RMSE, DS and forecast bias were used for comparing the predictive abilities of the models which showed and confirmed the superiority of LSTM. However, one important notion was made in the study: while the daily one-step-ahead predictions seemed promising, both of the models were making point-by-point predictions meaning that the predictions were not based on any previous predictions but on the last true value. A similar finding was made in Leccese (2019) where it was stated that for predicting the next day value, LSTM uses a value very close to the previous day closing price meaning that a model has no predictive ability. Therefore, predicting multiple time horizons ahead would better qualify as testing the models' capabilities. The 7-days-ahead forecasts demonstrated that the LSTM can handle better weekly movements: in more than half of the cases LSTM was able to predict volatile movements in the true data. Measures for ensuring and verifying that the models generalize well on new data were also taken by evading overfitting and testing with a new stock dataset. For future works, the authors identified the opportunity of adding more variables, as for now only price data were used for predictions.

Sezer and Ozbayoglu (2018) proposed a novel algorithmic trading model based on CNN. The authors elaborated that in recent years for financial time series forecasting, recurrent type of neural networks have been most commonly used from the deep learning spectrum and models integrating technical analysis data with DNNs occurs rarely in literature. With the use of CNN, the authors combine technical analysis and deep learning: the main difference between the proposed model and other methods was that the technical analysis data was applied on prices to create two-dimensional feature vectors and matrices that were fed to the CNN as input. In this way, the financial time series forecasting problem was turned into an image classification problem. More specifically, daily stock prices and ETFs (Exchange-Traded Fund) were collected from 2002 to 2017 and the size of 15x15 images were created daily using 15 mostly oscillator- and trend-based technical indicators. The architecture of the proposed CNN was described in detail and the model's performance was evaluated both computationally and financially, where the financial evaluation included some benchmarks such as BaH (Buy & Hold -strategy), RSI (Relative Strength Index), SMA (Simple Moving Average), LSTM and MLP methods as well as several metrics. The proposed model outperformed other baselines over the long run and demonstrated stable and robust operating characteristics. For future works, the authors suggested further optimisation in the proposed model's hyperparameters either manually or using an evolutionary algorithm. They also recognised the need for a bigger and more granular dataset in order to build deeper models as more training data would be available. However, the technical indicators used in the work are fairly simple and assume certain characteristics about the data which may weaken the model's performance. It is also worth mentioning that according to the authors, that in overall, the application of DNNs on financial forecasting models is very limited in the public domain.

The study from Chong et al. (2017) attempted to provide a comprehensive and objective assessment on both the pros and cons of deep learning algorithms for stock market analysis and prediction. In their work, high-frequency intraday stock returns data from the Korean stock market was used as an input data to several deep learning algorithms. The effects of three unsupervised feature extraction methods, namely PCA (Principal Component Analysis) (linear), AE (Autoencoder) and RBM (Restricted Boltzmann Machine) (nonlinears), were examined on the overall ability to predict future market behaviour. The created features by these methods were then fed to standard AR and DNN models for comparison. More specifically, stock prices from 38 different stocks were collected every five minutes between January 2010 and December 2014 and then logarithmic returns were calculated. This procedure resulted in 73,041 five-minute returns over 1239 trading days. Using the mean and standard deviation, all stock returns in the training dataset were normalised. The performance of the methods was evaluated with NMSE (Normalized Mean Squared Error), RMSE, MAE and MI (Mutual Information) while representations created by extracting features were assessed by reconstruction errors and prediction accuracies. From the results authors find that the DNN can extract additional information from input data and improve predictions. They also continue that while any conclusive deduction cannot be derived in favour to the superiority of DNNs, they do suggest promising extensions and directions for further investigation. The ability to extract raw features from a large set of raw data without relying on prior knowledge on predictors was recognised as the main advantage of DNNs but applying deep learning and finding the most appropriate model can be challenging. The authors mention for future works that the input could be augmented with variables that carry information about the future price movement and other data representation methods could be utilised. It would have also been interesting to see how, for example, RNN-based method would have performed in this study. Lastly, according to the authors, while there has been growing interest whether deep learning can be effectively applied to problems in finance, the literature still remains limited.

Fischer and Krauss (2018) deployed LSTM networks against memory-free classification methods such as RAF (Random Forest), DNN and LOG (Logistic Regression Classifier) in a large-scale financial market prediction task. Daily total cum-dividend prices from January 1990 until October 2015 were extracted from all participants of S&P 500. The authors provided an in-depth guide on how to frame a prediction task, extract representative features and pre-process the data, construct a suitable architecture for the models and derive a trading strategy based on the predictions. The LSTM was found to outperform the standard DNN and logistic regression by a clear margin while being most of the times superior to random forest with the exception of global financial crisis period. The findings also challenge the semi-strong form of market efficiency with statistically and economically significant returns of 0.46 percent per day. Another attempt to predicting stock prices was taken by Weng et al. (2018) where their goal was to develop a novel financial expert system incorporating historical stock prices, well-known technical indicators, counts and sentiment scores of published news articles for a given stock, trends in Google searches for the given stock ticker and number of unique visitors for pertinent Wikipedia pages. Then, after data preparation, the system trained four separate machine learning ensemble regression methods based on ANN (Artificial Neural Network), SVR, boosted regression tree and RaF and then the platform selected the best-performing ensemble for a given stock in the cross-validation phase. PCA was also experimented as a dimensionality reduction technique in the data preparation process. Model evaluation was done with RMSE, MAE and MAPE. Superior results compared to the literature that uses either single data sources or individual/ensemble learning models were achieved in the study and it was verified that, with varying importance, online data features contributed significantly to the prediction accuracy. Many opportunities for future research were identified, such as differing time-horizons for predictions, dealing with non-US markets, adding new features, utilising a recurrent architecture such as LSTM and taking advantage of a firm’s location information.

By Du et al. (2018), a more end-to-end type of approach was taken in solving time series forecasting problems. They propose a deep sequence-to-sequence model which is a more generalised approach for learning from sequential data and makes minimal assumptions on the sequence structure. The method involves an architecture familiar from autoencoders: an encoder first compresses the input sequence to a fixed-length vector with lower dimensionality and then a decoder produces the output by attempting to reconstruct the original input from the compressed data. According to the authors, they applied sequence-to-sequence learning for the first time to time series forecasting. As the encoder and decoder, the authors applied the LSTM model in order to account for spatial-temporal dependencies in the multivariate time series input data. Real air quality time series data with multiple variables were used for conducting experiments. Input data were also normalised to [0,1]-scale using min-max procedure. The proposed model was compared with SVR using three different kernels and three recurrent models, namely standard RNN, LSTM and GRU, via MSE. The results illustrate the superiority of the proposed model in single- and multi-step forward predictions. The approach taken by the authors is intriguing and it would be interesting to see how it performs in forecasting multivariate financial time series. It could also be worthwhile to verify the necessity for input normalisation since data manipulation can lead to unintentional costs in terms of model performance. One could also argue whether LSTM is able to extract spatial features in addition to temporal ones.

One of the challenges for neural network-based systems, as found in reviewed works, is to find the best optimal architecture. Tran et al. (2019) proposed a data-driven learning scheme where the architecture of the fully connected FFNN is constructed by adopting a progressive learning paradigm that gradually extends the network topology. The algorithm takes advantage of GOPs (Generalized Operational Perceptron) which have a richer set of functionals when comparing to traditional building blocks of neural networks, the perceptron. Simply put, the network topology gets extended by adding a new GOP-block on the current hidden layer if it improves the network’s performance and if the performance saturates, the block is added to a new hidden layer. Finally, when the learning procedure is over, the parameters of the newly created network are optimised. Using a large-scale imbalanced LOB-dataset (Limit Order Book) and predicting the movement of mid-price in the future, the proposed algorithm was evaluated by the means of F1 score, recall, precision and accuracy against 12 models, including machine learning methods using vector and tensor inputs, and three other progressive learning algorithms. The empirical study verified the superiority over the benchmarks, including multilinear methods that make use of privileged information. It would be interesting to see how these neural architecture learning algorithms perform with different multivariate financial time series of varying time horizons, spans and frequencies.

There are also hybrid approaches in the literature that attempt to combine econometric and deep learning models. In their study, Kim and Won (2018) conjectured that for predicting financial market volatility, combining various information from several econometric models with a neural network would be more effective. To that end, the authors proposed a hybrid LSTM model where the parameters of two or more GARCH-type (Generalized Autoregressive Conditional Heteroskedasticity) models were entered as inputs to the LSTM. They argued that as capturing heteroskedastic and leptokurtic nature of financial time series data as well as long-term dependencies are difficult to reproduce in out-of-sample predictions by econometric methods, a neural network could alleviate this issue. The dataset used considers the volatility of KOSPI (Korea Composite Stock Price Index) 200 stock index returns from January 2001 to September 2011, and predictions were made until January 2017. The presence of heteroskedasticity (until the fifth lag), non-stationarity and non-normality in the dataset was verified with ARCH-LM (ARCH-Lagrange Multiplier), ADF (Augmented Dickey-Fuller) and Jarque-Bera tests, respectively. Further computations revealed high kurtosis and negative skewness. The authors explained that the parameters of GARCH and EWMA (Exponential Weighted Moving Average) represent for example different properties of volatility shocks, such as its magnitude, direction and persistence. For the empirical study, GARCH, EGARCH (Exponential GARCH) and EWMA were leveraged as “GARCH-type” models for constructing LSTM-models that integrate two out of three of them, and finally an LSTM that integrated all three of them. For comparison, a DNN was also composed with one GARCH-type model separately, and all of the models were evaluated as individual models too. For measurements and validation, RV (Realised Volatility), MAE, MSE, HMAE (Heteroskedasticity-adjusted MAE) and HMSE (Heteroskedasticity-adjusted MSE) were used as well as the DM and WS (Wilcoxon Signed Rank) tests. Various forecast horizons were examined, namely 1, 14 and 21 days ahead with differing window lengths of 7, 14 and 22. The results confirmed that the LSTM model which integrated three GARCH-type models demonstrated superior out-of-sample performance in all scenarios and measures to all other benchmark models. Moreover, the other hybrid models outperformed single individual models. For future works authors recognized the inclusion of more diversified information in various forms such as text or images, since only 7 to 14 financial numerical inputs were used in the empirical part.

Another attempt on combining econometric and deep learning models was recently proposed by León and Ortega (2018). Their aim was to test if highly frequent electronic payments among individuals, firms and the central governments, instead of using traditional macroeconomic and financial variables, could be useful for incorporating signals of economic activity to a nowcasting model. An important distinction of their work is that unlike a traditional econometric approach, the purpose was not to explain which variables change the state of an economy and with what magnitude, but to construct an effective out-of-sample prediction model that tries to capture and leverage the complex information that underlies in financial data. In other words, as the authors summarised, their proposal is more of a machine learning approach instead of explanatory modelling for policy purposes. The authors decided to nowcast ISE (Colombian Economic Monitoring Index) which is a monthly indicator of economic activity that combines information about the production of goods and services pertaining to the most important activities in Colombia. Through the electronic payments, their economic activity nowcasting model seeks to make use of the information related to consumption, investment and government expenditure. More specifically, the electronic payment input included the value and number of operations from different FMIs resulting in three time series’ that represent values and three time series’ representing the number of operations in the corresponding FMIs. Closer inspection of the inputs revealed the presence non-normality and heteroskedasticity. The authors used a NARX-ANN (Non-Linear-Autoregressive-Exogenous-ANN), which essentially involves approximating the non-linear function of ARX with an ANN, with the aim of modelling the underlying nonlinear dynamic system. The out-of-sample performance was measured with RMSE and correlation was also used between the observed and the predicted logarithmic returns of ISE. The contribution of electronic payments data to the error of economic activity nowcast was verified against a NAR-ANN which showed that the payments data significantly reduced the nowcast error. The authors also demonstrated the competitiveness of their method against UK's economic quarterly growth nowcast performance. For future works, they suggested forecasting several periods of ahead instead of nowcasting and the use of produced nowcasts in these forecasting models. Moreover, the authors continue that financial and macroeconomic variables could be tested and as debit, credit and cash payments were excluded from the study, they could be added too.

Many scientific fields research networked data such as in biological, chemical, social, economic and financial applications, where the system under investigation has interdependent entities and components interacting with each other. In fact, these systems' underlying structure is a non-Euclidean space[17]. This is an important notion since initially, the power of deep learning was demonstrated in variety of applications with Euclidean or grid-like structure such as analysing speech, image and video signals, and later on, the success led to the wide adoption of deep learning in many other fields as one could see in the reviewed studies so far (BRONSTEIN et al. (2017)). However, in non-Euclidean applications the implementation of these tools has required the conversion from non-Euclidean data to Euclidean form - the familiar operations used with Euclidean data are not well defined in non-Euclidean domains (XU et al. (2018)). The conversion[18] essentially results into information loss which can have unintended consequences. This is also the case with the works reviewed and with the empirical study conducted in this thesis.

In practice, networked systems can be expressed with graphs which translate to non-Euclidean spaces. Through graphs it is possible to monitor and inspect the target system's

[17] Intuitively speaking Euclidean geometry is flat, or planar, and non-Euclidean is curved. In Euclidean space it is possible to draw lines directly between two points but in non-Euclidean space, it is possible to draw only on the surface of the curvature, leading to different properties. Non-Euclidean geometry encompasses data types such as manifolds, curvatures, elliptic and hyperbolical functions, graphs or shapeless 3D-meshes. For example, the Global Positioning System (GPS) works with non-Euclidean data since the surface of the earth is curved, and in Einstein's relativity theory, the universe's geometrical shape is presumed as non-Euclidean.
[18] A classic example of this type of conversion is expressing an image, taken from a real-world object, as a function over 2D-plane by enumerating the image colour intensities in each pixel to a square grid.

structural changes over different domains as well as nonlinear and complex interactions between participants. Graphs have also more capacity to encode complicated relationships in the data (XU et al. (2018)). Thus, there has been an accelerating growth in trying to apply machine learning on non-Euclidean geometric data, which covers data structures such as graphs and manifolds. The main idea in machine learning on graphs is to find out how all of the information encoded in a graph such as its structure, entities and interactions, can be incorporated into a machine learning model (HAMILTON et al. (2017)). New generalisations and definitions for important operations have been under development over the past few years by researchers for handling the complexity of graph data and extending the ideas of Euclidean-proven neural network architectures to graph-based machine learning. A comprehensive overview[19] on graph neural networks was recently provided by Wu et al. (2019). A couple of selected recent studies applying machine learning on graphs, focusing on graph-based deep learning, in financial and other applications are presented, as follows.

3.3 Graph-based deep learning

Chen and Wei (2018) argued that although improvements have been made in stock price prediction by leveraging news information in deep learning approaches, many factors from financial markets are still ignored. To that end, the authors investigated whether including information on corporation relationships to the forecasting model leads to enhanced predictions on stock prices. The authors used two types of information, namely historical time series on stocks and an adjacency matrix for encoding a weighted graph where each node stands for a company and the edges between the nodes are formed via shareholding ratio between companies. They introduced two models, namely a pipeline prediction model and a joint prediction model. The former integrated corporation relationships via node embeddings where the goal is to encode nodes as low-dimensional vectors summarising their positions in the graph and the structure of their local neighbourhood - a review on node embedding methods can be found in Hamilton et al. (2017). More specifically, at first, each company obtained a representation vector through the node embedding layer based on the constructed graph of corporation relationships. Then, using cosine similarity, the most related nodes to the target node in question were selected by averaging the features of the top N related companies. Lastly, the computed

[19] For more information, see e.g. Battaglia et al. (2018) as well as the GitHub repositories https://github.com/thunlp/GNNPapers and https://github.com/DeepGraphLearning/LiteratureDL4Graph.

average was combined with the target company's feature vector. The resulting vector, where the relevant companies' information is included, was fed as an input to an LSTM-based encoder. Three different node embedding approaches were also used for comparisons. While the pipeline model considers features of the top N relevant companies, the joint prediction model based on GCN (Graph Convolutional Neural Network) (KIPF and WELLING (2017)) was able to incorporate information for all relevant companies. The GCN takes as an input an adjacency matrix representing the graph and node-specific features in a vector that was constructed through the LSTM-based encoder. In total of seven models were experimented in the empirical study: the joint model, the three variants of the pipeline model, LSTM and GCN separately as well as a logistic regression-based method. From the node embedding methods, namely DeepWalk (PEROZZI et al. (2014)), node2vec (GROVER e LESKOVEC (2016)) and LINE (TANG et al. (2015)), LINE performed best in combination with the LSTM. The proposed joint prediction model outperformed all the baselines in predicting stock price movements, and the prediction accuracy of pipeline and joint models, was better than the other baselines. Daily historical price movement data from CSI300 stocks were used in the empirical analysis. For future works the authors identify different ways for establishing the relationship graph and including heterogeneous information sources, in addition to stock price movement data. The main drawback of the study is the used GCN: high computational cost, missing support for edge features, suitable only for undirected and static graphs, and inapplicable for large and densely connected graph datasets (KIPF and WELLING (2017)). Nevertheless, the study successfully demonstrates the superiority of a graph-based deep learning approach.

As discussed before, Du et al. (2018) proposed a sequence-to-sequence scheme for time series forecasting. However, Xu et al. (2018) argued that as many machine learning tasks have inputs naturally represented as graphs, existing sequence-to-sequence models have difficulties in accurately converting underlying graph-structured data into sequential form. Therefore, the authors come up with a novel graph-to-sequence approach that encodes a graph directly, and also implements the conventional encoder-decoder framework. The proposed model maps an input graph to a sequence of vectors and from these, an attention-based LSTM is used for decoding the target sequence. The model can also cope with raw inputs that are originally expressed in a sequential form since these inputs can be further enhanced with additional information resulting in constructing graph inputs. The ideal situation would be, according to the authors, to build a powerful method that is capable of learning representations of input data regardless of its inherent structure.

The proposed model includes a graph encoder that generates node embeddings and then graph embeddings are constructed based on the learned embeddings. A sequence decoder takes both the graph and node embeddings as input and sequences are generated by an attention mechanism employed over the node embeddings. The experiments were not conducted with financial data, but the model demonstrated impressive capabilities in reasoning (path finding) and natural language generation (translating SQL queries to text). The efficacy of the model's architecture was also verified through an ablation study. The proposed model outperformed existing graph neural networks and sequence-to-sequence models in real-world and synthetic settings. A financial or economical application of the model with networked data could be an interesting avenue for future research.

Many of the studies in graph-based deep learning and graph representation learning literature are limited by their applicability to dynamic environments while in real-world applications, one often encounters underlying graph data structures that evolve over time. Built on recent success of GNNs (Graph Neural Network) for static graphs, Pareja et al. (2019) approach the model adaptation problem to changing environment by implementing the commonly used recurrent mechanism directly in the model. In other words, while earlier studies focus on learning time-dependent representations from node embeddings by GNN+LSTM-setup, the proposed model's network parameters are updated based on the dynamism of incoming input graphs. In this way, the recurrent mechanism is built inherently to the model - in every iteration, the recurrence injects the dynamism into the parameters of GNN thus the model keeps evolving. The chosen graph-based method was GCN (KIPF and WELLING (2017)) and while it has the drawbacks as discussed earlier, the authors root for its simplicity and effectiveness. The recurrence was achieved via modified GRU. Paired with a task-specific classifier, the proposed model's performance was empirically tested in node and edge classification tasks as well as in link prediction against a GCN without temporal modelling and a GCN+GRU that utilised learned node embeddings. The proposed model outperformed the baselines proving that a model-oriented approach to dynamically changing graphs is a viable solution. For future works, the authors recognize that the proposed model does not scale up well to larger datasets. They recognize some remedies from emerging architectures such as alternatives for the GRU and the convolution operation in the chosen GCN. The authors will investigate how to adapt the model to an anomaly detection task where the data may be imbalanced. The proposed model has an appealing property because in non-stationary environments there always exists the question of how often and with what mechanism the model should be updated. Also, the model-oriented approach does not require any historical information for nodes that did not appear before. These properties could prove to be effective in forecasting financial or economic time series.

3.4 Remarks on recent works

The common trend in the current macroeconomic forecasting literature seems to be developing more efficient data-driven methods that mix high- and low-frequency data from different markets as well as include different types of innovations for accounting complexities and nonlinearities. The same ideas are present also in the intelligent computing literature, where the notion of limited applications of deep learning methods in economic and financial time series forecasting literature is repeatedly apparent. As it was stated, the issue with many traditional approaches is that they do not have the capacity to accurately model complex time series data whereas deep learning-based techniques are able to interpret complex structures from data without prior specifications. Therefore, efforts and contributions have been made towards applying methods that attempt to understand underlying data generating processes as they are. However, while deep learning has demonstrated promises in doing so, applying deep learning and finding the most appropriate model is indeed challenging. One of the most recent developments in deep learning research is the late birth of graph-based deep learning which is currently evolving rapidly.

4 TECHNICAL BACKGROUND

Machine learning is an umbrella term for algorithms and techniques that give computers the ability to act and learn without being explicitly programmed. More specifically, machine leaching can be understood[20] as the training of a model from data that generalises a decision against a performance measure (BROWNLEE, (2013)). In other words, the strength of these methods lies in their ability to provide generalised and adaptable solutions through architectures that can learn to improve their performance (KUMAR et al. (2019)).

Machine learning methods can be roughly divided into supervised and unsupervised learning algorithms. In supervised learning, a dataset $D = \{x, y\}_{n=1}^{N}$ consisting pairs of input features x and corresponding output labels or values y, is presented to the model. In classification tasks, the output is commonly a fixed set of classes, whereas in regression problems, y can also be a vector with continuous values. There may be one or more input features to learn from and one or more output targets to estimate simultaneously. In practice, supervised learning is typically about finding a set of optimal model parameters Θ that best predict the data based on a loss function $L(y, \hat{y})$, where the estimate $\hat{y}$ denotes the output of the model. The estimation is ultimately obtained by feeding an observation x, or a data point, to the function $f(x, \Theta)$ that represents the model while attempting to approximate the underlying data generating process (LITJENS et al. (2017)).

Unsupervised learning algorithms differ from supervised ones in that there are no output labels or values available. In other words, the purpose of the learning process is to find patterns without guidance. One example of unsupervised learning is by the means of reconstruction loss $L(x, \hat{x})$, where the idea is to learn to reconstruct the input, often through a lower-dimensional or noisy representation. While the reconstruction loss is commonly used in deep learning methods, some traditional unsupervised machine learning approaches include clustering algorithms and PCA (LITJENS et al. (2017)).

Another approach commonly present in the literature is semi-supervised learning, which bridges the gap between supervised and unsupervised approaches. In semi-supervised learning, only a small subset of the observations has corresponding class labels or values, and the rest of the data is unlabelled. The semi-supervised approach is an attractive option when obtaining class labels for the entire dataset is expensive or even

[20] Alternative definition of machine learning: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, measured by P, improves with experience E”, by Mitchell (1997).

impossible (KINGMA et al. (2014)). Other approaches include reinforcement learning, which is a technique that enables sequential decisions from evaluative feedbacks using unlabelled data. Reinforcement learning is essentially the task of learning how agents ought to take sequences of actions in an environment for maximising cumulative rewards (FRANÇOIS-LAVET et al. (2018)).

Machine learning techniques can be further divided into shallow and deep learning. The technical difference between these is subtle: deep learning is a term used to describe different types of ANN architectures that consists of more than one hidden layer and shallow learning can be understood as all other machine learning techniques that does not fit the description of deep learning. While shallow learners are limited by hand-crafted features, strong prior assumptions about the underlying data and application-specific expert knowledge, deep learning architectures are capable of learning representation and features with little or no prior knowledge directly from high-dimensional input data (BALL et al. (2017)).

In the following sections, deep learning methods used in the experiments are briefly introduced. The selection of methods is largely guided by the literature review and the intention is to choose models with different capabilities for gaining breadth in the empirical analysis. The chosen models[21] include DNN, LSTM and CNN. While not covered in the literature review, Convolutional LSTM is also added which is a certain type of combination of the two latter models, which will be discussed in section 4.4.

4.1 Artificial Neural Networks

ANNs are a type of learning algorithm which forms the basis of most deep learning methods. A neural network comprises of neurons[22] with some activation a and parameters $\Theta = \{W, B\}$, where W is a set of weights and B a set of biases. The activation is the result of a linear combination of the input x to the neuron and the parameters, followed by an element-wise nonlinearity $\sigma(\cdot)$, which is commonly known as the activation function or transfer function. A single neuron can be mathematically defined, in matrix[23] notation, and conceptually visualised, as follows (LITJENS et al. (2017)[24]):

[21] See e.g. SIAMI-NAMINI et al. (2018), SHA et al. (2018), SEZER and OZBAYOGLY (2018) and FISCHER and KRAUSS (2018).
[22] Also called as perceptrons, which were first introduced by Rosenblatt (1958).
[23] For a summary of matrix calculus required for deep learning, see e.g. PARR and HOWARD (2018).
[24] Unless stated otherwise, all illustrations in the thesis are elaborated by the author and referenced accordingly.

$$a = \sigma(w_1x_1 + w_2x_2 + \ldots + w_nx_n + b) = \sigma\left(\sum_{n=1}^{N} w_nx_n + b\right) = \sigma(w^Tx + b) \qquad (1)$$

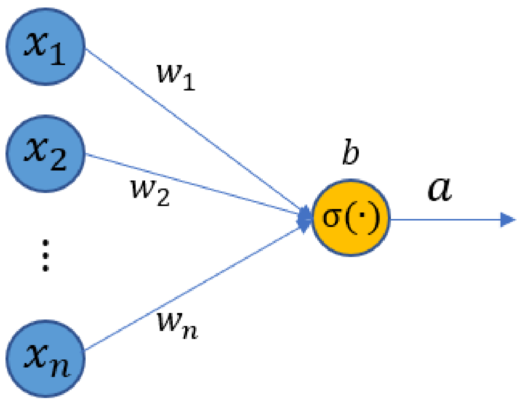


Figure 1: The concept of a single neuron with related variables.

For traditional neural networks, typical activation functions are the sigmoid and tanh function. However, since the inherent properties of sigmoid and tanh regarding their derivatives can result into slow learning rates, deep architectures based on new non-saturating activation functions have been suggested to be more effectively trainable during the learning process of a deep learning model. One of the most successful and widely popular activation functions is the ReLU (Rectified Linear Unit) activation, $\sigma(x) = \max\{0, x\}$ (ARORA et al. (2018)). The traditional neural network MLP has several layers of activations, or transformations. The activations from the previous layer serve as a part of the input for the next layer's activations, all the way until the final estimate is produced (LITJENS et al. (2017)):

$$y \approx \hat{y} = f(x, \Theta) = \sigma\left(W^T\sigma\left(W^T \ldots \sigma(W^Tx + b)\right) + b\right) \qquad (2)$$

Now, the weight columns w_k associated with activation k, generates the weight matrix W. Note that in each layer there are commonly many neurons[25]. These sequential layers between the input and output are often called as hidden layers which refers to the fact that during learning, the parameters related to the hidden layers are automatically determined

[25] Recall that more than one hidden layer in an ANN is effectively a DNN.

via an iterative indirect optimisation procedure[26] and they are not seemingly visible to the end user, in the same way as inputs and outputs are (LITJENS et al. (2017)). In Figure 2, an example of a deep neural network is demonstrated:

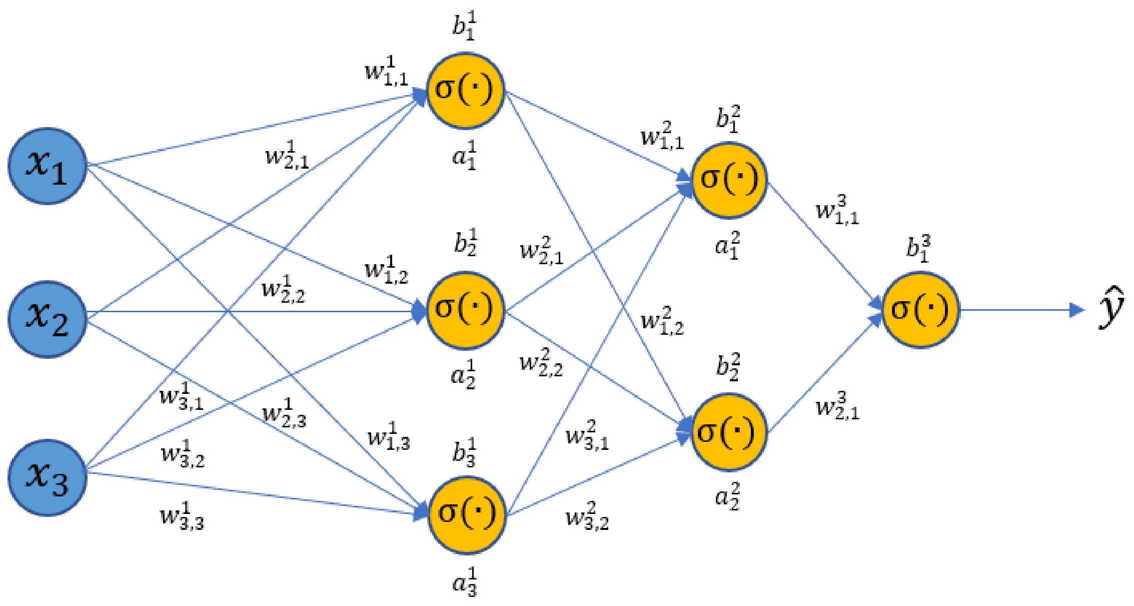


Figure 2: A deep neural network with three input variables and two hidden layers.

Using the formulation in (2), the output of the DNN showed in Figure 2 can be computed as follows[27]:

$$\hat{y} = \sigma(w_{1,1}^3 a_1^2 + w_{2,1}^3 a_2^2 + b_1^3) \tag{3}$$

where $\begin{bmatrix} a_1^2 \\ a_2^2 \end{bmatrix} = \begin{bmatrix} \sigma(w_{1,1}^2 a_1^1 + w_{2,1}^2 a_2^1 + w_{3,1}^2 a_3^1 + b_1^2) \\ \sigma(w_{1,2}^2 a_1^1 + w_{2,2}^2 a_2^1 + w_{3,2}^2 a_3^1 + b_2^2) \end{bmatrix}$ & $\begin{bmatrix} a_1^1 \\ a_2^1 \\ a_3^1 \end{bmatrix} = \begin{bmatrix} \sigma(w_{1,1}^1 x_1 + w_{2,1}^1 x_2 + w_{3,1}^1 x_3 + b_1^1) \\ \sigma(w_{1,2}^1 x_1 + w_{2,2}^1 x_2 + w_{3,2}^1 x_3 + b_2^1) \\ \sigma(w_{1,3}^1 x_1 + w_{2,3}^1 x_2 + w_{3,3}^1 x_3 + b_3^1) \end{bmatrix}$

To illustrate a supervised setting, given a dataset of inputs x and corresponding targets y, and a loss function $L(y, \hat{y})$ that measures the difference between the output $\hat{y} = f(x, \Theta)$ and the target y, the model parameters $\Theta = \{W, b\}$ can be learned[28] and chosen via minimising the sum of errors:

[26] Optimisation algorithms used for training deep models are different from pure optimisation problems in a sense that in most cases, a separate performance measure is defined with respect to the test set which is then controlled by minimising a loss function (GOODFELLOW et al. (2016)).

[27] Recall that $Ax = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + & \cdots & + a_{1n}x_1 \\ & \vdots & \\ a_{m1}x_1 + & \cdots & + a_{mn}x_n \end{bmatrix}$.

[28] The learning procedure is also called as training. As ANNs are fully connected, the learnable parameters for a hidden layer equal to $(n+1)m$, where n is the number of inputs and m is the number of outputs. Each output node is also associated with a bias term.

$$\min_{\Theta} J = \sum L(y, \hat{y}) \tag{4}$$

The minimisation problem can be solved using first-order gradient descent-based backpropagation[29] given that appropriate activation functions and the loss function $L(\cdot)$ are provided (CHONG et al (2017)).

4.2 Convolutional Neural Networks

CNNs[30] are a certain type of neural network architecture for processing grid-like topological data structures. Time-series data can be thought of as a one-dimensional grid, or a vector, where samples are taken at regular time intervals. Image data can be structured as a two-dimensional grid of pixels, or a matrix. Images can also be three-dimensional, where the 2D pixel values of each RGB-channel (Red-Green-Blue) are stacked in the third dimension. CNNs are similar to basic neural networks, but instead of employing general matrix multiplication in (at least one) of their layers, a convolution operation takes place. Oftentimes, the convolutional procedure used in neural networks differs from its counterparts in other fields. The basic discrete convolution operation can be formulated as (GOODFELLOW et al. (2016)):

$$(x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{5}$$

Generally speaking, convolution is an operation between two functions that produces a real value. In this case, x and w represent functions with an input parameter, and a convolution operation $*$ is performed at a time step t, where a is the age of a measurement. The terminology here is that x is an input whereas w is a kernel, or a filter[31]. A so-called feature map is produced from the convolution. In more practical terms, assume that a two-dimensional image I is used as an input and the kernel K has two dimensions as well, but it

[29] Backpropagation is based on applying the chain rule when calculating gradients, see e.g. LE (2015a) and NIELSEN (2019). For an overview of gradient descent optimisation algorithms and their variants designed to train deep models, among others, see RUDER (2018).
[30] Influential papers on CNNs from the past include e.g. FUKUSHIMA (1980) and LECUN et al. (1999).
[31] The convolution has a fundamental role in digital signal processing as a general filtering operation (PRANDONI and VETTERLI (2013)).

is greatly smaller than the image itself. Then, the convolution can be expressed as (GOODFELLOW et al. (2016)):

$$(I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) \tag{6}$$

With images, convolution[32] means passing a small matrix consisting of numbers, which is the kernel, over the image. At a given position of the kernel, element-wise multiplication of each cell value in the kernel is performed with the corresponding image pixel value that overlaps the kernel cell, and the sum of these multiplications is taken, resulting in a single value. The type of multiplication applied between the filter and the filter-sized area in the image is the dot product. The equation above depicts this procedure: the filter is systematically applied step by step across the grid-shaped image, which eventually produces the feature map. Then, each value in the feature map is passed through a nonlinearity $\sigma(\cdot)$, similar to ANNs. The following toy example visualises the procedure[33]:

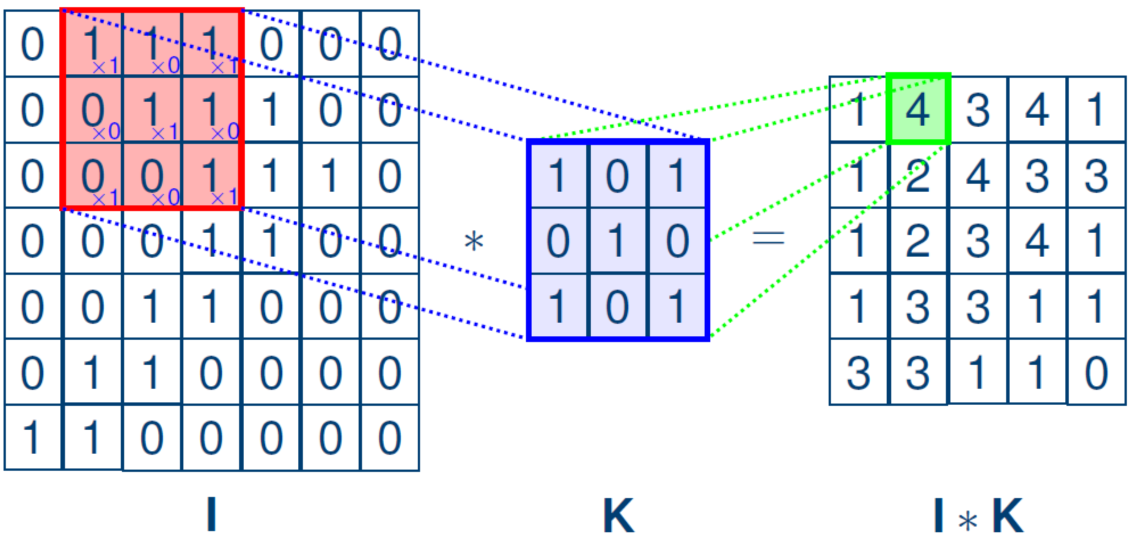


Figure 3: A toy example of convolution on images.

One key difference between CNNs and ANNs can be noted here: while in a traditional neural net, each element of the weight matrix is used once when computing the output of a layer, the weights in the convolutional neural network are shared, meaning that the value of weight applied to one input is tied to the value of a weight applied elsewhere

[32] In machine learning applications, the basic convolution equals to cross correlation.
[33] The visualisation is from Veličković (2018).

(GOODFELLOW et al. (2016)). Another key difference is the use of pooling layers in CNNs which commonly follow the convolutional layers. Pooling is the aggregation of neighbouring values typically through recording the mean or maximum value of a given mapping. A certain amount of translation invariance is induced through pooling. Both of these key differences contribute to the property of not needing to learn separate detectors for the same object occurring at different positions in an image (LITJENS et al. (2017)). Moreover, the convolution operation leverages the spatial properties of the data meaning that for example in the case of images, pixels that are closer together are treated differently than ones being further apart (LECUN et al. (2015)). A bias value is also associated with a kernel. Finally, a basic neural network structure is commonly added to the successive convolutional and pooling layers before calculating the output of the network.

What the CNN needs to figure out, or learn[34], are the values in the kernels, which determine how the feature maps are produced. In principle, the training of CNNs[35] follow the same pattern as training ANNs. However, it is worth mentioning that when comparing to ANN, the calculation of derivatives on the convolutional layers is more complicated due to the shared weight parameters.

4.3 Long Short-Term Memory

RNNs are a class of more dynamic models that are better suited for modelling temporal relationships in sequential data. Intuitively speaking, RNNs pass input data to the network across time steps and allows for information to persist, which is enabled through internal loops in the network design. This is a property that the forward feeding ANNs and CNNs do not have. The original RNN performs well when modelling short sequences, but it fails to capture long-term dependencies due to the problem of vanishing and exploding gradients[36] during learning. LSTM[37] is a variant[38] of the original RNN designed to relieve its learning issues (SHAH et al. (2018); KIM and WON (2018)).

[34] The amount of learnable parameters in a convolutional layer equals to $(mnl + 1)k$, where mn are the dimensions of a kernel, l is the number of feature maps in the previous layer, k is the number of resulting feature maps and each feature map has a bias term.
[35] For a tutorial on CNNs, see e.g. OLAH (2014) and LE (2015b).
[36] Due to chain rule, gradients coming from deeper layers go through multiple matrix multiplications and as they approach earlier layers, the gradients may shrink or grow exponentially. Shrinking may occur if the gradients have small values (<1) and growing if gradients are greater than one.
[37] For the original paper, see HOCHREITER and SCHMIDHUBER (1997).
[38] Another important variant in RNN literature is the GRU, proposed in CHO et al. (2014). While it is simpler and computationally efficient, LSTM is still widely used due to its performance in empirical applications.

The LSTM is composed of an input layer, one[39] or more hidden layer(s) and an output layer, similar to the ANNs. But, differently from neurons, the key to LSTMs is memory cells. The LSTM has the ability to remove or add information to its state in the memory cell, or cell state, by using components called gates. The responsibility of the gates is to decide whether to include new information in the model or not. The LSTM has three gates for controlling the cell state: the forget gate defines how much information is removed, the input gate specifies how much information is added and the output gate decides which information from the cell state is used as output. The forget and input gates output a value between zero and one, which translates to the proportion of absorbing information, one considering all information and zero forgetting everything (OLAH (2015)). Together, these gates and the cell state form a memory cell which is the building block of hidden layers. Next, the inner workings of the LSTM are detailed, following Fischer and Krauss (2018) and Olah (2015).

At first, the LSTM layer seeks to find out which information should be removed from its previous cell state s_{t-1}. The activation values in the forget gate f_t are thus calculated based on three values: the current input x_t, the output h_{t-1} of the memory cells at the previous timestep, and the bias terms b_f. The activation function used is sigmoid:

$$f_t = \sigma\left(W_{f,x}x_t + W_{f,h}h_{t-1} + b_f\right) \tag{7}$$

Then the LSTM decides how much information will be stored in the cell state. The activation values in the input gate i_t are calculated through sigmoid and candidate values $\tilde{s}_t$, that could be added to the cell states, are computed with tanh:

$$i_t = \sigma\left(W_{i,x}x_t + W_{i,h}h_{t-1} + b_i\right) \tag{8}$$

$$\tilde{s}_t = \tanh\left(W_{\tilde{s},x}x_t + W_{\tilde{s},h}h_{t-1} + b_{\tilde{s}}\right) \tag{9}$$

In the next step, the candidate values and the calculated activation values in forget and input gates are used for computing the new cell states. In other words, the old cell state gets multiplied by the forget gate's values and the values from the input gate are multiplied with the candidate values:

[39] Technically deep learning constitutes more than one hidden layer, but LSTM's building blocks are inherently deep.

$$s_t = f_t \circ s_{t-1} + i_t \circ \tilde{s}_t \tag{10}$$

Note that ○ denotes the elementwise-product (Hadamard). Finally, the activation values for the output gate o_t are calculated with sigmoid. The values are then leveraged in producing the output values h_t together with the newly established cell states, which go through tanh first:

$$o_t = \sigma(W_{o,x}x_t + W_{o,h}h_{t-1} + b_o) \tag{11}$$

$$h_t = o_t \circ \tanh(s_t) \tag{12}$$

For better intuition, the dynamic data flow in a memory cell can be visualised following Fischer and Krauss (2018), in line with the equations presented:

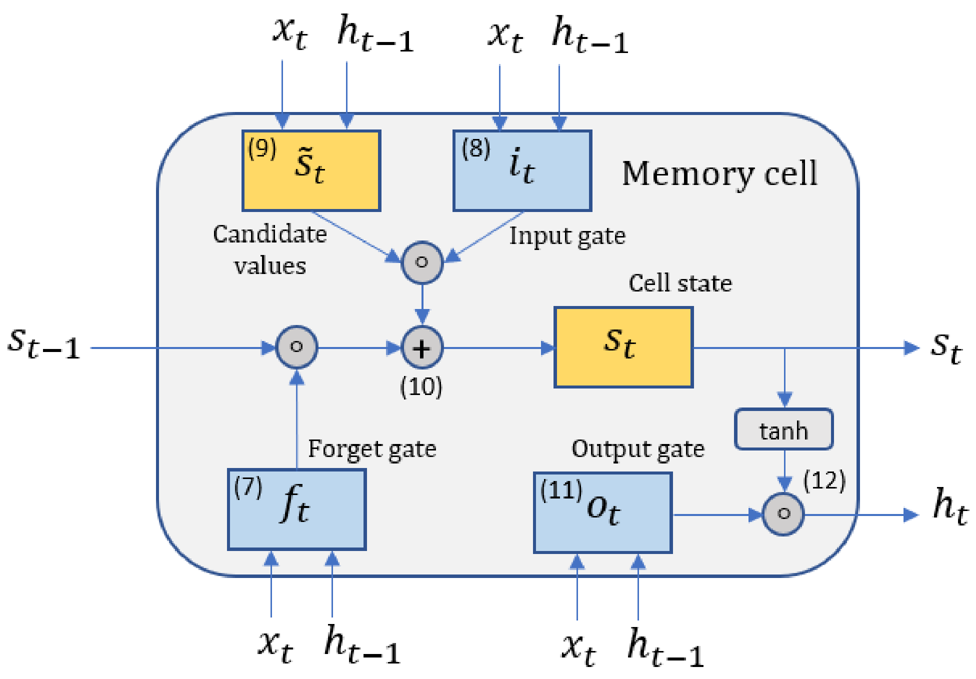


Figure 4: The dynamic data flow of one LSTM memory cell.

Note that the equations are in vectorised format and describe the update of the memory cells in the LSTM layer at every timestep. There can be one or more memory cells in one hidden layer, and multiple hidden layers can be stacked. A standard feed-forward hidden layer commonly follows the LSTM layer(s). Increasing the depth of LSTM network potentially allows the hidden state at each level to operate at a different timescale (PASCANU et al. (2014)).

The training[40] of LSTM is similar to the traditional ANN: the minimisation problem of the specified loss function is done via adjusting the weights, namely the kernel weights for inputs and recurrent weights for hidden states, and biases of the LSTM. The backpropagation method used for training RNNs is often referred to as Backpropagation Through Time (BPTT) where the RNN is “unfolded” by creating several copies of the recurrent units which can then be treated like a feed-forward network with tied weights. Then, the standard backpropagation techniques can be used for model training. The disadvantage of the standard BPTT is that, while being computationally efficient, it is also fairly intensive in memory[41] usage due the requirement of storing internal states of the unfolded network core at every time-step in order to be able to evaluate correct partial derivatives (GRUSLYS et al. (2016)).

4.4 Convolutional LSTM

Both CNNs and LSTMs have their own advantages, as they are capable of dealing with spatial and temporal data, respectively, but they also lack the properties of one another. To that end, Shi et al. (2015) proposed the convolutional LSTM in order to leverage both spatial and temporal information from the input data in the model. Their model is based on a modified LSTM presented in Gers et al. (2002) and follows the formulation of Graves (2014). The slight difference between the modified and the original LSTM is the inclusion of “peephole connections” meaning that in the equations for input and forget gates, a term is added which includes the cell state from the previous time step, and the information of the current state is added to the output gate. In other words, the LSTM becomes more connected increasing the awareness of the cell state information across different components, which in turn, enhances network performance[42]. The governing equations for the convolutional LSTM are as follows, in line with the LSTM notation. Recall also the notation for convolution: the internal matrix multiplications of LSTM are exchanged with convolution operations:

[40] The number of learnable parameters can be derived from $4hi + 4h + 4h^2 = 4(h(i+1)) + h^2$ where h denotes the number of hidden units of the LSTM layer and i the number of input features. $4hi$ corresponds to the dimensions of the four weight matrices applied to the inputs at each gate, $4h$ refers to the dimensions of the four bias vectors and $4h^2$ represents the dimensions of the weight matrices applied to the outputs at the previous timestep.

[41] GRUSLYS et al. (2016) present a more memory-efficient BPTT algorithm which uses dynamic programming for balancing the trade-off between storing and recomputing intermediate results.

[42] The convolutional LSTM used in the experiments does not include the peephole connections due to the limitation of the API leveraged in implementations. For details, see https://keras.io/layers/recurrent/.

$$f_t = \sigma\left(W_{f,x} * x_t + W_{f,h} * h_{t-1} + W_{f,c} * s_{t-1} + b_f\right) \tag{13}$$

$$i_t = \sigma\left(W_{i,x} * x_t + W_{i,h} * h_{t-1} + W_{i,c} * s_{t-1} + b_i\right) \tag{14}$$

$$\tilde{s}_t = \tanh\left(W_{\tilde{s},x} x_t + W_{\tilde{s},h} h_{t-1} + b_{\tilde{s}}\right) \tag{15}$$

$$s_t = f_t \circ s_{t-1} + i_t \circ \tilde{s}_t \tag{16}$$

$$o_t = \sigma\left(W_{o,x} * x_t + W_{o,h} * h_{t-1} + W_{o,c} * s_t + b_o\right) \tag{17}$$

$$h_t = o_t \circ \tanh(s_t) \tag{18}$$

As it can be seen, there exist an additional set of weights for each gate equations. Note also that one of the drawbacks of LSTM in handling spatiotemporal data, which is commonly presented in three dimensional tensors for dynamical systems, is that the inputs must be flattened out to 1D vectors before processing thus all spatial information in the original data will be lost. Therefore, the equations above, are capable of dealing with 3D tensors where last two dimensions are spatial dimensions, namely the rows and columns.

4.5 Remarks on deep learning

The history of deep learning began already in 1940s and since then, it has gone by many names. The current resurgence under the name of deep learning began in 2006 and the deep learning renaissance can be dated only 5 to 10 years back. Until 2006, deep networks were generally believed to be highly difficult to train requiring large computational costs when comparing to the hardware available at the time. Then, due to algorithmic advances made in training neural networks, deep neural networks started to outperform competing intelligent systems based on other machine learning technologies as well as hand-designed functionality (BROWNLEE (2019b) and GOODFELLOW (2016)). Nowadays, deep learning-based technologies are all around us.

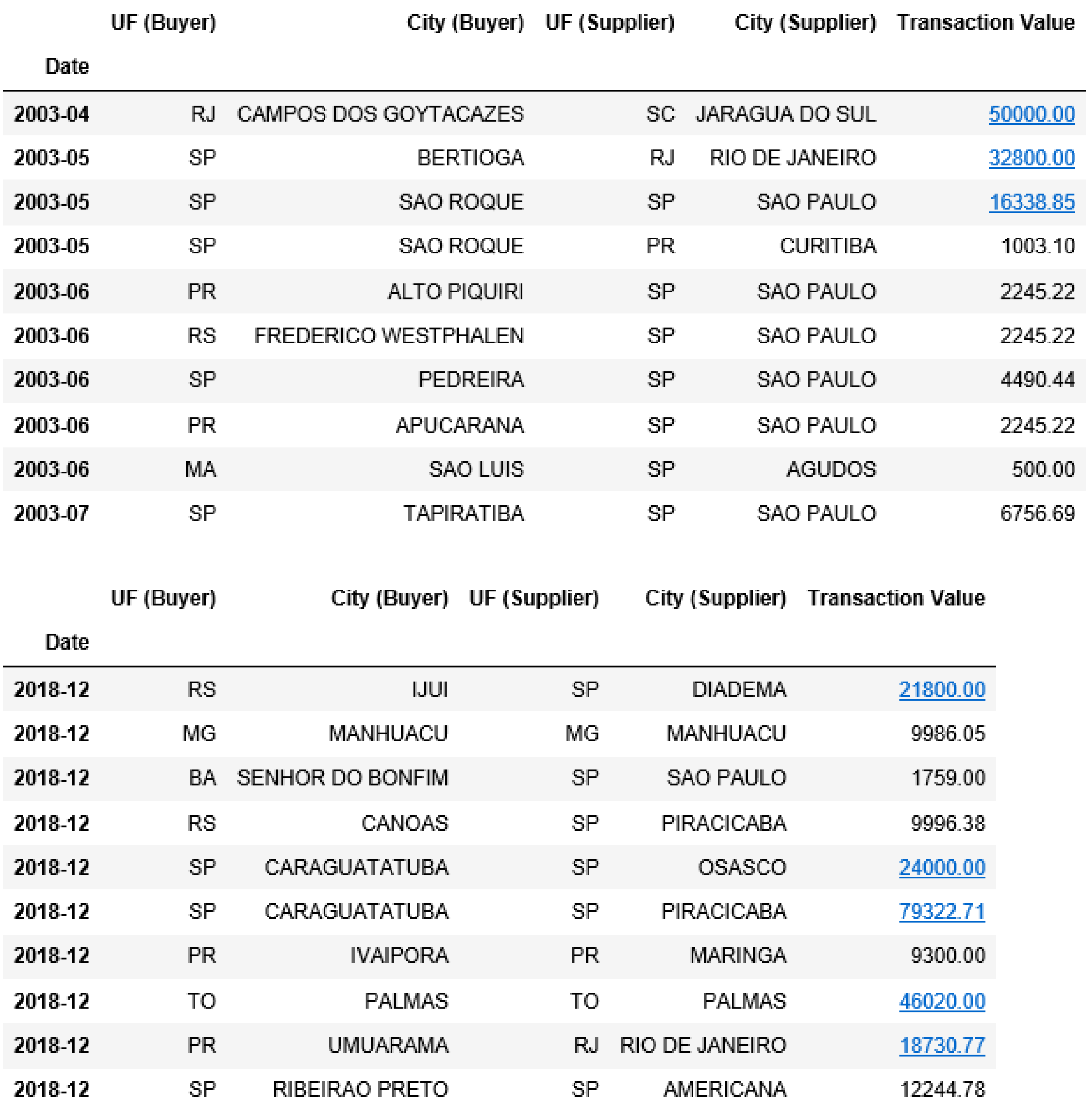
The learning algorithms reaching and exceeding human performance on complex tasks today are almost identical to the algorithms that were used to solve problems in the 1980s. Key reasons why neural networks are wildly successful after enjoying relatively little success since the 1980s include having the computational resources to run much larger models, the availability of data and further advances in algorithm development. Perhaps the most dramatic moment in the history of deep learning was when a CNN won the largest contest in object recognition in 2012 with over 10 percent error rate margin and since then, these competitions are consistently won by deep convolutional nets (KRIZHEVSKY et al. (2012) and GOODFELLOW (2016)). Deep learning techniques[43] have also been successful in other applications winning numerous contests in pattern recognition and machine learning (SCHMIDHUBER (2014)). In general, deep learning is being widely used to automate processes, improve performance, detect patterns, and solve problems.

[43] The methods reviewed in this section, while being current state-of-the-art, constitute only a small portion of available and developed deep learning architectures in the literature. For a summary of different architectures, see e.g. VAN VEEN and LEIJNEN (2019).

5 METHODOLOGY

5.1 Data

As discussed earlier, the data for the empirical study was provided by BNDES. The main two basic entities to consider are buyers and suppliers in Brazil. A buyer is a company that owns and uses the BNDES Credit Card. A supplier is a company registered to the network of suppliers (in the BNDES Credit Card's portal of operations) offering products that follow the standards of BNDES. A supplier can be a producer or a distributor selling items fabricated by others. The data records transactions between suppliers and buyers inter- and intrastate, where an observation consists of i) a timestamp in month-year format, ii) from which state and iii) city the purchase was made (buyer), iv) to which state and v) city the money was transferred (supplier), and finally vi) the transaction value in Brazilian reals. Figure 5 gives a glimpse on the data structure by displaying the first and last 10 observations from the dataset. Further descriptive statistics are provided in the next chapter. For following sections, the city-specific information is excluded.



	UF (Buyer)	City (Buyer)	UF (Supplier)	City (Supplier)	Transaction Value
Date					
2003-04	RJ	CAMPOS DOS GOYTACAZES	SC	JARAGUA DO SUL	50000.00
2003-05	SP	BERTIOGA	RJ	RIO DE JANEIRO	32800.00
2003-05	SP	SAO ROQUE	SP	SAO PAULO	16338.85
2003-05	SP	SAO ROQUE	PR	CURITIBA	1003.10
2003-06	PR	ALTO PIQUIRI	SP	SAO PAULO	2245.22
2003-06	RS	FREDERICO WESTPHALEN	SP	SAO PAULO	2245.22
2003-06	SP	PEDREIRA	SP	SAO PAULO	4490.44
2003-06	PR	APUCARANA	SP	SAO PAULO	2245.22
2003-06	MA	SAO LUIS	SP	AGUDOS	500.00
2003-07	SP	TAPIRATIBA	SP	SAO PAULO	6756.69

	UF (Buyer)	City (Buyer)	UF (Supplier)	City (Supplier)	Transaction Value
Date					
2018-12	RS	IJUI	SP	DIADEMA	21800.00
2018-12	MG	MANHUACU	MG	MANHUACU	9986.05
2018-12	BA	SENHOR DO BONFIM	SP	SAO PAULO	1759.00
2018-12	RS	CANOAS	SP	PIRACICABA	9996.38
2018-12	SP	CARAGUATATUBA	SP	OSASCO	24000.00
2018-12	SP	CARAGUATATUBA	SP	PIRACICABA	79322.71
2018-12	PR	IVAIPORA	PR	MARINGA	9300.00
2018-12	TO	PALMAS	TO	PALMAS	46020.00
2018-12	PR	UMUARAMA	RJ	RIO DE JANEIRO	18730.77
2018-12	SP	RIBEIRAO PRETO	SP	AMERICANA	12244.78

Figure 5: First and last 10 observations displayed from the BNDES Credit Card dataset.

5.2 Setup and data preparation

In the experiments, the regression problem of forecasting multiple steps ahead with uni- and multivariate inputs is being considered. In total of five deep learning methods are implemented in the univariate setting and three methods in the multivariate setting. The forecasting performance of the deep learning methods are compared to a baseline which is formed by ARIMA, as it was commonly used in the reviewed works (e.g. SIAMI-NAMINI (2018); SUN et al. (2018); MAGGIOLO e SPANAKIS (2019)).

The univariate setup is constructed by transforming the raw data into 189 monthly observations[44], where each value represents the sum of all transactions[45] made in the corresponding month. The deep learning models implemented with univariate input include DNN, CNN and LSTM as well as CNN+LSTM[46], which is a hybrid from the two models meaning that the output from the CNN is entered as an input to an LSTM, and the convolutional LSTM, which inherently employs convolutions as part of the LSTM model. The CNN+LSTM hybrid was the basis for the approach proposed by Maggiolo and Spanakis (2019), and the authors also argued that while LSTM is able to model long-term and infrequent dependencies over time, CNN can model short-term highly frequent patterns. Thus, it would be interesting to see how the hybrid model performs when compared to the convolutional LSTM which presumably has similar characteristics. One key difference between the two methods, in addition to how convolutions are leveraged, is that the hybrid model is computationally more expensive.

The multivariate setup is inspired by the methodologies presented in Triepels et al. (2017) and Sezer and Ozbayogly (2018). Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of n states participating in the credit card payments and $T = < t_1, t_2, \ldots, t_m >$ an ordered set of m consecutive time intervals of equal duration, where $t_1 = [\tau_0, \tau_1), t_2 = [\tau_1, \tau_2)$ and so on. The amount of money transferred from one state to another through the credit card system over time is recorded. Let $D = \{A^{(1)}, A^{(2)}, \ldots, A^{(m)}\}$ be a set of m matrices where each $A^{(k)} \epsilon D$ is the n by n matrix:

$$A^{(k)} = \begin{bmatrix} a_{11}^{(k)} & \cdots & a_{1n}^{(k)} \\ \vdots & \ddots & \vdots \\ a_{n1}^{(k)} & \cdots & a_{nn}^{(k)} \end{bmatrix} \tag{19}$$

Each element $a_{ij}^{(k)} \epsilon [0, \infty)$ denotes the total amount of money transferred from state s_i to s_j in time interval t_k, including the cases where $i = j$. The matrix $A^{(k)}$ is essentially an adjacency matrix with the total value of payments as link weights, forming a payment network at time interval t_k. Moreover, a vectorised representation $a^{(k)}$ from the adjacency matrix $A^{(k)}$ can be derived, which results in:

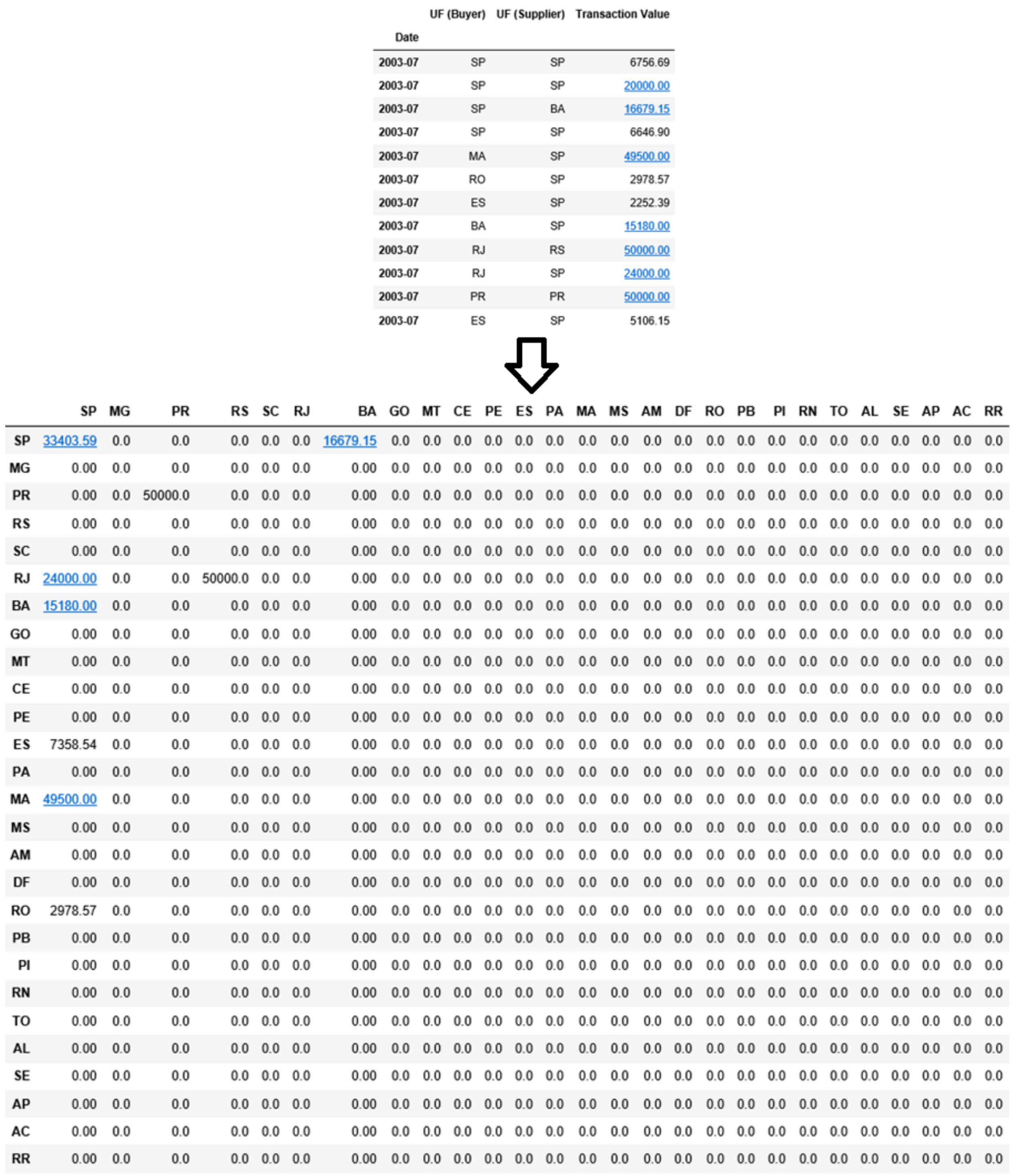
[44] The original data spanned from 2003-04 to 2018-12, resulting in 15 years and 9 months, i.e., 189 months.
[45] The sum is formed from the transaction values. Transaction data aggregation is present in other studies related to BNDES Credit Card, see e.g. MARTINI and TEIXEIRA (2016) and CORSEUIL (2019).
[46] Note that in the univariate case CNN and CNN+LSTM employ one-dimensional convolutions.

$$a^{(k)} = \left[a_{11}^{(k)}, \dots, a_{n1}^{(k)}, \dots, a_{1n}^{(k)}, \dots, a_{nn}^{(k)}\right]^T \tag{20}$$

where $a^{(k)}$ is a column vector of size n^2 consisting of all columns of $A^{(k)}$ vertically enumerated. In practice, 189 adjacency matrices of size 27x27 and corresponding vector representations of size 729 can be formed from the BNDES data in the way described above[47]. The resulting adjacency matrices and vector representations are used as an input for deep learning models in the multivariate setting. This allows for including state-specific information in the model, in contrast to the univariate setting. The models implemented with the vectorised input were the convolutional LSTM and bidirectional[48] LSTM, and the adjacency matrices were fed to a CNN. One could think that the CNN processes the adjacency matrices as 2D-images. Below is a sample adjacency matrix constructed from the observations dated to 2003-07:

[47] The validation phase in model development revealed that ordering the columns and rows of adjacency matrices (states) according to the sum of transactions in each state (descending) for the training period (defined on p. 43) resulted in enhanced performance and stability. Other schemes could be also applied.

[48] The bidirectional LSTM is an LSTM at its core, but the input sequence is processed both forward and backward directions.



Date	UF (Buyer)	UF (Supplier)	Transaction Value
2003-07	SP	SP	6756.69
2003-07	SP	SP	20000.00
2003-07	SP	BA	16679.15
2003-07	SP	SP	6646.90
2003-07	MA	SP	49500.00
2003-07	RO	SP	2978.57
2003-07	ES	SP	2252.39
2003-07	BA	SP	15180.00
2003-07	RJ	RS	50000.00
2003-07	RJ	SP	24000.00
2003-07	PR	PR	50000.00
2003-07	ES	SP	5106.15

	SP	MG	PR	RS	SC	RJ	BA	GO	MT	CE	PE	ES	PA	MA	MS	AM	DF	RO	PB	PI	RN	TO	AL	SE	AP	AC	RR
SP	33403.59	0.0	0.0	0.0	0.0	0.0	16679.15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MG	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PR	0.00	0.0	50000.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RS	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SC	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RJ	24000.00	0.0	0.0	50000.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
BA	15180.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
GO	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MT	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
CE	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PE	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ES	7358.54	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PA	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MA	49500.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MS	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AM	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
DF	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RO	2978.57	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PB	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PI	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RN	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
TO	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AL	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SE	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AP	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AC	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RR	0.00	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 6: An example adjacency matrix constructed from the observations dated to 2003-07.

Note that while the given sample in Figure 6 is sparse, there are also matrices that are relatively dense. For example, in the corresponding adjacency matrix for 2014-10 which consists almost 31000 observations, more than two thirds are nonzero values.

In both settings, the purpose is to forecast the monthly sum of transaction values in the future. In order to use machine and deep learning methods for time series forecasting, the forecasting problem is commonly formulated as a supervised learning problem. This means that given a sequence of numbers for a time series dataset, the data is restructured by using previous timesteps as input variables and the next timestep or multiple steps as the output, depending on the problem formulation. The use of prior time steps to predict the next time step(s) is called the sliding window or the rolling forecast method, or in terms of statistics, a lag or lag method. As it is assumed that there are dependencies between observations in the dataset, the rolling forecast method is required (BROWNLEE (2019a); SIAMI-NAMINI et al. (2018)).

A supervised learning algorithm requires that data is provided as a collection of samples, where each sample has an input component and an output component. Consider for example the univariate time series represented as a vector of observations $[1, 2, 5, 6, 4, 3, 8, 9, 10, 12]$, depicting some sums for ten timesteps. Then, if using three prior timesteps to predict the next three timesteps, the first and second sample would look like $[1, 2, 5]$ $[6, 4, 3]$ and $[2, 5, 6]$ $[4, 3, 8]$, and so on:

$$\begin{matrix} [1 & 2 & 5] & [6 & 4 & 3] \\ [2 & 5 & 6] & [4 & 3 & 8] \\ [5 & 6 & 4] & [3 & 8 & 9] \\ [6 & 4 & 3] & [8 & 9 & 10] \\ [4 & 3 & 8] & [9 & 10 & 12] \end{matrix}$$

The first row in the left side and the corresponding row in the right side is the first input-output sample, the second row similarly the second input-output sample, and so on. Note that only one variable is considered. The supervised learning algorithm can be trained by using these input-output samples. It is important to maintain the order of samples when training the model. This way the model learns from input samples what it is expected to output, and the procedure also defines what the model requires for making forecasts. In order words, if the model is trained with the input-output samples described above, it takes three values as input to predict three timesteps ahead (BROWNLEE (2019a)).

For multivariate setting, assume a multivariate time series represented as:

$$\begin{array}{cccc}
[x_1 & x_2 & x_3 & y] \\
[1 & 2 & 5 & 8] \\
[2 & 5 & 6 & 13] \\
[5 & 6 & 4 & 15] \\
[6 & 4 & 3 & 13] \\
[4 & 3 & 8 & 15] \\
[7 & 4 & 9 & 20]
\end{array}$$

where the first three columns are three variables and the last column is the sum of these three variables. A row represents an observation. Now, by using the three variables, the sum is predicted. Then, if using three prior timesteps to predict the next three timesteps, the input-output samples are:

$$\begin{array}{cccc}
[1 & 2 & 5] & \\
[2 & 5 & 6] & \\
[5 & 6 & 4] & [15 \quad 13 \quad 15] \\
[2 & 5 & 6] & \\
[5 & 6 & 4] & \\
[6 & 4 & 3] & [13 \quad 15 \quad 20]
\end{array}$$

During training, when the first three rows from the left side are entered as the input for the model, it is expected to output the numbers $[15\ 13\ 15]$. This is the first input-output sample, and the second one follows similarly. Note that in both cases, uni- and multivariate, the number of prior timesteps considered and the forecast horizon is related to the size of the whole sample. For example, even though the univariate series had ten observations, five input-output samples were created, and in the multivariate case, the six observations lead to only two input-output samples (BROWNLEE (2019a)). The time series forecasting problem for the both cases is transformed to a supervised learning problem leveraging the schemes presented. Adjacency matrices do not raise an issue, since the vectorised[49] representations can be easily converted back to the matrix form.

Finally, instead of making a forecast once, forecasting models are commonly tested and validated by making multiple sequential or rolling forecasts further to the future against real data. This is called as walk-forward validation. This is convenient due to the fact that as newer information becomes available, the earlier fitted models may not be valid anymore. The walk-forward validation also gives the best opportunity for the model to make good

[49] For a 729-long sequence, consider $\{x_n\}_{n=1}^{729}$ and y as the sum for that month. Note also that $\sum_{n=1}^{729} x_n = y$.

forecasts at each time step (BROWNLEE (2019a)). It is worth mentioning too that the forecasts should be evaluated with unseen test data.

In the experiments, the last 36 months are left as test data and the training data (153 observations)[50] is transformed in a way that three prior months are required for creating a forecast three months ahead. This results in 12 separate forecasts with the test data[51]. During walk-forward validation, the ARIMA model is re-fitted before each forecast is made with all data it has available in the timestep in question[52]. The deep learning models are given the most recent three observations before each prediction is made and they are fitted only once before the first forecast is made due to their computational cost. On a side note, one of the reasons why predictions are made for multiple steps ahead is due to the findings of Sha et al. (2018) and Leccese (2019): DNN and LSTM do not demonstrate predictive abilities when forecasting one step ahead. Therefore, for testing the models' capabilities, using multiple steps ahead is more adequate. On the other hand, as the size of the training data is relatively small, the timesteps for back and forth should not be increased too much, as it affects on the number of input-output samples and ideally, more samples to train on is better for the deep learning models. Finally, while only for convenience, three-steps ahead gives quarterly forecasts which can be beneficial in businesses' point of view.

5.3 Performance metrics and statistics

One of the most important and also challenging aspects of forecasting problems is to determine how much the forecasts were off from the actual values in a reliable and informative manner. In the experiments, RMSE, MAE, MAPE, MSLE and MDA (Mean Directional Accuracy) are used for measuring forecasting performance as well as the DM-test[53] for verifying that two forecasts are statistically different from each other. The DM-test can be commonly encountered from the forecasting literature for checking the statistical significance of forecasts (e.g. CARRIERO et al. (2019) and DOMIT et al. (2019)). Assume

[50] The split to training and test sets is often guided by proportional rules such as 70:30, 80:20 and 90:10, depending on the size of the original sample. For convenience, data until 2016 is selected for training (of which last 12 months are used for validation) and the rest is left for test data (by the end of 2018), which results into splitting rule around 80:20.

[51] Note that some of the models, as will be discussed in section 5.4, had to be implemented forecasting four steps ahead (requiring four prior months) due to their construction, resulting in 9 separate forecasts with the test data.

[52] The approach where all observations available up to each forecasting origin are used, is also known as expanding samples, as described in Carriero et al. (2019).

[53] The modified test proposed by Harvey et al. (1997) is used, since predictions are done multiple steps ahead and the sample is relatively small. See the paper also for the original DM statistic.

that y_t is the realised value at time t and $\hat{y}_t$ the predicted value. Then, the metrics are as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \tag{21}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \tag{22}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\frac{|y_t - \hat{y}_t|}{y_t}100\% \tag{23}$$

$$\mathrm{MSLE} = \frac{1}{n}\sum_{t=1}^{n}\log\left(\frac{\hat{y}_t + 1}{y_t + 1}\right)^2 \tag{24}$$

$$\mathrm{MDA} = \frac{1}{n}\sum_{t=1}^{n}\mathbf{1}_{sign(y_t - y_{t-1}) == sign(\hat{y}_t - y_{t-1})} \tag{25}$$

And the modified DM-test statistic S, where DM is the original test statistic, h is the horizon and n the amount of resulted errors between two forecasts:

$$\mathrm{S} = \mathrm{DM}\sqrt{\frac{1}{n}[n + 1 - 2h + n^{-1}h(h-1)]} \tag{26}$$

The null hypothesis of the test is that the two forecasts has equal predictive accuracy and the alternative hypothesis depends on whether the test is run as one- or two-sided. The statistic is compared to the critical values from Student-t distribution with $(t - 1)$ degrees of freedom.

The selected measures are not without their caveats. In fact, many error measures have drawbacks that can lead to inaccurate evaluation of forecasting results. Therefore, more than one measure is highly recommended to be used but still, it is a difficult task to choose the most appropriate set of measures. RMSE, MAE and MAPE are selected for their frequent usage in forecasting papers (e.g. CARRIERO et al. (2019) and CHONG et al. (2017) and SUN et al. (2018)). However, these error measures are sensitive to different scales in the data and to outliers, RMSE more than MAE. They can also give different results on different fractions of data (SHCHERBAKOV et al. (2013)). With MAPE, while being easy to interpret, there exist the danger of division by zero, it is sensitive to outliers, it is biased towards methods whose predictions are too low and it is not symmetric in a sense that interchanging y_t and $\hat{y}_t$ does not lead to the same answer, despite the fact that the absolute error remains the same[54]. Therefore, in addition to MAPE, MSLE is also applied to the analysis as suggested by Tofallis (2015). MSLE is dimensionless and it treats positive and negative errors with an appropriate symmetry, and the possible ranges are equal on both sides. Lastly, the MDA is rather simple but effective measure on determining whether the forecasts have captured the overall trend of the actual data. It essentially illustrates how often the model got the trend right by comparing the sign produced by subtracting the current prediction from the previous one, to the sign yielded by differencing the corresponding actual data.

5.4 Modelling

Traditionally, the ARIMA model is fitted manually via the Box-Jenkins methodology. In this case, as explained before, 12 sequential three-steps-ahead out-of-sample forecasts are made and after each forecast, the realised data along with the full history is made available for producing the next forecast with a re-fitted ARIMA model. As the nature of the data generating process may change as time evolves, the optimal solution would be choosing the model parameters $(\mathrm{p}, \mathrm{q}, \mathrm{d})$[55] before each forecast is made. However, this is rather cumbersome, thus the model parameters are determined using the training data and these parameters are then used each time when the model is re-fitted for developing a new forecast. This is also in line with the deep learning models, where the model fitting occurs only once. Furthermore, in order to determine which model parameters results in best performance over the whole forecasting scenario on average, a systematic grid search[56] guided by correlograms and statistical testing is employed. The grid search is also run for Box-Cox transformed data, where the transform is re-executed before each forecast during

[54] Consider the example provided in Makridakis (1993), where the forecast error is 50 units in both cases: $\frac{|150-100|}{150} \neq \frac{|100-150|}{100}$.

[55] Recall that $\mathrm{ARIMA}(\mathrm{p}, \mathrm{d}, \mathrm{q})$ model is an approach where dependencies between observations are captured by an autoregressive process with p lags, dependencies between residual error terms by a moving average process with q lags and d represents the order of differencing required.

[56] For p and q, the list of parameters [0, 1, 2, 3, 4, 5, 6] is considered, and for d $[0, 1, 2]$. From these numbers, every possible combination $(\mathrm{p}, \mathrm{d}, \mathrm{q})$ is evaluated against the metrics introduced earlier as well as AIC and BIC. This results in 147 individual configurations. Parameter settings $(12{,}0{,}0), (12{,}1{,}0), (12{,}0{,}1), (12{,}1{,}1)$ are also evaluated for comparison and completeness.

model fitting. The grid search may not result in the most optimal model, but it provides an effective representative baseline for analysis purposes. Nevertheless, for completeness, evidence for inherent characteristics present in raw and transformed training data are investigated, and the results are provided in Appendix I. Interestingly, after grid search, the best performing model on average over the in-sample validation forecasting period, taking the parsimony principle into account, is ARIMA(4,0,0) with raw input data. As the correlograms and stationarity tests indicated, at least differencing would have been expected. However, all difference-equipped models performed significantly worse than models with no differencing. It is worth mentioning that the data to be modelled in this case is relatively difficult due to the suggested presence of nonlinearity, heteroskedasticity, heavy-tailed distribution, serial correlation and outliers by Appendix I.

For each of the deep learning methods, a similar grid search framework is developed in the same sense that the model's forecasting performance is evaluated against the metrics as before. However, there is more freedom now in the validation phase and in finding the optimal parameters as by definition, neural networks are capable of learning arbitrary complex mappings from inputs to outputs. In other words, neural network models may not require transformed and stationary inputs. Moreover, they are capable of directly inferring trend and seasonality, among other characteristics commonly present in real-world time series data. Deep learning methods are also highly versatile being able to handle uni- and multivariate inputs as well as multi-step outputs, and different types of methods can automatically extract possibly nonlinear spatial and temporal dependencies (BROWNLEE (2019a)). On the other hand, this freedom comes with many parameters that can be optimised when fitting deep learning models. There does not exist deterministic rules on how to optimize deep learning models and which model architectures deliver the best results meaning that dealing with deep learning methods is somewhat of a dark art (BROWNLEE (2019b)). In many scientific papers in the field, the model parameters are found by trial-and-error, grid search or thorough experimentation, or applying all of them at the same time. However, for starters, one can find good suggestions for model parameters by, for example, finding architectures used in the literature, and there also exist a myriad of recommendations that have been found to be useful in practice when optimising deep learning models.

While grid search helps in finding model parameters for deep learning methods, it is still mechanical. The performance of a deep learning model is uniquely application dependent. Oftentimes, the deep learning practitioner must assess quanti- and qualitatively the type of problem that is possibly occurring and evaluate proper remedies for the issue.

The poor performance of a deep learning based neural network model is generally accounted to problems with learning, generalisation and predictions (BROWNLEE (2019b)). In other words, the deep learning model should be able to effectively learn from training data without overfitting. This practically means that as neural networks are universal (continuous) function approximators[57], they tend to fit any dataset ideally. This may not be desired behaviour since real-world data commonly contains noise and underlying continuity cannot be guaranteed in a training sample, especially when dealing with small sample sizes. In essence, it cannot be trusted that the samples at hand fully reflect the underlying physical system or environment, so it is better to not fit too strongly but to generalize and try to learn the most essential features. Overfitting manifests when a model is performing well with a training set but performance drops against the test set. Moreover, the trainable model parameters such as weights are commonly initialised in a randomised way, and by construction more randomness may be injected in the model. This means that neural networks are stochastic in nature which may result in high variability in predictions. To summarize, an ideal model performs well with training and test data, gives stable and reliable results, while learning fast. This was also the main goal when fitting the models, and a couple of techniques were used in trying to accomplish it.

Firstly, the modern AMSGrad was selected as the gradient-based optimisation algorithm due to the fact that while Adam is widely used, there exists a stochastic convex optimisation problem for which Adam does not converge to the optimal solution (REDDI et al. (2018)). AMSGrad also attempts to avoid some of the pitfalls recognised in other gradient-based algorithms for stochastic optimisation. The best-performing loss functions as per learning rate, stability and metrics, were MAPE and MAE. Model capacities were systematically varied in terms of layer and node sizes. As it is common in the literature, data scaling techniques such as logarithmic transformation and minmax-scaling to [0,1] range were tested also, as well as the combination of both[58]. Moreover, some regularisation techniques were applied and evaluated in different components of the models. Intuitively, regularisation means penalising high-valued weights in proportion to the norm of the model weights, resulting in changes in the loss function. Practically, all the weights are summed up

[57] According to the universal approximation theorem, arbitrary decision functions can be arbitrarily well approximated by a neural network with only one internal layer and a continuous sigmoidal nonlinearity (Cybenko (1989)). In other words, neural nets can be used to approximate any continuous process. However, the theorem focuses only on existence, which leaves questions such as how many neural nodes are required to yield an approximation of a given quality unanswered. Nevertheless, the approximation properties of neural nets are very powerful.

[58] See e.g. Shah et al. (2018), Du et al. (2018) and Triepels et al. (2017).

and either the absolute value is taken, or the sum is squared (L1 and L2 norms, respectively), and the resulting value is added to the overall loss. The regularisation parameter determines the magnitude of the penalisation effect. Regularisation promotes smaller weights and, in turn, lower complexity. Dropout and input noise were also applied: in the former, units such as neurons (and the parameters attached to it) are probabilistically ignored from the model and the latter reduces the model's dependence on specific input values. The batch sizes and number of epochs were also systematically searched[59]. Due to the stochastic nature of the models, 15 consecutive stable predictions were run in the experiments, and then the final prediction was formed by taking the means[60]. Finally, there exists a rule-of-thumb that when developing deep learning models, there should be ten times more data available than the model has trainable parameters. This further highlight the small sample size and the risk of overfitting.

The following list describes the univariate model architectures used in the analysis:

- All models use AMSGrad as optimizer, ReLU as activations and log-transformed input data.
- The ANN has 5 hidden layers with [10, 20, 10, 20, 10] neurons, respectively. The last four layers are equipped with L2(10^{-3}) regularizer. The loss function used is MAE. The number of epochs is set up to 150 and the batch size to 24. Total trainable parameters equal to 933.
- The CNN has two one-dimensional convolutional layers with 512 and 256 filters, where kernel sizes are 6 and 2, respectively. The first layer has the same padding. After the convolutional layers, 1D-maxpooling is employed with a pool size of 2 and valid padding. Then, before output, the data is flattened out and fed through a 3-layer [10, 10, 10] ANN. The number of epochs is 300 and the batch size 16. The loss function used is MAPE. There are 268,807 trainable parameters in total.
- The LSTM has one hidden layer with 80 memory cells. Before the hidden layer's activation, Gaussian Noise (0.65) layer is applied. Before output, there is also one additional basic hidden layer with 10 neurons, before which dropout layer (0.05)

[59] Batch size refers to the number of samples used at a time from the training set to update the model weights. One training epoch means one pass through the training set. For an example, say 30 observations are used for training. Then, using a batch size of 10 results in three model updates per epoch. Lower batch sizes result in less accurate and noisy estimates, but they offer a regularising effect, lower generalisation error, faster learning and consume less memory.

[60] Achieving stability with more complex models tends to be challenging which the experiments also demonstrated. Only 15 runs were run due to limited computational resources.

is applied. The model is trained with 300 epochs and a batch size of 24. The loss used is MAPE. The model requires 27,083 trainable parameters.

- For the CNN+LSTM, by the way it is implemented due to LSTM, the number of prior timesteps had to be increased to 4. Similarly, the forecasting horizon was increased to 4 timesteps. The model has three 1D-convolutional layers with same padding, each having 64 filters with kernel size of 2. Then, 1D-maxpooling with pool size of 2 is employed and data is flattened out for the one-layer LSTM with 20 memory cells. Before output, there is a basic hidden layer with 20 neurons. The loss function used is MAPE. The model is trained with 200 epochs and a batch size of 24. There are 24,008 trainable parameters in this specification.
- The timestep-setup had to be also changed for the convolutional LSTM to 4 predictions for each 4 prior timesteps. The model has one ConvLSTM layer with 100 filters and a kernel size of (1,2). L2(10^{-4}) regularizer is employed also in the first layer for kernel weights. A dropout (0.15) layer precedes flattening the data, after which the data goes through a two-layer neural network with 20 neurons each layer. A dropout (0.05) layer is also between the two layers. 200 epochs with batch size of 32 was selected. The loss function used is MAE. There are 83,724 trainable parameters in total.

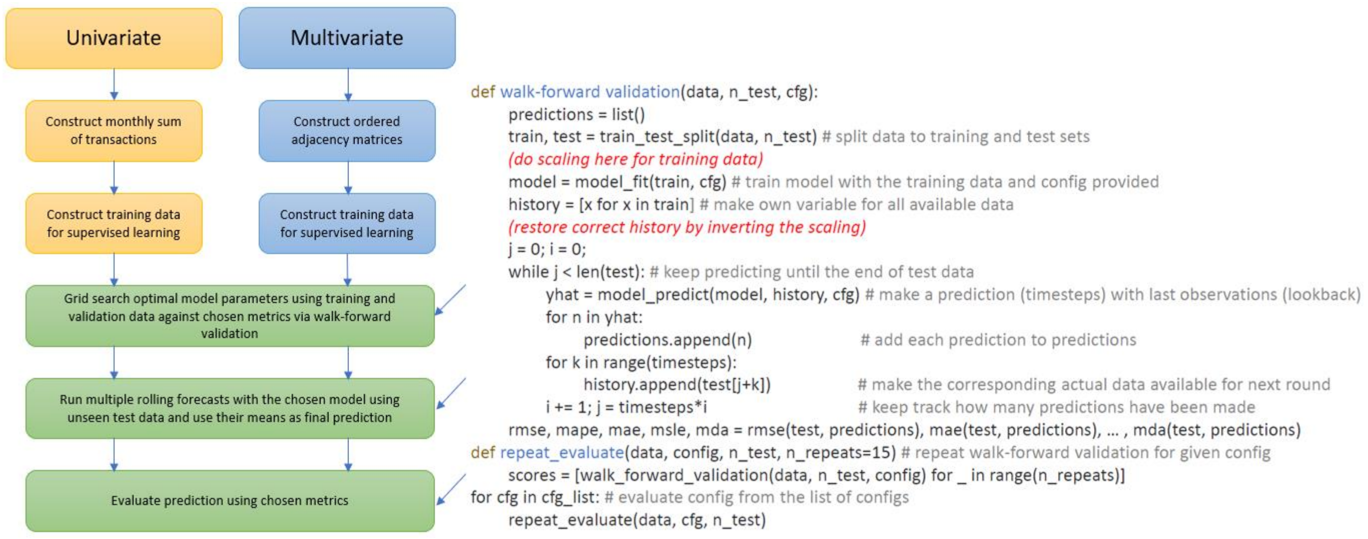
The multivariate models have the following specifications:

- All models use AMSGrad as optimizer, MAE as loss function, ReLU as activations and raw input data.
- The convolutional LSTM uses again 4 prior time steps to predict 4 steps ahead. The model has one ConvLSTM layer with 20 filters and a kernel size of (1,2). L2(10^{-5}) regularizer is employed also in the first layer for kernel weights. A dropout (0.15) precedes flattening the data, after which the data goes through a three-layer neural network with [40, 40, 10] structure neuron-wise. A dropout (0.10) is again applied between the first two layers. The model is trained using 120 epochs and with a batch size of 16. The loss function used is MAE. The number of trainable parameters is 122,854.
- The CNN model, which takes the adjacency matrices as input, is constructed as follows. The first convolutional layer has 64 filters with a kernel size of (3,3). Then, three convolutional-2D-maxpooling layers are added, where first two layers has 128 filters and the last one 64. Pool-size is kept at (2,2) and kernel size remains at (3,3). Then a dropout (0.2) is added, and the data gets flattened out for a 128-

neuron basic hidden layer. Before the output, a dropout (0.1) is added. The model is trained using 100 epochs and a batch size of 16. This model has 305,731 trainable parameters.

- The bidirectional LSTM has two hidden layers with 90 memory cells and L2(10^{-3}) regularizer each. The results from the bidirectional layers are merged via averaging. 350 epochs are used with a batch size of 24. In this case, 720,811 trainable parameters are in effect.

For clarification, Figure 7 represents a conceptual high-level flowchart for applying the methodology presented:


Univariate
Multivariate
Construct monthly sum of transactions
Construct ordered adjacency matrices
Construct training data for supervised learning
Construct training data for supervised learning
Grid search optimal model parameters using training and validation data against chosen metrics via walk-forward validation
Run multiple rolling forecasts with the chosen model using unseen test data and use their means as final prediction
Evaluate prediction using chosen metrics

```
def walk-forward validation(data, n_test, cfg):
    predictions = list()
    train, test = train_test_split(data, n_test) # split data to training and test sets
    (do scaling here for training data)
    model = model_fit(train, cfg) # train model with the training data and config provided
    history = [x for x in train] # make own variable for all available data
    (restore correct history by inverting the scaling)
    j = 0; i = 0;
    while j < len(test): # keep predicting until the end of test data
        yhat = model_predict(model, history, cfg) # make a prediction (timesteps) with last observations (lookback)
        for n in yhat:
            predictions.append(n)                # add each prediction to predictions
        for k in range(timesteps):
            history.append(test[j+k])            # make the corresponding actual data available for next round
        i += 1; j = timesteps*i                  # keep track how many predictions have been made
    rmse, mape, mae, msle, mda = rmse(test, predictions), mae(test, predictions), ... , mda(test, predictions)
def repeat_evaluate(data, config, n_test, n_repeats=15) # repeat walk-forward validation for given config
    scores = [walk_forward_validation(data, n_test, config) for _ in range(n_repeats)]
for cfg in cfg_list: # evaluate config from the list of configs
    repeat_evaluate(data, cfg, n_test)
```

Figure 7: Flowchart illustrating the deep learning-based forecasting scenario with pseudo-code for walk-forward validation/rolling forecasts.

The pseudo-code[61] for walk-forward validation in the figure above is valid for training and validating as well as testing and evaluating deep learning models.

Python (3.4.9) was used in model development. Deep learning models were implemented using Keras (2.2.4) with TensorFlow (gpu-1.14.0) as the backend. ARIMA was implemented using statsmodels (0.10.1).

[61] Note that the purpose of the given concise pseudo-code is to illustrate only the essentials of the procedure.

6 RESULTS AND ANALYSIS

Before going into the results given by the presented methodology applied, descriptive statistics and illustrations about the BNDES data are shortly provided. Recall Figure 5 which demonstrated the data structure. As was discussed, the data contains transactional information with state- and city-specific details and corresponding dates in month-year format. In the dataset, there are buyers and suppliers from all 27 states in Brazil. 5129 different cities can be found from the buyers' side whereas 2452 cities among the suppliers. From 2003-04 to 2018-12 a total of approximately 72.8 billion in reals was transferred in 2103674 transactions, which is also the amount of observations in the whole dataset. The largest amount transferred in the whole period was 28089562.14 from São Paulo to São Paulo in 2014-10, and the minimum of 12 occurred twice (2013-05 and 2014-07), in both cases the buyer being from Londrina, Paraná. In 2013-05, the supplier was from Santo Andre, São Paulo and in 2014-07 from Brasilia, Distrito Federal.

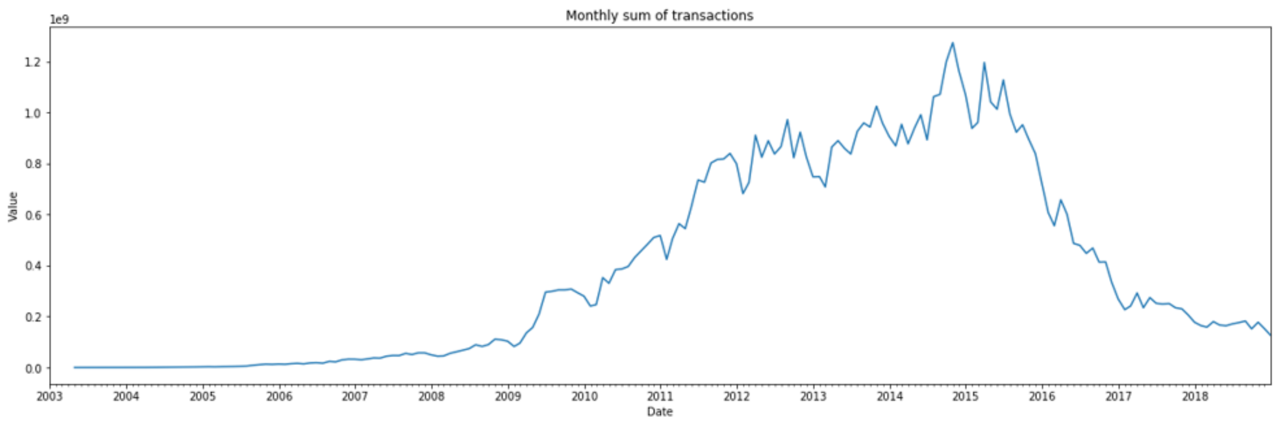


Figure 8: Visualisation of the monthly sum of transactions over the whole period.

Above[62] the monthly sum of transactions is visualised. One can notice that there exists plenty of variability in different periods. As the BNDES Credit Card is relatively young being launched in 2002, the first years were markedly steady until 2009. Then, the flow of payments started to experience steep growth, which interestingly coincides for example with the growth of Brazilian GDP of over 7.5 percent in 2010. The trend remained upwards until 2014, as the growth of GDP decelerated to zero. Around this time, the corruption scandal related to the state-owned oil company Petrobras started to disentangle and shake the

[62] Note that the scale is in billions of reals $(1e9 = 10^9)$. In the upcoming line and bar plots, a similar convention is used.

political atmosphere of Brazil. These shocks eventually propagated strongly throughout the Brazilian economy leading to crisis, which manifested later on as investment failures, plummeting exchange rates and increasing unemployment, to mention a few. These events have seemingly had a strong impact to the progress of payment flows[63]. In the next table, summary statistics from each year are provided:

	Sum	Cum%	Count	Cum%	Mean	Std	Min	25%	50%	75%	Max
2003	1578229	0.0	97	0.0	16270	15719	187	4165	9380	24000	50000
2004	15282494	0.0	952	0.0	16053	14906	535	5000	10000	22992	80841
2005	85277353	0.1	4289	0.3	19883	29720	150	4772	10700	25750	670397
2006	253028715	0.5	11711	0.8	21606	44696	99	4462	10156	25500	1875135
2007	546861278	1.2	23729	2.0	23046	56806	76	4402	10000	25900	2454413
2008	933484383	2.5	36272	3.76	25736	68654	50	4728	11210	28850	4446862
2009	2767410041	6.3	93076	8.1	29733	105697	88	5014	12000	31112	11310453
2010	4737282889	12.8	152522	15.3	31060	133521	18	5053	11941	30500	15940780
2011	8214375864	24.1	231199	26.3	35529	172345	28	5300	12798	338400	23272182
2012	10031153431	37.9	283760	39..8	35351	171338	30	5252	12650	33122	26438670
2013	10626937233	52.5	300324	54.1	35385	168596	12	5443	12729	32682	23336693
2014	12365054030	69.5	315809	69.1	39154	180194	12	5840	13854	35800	28089563
2015	11606817300	85.5	295470	83.2	39283	170479	17	5890	13741	35546	25880041
2016	5733365856	93.3	185487	91.8	30910	114980	22	5400	12000	28711	14318974
2017	2870579913	97.3	99448	96.7	28865	96776	32	5355	11820	27900	9079755
2018	1970043346	100	69529	100	28334	80489	27	5691	12332	28810	5574301
tot.	72758532355		2103674								

Table 1: Summary statistics for each year.

Table 1 demonstrates further how the data exhibits strong trends and differences in scales. The period from 2010 to 2016 is responsible for over 80 percent in terms of money transferred, and the amount of transactions follow closely with 76.5 percent in the same period. Between 2012-2016 alone, 1380850 transactions were made worth of over 50 billion. Next, Tables 2 and 3 focus on state-specific data over the whole period as follows:

[63] Discussions with BNDES personnel revealed that until 2009, the credit card financed mainly equipment acquisitions and had only few card-providing banks in its network. As the Brazilian economy started to grow, the credit card expanded its financing operations beyond equipment and increased the network of credentialised banks. However, as the Brazilian economy entered in recession after 2016, the network of banks diminished leading to a significant decrease in the scale of operations.

	Suppliers			
	States		Cities	
	Sum	Count	Sum	Count
1.	São Paulo (28782582633)	São Paulo (742953)	São Paulo (8342532138)	São Paulo (103650)
2.	Minas Gerais (8244108119)	Rio Grande do Sul (269145)	Betim (3261878892)	Curitiba (45978)
3.	Rio Grande do Sul (7913309698)	Paraná (257566)	São Bernando do Campo (2437651534)	Betim (40341)
4.	Paraná (7797534983)	Minas Gerais (220157)	Curitiba (2045921645)	São Bernando do Campo (38160)
5.	Santa Catarina (5626398560)	Santa Catarina (210456)	Caxias do Sul (1342489116)	Eldorado do Sul (34604)
6.	Rio de Janeiro (1982138690)	Bahia (50597)	São Caetano do Sul (1157775010)	Sapucaia do Sul (32211)
7.	Goiás (1588930773)	Rio de Janeiro (48588)	Porto Alegre (1080269761)	Campinas (31374)
8.	Bahia (1514672313)	Goiás (41935)	Belo Horizonte (1058550021)	Piracicaba (31307)
9.	Mato Grosso (1100287159)	Espírito Santo (37416)	Rio de Janeiro (1039153006)	Caxias do Sul (30783)
10.	Espírito Santo (1082242482)	Pernambuco (31170)	Eldorado do Sul (1008193456)	Maringa (29081)

Table 2: Top 10 states and cities according to the sum of transactions and total count over the whole period (suppliers).

	Buyers			
	States		Cities	
	Sum	Count	Sum	Count
1.	São Paulo (20460963062)	São Paulo (487159)	São Paulo (4964249696)	São Paulo (30310)
2.	Minas Gerais (7266954392)	Minas Gerais (255418)	Rio de Janeiro (1485051080)	Rio de Janeiro (16721)
3.	Paraná (7058415932)	Rio Grande do Sul (208516)	Curitiba (1338856597)	Belo Horizonte (14845)
4.	Rio Grande do Sul (5793658427)	Paraná (201800)	Belo Horizonte (1197110144)	Curitiba (14622)
5.	Santa Catarina (5296387495)	Santa Catarina (171175)	Goiânia (1053324745)	Porto Alegre (13496)
6.	Rio de Janeiro (3487269693)	Bahia (119444)	Porto Alegre (922806651)	Goiânia (13076)
7.	Bahia (3457788586)	Rio de Janeiro (87576)	Brasília (883526412)	Salvador (11427)
8.	Goiás (2724928015)	Goiás (73374)	Fortaleza (818884529)	Fortaleza (10732)
9.	Mato Grosso (2079309415)	Mato Grosso (67707)	Salvador (714912614)	Brasília (10527)
10.	Ceará (1778638851)	Ceará (49863)	Campinas (617754058)	Ribeirão Preto (9759)

Table 3: Top 10 states and cities according to the sum of transactions and total count over the whole period (buyers).

In Tables 2 and 3, the sum and amount of transactions are calculated over the whole period for states and cities differentiating suppliers and buyers. As said before, there are suppliers from 2452 different cities and buyers from 5129, making purchases more diversified by over two-fold when considering the origin. Nevertheless, there are not many differences among the top 10 states and cities from both sides. In other words, states and cities that tend to trade often and high in value do not change. This finding aligns also with the study[64] made by Martini and Teixeira (2016). To further demonstrate payment flows in different periods, the evolution of payments from the top 10 states by total value transferred per month is visualised in Figures 9 and 10:

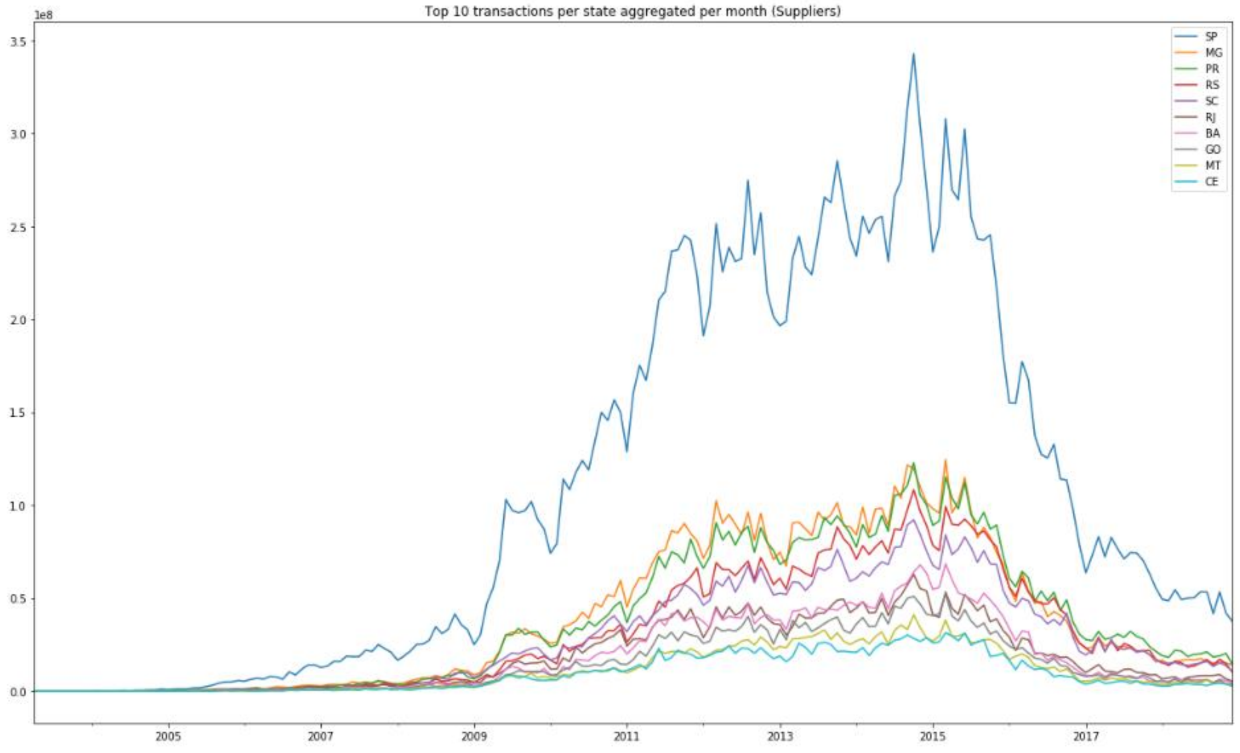


Figure 9: Evolution of payment flows aggregated per month according to top 10 states (suppliers).

[64] Note that Martini and Teixeira (2016) used BNDES Credit Card data only from 2014. However, it is the year when most transactions are recorded both in value and count, being responsible for almost 20% of transactions in value in the whole sample.

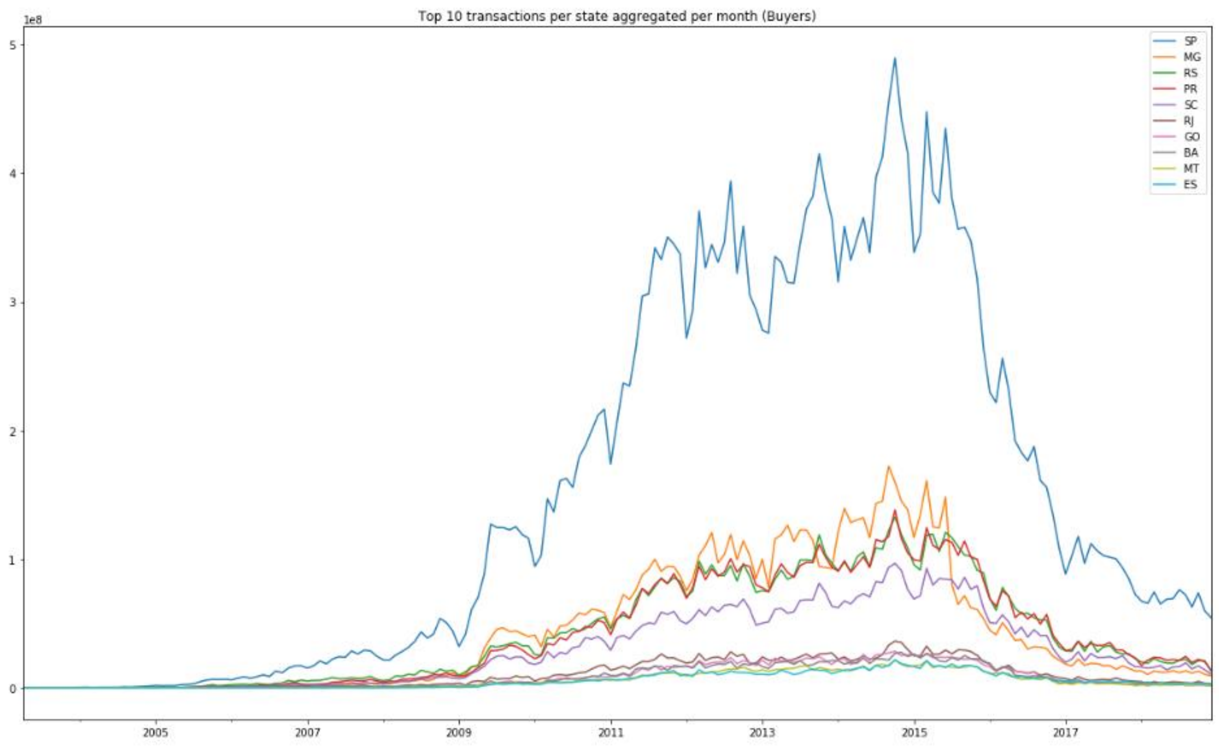


Figure 10: Evolution of payment flows aggregated per month according to top 10 states (buyers).

São Paulo seems to play a significant role in both suppliers' and buyers' side being responsible to a great proportion of value transferred, which the Tables 2 and 3 demonstrated as well. Martini and Teixeira (2016) made also similar findings in their study. Figures 11 and 12 give a better notion of the proportions exhibited by all states for the whole period, including the top 10 ones. The charts reveal that the suppliers' side relies more in São Paulo and in a few states from the southeast and south of Brazil, again agreeing with Martini and Teixeira (2016). In buyers' side, São Paulo's role is more discreet, and the purchasing activity is more evenly distributed across states. The Figures 13-18 further reveal the evolution of these proportions among suppliers and buyers in different time periods, namely 2003-2008, 2009-2016 and 2017-2018. Finally, Figures 19-21 represent the network structure[65] formed by payment activity in the mentioned periods. One can notice that among the suppliers, São Paulo's position remains strong across time periods, being responsible for almost half of the value transferred in 2003-2008. In the buyers' side, as already noted, São Paulo's status is more subtle, and the value transferred is more evenly spread. The network graphs demonstrate the connectedness among the states effectively showing which states are more central and which states contribute less in terms of the whole network

[65] The directed graphs were constructed with Gephi (network visualization software) leveraging the force atlas layout algorithm and using weighted degrees as the condition for node partitioning.

structure[66]. The graphs also give a good hierarchical sense on the monetary flows between states: in addition to São Paulo, three tiers are clearly distinguishable demonstrating the importance of a state when considering credit card usage. For example in Figure 18, where payment flows between states in 2009-2016 are illustrated, the most important states are São Paulo, Bahia, Paraná, Minas Gerais, Rio Grande do Sul and Santa Catarina, forming the first tier. These findings are also in line with Martini and Teixeira (2016) with the exception of Santa Catarina. The states mentioned are present in other time periods as well in the first tier. The similarities between this study and the study made by Martini and Teixeira (2016) highlights the fact that market dynamics among BNDES Credit Card users (or states) have not experienced significant changes over time, even though the scale of operations in terms of transaction values and counts has changed drastically.

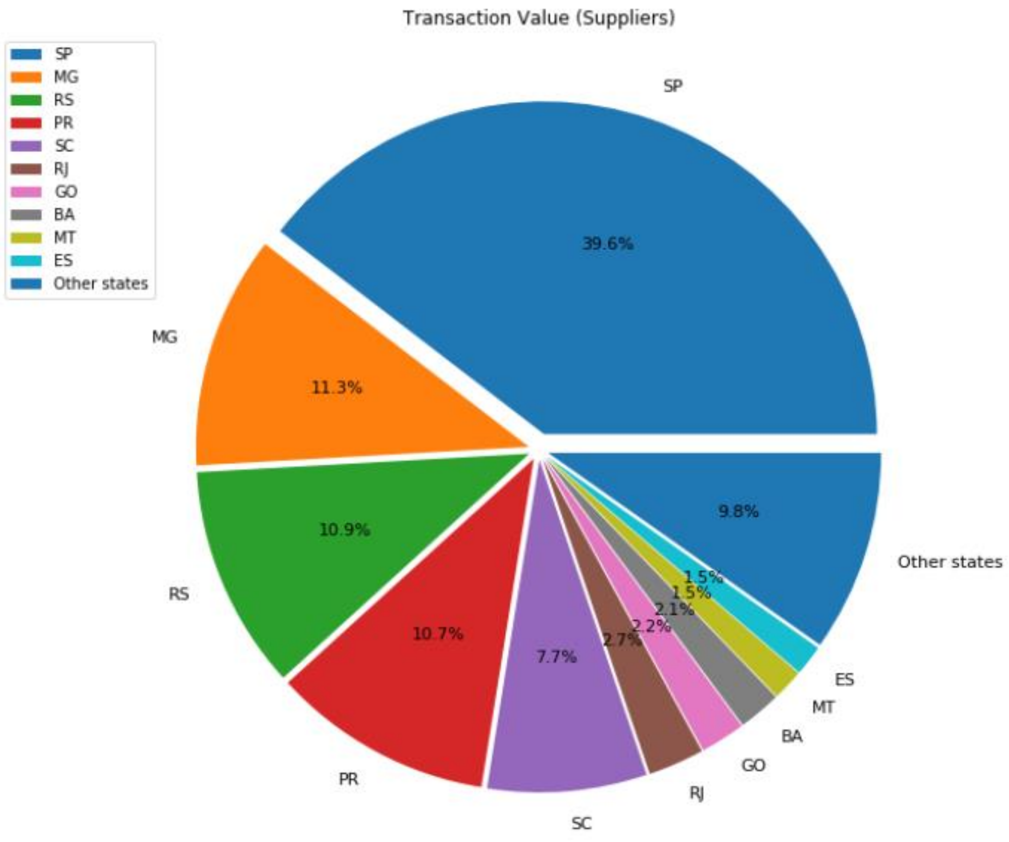


Figure 11: Share of transactions according to states for the whole period (suppliers).

[66] While drawn for illustrative purposes, the graphs visualise and capture the strength of interrelations between states in such a compact manner that earlier tables and figures cannot.

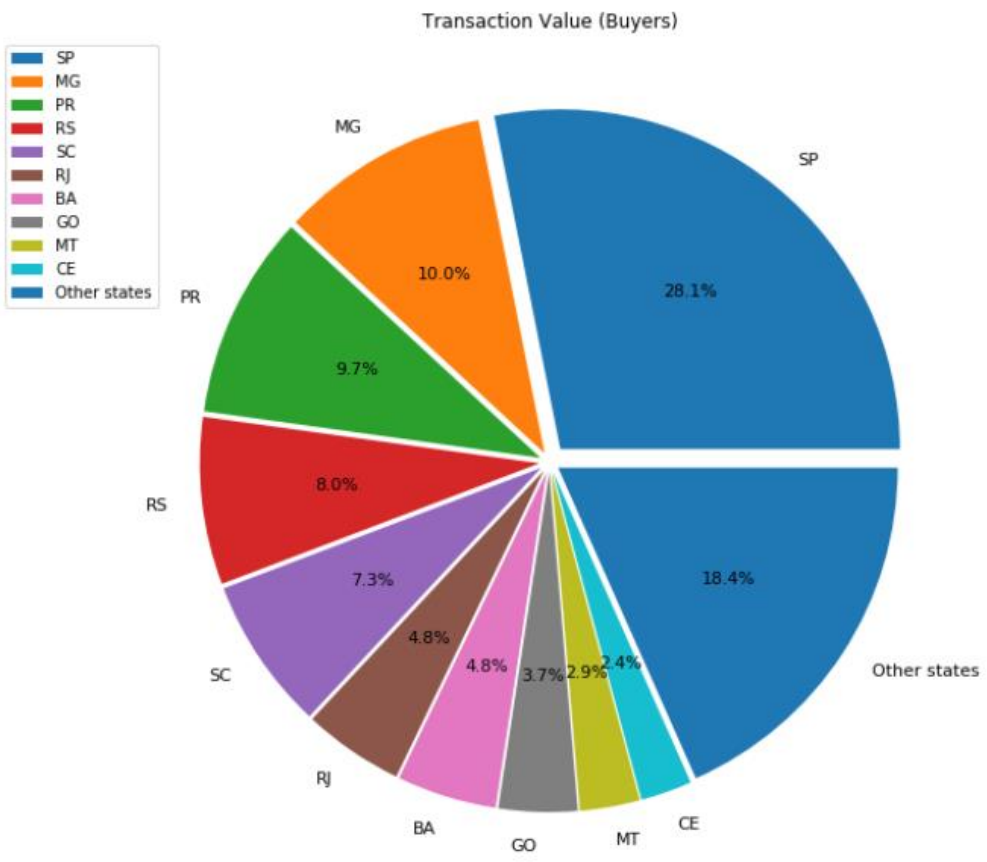


Figure 12: Share of transactions according to states for the whole period (buyers).

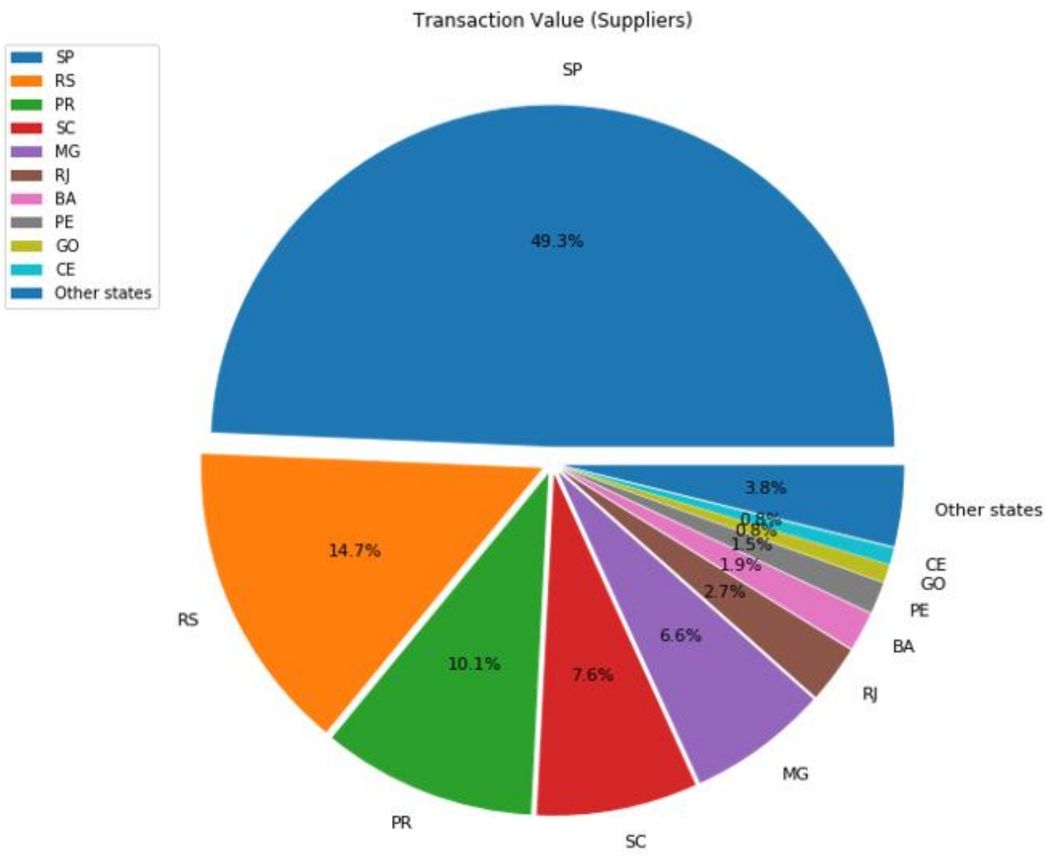


Figure 13: Share of transactions according to states for 2003-2008 (suppliers).

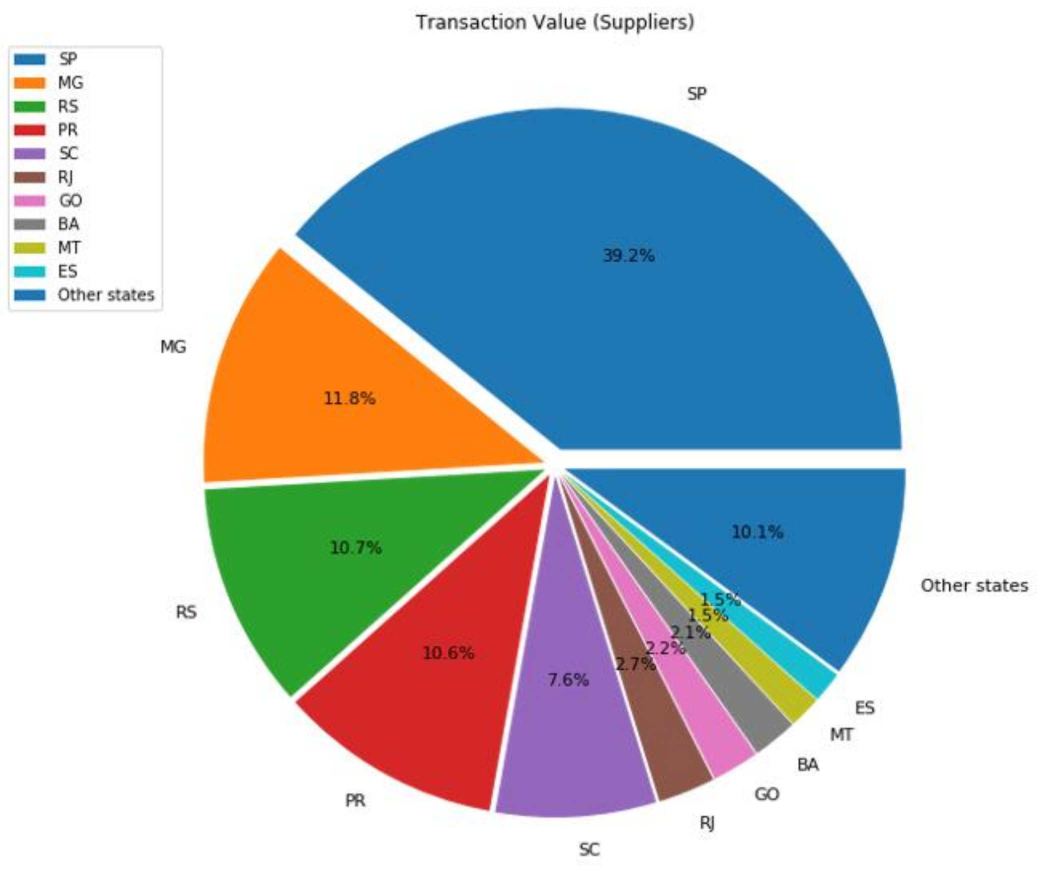


Figure 14: Share of transactions according to states for 2009-2016 (suppliers).

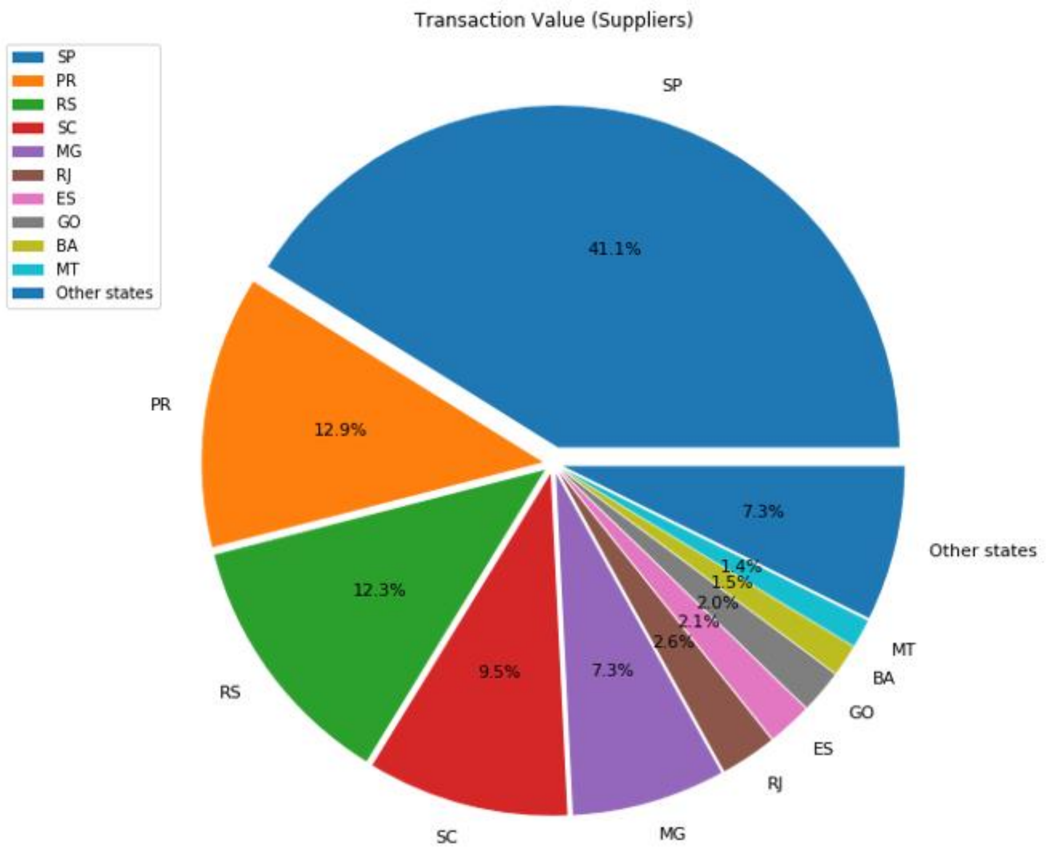


Figure 15: Share of transactions according to states for 2017-2018 (suppliers).

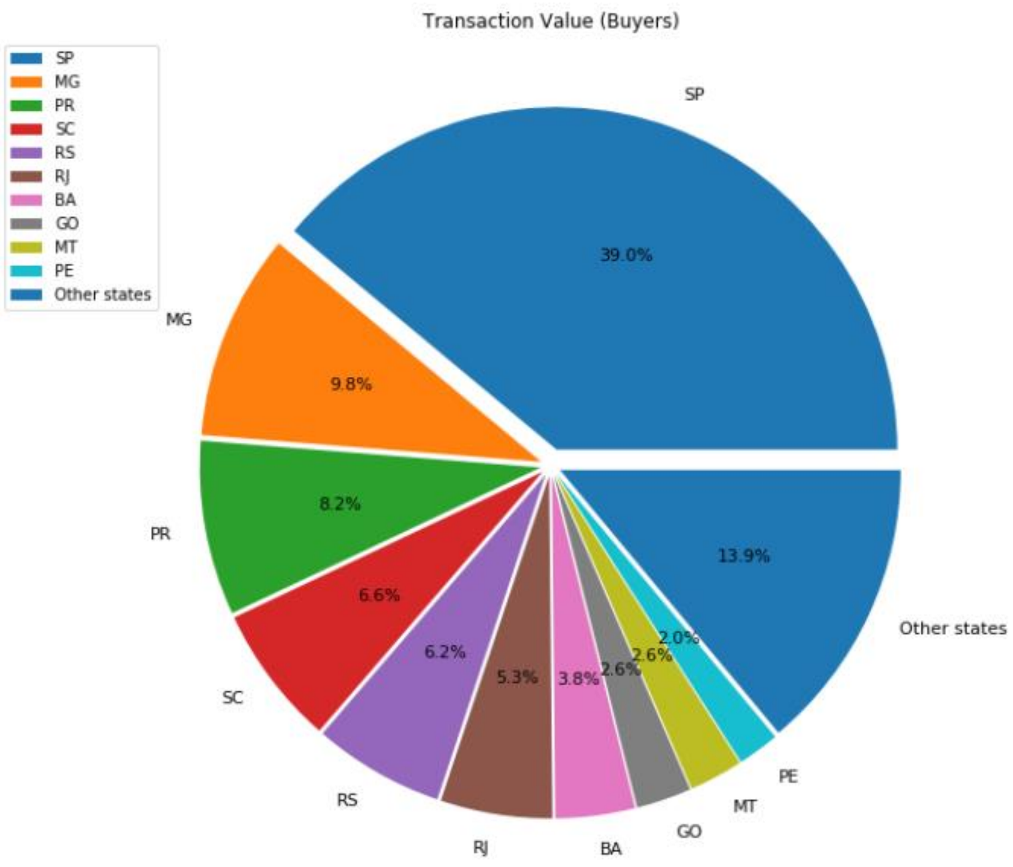


Figure 16: Share of transactions according to states for 2003-2008 (buyers).

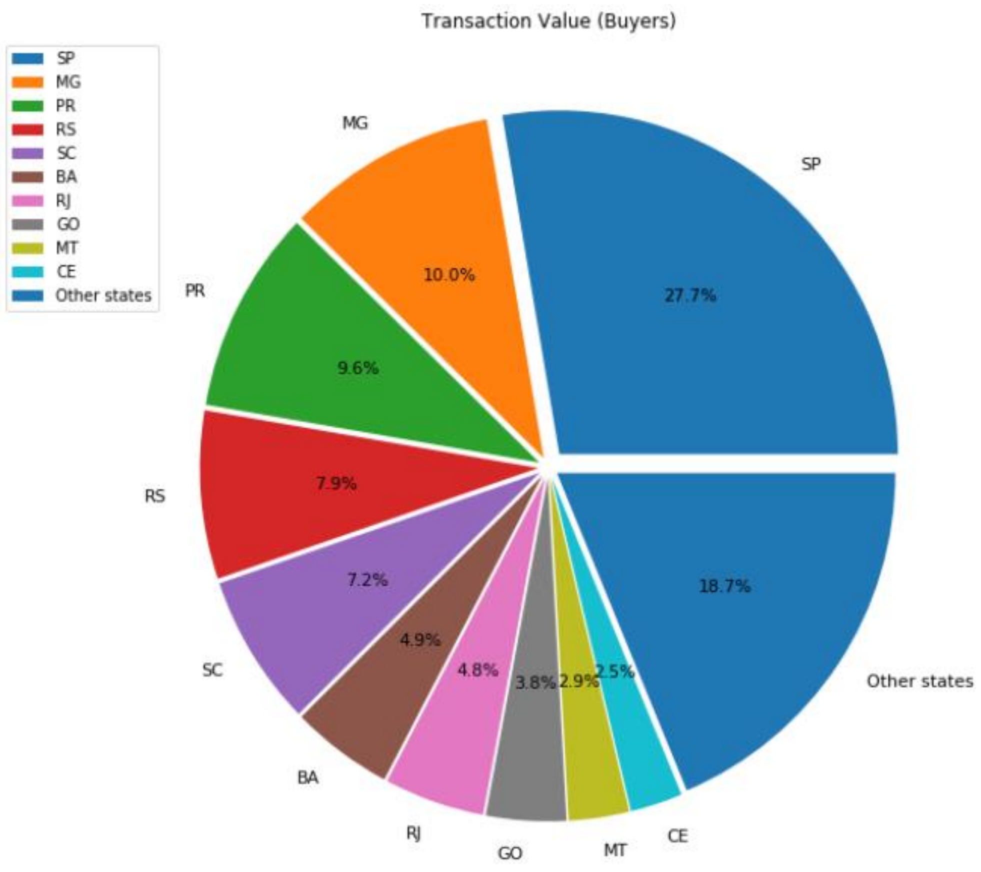


Figure 17: Share of transactions according to states for 2009-2016 (buyers).

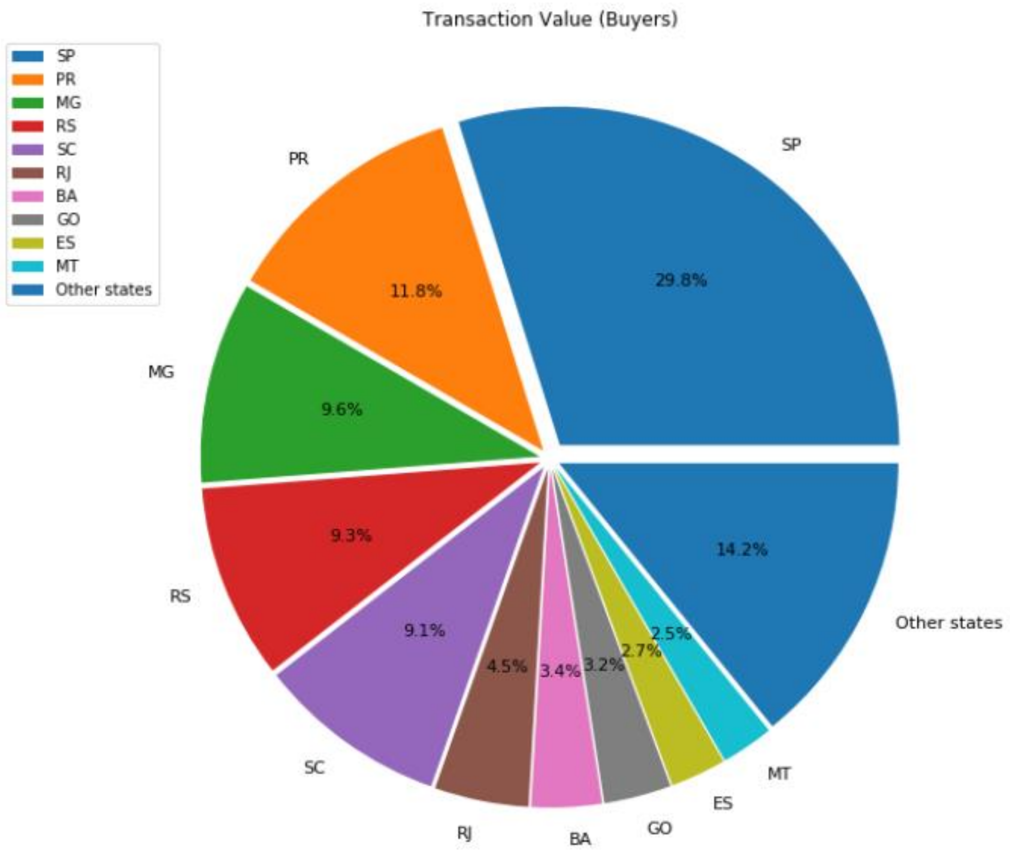


Figure 18: Share of transactions according to states for 2017-2018 (buyers).

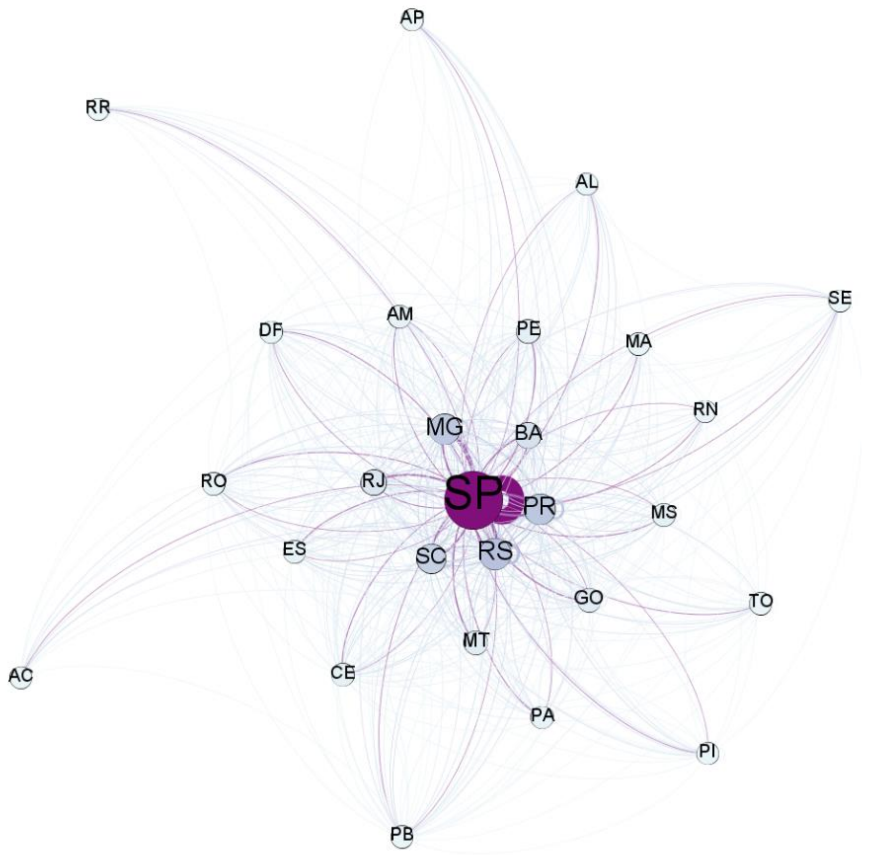


Figure 19: Network graph illustrating payment flows between states in 2003-2008.

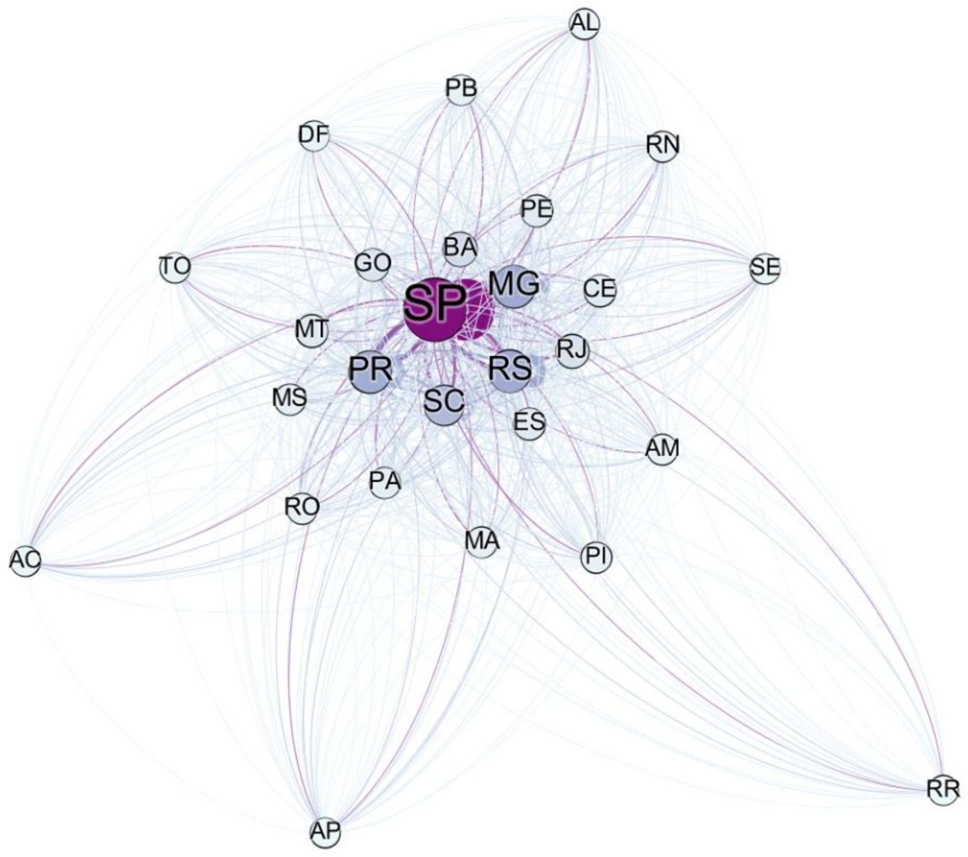


Figure 20: Network graph illustrating payment flows between states in 2009-2016.

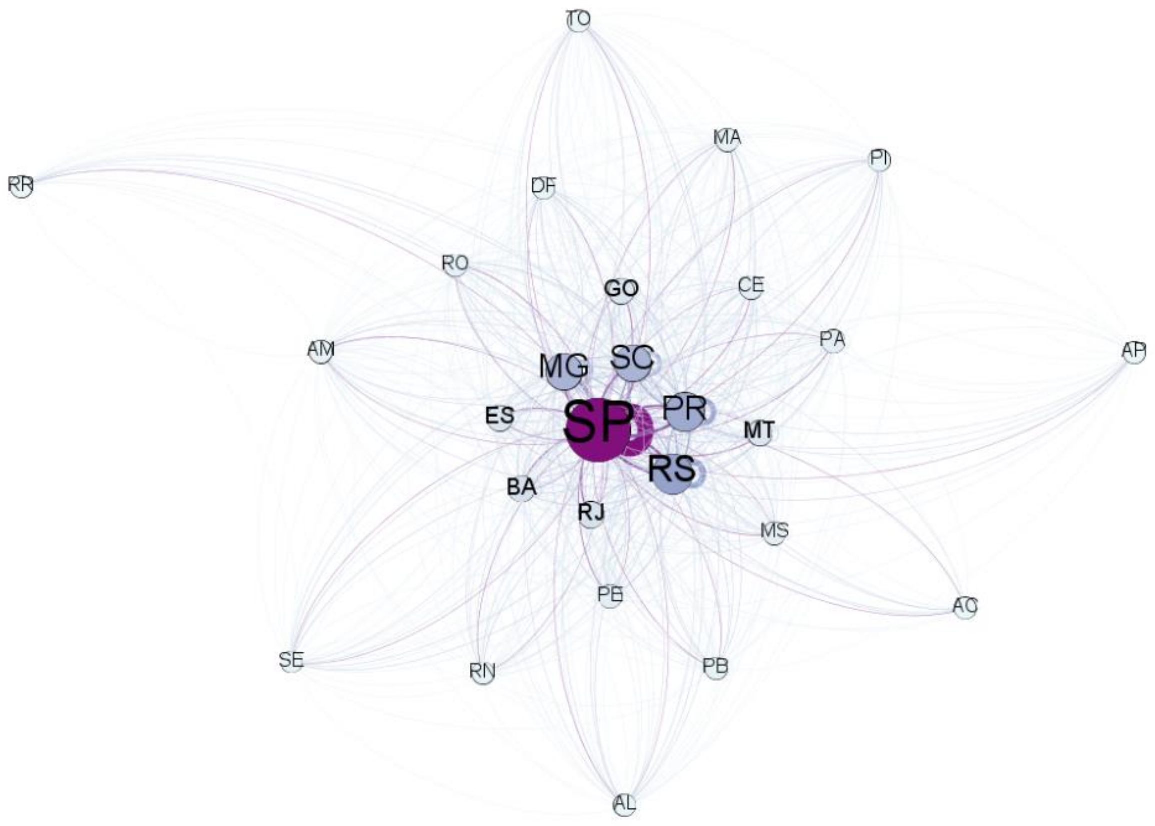


Figure 21: Network graph illustrating payment flows between states in 2017-2018.

To summarize, São Paulo seems to play a significant role among BNDES Credit Card users over the whole period. When compared to buyers, the network of suppliers seems to be more compact and relies largely on couple of states especially in south-eastern and southern Brazil. There also exists clear hierarchy between states in terms of regional activities and their relative magnitudes. Lastly, the underlying structure of market dynamics has stayed rather static even though the scale of operations has changed greatly over time. Next, the results from the time series forecasting problem are discussed. Recall that the monthly sum of transactions (in value) is being predicted with the uni- and multivariate models implemented according to the methodology. Figures 22 and 23 illustrate[67] the forecasts from each model against actual data:

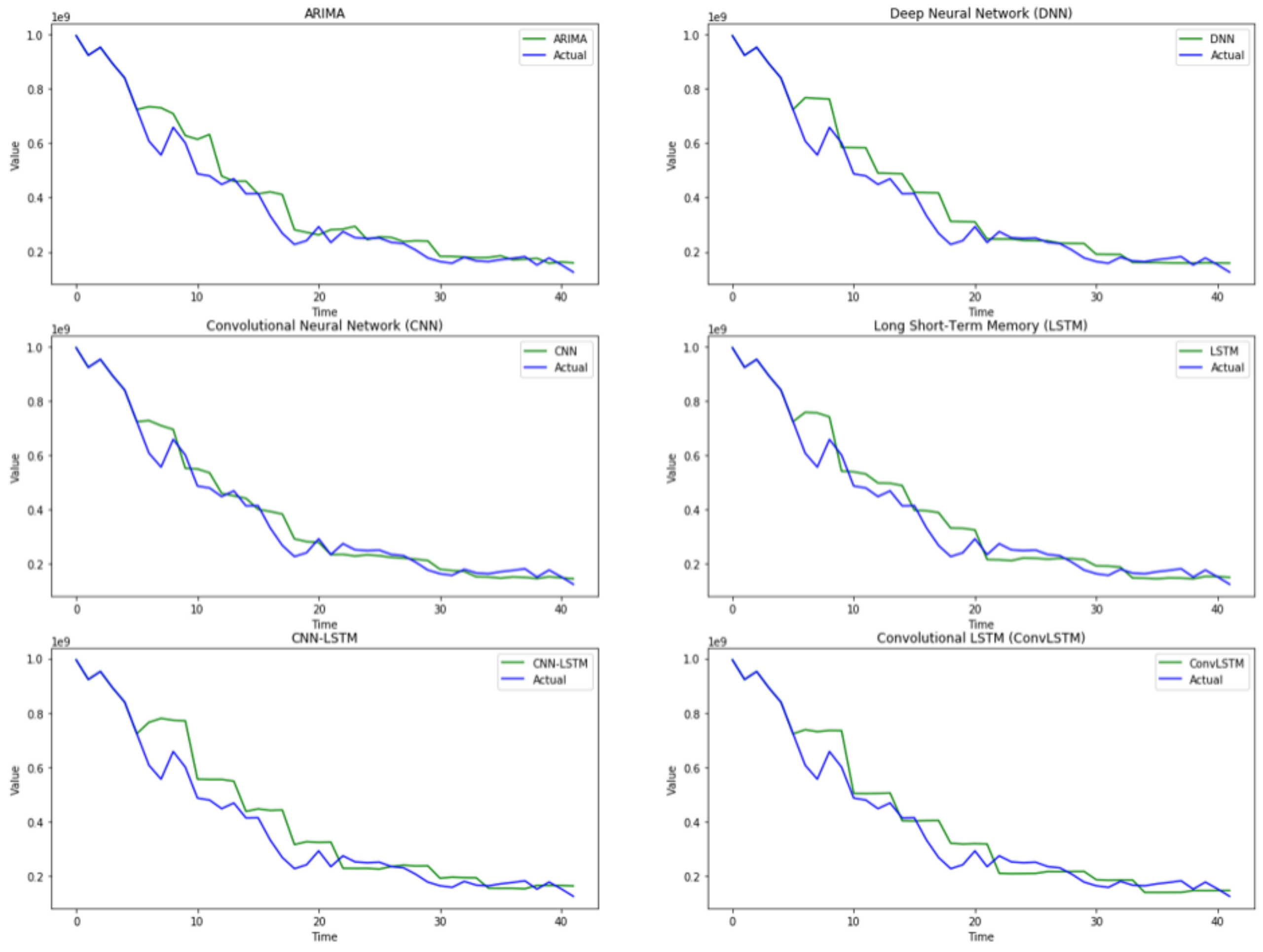


Figure 22: Univariate prediction results.

[67] Note that the actual data starts 42 months away (corresponding to 2015-06) and the predictions from 36 months away (corresponding to 2016-01) from 2018-12.

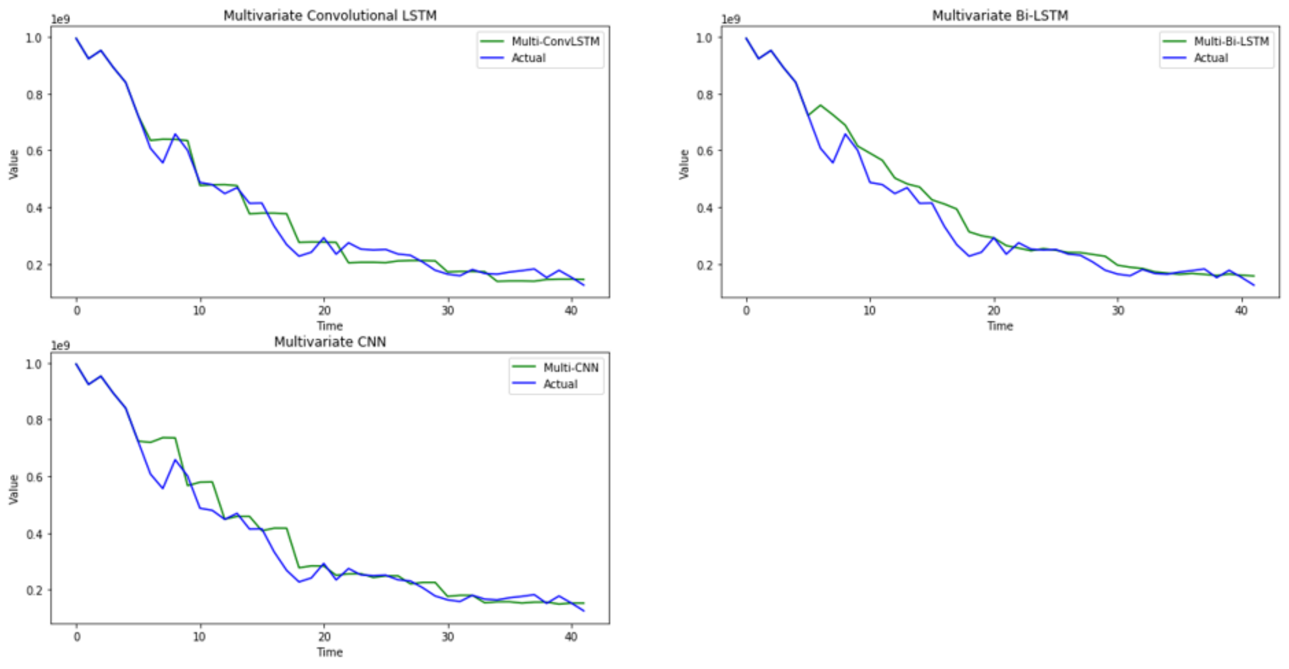


Figure 23: Multivariate prediction results.

While the visualisations of the predictions do not offer much information for analysis purposes, some notions can be made based on them. In the univariate case, ARIMA, DNN and CNN-LSTM seem to consistently over-estimate the actual data, whereas other univariate models demonstrate more favourable variability in terms of forecasting bias. All univariate models have difficulties in capturing the strong downward trend in the beginning of the test period. Interestingly, even though the multivariate models are much more complex in terms of input data and model architecture, they show steadiness and competitiveness as well as relatively neutral results when considering forecasting bias.

	RMSE	MAPE	MAE	MSLE	MDA
ARIMA	61921182.447	13.794	41853079.334	0.02513585	0.51428571
DNN	66229944.532 (-6.96%)	13.881 (+0.63%)	43690546.354 (+4.39%)	0.02675362 (+6.44%)	0.65714286 (+27.78%)
CNN	47520703.748 (-23.26%)	**11.292 (-18.14%)**	33551821.985 (-19.83%)	**0.01743506 (-30.64%)**	**0.71428571 (+38.89%)**
LSTM	62619773.038 (+1.13%)	15.574 (+12.90%)	46529724.597 (+11.17%)	0.02978695 (+18.50%)	0.65714286 (+27.78%)
CNN-LSTM	79786289.111 (+28.85%)	18.842 (+36.60%)	58638034.662 (+40.10%)	0.03999117 (+50.10%)	0.42857143 (-16.67%)
ConvLSTM	62922734.643 (+1.62%)	16.365 (+18.64%)	47326546.805 (+13.08%)	0.03401049 (+35.31%)	0.57142857 (+11.11%)
MultiCNN	56079607.704 (-9.43%)	11.687 (-15.27%)	36279434.928 (-13.32%)	0.02065243 (-17.84%)	0.48571429 (-5.56%)
MultiConvLSTM	**38166738.569 (-38.36%)**	11.988 (-13.09%)	**30740556.82 (-26.55%)**	0.02209509 (-12.10%)	0.48571429 (-5.56%)
MultiBiLSTM	57377753.786 (-7.34%)	12.428 (-9.90%)	37941839.405 (-9.35%)	0.02226637 (-11.42%)	0.68571429 (+33.33%)

Table 4: Prediction results by metrics.

The performance results (Table 4) indicate that the best performing model is CNN being superior to the other models by a clear margin in three measures, as bolded. Note that lower result is better with the first four measures while higher is better with MDA. The CNN outperforms ARIMA in all measures by 19 to 38 percent margins. Another model outperforming ARIMA when all measures are considered is the MultiBiLSTM with 7 to 33 percent margins. The bidirectional functionality may have assisted in interpreting the 729-variable long input sequences. Other multivariate models, while being slightly more inaccurate than ARIMA, also show competitive results beating the baseline in the first four measures, and notably the MultiConvLSTM has the lowest RMSE and MAE, as bolded. The multivariate models demonstrate promising capabilities rem,00embering that the data is not transformed, and the input is high dimensional.

DNN is also competitive when comparing to the baseline, having especially high MDA. The LSTM and ConvLSTM also show better accuracy, while being otherwise relatively incompetent. The worst model is the CNN-LSTM demonstrating the poorest performance in all measures when compared to the baseline and seems to have great difficulties capturing the overall trend being wrong in more than half of the cases on average. This may be, among other reasons, due to model complexity (as two different models are sequentially combined) and suboptimal model architecture.

j =	ARIMA	DNN	CNN	LSTM	MultiCNN	MultiBiLstm
i	-	-	-	-	-	-
ARIMA	-	0.6596	**0.03317**	0.8189	0.05819	**0.0494**
DNN	0.3405	-	**0.04551**	0.7566	**0.02928**	**0.02407**
CNN	0.9668	0.9545	-	0.9985	0.8337	0.827
LSTM	0.1811	0.2434	**0.001534**	-	**0.03014**	**0.007086**
MultiCNN	0.9418	0.9707	0.1663	0.9699	-	0.2754
MultiBiLSTM	0.9506	0.9759	0.173	0.9929	0.7246	-

Table 5: p-values of the modified DM test for methods with h = 3.

j =	CNN-LSTM	ConvLSTM	MultiConvLSTM
i			
CNN-LSTM	-	0.1184	0.06648
ConvLSTM	0.8816	-	0.064045
MultiConvLSTM	0.9335	0.9396	-

Table 6: p-values of the modified DM test for methods with h = 4.

In the first row of Table 5[68], when forecasting three steps ahead, it can be noticed that the DM-test suggests superior forecasting accuracy of CNN and MultiBiLSTM to the baseline ARIMA under 5% significance level. The predictive ability of MultiCNN against the baseline is also statistically significant under 10% level. Statistical significance for the efficacy of CNN, MultiCNN and MultiBiLSTM can be found also from the 2th and 4th row, when compared to DNN and LSTM (under 5% level). Interestingly, the test results cannot support the hypothesis that ARIMA is more accurate than any of the other models with forecast horizon of three timesteps ahead confirming that all models tested against the baseline are competitive in this case. In Table 6, the p-values of the models with 4-month horizon demonstrate the superiority of MultiConvLSTM under 10% significance level which supports the performance results in Table 4.

A separate but similar forecasting experiment was also conducted with additional BNDES Credit Card payment data, namely the first semester of 2019, using three well-performed models from the previous experiment. The selected models were CNN, MultiCNN and MultiBiLSTM with some minor modifications. The experiment included making predictions with the same forecasting horizons as before and also using six prior time steps

[68] The alternative hypothesis is that model j is better than model i in terms of forecasting accuracy. Recall also that the forecasting horizons differ according to the model.

to predict six steps ahead. In other words, when predicting three steps ahead, two sequential forecasts were done, as only one forecast was required for predicting six steps onward. In addition to the training scheme as earlier, the models were also updated using data for training until the end of 2018, for further comparison. The following charts visualise the prediction results, where the names of the models indicate the forecasting horizon applied and whether an updated version was used[69]:

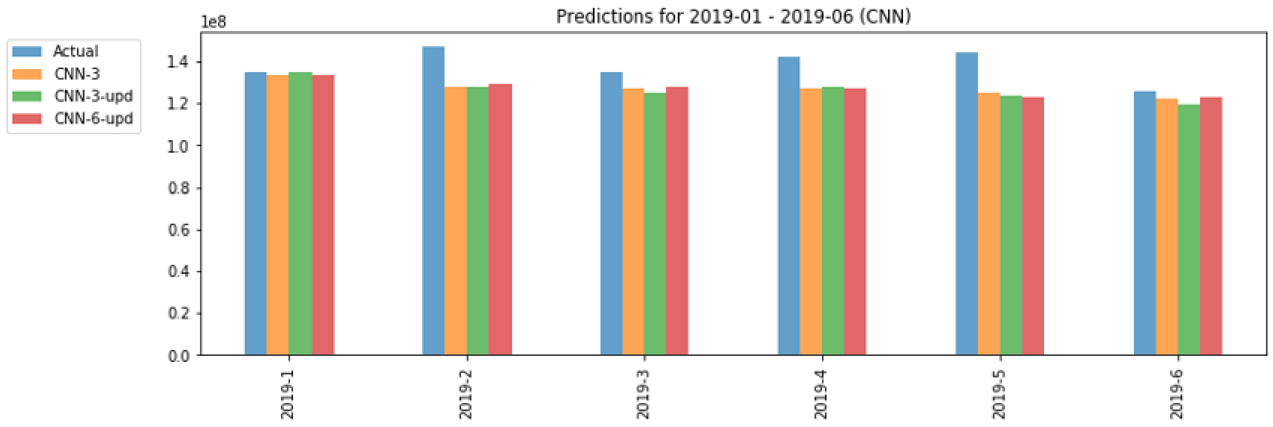


Figure 24: Prediction results for the period 2019-01 – 2019-06 (CNN).

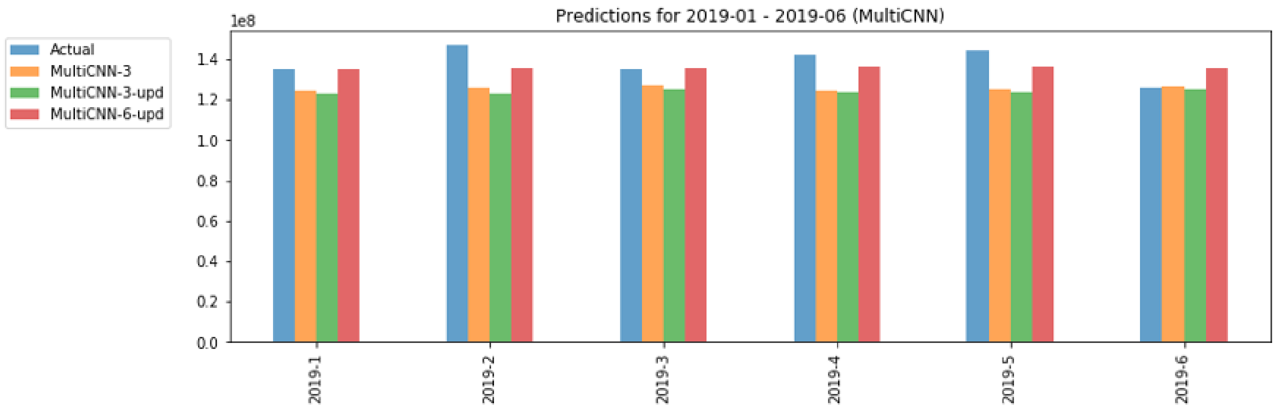


Figure 25: Prediction results for the period 2019-01 – 2019-06 (MultiCNN).

[69] The updated model of MultiCNN with h = 6 (MultiCNN-6-upd) had a batch size of 24 and was run with 130 epochs. Both updated models of MultiBiLSTM (MultiBiLSTM-3-upd & MultiBiLSTM-6-upd) had a batch size of 16. Otherwise, model architectures remained untouched. These changes were necessary for producing stable predictions. Note also that the results from h = 6 without updating are not reported since it did not produce satisfactory results initially, that is, it would have required the optimisation of model architectures again.

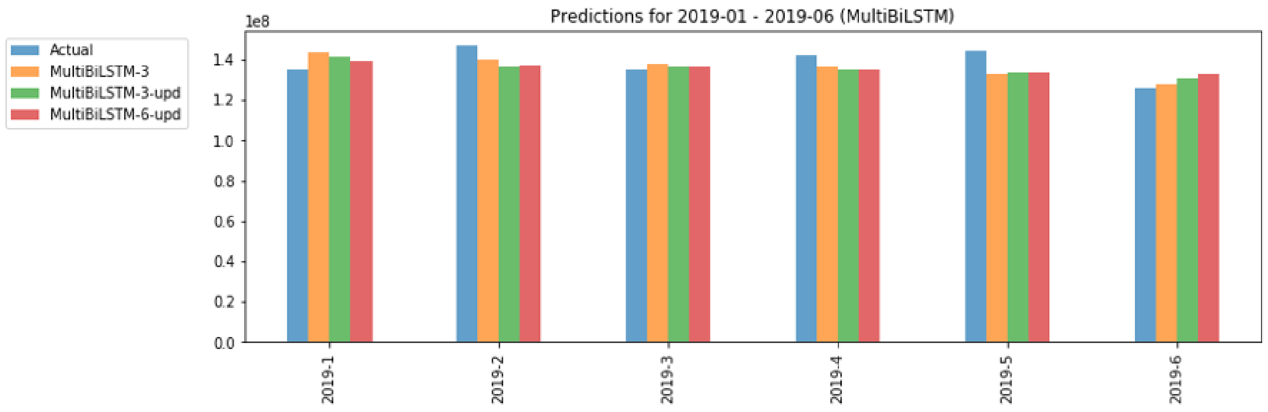


Figure 26: Prediction results for the period 2019-01 – 2019-06 (MultiBiLSTM).

From the bar charts one can notice that the univariate CNNs, while demonstrating fairly stable results, could not track the fluctuations of the actual data. The MultiBiLSTM and its different versions showed better ability in keeping up with the realisations. Interestingly, among CNNs and MultiBiLSTMs, the model updates did not seem to have much of an effect on the model's forecasting performance. Also, even though the updated MultiCNN with 6-month horizon appears to follow well the realisations, the predictions are very similar to each other thus demonstrating rather conservative results. Performance results of the models were also calculated using the same metrics as in the previous experiment, which are collected in the following table. Note that the CNN (CNN-3) from the former experiment is used as a baseline:

	RMSE	MAPE	MAE	MSLE	MDA
CNN-3	13111610.338	7.762	11015911.510	0.00944295	0.40
CNN-3-upd	13637202.463 (+4.01%)	8.261 (+6.43%)	11661635.466 (+5.86%)	0.01033221 (+9.42%)	0.60 (+50.00%)
CNN-6-upd	13058076.955 (-0.41%)	7.531 (-2.98%)	10710861.110 (-2.77%)	0.00938962 (-0.56%)	0.40 ($\pm 0.00\%$)
MultiCNN-3	14651362.838 (+11.74%)	9.059 (+16.71%)	12844263.724 (+16.60%)	0.01195106 (+26.56%)	0.40 ($\pm 0.00\%$)
MultiCNN-3-upd	16261953.800 (+24.03%)	10.050 (+29.48%)	14254070.510 (+39.40%)	0.01495003 (+58.32%)	0.20 (-50.00%)
MultiCNN-6-upd	7417893.457 (-43.43%)	**4.322 (-44.32%)**	**5982255.944 (-45.69%)**	0.00293072 (-68.96%)	**1.00 (+150.00%)**
MultiBiLSTM-3	**7026743.014 (-46.41%)**	4.413 (-43.16%)	6191763.922 (-43.79%)	**0.00255168 (-72.98%)**	0.40 ($\pm 0.00\%$)
MultiBiLSTM-3-upd	7504390.035 (-42.77%)	4.876 (-37.18%)	6820265.548 (-38.09%)	0.00293746 (-68.89%)	0.40 ($\pm 0.00\%$)
MultiBiLSTM-6-upd	7428622.549 (-43.34%)	4.824 (-37.85%)	6718918.214 (-39.01%)	0.00291029 (-69.18%)	0.40 ($\pm 0.00\%$)

Table 7: Prediction results for the period 2019-01 – 2019-06 by metrics.

The performance results confirm the deductions made from the bar charts. However, while the MultiCNN-6-upd demonstrated cautiousness, it was able to capture the trends perfectly. Updating models with most recent data especially in the case of longer forecasting horizons may thus be a crucial factor for forecasting performance. Intuitively, this makes sense since more situation-aware models should indeed produce better predictions. Moreover, four multivariate models (representing half of the models) showed improvements of 37 to 73 percent in first four metrics when compared to the (univariate) baseline. Finally, the most accurate predictions from both experiments, namely from models CNN and MultiCNN-6-upd, are collected to the following figure[70]:

[70] In this case, the actual data starts 48 months away (corresponding to 2015-06) and the predictions from 42 months away (corresponding to 2016-01) from 2019-06.

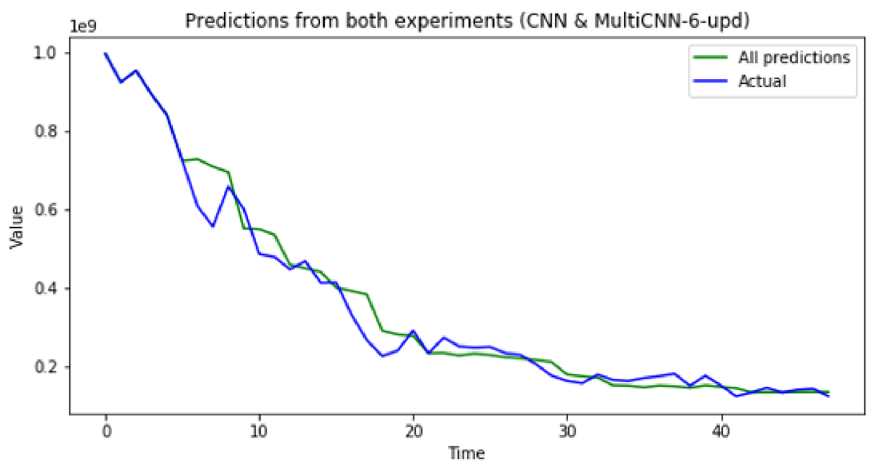


Figure 27: Most accurate predictions from both experiments.

When considering the results from both experiments and the figure above, the best performing models demonstrate impressive capabilities in forecasting aggregated payment flows. The contribution of the multivariate setting is also highly relevant and shows very promising results.

Some important indications can be derived from the forecasting study and results presented. As it was a preliminary study, there are many extensions to it which will be detailed in the next chapter. Nevertheless, a systematic methodology to forecasting time series data with deep learning methods was implemented successfully. Moreover, a methodology to include transaction data where granularities such as sources and destinations was developed and carried out showing promising results. It is worth noting also that the created adjacency matrices of payment flows provide a natural stepping point to graph-based methods. The results demonstrate the efficacy and further potential of deep learning methods in forecasting transactional data of different forms. Also, based on the literature review, there still does not exist an application of the convolutional LSTM to forecasting payment data, as performed in this study[71]. Note that some deep learning techniques have been applied to payment data before, but mostly in the problems of detecting anomalies, as e.g. in Roy et al. (2018), Ziegler et al. (2017) and Triepels et al. (2017).

Forecasting transaction and other types of financial time series data can be helpful when foreseeing upcoming events are important for example in terms of predictive business process monitoring. With accurate and reliable forecasts, public and private organisations

[71] This can be due to the fact that the model is challenging to implement from scratch. Recall also that the model implemented in Keras does not fully represent the model in the original paper.

and enterprises can align their own strategies and policies towards more desirable and perhaps profitable outcomes while ensuring continuity of operations. For example, in the case of foreseeing the BNDES Credit Card usage, BNDES may be able to make more informed decisions regarding their capital reserved for financing, using and further enhancing the models developed in this work. Moreover, forecasting is one of the major aims of economic and econometric analysis (CARRIERO et al. (2019)). Also, the discovery of complex structures in raw data with no prior assumptions that may result in better out-of-sample prediction models plays an important role in econometric approaches, as the traditional approaches commonly seek for understanding causal theories or explaining determinants of economic activity (CARRIERO et al. (2019); LEÓN and ORTEGA (2018)).

There were some limitations in the preliminary forecasting experiments. There was a lack of both, experience and computational resources, which impacted the development of deep learning models. The data sample was also relatively small – deep learning methods perform better when more data is available. Nonetheless, the methodologies developed are versatile and flexible in many directions. One clear extension is deeper investigation of different forecasting horizons with varying prior timesteps and proportions for splitting test and training data, and quarterly data could be applied instead of monthly. Model updates could also occur in each forecasting origin meaning that the deep learning models could be retrained with all available data, hypothetically resulting in more accurate forecasts, similarly to the second experiment. Instead of predicting only one variable, the prediction of multiple variables at once is possible as well. In this study, only the available payment data was considered, thus the study could be extended to verifying whether i) adding financial or macroeconomic variables[72] of different frequencies and ii) incorporating information from news and sentiments or satellite imagery data could improve the forecasting accuracies. The payment data itself could be also used as an auxiliary variable for predicting macroeconomic variables as was done in León and Ortega (2018). Deep learning models could be also combined with macroeconomic models through hybrid- or ensemble-based approaches, and as mentioned before, the methodology can be easily adapted to graph-based approaches[73]. More traditional methods could also be added for wider comparison. With more computational resources, it might be possible to find a model configuration for

[72] Recall the similarities in trends between the state of Brazilian economy (represented by the GDP) and the monthly sum of transactions.
[73] Note that the forecasting study was executed in the Euclidean setting. Recall the end of section 2.2 and 2.3.

the simpler models that does not require data transformation techniques which can have undesired effects on forecasting performance - working with raw data ensures that overall complexity or randomness of the data is not reduced (BODSTRÖM and HÄMÄLÄINEN (2019)). For completeness, sensitivity and robustness analysis should be added as well as efforts made in terms of interpreting and characterising the behaviour of deep learning models while making predictions (FRANCA LEPPÄNEN and HÄMÄLÄINEN (2019)). Finally, the methodologies can be used with all kinds of data that are transactional in nature.

The descriptive statistics and the brief study of participant dynamics in the usage of BNDES Credit Card could also largely benefit from further extensions. One interesting aspect would be incorporating currency fluctuations in transaction values and comparing whether it has some effect in the results. One could also investigate how the flow of payments follow the trends and dynamics of other macroeconomic and financial variables. Complex network theory may also offer useful tools[74] for deriving structural and behavioural insights from payment data – the theory is strongly present for example in the stream of literature regarding transactional data producing FMIs in Brazil (see e.g. SILVA et al. (2017) and SILVA et al. (2018)). The city-specific data could also be more explored, and certainly if more granular such as firm- and employment-level data were available, more avenues for further studies could be explored.

[74] Summary statistics derived from complex network theory may be limited in some cases because they cannot adapt to changes in data, paving the way for more flexible techniques such as the deep learning methods presented in this thesis. Nevertheless, some good introductory references about data mining and complex network theory include Zaki and Meira (2016), Silva and Zhao (2016) and Soramäki and Cook (2016).

7 CONCLUDING REMARKS

In this thesis, a brief introduction to FMIs and BNDES was given followed by a literature review on forecasting in the fields of macroeconomics, machine and deep learning as well as graph-based deep learning. Common factors, challenges and further research directions were also identified. Moreover, a methodology for forecasting transactional time series data was developed and successfully implemented in uni- and multivariate cases taking advantage of state-of-the-art deep learning algorithms. A technical background for the deep learning methods used in the experiments was also provided. Moreover, another study direction was taken by providing descriptive statistics and insights on the dynamics underlying the BNDES Credit Card payment data used in experiments. In addition, the limitations experienced during the empirical part and suggestions for future works were discussed.

The development of more efficient data-driven methods with more degrees of freedom mixing data from different frequencies and sources seems to be the main driver in both macroeconomic and deep learning-based forecasting literature. In recent years, deep learning has gained a massive amount of interest in various disciplines due to its capability of learning arbitrary complexities directly from high-dimensional input data possibly without prior specifications. The competitiveness and utility of deep learning in predicting payments was confirmed by clear margins through various metrics and statistical testing, when forecasting the monthly sum of transaction values (derived from BNDES Credit Card data) multiple steps ahead in the future in uni- and multivariate setting. Interestingly, while the univariate CNN was found as the best performing model in the main experiment, the multivariate models with raw payment data input also consistently outperform the baseline ARIMA.

The brief exploration of interdependencies among BNDES Credit Card participants revealed the strong influence of São Paulo and the role of south-eastern and southern states in suppliers’ network. The regional activities and their proportional strength can be characterised by a hierarchical structure, which stays relatively steady over the whole period despite the substantial changes in the amount trading.

Various and diverse research avenues for future works were identified which are hoped to be explored by the researchers of BNDES and the scientific community in general in their respective studies related to transactional data.

REFERENCES

ARORA, R.; BASU, A.; MIANJY, P.; MUKHERJEE, A. (2018). **Understanding Deep Neural Networks with Rectified Linear Units.** International Conference on Learning Representations (ICLR'18).

BALL, J.E., ANDERSON, D.T. & CHAN, C.D. (2017). **A Comprehensive Survey of Deep Learning in Remote Sensing**: Theories, Tools, and Challenges for the Community. CoRR, abs(1709.00308). arXiv.

BANCO CENTRAL DO BRASIL. (BCB 2017). **Sistema de Transferência de Reservas** (Reserves Transfer System). https://www.bcb.gov.br/content/financialstability/reservestransfersystem_docs/str_annual_reports/4_Annual_Report_STR_2017.pdf. Accessed on 23/8/2019.

BANCO CENTRAL DO BRASIL. (BCB 2019). **Relação de participantes do STR**. https://www.bcb.gov.br/pom/spb/estatistica/port/ASTR003.pdf. Last accessed 23/8/2019.

BATTAGLIA, P., BLAKE, J., HAMRICK, C., BAPST, V., SANCHEZ, A., ZAMBALDI, V., MALINOWSKI, M., TACCHETTI, A., RAPOSO, D., SANTORO, A., FAULKNER, R., GULCEHRE, C., SONG, F., BALLARD, A., GILMER, J., DAHL, G.E., VASWANI, A., ALLEN, K., NASH, C., LANGSTON, C.J., DYER, C., HEESS, N., WIERSTRA, D., KOHLI, P., BOTVINICK, M., VINYALS, O., LI, Y. & PASCANU, R. (2018). **Relational inductive biases, deep learning and graph networks.** Google AI. arXiv. https://arxiv.org/abs/1806.01261. Last accessed 16/10/2019.

BNDES. (2014). **BNDES adere ao Sistema de Pagamentos Brasileiro** (SPB). https://www.bndes.gov.br/wps/portal/site/home/imprensa/noticias/conteudo/20140618_spb. Last accessed 25/8/2019.

BNDES. **Relatório de Efetividade 2017**: efetividade para um novo ciclo de crescimento econômico. Rio de Janeiro, 2018a.

BNDES. **Agreement between SEBRAE and BNDES expands access to credit for micro and small entrepreneurs**, 2018b. https://www.bndes.gov.br/SiteBNDES/bndes/bndes_en/Institucional/Press/Noticias/2018/20180117_bndes_sebrae.html. Last accessed 19/9/2019.

BODSTRÖM, T., HÄMÄLÄINEN, T. (2019). **A Novel Deep Learning Stack for APT Detection.** MDPI Applied Sciences, 9(6), 1055.

BRONSTEIN, M., BRUNA, J., Y, LECUN., SZLAM, A. & VANDERGHEYNST, P. **Geometric deep learning: going beyond Euclidean data**. IEEE Signal Processing Magazine, 34(1), 18-42, 2017.

BROWNLEE, J. **What is Machine Learning?** https://machinelearningmastery.com/what-is-machine-learning/. Last accessed on 22/8/2019, 2013.

______, **Deep Learning for Time Series Forecasting**. Edition: v1.5. Machine Learning Mastery, 2019a.

_______. **Better Deep Learning**. Edition: v1.3. Machine Learning Mastery, 2019b.

BURGESS, S., FERNANDEZ-CORUGEDO, E., GROTH, C., HARRISON R., MONTI, F., THEODORIDIS, K. & WALDRON, M. The Bank of England's forecasting platform: COMPASS, MAPS, EASE and the suite of models. **Working Paper No. 471**, 2013.

CARRIERO, A., GALVÃO, A.B. & KAPETANIOS, G. A comprehensive evaluation of macroeconomic forecasting methods. **International Journal of Forecasting**, 35(1), 1226-1239, 2019.

CAVALCANTE, R.C., BRASILEIRO, R.D., SOUZA, V.L.F & NOBREGA, J.P. **Computational Intelligence and Financial Markets: A Survey and Future Directions.** Expert Systems With Applications, 55(1), 194-211, 2016.

CHEN, S. & RANCIERE, R. **Financial information and macroeconomic forecasts.** International Journal of Forecasting, 35(1), 1160-1174, 2019.

CHEN, Y. & WEI, Z. **Incorporating Corporation Relationship via Graph Convolutional Neural Networks for Stock Price Prediction.** International Conference on Information and Knowledge Management (CIKM'18), 2018.

CHO, K., VAN MERRIENBOER, B., CULCEHRE, C., BAHDANAU, D., BOUGARES, F., SCHWENKS, H. & BENGIO, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. **Conference on Empirical Methods on Natural Language Processing** (EMNLP'14), 2014.

CHONG, E., HAN, C. & PARK, F.C. **Deep learning networks for stock market analysis and prediction: Methodology, data representations and case studies.** Expert Systems With Applications, 83(1), 187-205, 2017.

COMMITTEE ON PAYMENT AND SETTLEMENT SYSTEMS & TECHNICAL COMMITTEE OF THE INTERNATIONAL ORGANIZATION OF SECURITIES COMMISSION, Bank for International Settlements. (CPSS-TCIOSC, 2012). **Principles for Financial Market Infrastructures.**
https://www.bis.org/cpmi/publ/d101a.pdf. Last accessed on 23/8/2019.

COMMITTEE ON PAYMENT AND SETTLEMENT SYSTEMS, Bank for International Settlements. **Payment, clearing and settlement systems in the CPSS countries**. (CPSS 2011). Volume 1. https://www.bis.org/cpmi/publ/d97.pdf. Last accessed on 23/8/2019.

______, Bank for International Settlements. (CPSS 2005). **New developments in the Large-Value Payment Systems.** https://www.bis.org/cpmi/publ/d67.pdf. Last accessed on 23/8/2019.

______, Bank for International Settlements. **A glossary of terms used in payments and settlements systems**. (CPSS 2016). https://www.bis.org/cpmi/publ/d00b.htm. Last accessed on 23/8/2019.

CORSEUIL, C.H.L., ROITMAN, F.B., ULYSSEA, G. & MACHADO, L. **Uma análise do perfil e da dinâmica das empresas que usam o Cartão BNDES**. Rio de Janeiro: Banco NACIONAL DE DESENVOLVIMENTO ECONÔMICO E SOCIAL, 2019. 30 p. **(Textos para Discussão)**; 142), 2019.

CYBENKO, G. **Approximation by Superpositions of a Sigmoidal Function. Mathematics of Control**, Signals and Systems, 2(1), 303-314.

DE RESENDE, C. **An assessment of IMF medium-term forecasts of GDP growth**. IEO Background Paper No. BP/14/01. Independent Evaluation Office of the International Monetary Fund, 2014.

DOMIT, S., MONTI, F. & SOKOL, A. **Forecasting the UK economy with a medium-scale Bayesian VAR.** International Journal of Forecasting (Article in Press), 2019.

DU, S., LI, T. & HORNG, S. **Time Series Forecasting using Sequence-to-Sequence Deep Learning Framework.** 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP'18), 2018.

FISCHER, T. & KRAUSS, C. **Deep learning with long short-term memory networks for financial market predictions.** European Journal of Operational Research, 270(1), 654-669, 2018.

FRANCA LEPPÄNEN, R. & HÄMÄLÄINEN, T. **Network Anomaly Detection in Wireless Sensor Networks: A Review.** In Proceedings: 19th International Conference on Next Generation Wired/Wireless Advanced Networks and Systems (NEW2AN), St. Petersburg, Russia. In Book: Internet of Things, Smart Spaces, and Next Generation Networks and Systems, pp. 196-207, Springer International Publishing, 2019.

FRANÇOIS-LAVET, V., HENDERSON, P., ISLAM, R., BELLEMARE, M.G. & PINEAU, J. **An Introduction to Deep Reinforcement Learning**. Foundations and Trends in Machine Learning, 11(3-4), 2018.

FUKUSHIMA, K. NEOCOGNITRON: **A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position**. Biological Cybernetics, 36, 193-202, 1980.

GERS, F.A., SCHRAUDOLPH, N.N. & SCHMIDHUBER, J. **Learning Precise Timing with LSTM Recurrent Networks.** Journal of Machine Learning Research, 3(1), 115-143, 2002.

GOODFELLOW, I., BENGIO, Y. & COURVILLE, A. **Deep Learning**. MIT Press, 2016.

GORGI, P., KOOPMAN, S.J. & LI, M. **Forecasting economic time series using score-driven dynamic models with mixed-data sampling.** International Journal of Forecasting (Article in Press), 2019.

GRAVES, A. **Generating Sequences with Recurrent Neural Networks.** arXiv:1308.0850, 2014.

GROVER, A. & LESKOVEC, J. **node2vec: Scalable Feature Learning for Networks.** International Conference on Knowledge Discovery and Data Mining (KDD'16), 2016.

GRUSLYS, A., MUNOS, R., DANIHELKA, I., LANCTOT, M. & GRAVES, A. **Memory-Efficient Backpropagation Through Time.** Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16), 4132-4140, 2016.

HAMILTON, W.L., BENGIO, Y. & LESKOVEC, J. **Representation Learning on Graphs: Methods and Applications**. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 40(1), 52-74, 2017.

HARVEY, D., LEYBOURNE, S. & NEWBOLD, P. **Testing the equality of prediction mean squared errors**. International Journal of Forecasting, 13(2), 281-291, 1997.

HILL, S.L., WANG, Y., RIACHI, I., SCHÜRMANN, F. & MARKRAM, H. **Statistical connectivity provides a sufficient foundation for specific functional connectivity in neocortical neural microcircuits.** Proceedings of the National Academy of Sciences, 109(42), E2885-E2894, 2012.

HOCHREITER, S. & SCHMIDHUBER, J. **Long Short-Term Memory**. Neural Computation, 9(8), 1735-1780, 1997.

HUA, Y., ZHAO, Z., LI, R, CHEN, X., LIU, Z. & ZHANG, H. **Deep Learning with Long Short-Term Memory for Time Series Prediction**. IEEE Communications Magazine, 2019.

KIM, H.Y. & WON, C.H. **Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models**. Expert Systems With Applications, 103(1), 25-37, 2018.

KINGMA, D.T., REZENDE, D.J., MOHAMED, S. & WELLING, M. **Semi-supervised Learning with Deep Generative Models**. Advances in Neural Information Processing Systems 27 (NIPS'2014), 2014.

KIPF, T., & WELLING, M. **Semi-Supervised Classification with Graph Convolutional Neural Networks.** International Conference on Learning Representations (ICLR'17), 2017.

KNOTEK II., E.S. & ZAMAN, S. **Financial nowcasts and their usefulness in macroeconomic forecasting.** International Journal of Forecasting (Article in Press), 2019.

KOROBILIS, D. & PETTENUZZO, D. **Adaptive hierarchical priors for high-dimensional vector autoregressions.** Journal of Econometrics (Article in Press), 2019.

KRIZHEVSKY, A., SUTSKEVER, I. & Hinton, G.E. **ImageNet Classification with Deep Convolutional Neural Networks.** Advances in Neural Information Processing Systems (NIPS'12), 2012.

KUMAR, P.D., AMGOTH, T. & ANNAVARAPU, C.S.R. **Machine learning algorithms for wireless sensor networks: A survey.** Information Fusion, 49(1), 1-25, 2019.

LAI, G., YANG, Y., CHANG, W. & LIU, H. **Modelling Long- and Short-Term Temporal Patterns with Deep Neural Networks.** Conference on Research and Information Retrieval (SIGIR'18), 2018.

LÄNGKVIST, M., KARLSSON, L. & LOUTFI, A. **A review of unsupervised feature learning and deep learning for time-series modelling.** Pattern Recognition Letters, 41(1), 11-24, 2014.

LE, Q.V. **A Tutorial on Deep Learning Part 1**: Nonlinear Classifiers and The Backpropagation Algorithm. http://ai.stanford.edu/~quocle/tutorial1.pdf. Last accessed on 1/9/2019. 2015a.

_______. **A Tutorial on Deep Learning Part 2:** Autoencoders, Convolutional Neural Networks and Recurrent Neural Networks. http://robotics.stanford.edu/~quocle/tutorial2.pdf. Last accessed on 1/9/2019, 2015b.

LECCESE, A. **Machine Learning in Finance**: Why You Should Not Use LSTM's to Predict the Stock Market. Bluesky Capital Management. https://www.blueskycapitalmanagement.com/machine-learning-in-finance-why-you-should-not-use-lstms-to-predict-the-stock-market/. Last accessed on 31/07/2019, 2019.

LECUN, Y., HAFFNER, P., BOTTOU, L. & BENGIO, Y. **Object Recognition with Gradient-Based Learning**. In: Shape, Contour and Grouping in Computer Vision. Lecture Notes in Computer Science, vol 1681, 1999.

LECUN, Y., BENGIO, Y. & HINTON, G. **Deep learning.** Nature, 521(7553), 436-444, 2015.

LEÓN, C. & ORTEGA, F. **Nowcasting economic activity with electronic payments data: A predictive modelling approach**. Borradores de Economía 1037, Banco de la Republica de Colombia, 2018.

LITJENS, G., KOOI, T., BEJNORDI, B.E., SETIO, A.A.A., CIOMPI, F., GHAFOORIAN, M., VAN DER LAAK, J.A.W.M., VAN GINNEKEN, B. & SÁNCHEZ, C.I. **A Survey on Deep Learning in Medical Image Analysis**. Med Image Anal, 42(1), 60-88, 2017.

LIU, W., WANG, Z., LIU, X., ZENG, N., LIU, Y. & ALSAADI, F.E. **A survey of deep neural network architectures and their applications.** Neurocomputing, 234(1), 11-26, 2017.

LOUZIS, D.P. **Steady-state modelling and macroeconomic forecasting quality.** Journal of Applied Econometrics, 34(1), 285-314, 2019.

MACHADO, L., PARREIRAS, M.A. & PEÇANHA, V.R. **Avaliação de impacto do uso do Cartão BNDES sobre o emprego nas empresas de menor porte.** Revista do BNDES, Rio de Janeiro, n. 36, p. 5-42, dez. 2011.

MAGGIOLO, M. & SPANAKIS, G. **Autoregressive Convolutional Recurrent Neural Network for Univariate and Multivariate Time Series Prediction**. Expertise and Skill Acquisition Network (ESAN'19), 2019.

MAKRIDAKIS, S. **Accuracy measures: theoretical and practical concerns.** International Journal of Forecasting, 9(4), 527-529, 1993.

MARTINI, R.A. & TEIXEIRA, L.A.S. **Head/tail breaks: uma proposta metodológica para a classificação de dados regionalizados dos fluxos do cartão BNDES.** Revista do BNDES, Rio de Janeiro, n. 45 , p. [5]-34, jun. 2016.

MITCHELL, T. **Machine learning.** McGraw-Hill Education; 1st edition, 1997.

MODUGNO, M. **Now-casting inflation using high-frequency data.** International Journal of Forecasting, 29(4), 664-675, 2013.

NASSIRTOUSSI, A.K., AGHABOZORGI, S., WAH, T.Y. & NGO, D.C.L. **Text mining for market prediction: A systematic review.** Expert Systems With Applications, 40(16), 7653-7670, 2014.

NIELSEN, M. **How the backpropagation algorithm works.** http://neuralnetworksanddeeplearning.com/chap2.html. Last accessed on 1/9/2019, 2019.

OLAH, C. (2014). **Conv Nets: A Modular Perspective**. http://colah.github.io/posts/2014-07-Conv-Nets-Modular/. Last accessed on 1/9/2019.

OLAH, C. **Understanding LSTM Networks**. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Last accessed on 19/8/2019. 2015.

PAREJA, A., DOMENICONI, G., CHEN, J., MA, T., SUZUMURA, T., KANEZASHI, H., KALER, T. & LEISERSEN, C.E. EvolveGCN: **Evolving Graph Convolutional Networks for Dynamic Graphs**. MIT-IBM, arXiv. 2019.

PARR, T. & HOWARD, T (2018). **The Matrix Calculus You Need For Deep Learning**. https://arxiv.org/abs/1802.01528. Last accessed on 22/8/2019.

PASCANU, R., GULCEHRE, C., CHO, K. & BENGIO, Y. **How to Construct Deep Recurrent Neural Networks.** International Conference on Learning Representations (ICLR'13), 2013.

PEROZZI, B., AL-RFOU, R. & SKIENA, S. **DeepWalk: online learning of social representations.** International Conference on Knowledge Discovery and Data Mining (KDD'14), 2014.

PRANDONI, P. & VETTERLI, M. **Signal Processing for Communications**. EPFL Press, 2013.

REDDI, S., KALE, S. & KUMAR, S. **On the convergence of Adam and beyond.** International Conference on Learning Representations (ICLR'18), 2018.

ROSENBLATT, F. **The perceptron: A probabilistic model for information storage and organization in the brain**. Psychological Review, 65(6), 386-408, 1958.

ROY, A., SUN, J., MAHONEY, R., ALONZI, L., ADAMS, S. & BELING, P. (2018). **Deep Learning Detecting Fraud in Credit Card Transactions.** Systems and Information Engineering Design Symposium (SIEDS'18).

RUDER, S. **An overview of gradient descent optimization algorithms**. http://ruder.io/optimizing-gradient-descent/. Last accessed on 19/08/2019, 2018.

SCHMIDHUBER, J. **Deep Learning in Neural Networks: An Overview.** Neural Networks, 61(1), 85-117, 2015.

SEBRAE. **Participação das Micro e Pequenas Empresas na Economia Brasileira**, 2014.

_______. **Anuário do trabalho nos pequenos negócios: 2016.** 9.ed / Serviço Brasileiro de Apoio às Micro e Pequenas Empresas; Departamento Intersindical de Estatística e Estudos Socioeconômicos, 2016.

SEZER, O.M. & OZBAYOGLU, A.M. **Algorithmic financial trading with deep convolutional neural networks**: Time series to image conversion approach. Applied soft computing, 70(1), 525-538, 2018.

SHAH, D., CAMPBELL, W. & ZULKERNINE, F.H. **A Comparative Study of LSTM and DNN for Stock Market Forecasting.** IEEE International Conference on Big Data, 2018.

SHCHERBAKOV, M.V., BREBELS, A., SHCHERBAKOVA, N.L., TYUKOV, A.P., JANOVSKY, T.A. & KAMAEV, V.A. **A Survey of Forecast Error Measures**. World Applied Sciences Journal, 24(1), 171-176.

SHI, X., CHEN, Z., WANG, H. & YEUNG, D. **Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting.** Advances in Neural Information Processing Systems 28 (NIPS'15), 2015.

SIAMI-NAMINI, S., TAVAKOLI, N. & NAMIN, A.S. **A Comparison of ARIMA and LSTM in Forecasting Time Series.** 17th International Conference on Machine Learning and Applications, 2018.

SILVA, T.C. & ZHAO, L. **Machine Learning in Complex Networks**. Springer International Publishing, 2016.

SILVA, T.C., SOUZA, S.R.S. & TABAK, B.M. **Monitoring vulnerability and impact diffusion in financial networks.** Journal of Economic Dynamics and Control, 76(1), 109-135, 2017.

SILVA, T.C., ALEXANDRE, M.S. & TABAK, B.M. **Bank lending and systemic risk: A financial-real sector network approach with feedback.** Journal of Financial Stability, 38(1), 98-118, 2018.

SMETS, F. & WOUTERS, R. **The American Economic Review**, 97(3), 586-606, 2017.

SORAMÄKI, K. & COOK, S. **Network Theory and Financial Risk**. Risk Books, 2016.

SUN, S., WEI, Y. & WANG, S. **AdaBoost-LSTM Ensemble Learning for Financial Time Series Forecasting.** International Conference on Computational Sciences (ICCS'18), 2018.

TANG, J., QU, M., WANG, M., ZHANG, M., YAN, J. & MEI, Q**. LINE: Large-scale Information Network Embedding.** International Conference on the World Wide Web (WWW'15), 2015.

TOFALLIS, C. **A better measure of relative prediction accuracy for model selection and model estimation.** The Journal of Operational Research Society, 66(1), 1352-1362.

TRAN, D.T., KANNIAINEN, J., GABBOUJ, M. & IOSIFIDIS, A. **Data-Driven Neural Architecture Learning for Financial Time-Series Forecasting.** Digital Image & Signal Processing (DISP'19), 2019.

TRIEPELS, R., DANIELS, H. & HEIJMANS, R. **Anomaly Detection in Real-Time Gross Settlement Systems.** Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS'17), 1(1), 433-441, 2017.

VAN DEN BOOM, W., REEVES, G. & DUNSON, D.B**. Quantifying uncertainty in variable selection with arbitrary matrices.** IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP'15), 2015.

VAN DEN OORD, A., DIELEMAN, S., ZEN, H., SIMONYAN, K., VINYALS, O., GRAVES, A., KALCHBRENNER, N., SENIOR, A. & KAVUKCUOGLU, K. WAVENET: **A Generative Model for Raw Audio.** DeepMind, arXiv, 2016.

VAN VEEN, F. & LEIJNEN, S. (2019). **The Neural Network Zoo.** http://www.asimovinstitute.org/neural-network-zoo/. Last accessed on 22/8/2019.

VELIČKOVIĆ, P. (2018). O**verview of neural network architectures for graph-structured data analysis.** https://www.cl.cam.ac.uk/~pv273/slides/UCLGraph.pdf. Last accessed on 16/10/2019.

WENG, B., LU, L., WANH, X., MEGAHED, F.M. & MARTINE, W. **Predicting short-term stock prices using ensemble methods and online data sources.** Expert Systems With Applications, 112(1), 258-273, 2018.

WU, Z., PAN, S., CHEN, F., LONG, G., ZHANG, C. & YU, P.S. **A Comprehensive Survey on Graph Neural Networks**. arXiv, 2019.

XU, K., WANG, Z., WITBROCK, M., WU, L., FENG, Y. & SHEININ, V. **Graph2Seq: Graph to Sequence Learning with Attention-based Neural Networks.** IBM Research, arXiv, 2018.

ZAKI, M.J. & MEIRA, W. JR. **Data Mining and Analysis:** Fundamental Concepts and Algorithms. Cambridge University Press, 2016.

ZIEGLER, K., CAELEN, O., GARCHERY, M., GRANITZER, M., HE-GUELTON, L., JURGOVSKY, J., PORTIER, P. & ZWICKLBAUER, S**. Injecting Semantic Background**

Knowledge into Neural Networks using Graph Embeddings. International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE’17), 2017.

APPENDIX I

Figure 28 visualises the whole raw dataset, where the vertical line indicates the split for training[75] and test sets:

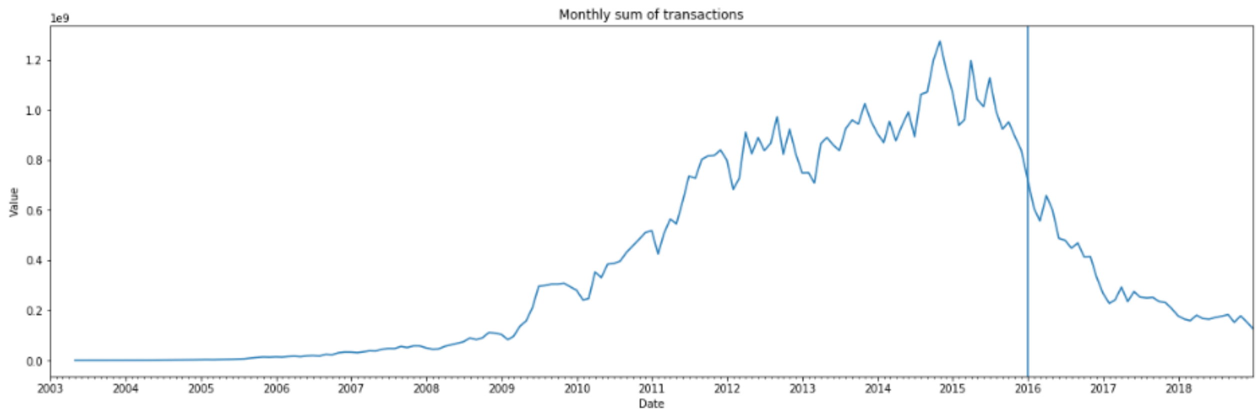


Figure 28: Monthly sum of transactions derived from the BNDES dataset.

One can notice that the scale varies greatly in different time periods: the difference for example from 2008 to 2015 is over 12-fold, when considering yearly totals. First, possible structural breaks are investigated via Zivot-Andrews test. While it suggests potential structural break point for the whole data (07-2015), the test statistic does not imply significance[76]. Next, in the figure below, histograms for raw[77] and Box-Cox transformed training data are plotted:

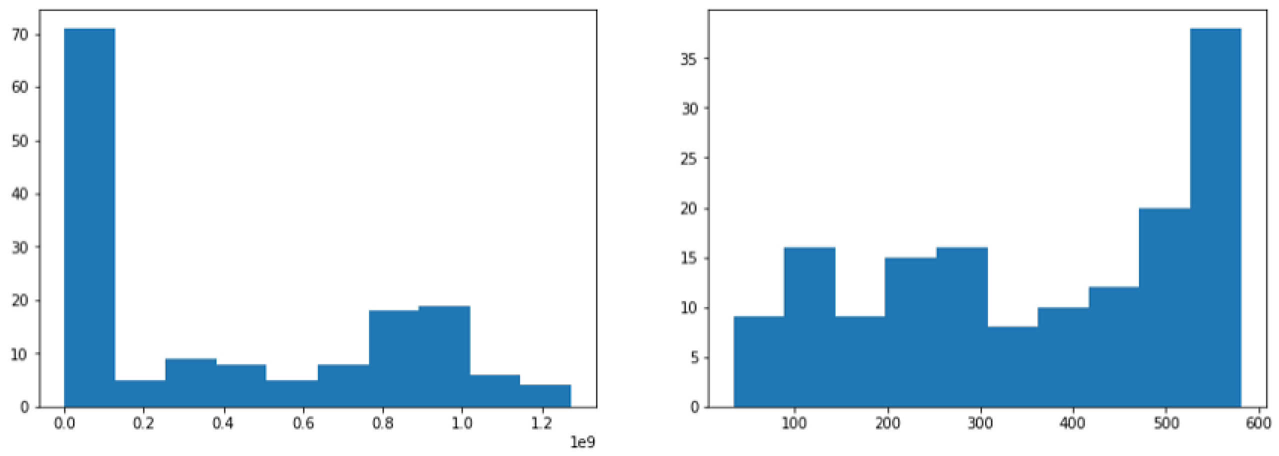


Figure 29: Histograms for raw (left) and transformed (right) training data.

[75] Recall that the last 12 months from the training set were used for validation purposes.
[76] The critical value for 10% significance level was -4.58 while the test statistic was -3.9393.
[77] The heavy-tailed distributions displayed by the histograms follow the findings in Martini and Teixeira (2016).

The figures do not offer much support for the presence of normality in both raw and transformed training data as histograms do not conform to normal distribution. One a side note, the Box-Cox transformation reduces a certain amount of skewness. In terms of statistical tests, Shapiro-Wilk, D'Agostino's K-squared and Anderson-Darling cannot support the null hypothesis that the raw or transformed data were drawn from a normal distribution[78]. The same applies to exponential distribution according to Anderson-Starling test[79]. Furthermore, even though KPSS (Kwiatkowski–Phillips–Schmidt–Shin) test supports the null hypothesis of stationarity for raw data, the ADF and PP (Phillips-Perron) tests cannot reject the null hypothesis of stationarity[80]. In the case of transformed data, ADF and PP tests cannot reject the null hypothesis of nonstationarity and KPSS cannot support the null hypothesis of stationarity[81]. In essence, stationarity cannot be assumed in both cases. Next, fitting a linear regression model to raw and transformed training data gives the opportunity to verify whether misspecification occurs.

[78] For Shapiro-Wilk and D'Agostino's K-squared, the highest p-value was in the order of 10^{-9}. While the critical value for 1% significance level in Anderson-Starling's test was in both cases 1.065, the lowest test statistic was 5.656.

[79] The lowest Anderson-Starling's test statistic was 17.753 while the critical value for 1% significance level was 1.949.

[80] ADF's p-values until 5 lags did not go lower than 0.468 for all specifications (no drift no trend, with drift no trend, with drift and trend). Lowest p-value for PP was 0.529 for the previous three configurations. KPSS suggests nonstationarity at 10% (0.0698) significance level for drift and trend, otherwise stationarity.

[81] ADF's p-values until 5 lags did not go lower than 0.346 for all specifications (no drift no trend, with drift no trend, with drift and trend). Lowest p-value for PP was 0.86 for the previous three configurations. KPSS suggests nonstationarity (for no drift no trend, and drift and trend) with a significance level of 5% (0.0133 and 0.0362, respectively).

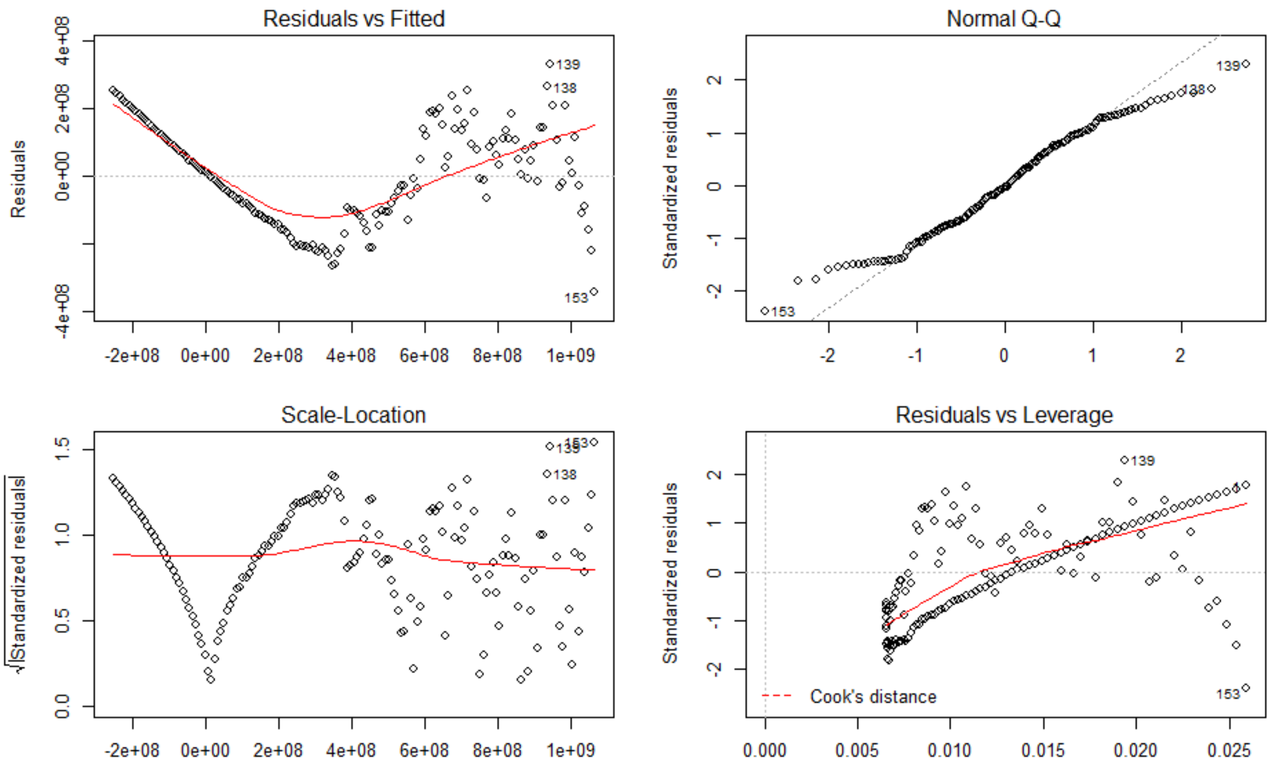


Figure 30: Diagnostics for the linear regression model fitted for raw data.

The first row of Figure 30 suggest that the residuals have a pattern indicating nonlinearity and do not exhibit normality. The second row shows non-constant variances in the residuals' errors suggesting heteroskedasticity, whereas the presence of outliers, while demonstrating weak evidence, cannot be confirmed. According to RESET test, the model suffers from misspecification with a p-value in the scale of 10^{-16}.

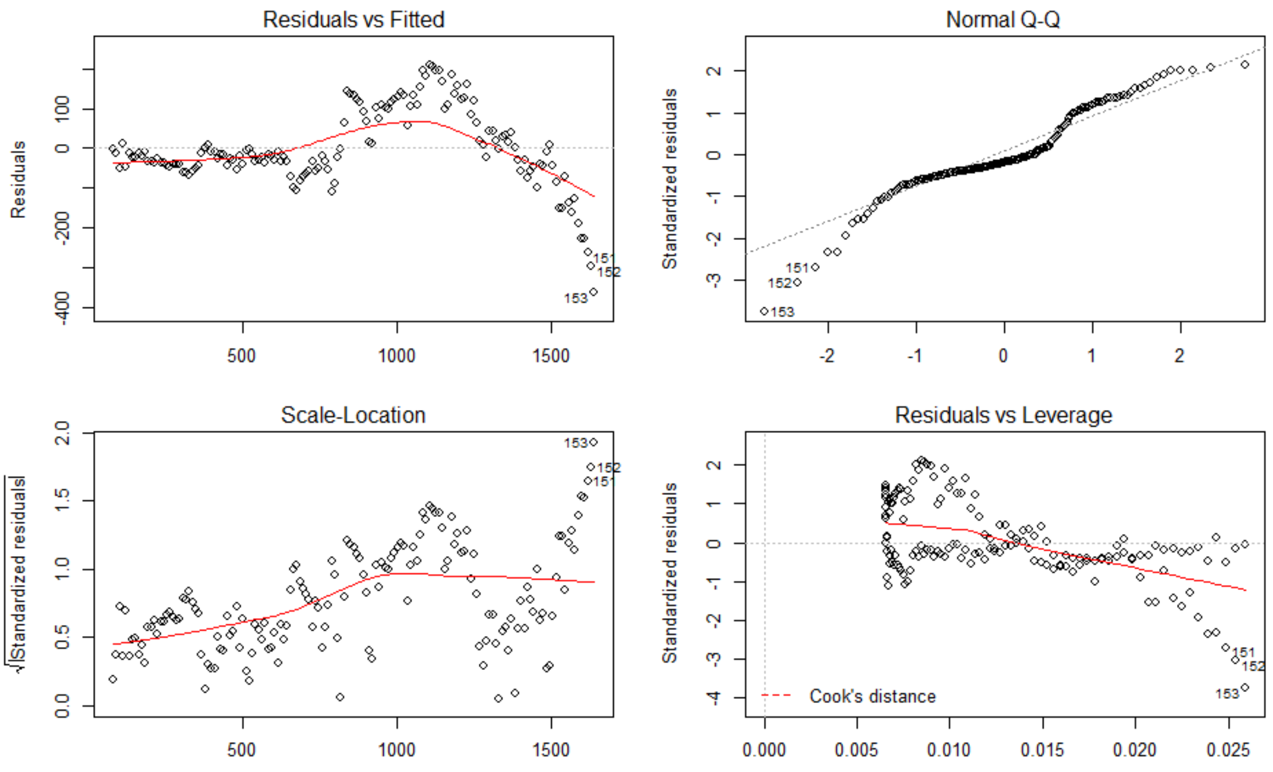


Figure 31: Diagnostics for the linear regression model fitted for transformed data.

From Figure 31, even though more subtle in nature, similar deductions can be made as previously. Misspecification is present by RESET test with a p-value again in the range of 10^{-16}. These findings suggest the presence of nonlinearity. Finally, the correlograms for the raw and transformed training data are as follows:

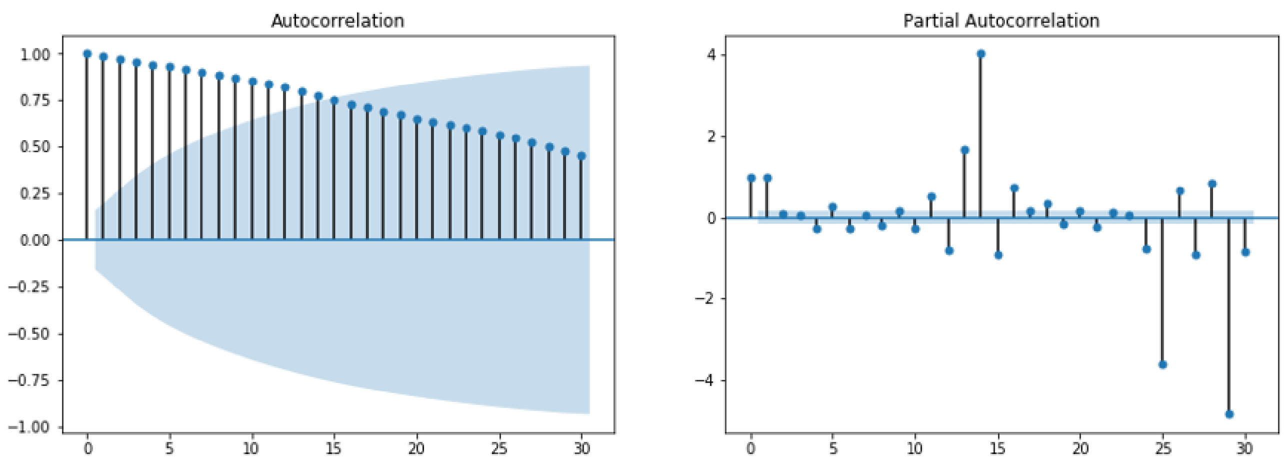


Figure 32: Correlograms for the raw training data.

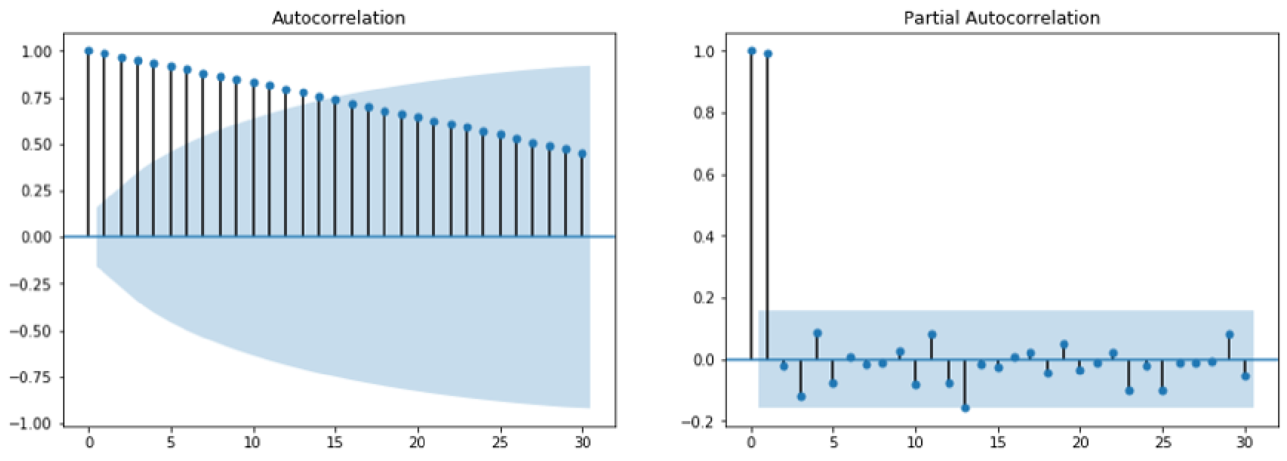


Figure 33: Correlograms for the Box-Cox transformed training data ($\lambda \approx 0.235$).

The correlograms indicate that without differencing the observations in the training data in both cases are serially correlated until the 14th lag whereas in residual errors there appears to be useful information until the first lag.