Long Term Forecasting of Industrial Electricity Consumption Data With GRU, LSTM and Multiple Linear Regression

Buzăţoiu, Roxana

Award date:
2020

Awarding institution:
Royal Institute of Technology

DEGREE PROJECT IN INFORMATION AND COMMUNICATION TECHNOLOGY
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2020

Long Term Forecasting of Industrial Electricity Consumption Data With GRU, LSTM and Multiple Linear Regression

ROXANA BUZATOIU

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Authors

Roxana Buzatoiu <buzatoiu@kth.se>
Information and Communication Technology
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden

Examiner

Vladimir Vlassov
KTH Royal Institute of Technology

Academic Supervisor

Shatha Jaradat
KTH Royal Institute of Technology

Industry Supervisor

Irina Dragunova

Vattenfall

Abstract

Accurate long-term energy consumption forecasting of industrial entities is of interest to distribution companies as it can potentially help reduce their churn and offer support in decision making when hedging.

This thesis work presents different methods to forecast the energy consumption for industrial entities over a long time prediction horizon of 1 year. Notably, it includes experimentations with two variants of the Recurrent Neural Networks, namely Gated Recurrent Unit (GRU) and Long-Short-Term-Memory (LSTM). Their performance is compared against traditional approaches namely Multiple Linear Regression (MLR) and Seasonal Autoregressive Integrated Moving Average (SARIMA). Further on, the investigation focuses on tailoring the Recurrent Neural Network model to improve the performance.

The experiments focus on the impact of different model architectures. Secondly, it focuses on testing the effect of time-related feature selection as an additional input to the Recurrent Neural Network (RNN) networks. Specifically, it explored how traditional methods such as Exploratory Data Analysis, Autocorrelation, and Partial Autocorrelation Functions Plots can contribute to the performance of RNN model.

The current work shows through an empirical study on three industrial datasets that GRU architecture is a powerful method for the long-term forecasting task which outperforms LSTM on certain scenarios. In comparison to the MLR model, the RNN achieved a reduction in the RMSE between 5% up to to 10%.

The most important findings include: (i) GRU architecture outperforms LSTM on industrial energy consumption datasets when compared against a lower number of hidden units. Also, GRU outperforms LSTM on certain datasets, regardless of the

choice units number; (ii) RNN variants yield a better accuracy than statistical or regression models; (iii) using ACF and PACF as dicovery tools in the feature selection process is unconclusive and unefficient when aiming for a general model; (iv) using deterministic features (such as day of the year, day of the month) has limited effects on improving the deep learning model's performance.

Keywords

Time Series Analysis, Recurrent Neural Networks, long-term Forecasting, Exploratory Data Analysis, Multiple Linear Regression, ACF, PACF, Energy Sector

Sammanfattning

Noggranna långsiktiga energiprognosprognoser för industriella enheter är av intresse för distributionsföretag eftersom det potentiellt kan bidra till att minska deras churn och erbjuda stöd i beslutsfattandet vid säkring.

Detta avhandlingsarbete presenterar olika metoder för att prognostisera energiförbrukningen för industriella enheter under en lång tids förutsägelsehorisont på 1 år. I synnerhet inkluderar det experiment med två varianter av de återkommande neurala nätverken, nämligen GRU och LSTM. Deras prestanda jämförs med traditionella metoder, nämligen MLR och SARIMA. Vidare fokuserar undersökningen på att skräddarsy modellen för återkommande neurala nätverk för att förbättra prestanda.

Experimenten fokuserar på effekterna av olika modellarkitekturer. För det andra fokuserar den på att testa effekten av tidsrelaterat funktionsval som en extra ingång till RNN -nätverk. Specifikt undersökte den hur traditionella metoder som Exploratory Data Analysis, Autocorrelation och Partial Autocorrelation Funtions Plots kan bidra till prestanda för RNN -modellen.

Det aktuella arbetet visar genom en empirisk studie av tre industriella datamängder att GRU -arkitektur är en kraftfull metod för den långsiktiga prognosuppgiften som överträffar ac LSTM på vissa scenarier. Jämfört med MLR -modellen uppnådde RNN en minskning av RMSE mellan 5 % upp till 10 %.

De viktigaste resultaten inkluderar: (i) GRU -arkitekturen överträffar LSTM på datauppsättningar för industriell energiförbrukning jämfört med ett lägre antal dolda enheter. GRU överträffar också LSTM på vissa datauppsättningar, oavsett antalet valenheter; (ii) RNN -varianter ger bättre noggrannhet än statistiska modeller eller regressionsmodeller; (iii) att använda ACF och PACF som verktyg för upptäckt i funktionsvalsprocessen är otydligt och ineffektivt när man siktar på en allmän modell;

(iv) att använda deterministiska funktioner (t.ex. årets dag, månadsdagen) har begränsade effekter på att förbättra djupinlärningsmodellens prestanda.

Nyckelord

Tidsserieanalys, återkommande neurala nätverk, långtidsprognoser, undersökande dataanalys, multipel linjär regression, ACF, PACF, energisektor

Acknowledgements

I wish to thank all the people whose assistance made possible the completion of this project. I would like to thank my supervisor Shatha Jaradat for providing guidance and feedback throughout this project.

I would like to acknowledge my colleagues from my internship at Vattenfall for their wonderful collaboration and to the RD department who kindly answered all my questions. I would particularly like to thank my supervisor at Vattenfall, Irina Dragunova, who guided and encouraged me to be professional, for her patience, interest and support she showed during this project.

Also, I would like to express my sincere gratitude to EIT Digital, for letting me be part of this incredible network of ambitious students, for creating a strong community of innovators and for facilitating the cross-border cooperation between top European Universities.

Finally, I want to thank Martina, Keyshav and my flatmate Nici for their support, affection and for their positive stand who always made me feel confident in my abilities after coming off the phone with them.

Acronyms

ACF Autocorrelation Function

ANN Artificial Neural Network

AR Autoregressive

ARIMA Autoregressive Integrated Moving Average

EDA Exploratory Data Analysis

FCRBM Factored Conditional Restricted Boltzmann Machine

FNN Feedforward Neural Network

GRU Gated Recurrent Unit

GD Gradient Descent

KNN K-Nearest Neighbors Algorithm

LSTM Long-Short-Term-Memory

LR Linear Regression

LSS Least Squares

LTNN Layerwise Tensorized Neural Network

MA Moving Average

MLP Multilayer Perceptron

MLPs Multilayer Perceptrons

MLR Multiple Linear Regression

MSE Mean Squared Error

MAE Mean Absolute Error

MAPE Mean Average Percentage Error

MWh Megawatt-hour

NRMSE Normalized Root Mean Squared Error

PACF Partial Autocorrelation Function

ReLu Rectified Linear Unit

RMSE Root Mean Squared Error

RNN Recurrent Neural Network

RNNs Recurrent Neural Networks

SARIMA Seasonal Autoregressive Integrated Moving Average

SGD Stochastic Gradient Descent

SVM Support Vector Machine

Contents

Chapter 1

Introduction

Prediction systems dedicated for short term forecasting of hours or days ahead are popular across various energy sectors. This thesis presents methods of analysing and forecasting time-series over a 1 year horizon using industrial datasets of five years length. A bridge between traditional methods and learning methods is proposed and evaluated. Experiments with various configurations of Recurrent Neural Networks (RNNs) [16] methods is assessed against a MLR[6] benchmarking model in terms of the application requirements. In particular, the following work compares GRU against LSTM variants of RNNs.

In this introductory chapter, I motivate the research, introduce the area under study, highlight the research gap, and position my research in the vast pool of approaches proposed to tackle forecasting problems.

1.1 Motivation

Time-series analysis and forecasting are concern with exploring the characteristics of time-related data and fitting models that can explain and predict the future behaviour of various processes. Traditionally, forecasting problems relied on applying stochastic models to the time-series (namely mathematical expression describing the probability structure of a time-series). Later, Artificial Neural Network (ANN)[1] and Support Vector Machine (SVM) methods have been introduced. With the last big wave of computing driven by embedded systems, more data are being generated than ever before. This causes an increased need for intelligent ways of processing it. One of

the data types that has enlarged is time-series, stemming from sensory data collected to boost industry performance. The energy industry relies heavily on sensory data and time-series analysis in both production and sales. Time-series analysis finds its applications in many aspects of energy management, from forecasting load to hedging electricity, to enhancing customer experience. While many researches and experiments address short term forecasting using modern methods, to my knowledge, minimal research has been done in the field of long-term forecasting. In addition, the evaluation of proposed models on industry datasets, especially of industry datasets of business customers (as opposed to households) is limited due to their constrained availability for research purposes. Following the increased data expansion, one of the new challenges imposed on time-series analysis is the model's scalability, which has made many of the traditional models obsolete.

Additional challenges addressed from an industry point of view include harnessing the field-specific datasets through data exploration and time-series mining that can lead to new discoveries and better data pre-processing prior to any model fitting. This comes in addition to the challenge of selecting a model for an industry-specific dataset while taking into account the field specifics and the business context. Despite their long time since their conception, stochastic models such as Autoregressive Integrated Moving Average (ARIMA)[8] and its variants are widely adopted methods, liked for their simplicity in usage and generally good performance. However, although many time-series applications can be solved using such models, it is debatable if these are still an efficient method to deal with the new industry requirements and challenges of continuous data acquisition. This thesis examines deep learning methods where historical datasets and insights from traditional methods are fully exploited in order to improve the accuracy of long-term forecasting.

There is a vast literature on ANN for time-series forecasting. Surveys [53] [2] [3] reviewing the related works existing in the literature identified that the most widely used ANNs in the past 30 years to solve forecasting problems have been MLR, with a single hidden layer Feedforward Neural Network (FNN) [52], where the inputs are unchanged. More recent studies show RNNs potential and strength in scaling to much longer sequences than would be practical for networks without sequence-based specialization. [14].

In this thesis, I compare deep learning methods (namely sequence to sequence models

[30]) against other methods available in the literature applied to long-term forecasting. In addition, I compare their suitability for the energy consumption application and draw a comparison to the traditional methods which are most of the time adopted in the industry. Nonetheless, I ran a thorough data exploration aimed to enhance the discovery potential of specifics of the dataset and use the newly found insights towards improving the machine learning models.

In recent years, RNNs techniques have demonstrated outstanding performance in modeling datasets from different domains. This thesis is motivated by evaluating the potential of RNNs techniques on time-series data, specifically for long-term forecasting applications, as to my knowledge, the research on these type of datasets is very limited. Another motivational factor is the identification of a suitable RNN architecture for the case application, as the current literature does not agree on a single best architecture. In the same time, the thesis is motivated to draw a comparison between the new learning methods and traditional modeling, where the manual extraction of features is required. Nevertheless, the motivation is fueled by bridging traditional techniques of time-series analysis to learning methods, an approach that has shown, to my knowledge, little interest in the research. Consequently, the value and insights gained from the traditional time-series analysis aim to enhance learning methods techniques.

This thesis analyzes different preprocessing and modeling strategies in order to get some insights about the impact of these strategies on the performance of RNN algorithm. Even though many of the strategies used in the proposed methodology are done according to a specific dataset, I believe some of the methods can be applied in other time-series forecasting applications such as heat and water consumption. For example, the exploratory data analysis techniques can be used on any time-series datasets from the energy sector to visually present characteristics of the consumers. Secondly, including deterministic features as an input to RNN models can be generalized to any time-series dataset.

1.2 Background

Two of the sectors with the highest electricity consumption are the industrial sector and households, each of them weighing as much as 30% of total electricity consumption of Europe according to Eurostat. In an industrial setting, total electricity consumption

can be energy predominantly used for production, as well as for lighting and other business uses in the manufacturing industries.

Forecasting problems can be subdivided into long-term forecasting, where the prediction window varies from 1 year up and short term including hourly, daily, and sometimes monthly predictions. Depending on the specific case scenarios, both short and long-term forecasting problems are common in the energy sector. Long term predictions of energy consumption of industrial entities can be used as a strategy of improving customer experience. Nonetheless, a reliable long-term forecast can be used as a tool for reducing the risks associated with the hedging process. Several research studies explore the applicability of RNNs on short term time-series forecasting. However, the number of studies evaluating the performance of RNNs on long-term prediction is limited.

A common practice within the utility market is hedging, which refers to fixing the prices of electricity for a specific future horizon. Hedging practices are conducted between distributors of electricity and electricity markets, but also between business consumers (industrial consumers) and distributors. In such settings, forecasted electricity consumption becomes the main reference in hedging for both parties. Thus, a poor forecasting strategy leads to higher risks and higher costs. In an industrial context where the consumption per user is high, in the orders of thousands of Megawatt-hour (MWh), the penalty for misprediction is also high and a small percent increase in the forecast accuracy has a large impact. However, a common practice is to use a persistence model when communicating their estimated forecast to the providers. Such a forecast is a naive forecasting version when the previous year's consumption is projected for the next year.

The work of this thesis emerged from a real business case provided by one of the large energy producers of Europe. The case energy provider delivers energy to hundreds of business customers, each of them owning a various number of smart-meters that signal back the localized consumption. Thus, the total energy consumption per business customer takes the form of a multivariate time-series.

Given the continuous expansion of datasets both horizontally (length of time-series) and vertically (increase in the number of time-series per business customer and an increase in the number of total business customers), the scalability, and training times are some of the requirements of the proposed solution.

1.3 Problem

In contrary to the wide acceptance and implementation of deep learning methods in various domain areas, there is a gap between the theory, research, and the implementation of such methods in the industrial sector of electrical energy. Several studies emphasize the potential of RNNs in short-term forecasting, however few, to my knowledge, highlight the potential of these techniques in long-term forecasting applications. It remains to be determined what optimization techniques and recurrent architectures can be used to best realize the potential of RNNs in the later case.

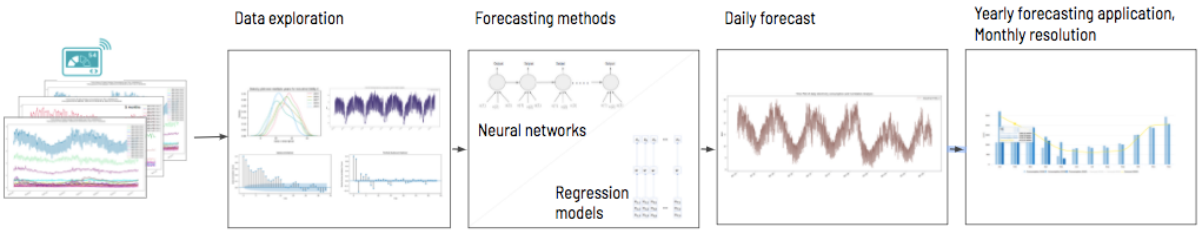
Two of the most popular RNN architectures are GRUs and LSTMs (see Section 2.3.4). When it comes to the architectural choice, the current literature does not clearly agree on which one is the most efficient. In particular, both architectures are deep structures with the potential to model complex nonlinear patterns and both of them are appropriate for modeling time series data. Alike other fields, the energy sector has not established yet an effective architecture for long term energy consumption predictions.

The electricity consumption data of the industrial sector has been less in the attention of researchers, possibly due to the limited access to industrial datasets. Thus there is a lack of exploration and model experimentation of such datasets.

In practice, traditional methods such as regression models are widely spread and utilized in the industry as compared to learning methods, mainly preferred for their model explainability and informing feature engineering. A natural question arises: which models can be used for long-term forecasting and whether the more complex models are significantly better in this context. Nevertheless, it is interesting to study how Exploratory Data Analysis (EDA) can be used in deciding the input for the learning methods. An overview of the research problem is visually presented in Figure 1.3.1.

This thesis addresses the problem of long-term forecasting of single entities in the industrial sector through the means of both traditional and deep learning methods. Considering that the scalability, faster training times, and dealing with the nonlinearity of datasets are known challenges of traditional models and requirements from the industry perspective, the research question of the following research summarizes to:

Figure 1.3.1: An overview of the research problem



How can we perform long-term electricity consumption forecasts using deep learning techniques in a scalable and accurate way? Which RNN architectures are suitable for the current case study? How do the results of traditional methods compare to learning methods for electricity consumption dataset? How can EDA be used to improve the performance of learning models?

1.4 Purpose

The purpose of the thesis is to bridge the gap between the latest developed machine learning models and their application in forecasting tasks in the energy industry. While these models showed promising results in other domains, this thesis explores their performance directly on industry datasets from the energy sector. Moreover, the project aspires to contribute to the development of the methods used for long-term forecasting, that have shown less interest to researches in opposition to the methods used in short-term realm.

1.5 Goal

This work aims to implement a forecasting procedure for industrial electricity consumption using smart-meter load data, for a time horizon of one year and monthly resolution. At the same time, the aim is to propose a solution that can be easily integrated into production and have a real impact at a business level. Additionally, the goal is to bring new insight into smart meter data of the industrial sector. Thus, a performance model comparison between a MLR and RNN model is conducted. In this thesis, I explore RNNs method to produce long-term forecasts as opposed to short-term forecasts of energy consumption.

1.5.1 Benefits, Ethics and Sustainability

Solving load forecasting problems is of big interest to utility companies providing services such as electricity, gas, water, heating to direct customers or businesses. At the same time, everyone from the seller, the consumer, and everyone else will be directly or indirectly affected by the forecasting capabilities of each energy company. Accurate load forecasting models can optimize the production and the buying process of these utilities and thus achieve leaner processes with a minimal amount of waste. There is no information, personal or non-personal, that could in any way be used to reveal identities of the industrial entities. Sustainability is another important criterion when computational expensive methods are proposed. The training costs must be taken into consideration and plan accordingly.

1.6 Contributions

This thesis project illustrates the research performed in order to tailor a forecasting model suitable to predict electricity consumption values for a period of one year ahead which can be further used for decision support in hedging, operations, and planning of energy. The thesis contributions include:

- An overview of the main techniques available in the literature to generate time-series predictions.
  First, an overview of state-of-the-art forecasting approaches is provided. Secondly, the main techniques are explained in detail, namely stochastic models, regression models, and deep learning models.
- An EDA of electricity consumption over a period of 5 years of three industrial entities in Sweden.
  The EDA is used to summarize the main characteristics of datasets using visual methods such as distribution plots, time-plots, Autocorrelation Function (ACF), Partial Autocorrelation Function (PACF).
- The design and implementation of a Multiple Linear Regression model meant to harness the insights of the EDA.
  Thus, the input features are solely time-related and chosen based on new insights. The MLR model was chosen as a benchmarking model and later compared to more complex models.

- The design and implementation of a Recurrent Neural Network Model. Different aggregations, time-related inputs, and RNN architectures were tested to better understand potential improvement directions. Using ACF and PACF as tools for feature selection showed error improvements only on one out of three datasets.
- An evaluation of deep learning methods (specifically RNNs) in comparison to statistical models alternatives, namely MLR and SARIMA for long-term forecasting tasks.
- A interpretation of the performance of GRU and LSTM architectures in relation with the specifics of the experimental datasets.
- A system that can scale easily and generalize to the addition of new customers.

The empirical study is, to my knowledge, the only study where the prediction models are evaluated on industry datasets and where the electricity consumption behaviour of industrial customers is evaluated. The current study is unique also by the fact that it explores the suitability of two popular RNN architectures, namely GRU and LSTM in relation with the specifics of the industrial electricity consumption datasets.

Chapter 2

Long-Term Forecasting of time-series

Considering the continuously increasing amount of sensory data stored by a plethora of businesses and the focus on data-driven decision making of the majority of 21st century industries, the call for insightful analysis and processing methods of time-series is high. Processing large historical time-series with traditional forecasting methods can be challenging and cumbersome especially when non-linearity variation or non-stationary of the time series is present. For this reason, in recent years it was common to turn to machine learning methods for forecasting time-series.

In this chapter, a detailed description about the background theory and related work will be covered. In Section 2.2, I introduce the field of time-series analysis and forecasting from my research perspective, then I introduce a selection of methods used in time-series analysis. In Section 2.3, I present a collection of both traditional and newer methods applied to tackle forecasting tasks and their individual challenges. Section 2.3.4 includes a closer look into the fundamentals that my research builds upon, namely MLR and RNN models for forecasting. In Section 2.4 I introduce the state-of-the-art in what concerns applied research in the energy sector.

2.1 Directional Approaches in Forecasting Time-Series

The research community proposes different focus areas for tackling forecasting problems using modern approaches such as learning methods. Three directional categories have been identified. Each of them is advocating for its importance in achieving improved forecast accuracy.

- Feature based methods
  Focus on finding explanatory variables that can improve the forecast. It implies conducting procedures such as feature selection, feature engineering, clustering of time-series and other methods aiming to maximize the input information fed to the forecasting model. The feature based methods can be further on divided depending on the type of explanatory features, namely structural features of time-series (e.g. trend, seasonality etc.) or exogenous variables (e.g. weather, economical factors etc.).

- Model based methods
  Aim to find a suitable model based on the use case requirements and data type. There are several criteria in selecting a forecasting model. If we consider the characteristics of time-series data as the main criterion, one can distinguish between: linear/non-linear models, stationary/non-stationary, univariate/multi-variate models. Models can also be grouped based on the used forecasting framework. Here, we have classical models versus modern methods. The forecasting models could also be subdivided into pure versus hybrid models.

- Hyper-parameters tuning methods
  Focus on improving the forecast in terms of accuracy or speed of a specific model by running a hyper-parameters tuning job. Hyper-parameters values are set before training starts, unlike model parameters, which are determined during training. It implies running many training jobs over a range of hyper-parameter values (e.g. optimizers, activation functions, loss functions, learning rates). The optimum set of values depends on the algorithm, the training data, and the specified metric objective.

2.2 Time-Series Analysis

According to Chatfield's work [9], there are four basic objectives of time-series analysis: (1) description of the main properties, often with visual methods, (2) explanation of time-series through other time-series; (3) prediction/forecasting; (4) control, as in quality control of a manufacturing process. In this report, the time-series analysis objectives include the description and prediction of time-series.

Time-series forecasting is concerned with predicting the future behaviour of a random variable based on its past behaviour as well as exogenous factors where available. Forecasting methods are further explained in Section 2.3. The procedure of fitting a time-series to a proper model is termed as time-series analysis [1]. A prime step in building a forecasting model is the selection of input features which can be achieved by running a thorough EDA.

2.2.1 Exploratory Data Analysis

Exploratory Data Analysis is used for finding out what the data can tell us beyond the formal statistical modeling or hypothesis testing task. The underlying assumption of the exploratory approach is that the more one knows about the data, the more effective data can be used to develop and refine the forecasting model [20].

Simple measures of the distribution of time-series values (which ignore their time ordering) can often be informative. Moreover, scanning through examples, visual analysis of time-series, understanding their distribution, and identifying patterns are useful methods in gaining more insights. More than examining just individual variables, looking at the relationship between two or more variables brings additional value.

2.2.2 ACF and PACF

The ACF and PACF are instruments used to identify dependencies between the values of a time-series at different times. "Just as correlation measures the extent of a linear relationship between two variables, autocorrelation measures the linear relationship between lagged values of a time-series" [23]. The ACF is answering the question of whether the successive observations are related by computing an autocorrelation coefficient at each time-lag k. The value r_k is the autocorrelation between any

two time-series values separated by a lag of k time units. Given measurements, $Y_1, Y_2, \ldots, Y_N$ at time $X_1, X_2, \ldots, X_N$, the lag k autocorrelation function is defined as [7]:

$$r_k = \frac{\sum_{i=1}^{N-k} \left(Y_i - \bar{Y}\right)\left(Y_{i+k} - \bar{Y}\right)}{\sum_{i=1}^{N} \left(Y_i - \bar{Y}\right)^2} \tag{2.1}$$

The ACF measures for the overall correlation between two values of the same variable at the different times X_i and X_{i+k}. Looking at ACF could be misleading in regard with which lags are significant. For example, if X_{i+k} is strongly correlated with X_{i+k-1}, this correlation could show up in the previous lags X_{i+k-2}, X_{i+k-3}.

In contrast, PACF measures for the direct correlation between X_t and X_{t-k}. In other words, PACF measures the linear relationship between X_t, and X_{t-k}, eliminating the influence of the intervening variables. Each partial autocorrelation can be obtained as a series of regressions of the form:

$$\tilde{y}_t = \phi_{21}\tilde{y}_{t-1} + \phi_{22}\tilde{y}_{t-2} + e_t \tag{2.2}$$

where $\tilde{y}_t$ is the original series minus the sample mean, $y_t - \bar{y}$. The estimate of ϕ_{22} gives the value of the partial autocorrelation of order 2. Extending the regression with k additional lags, the estimate of the last term gives the partial autocorrelation of order k. The mathematical details of computing the partial autocorrelation are explained in depth in Box et al. [7]. The significant lags are considered the ones which scores are above the confidence level.

The classical Box-Jenkins time-series model uses both ACF and PACF tools to construct the statistical model. Specifically, PACF are useful in identifying the order of AR(p) while ACF are used to identify the order of the moving average term MA(q). A classical model is considered adjusted when both ACF and PACF show no significant autocorrelation.

2.3 Forecasting Methods

Common methods used to produce forecasting models include time-series regression, exponential smoothing, ARMA (and its variants ARIMA, SARIMA, etc), Dynamic regression models[23]. The general setup of a forecasting problem can be expressed

as below, where the aim is to predict the future behaviour of a (univariate) time-series $z_{i,0}$ for item $i \in I$ given its past.

$$z_{i,0}, \ldots, z_{i,T-2}, z_{i,T-1}, z_{i,T} \Longrightarrow P\left(z_{i,T+1}, z_{i,T+2}, \ldots z_{i,T+h}\right) \tag{2.3}$$

Different techniques can be approached when computing the unknown value of future time-series. Depending on the scope of the problem, one can use point estimation techniques or probabilistic forecasts. A point estimator is a statistic that produces a single numerical value, while the interval estimator produces an interval within which the parameter is expected to fall.

2.3.1 Evaluation Metrics

Common metrics to evaluate point forecasts include:

- Mean Absolute Error (MAE) over the entire forecasting horizon h : mean(e_t), where $e_t = |z_t - \widehat{z}_t|$
- Mean Average Percentage Error (MAPE): $\frac{1}{h}\sum_t e_t / |z_t|$
- Root Mean Squared Error (RMSE) : $\sqrt{\text{mean}\,(e_t^2)}$
- Coefficient of determination, R^2. It measures the goodness of fit for regression; it usually ranges from 0 to 1, though it can take on negative values as well.

Domain specific considerations

The direction of errors often has an economic impact in long-term forecasting applications. In power system planning applications, positive forecasting errors can result in planning inadequate capacity, and in turn loss of load service. In hedging processes, the forecast provided by an industrial entity is used in defining the hedging contract with the energy provider. Positive forecasting errors resulting from underestimating the real energy consumption impose limitations to the industrial entity. In this case, the industrial entity cannot consume more than what it initially declared in the contract. Negative forecasting errors result in the industrial entity overbuying electricity from the electricity provider. In addition, the industrial entities are penalized by the energy provider for what they do not consume. The importance and impact of negative versus positive errors should be further evaluated by a domain

expert based on the forecasting application use case. Thus, reporting these values is important.

2.3.2 Stochastic Time-Series Models for Univariate Time-Series

The underlying idea of stochastic models assumes that a time-series $\{x(t), t = 0, 1, 2, \ldots\}$ follows a certain probability model. Thus, the sequence of observations of the series is a sample realization of the stochastic process that produced it. Stochastic time-series models can be categorized into linear and non-linear models. A time-series model is said to be linear or non-linear depending on whether the current value of the series is a linear or non-linear function of past observations. Models can be further divided into univariate versus multivariate models. The linear models for univariate time-series include:

- Autoregressive (AR)
  An *AR(p)* assumes that the observed time-series depends on a weighted linear sum of the p past values of X(t) and a random shock ε_t also known as white noise. This model formulates the predicted value X(t) as: $X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + \varepsilon_t$
- Moving Average (MA)
  The intuition behind a *MA(q)* is that the observed time-series X_t depends on a weighted linear sum of past, q , random shocks ε_t. Technically, the model can be formulated as follows: $X_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q}$
- Autoregressive moving average (ARMA)
  An ARMA(p,q) model consists of both an AR(p) term and a MA(q) term. In other words, the model consists of the weighted sum of past values (autoregressive component) and the weighted sum of past errors (moving average component). Mathematically, it can be formulated as: $X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q}$
- Autoregressive integrated moving average (ARIMA)
  While ARMA(p,q) models stationary time-series, ARIMA(p,d,q) model allows for modeling non-stationary time-series. The model assumes a series can become stationary by differentiating it d times.

- Seasonal Autoregressive Integrated Moving Average (SARIMA)
  SARIMA(p,d,q)(P,D,Q)s is used to model time-series that possess a seasonal component that repeats every s observations. The uppercase notation is used for the seasonal parts of the model, and lowercase notation for the non-seasonal parts of the model. The SARIMA model is composed of three trend elements that require configuration, which are the same as ARIMA. The parameter (p, d, q) stand *(Trend autoregression order, Trend difference order, Trend moving average order)*. The next four seasonal elements $(P, D, Q)s$ stand for *(Seasonal autoregressive order, Seasonal difference order, Seasonal moving average order)*, while m stands for the number of time steps for a single seasonal period.

The estimation of the model parameters is mostly done by maximum likelihood estimation.

2.3.3 Regression Models for Time-Series

Regression analysis is a set of statistical processes for estimating the relationship between the outcome variable and one or more predictors variables. One of the features of a regression model is that most of the predictors variables are continuous and generally use dummy coding.

From the pool of various regression models, linear regression is one of the common models used in prediction tasks. Linear regression models can be approached from a traditional perspective or by solving the system of equations using an artificial neural network, which automatizes the process. One of the advantages of solving regression models with neural networks is that they have more prediction power and they can learn a non-linear function approximator. The traditional regression model advantage is that they are supported by a proven theoretical background, and less complex thus easier to understand.

Regression models are used to forecast the value of interest by assuming that there is a linear relationship with another time-series x. One can include multiple time-series as predictors or explanatory random variables (see Figure 2.3.1 for visual representation). Such models are named multiple linear regression. Assuming our prediction $\hat{z}_t$ is a

weighted combination of features $x_{t,1}, \ldots, x_{t,D}$, we have:

$$\widehat{z}_t = \sum_{d=1}^{D} w_d x_{t,d}$$

where $\hat{z}$ is also called the fit for the **x** [13].

In linear regression models, the predicted output is continuous and has a constant slope. The goal is, given the data x (the predictors) and z (the forecast value) to find the weights (w) so that determine the hyperplane with the best fits for the training set. Common performance measures are RMSE, Least Squares (LSS) (also known as Mean Squared Error (MSE)). These are used to quantify how well the model fits the data. In traditional methods, the Normal Equation is implemented giving the mathematical results directly. Normal Equation is an analytical approach to Linear Regression with a Least Square Cost Function. The value for the model parameters can be computed without using Gradient Descent. Another way to train linear regression is to use gradient descent, which is an optimization algorithm. In opposition, in a linear regression model using neural networks, the weights are approximated. The best fit can be defined using the principle of least square regression which provides a way of choosing the feature's weights by minimizing the sum of the squared errors. First, weights are found by solving the following equation:

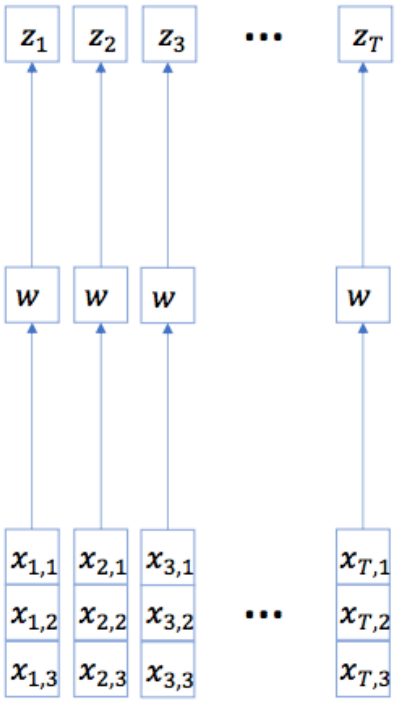


Figure 2.3.1: Multiple linear regression model representation with three time-series as as predictors ($x_{T,1}$, $x_{T,2}$, $x_{T,3}$) [11]

$$w^\star = \underset{w}{\operatorname{argmin}} \sum_{t=1}^{T} (z_t - \widehat{z}_t)^2 = \sum_{t=1}^{T} \left(z_t - \sum_{d=1}^{D} w_d x_{t,d} \right)^2$$

And later $w^\star$ can then be used to make forecasts (where h is the chosen forecasting

horizon):

$$z_t = \sum_{d=1}^{D} w_d^\star x_{t,d} \quad t = T+1, \ldots, T+h$$

Features for Linear Regression A disadvantage of linear regression is that the features need to be given, thus hand designed. Features used in linear models are themselves time-series, $x_{1,d}, x_{2,d}, \ldots, x_{T+H,d}$. Features of time-series data can be summarised as follows:

- Trend features
- Seasonal features: dummies (one-hot indicators), periodic features (e.g. Fourier)
- Lagged target values (e.g. use z_{t-1} and z_{t-2} as features to predict z_t)
- Seasonal lagged target values (e.g. use z_{t-S} to predict zt, with $S = 12$ for monthly data)
- (Weighted) average features (e.g. meanz$_{t-7:t-1}$)

2.3.4 Learning Methods

Under the umbrella of ANN, one can differentiate between two model architectures.

Feed forward neural networks

Feed forward neural networks, known in literature also as deep feedforward networks or Multilayer Perceptrons (MLPs) [43], are a type of neural network where the connections between the nodes do not form a cycle[14]. The information moves in only one direction, from the input nodes, through the hidden nodes, and to the output nodes (as shown in Figure 2.3.2). Each neuron in a hidden layer computes a

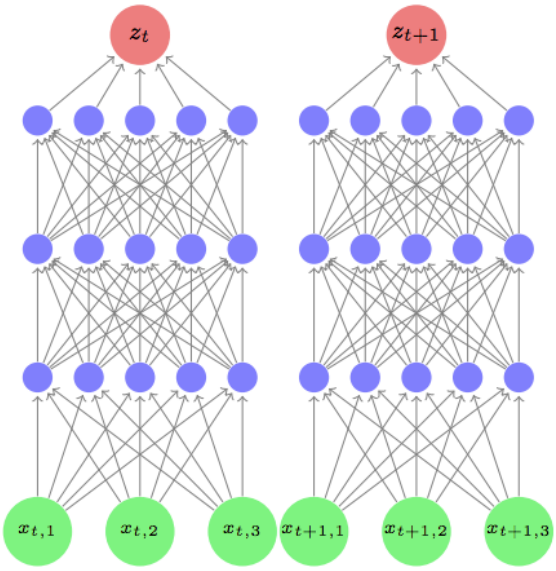


Figure 2.3.2: MLPs arhitecture [11]

function of the previous layer, followed by a non-linear activation function.

$$h_{l,j} = \sigma\left(\mathbf{w}_{l,j}^{\top}\mathbf{h}_{l-1} + b_{l,j}\right) \tag{2.5}$$

MLPs are robust baseline methods that can model non-linear patterns in the data, as opposed to linear models. Such features are enabled by the non-linear activation function σ of each neuron. Common activation functions include $sigmoid$, $tanh$, $ReLU$, etc. Each hidden layer outputs a new representation of its input vector which aims to synthesize the characteristics of the time-series. Thus, MLPs are able to learn complex input-output relationships. In this way, we can say that the feature extraction is automated. In comparison to linear models, MLPs require less manual feature engineering. However, the accuracy of the output depends on the choice of input data. In addition, adopting MLPs as a forecasting method implies hyperparameter tuning (e.g. of regularization, learning rate, activation function, etc.).

Recurrent Neural Networks

While previous models assume that all inputs (and outputs) are independent of each other, the concept of recurrent neural networks (RNN) lies in the fact that for certain tasks it is necessary to include information from the previous values and thus the design of the neuron has focused on leveraging the sequential information in a dataset. In other words, RNNs are meant to analyze sequences of data and model temporal patterns. This is achieved by augmenting the neuron with a memory, which captures information about what has been calculated so far in its hidden layer. Each recurrent neuron has three sets of weights: the weight u for the input $x^{(t)}$, the weight w for the hidden state at the previous time step $h^{(t-1)}$ which is the output vector of the previous time step, and the weight v for the hidden state at the current time step $h^{(t)}$ (see Figure 2.3.3). The output vector $y^{(t)}$ is computed with the following equations:

$$\begin{aligned} h^{(t)} &= \tanh\left(u^T x^{(t)} + w^T h^{(t-1)}\right) \\ y^{(t)} &= \text{sigmoid}\left(v^T h^{(t)}\right) \end{aligned} \tag{2.6}$$

Stacking multiple neurons forms a layer. Stacking layers of cells give a deep RNN. Depending on the forecasting task, one can choose between different RNN design patterns:

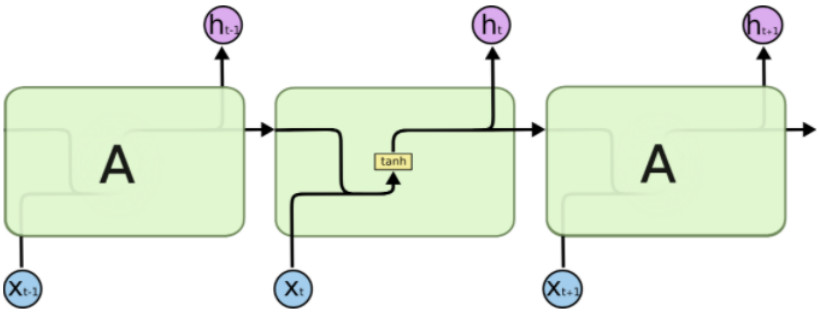


Figure 2.3.3: Repeating module in a standard RNN [38]

- Vector-to-Sequence: a network that takes a single input at the first time step, and it outputs a sequence: $x[1] \mapsto \{\hat{y}[t]\}$
- Sequence-to-Vector: a network that takes as input a sequence and ignores all the outputs except for the last one: $\{x[t]\} \mapsto \hat{y}[T]$
- Sequence-to-Sequence: takes a sequence of inputs and produce a sequence of outputs: $\{x[t]\} \mapsto \{\hat{y}[t]\}$
- Encoder-Decoder: it is a sequence-to-vector network (encoder), followed by a vector-to-sequence network (decoder): $\{x[t]\} \mapsto h \mapsto \{\hat{y}[t]\}$

The vanilla RNN explained above encountered the big problem of vanishing or exploding gradients, which in practice made learning possible only from a short past sequence. There have been several modifications to the RNN proposed to remedy these problem of which the most popular are Gated Recurrent Unit networks and Long Short Term Memory.

LSTM

LSTM [21] is a type of arhitecture explicitly designed to avoid the long-term dependency problem. While the basic RNN contains a single hidden layer in each cell (see Figure 2.3.3), an LSTM contains four interacting layers in each cell (see Figure 2.3.4). By using LSTMs, the network can learn what to store and what to throw away. The operational principle behind stand in splitting the memory it two states: the short-term state, which is represented by the vector $h^{(t)}$ and the long-term state, represented by $C^{(t)}$ vector. Then, LSTM can remove or add information to the cell state, regulated by three gates:

- *Forget gate:* decides what information will be deleted from the cell state by

applying a *sigmoid* over the previous hidden state $h^{(t-1)}$ and the input $x^{(t)}$.

$$f^{(t)} = \sigma\left(W_f \cdot \left[h^{(t-1)}, x^{(t)}\right] + b_f\right)$$

- *Input gate:* decides what new information will be stored in the cell state. First, the *sigmoid* input layer $i^{(t)}$ decides which values will be updated. Secondly, a *tanh* layer decides potential candidate values. Thirdly, the old cell state $C^{(t-1)}$ gets updated.

$$\begin{aligned} i^{(t)} &= \sigma\left(W_i \cdot \left[h^{(t-1)}, x^{(t)}\right] + b_i\right) \\ \tilde{C}^{(t)} &= \tanh\left(W_c \cdot \left[h^{(t-1)}, x^{(t)}\right] + b_c\right) \\ C^{(t)} &= f^{(t)} * C^{(t-1)} + i^{(t)} * \tilde{C}^{(t)} \end{aligned}$$

- *Output gate:* decides the final output by running a *sigmoid* on the hidden state and the input. Then, the output is multiplied with the long term state $C^{(t)}$.

$$\begin{aligned} o^{(t)} &= \sigma\left(W_o\left[h^{(t-1)}, x^{(t)}\right] + b_o\right) \\ h^{(t)} &= o^{(t)} * \tanh\left(C^{(t)}\right) \end{aligned}$$

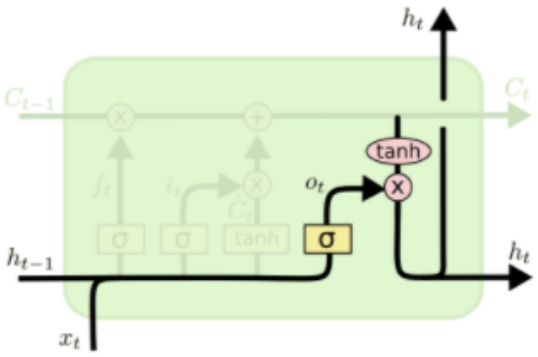


Figure 2.3.4: LSTM neuron arhitecture, last update step [38].

GRU

GRU [10] is newer and simplified version of the LSTM cell which has only two gates: reset gate and update gate. GRU removed the cell state $C^{(t)}$ present in LSTM cell and used only the hidden state $h^{(t)}$ to transfer information to the recurrent unit or to the next layer [41]. Compared to LSTM, the hidden state of the GRU can be interpreted as a merging between the cell state and hidden state.

- *Update gate:* it acts similar to the forget and input gate of LSTM. It decides what information to remove and what new information to add.

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

- *Reset gate:* decides how much past information to forget.

$$\begin{aligned} z_t &= \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \\ r_t &= \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \\ \tilde{h}_t &= \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \end{aligned}$$

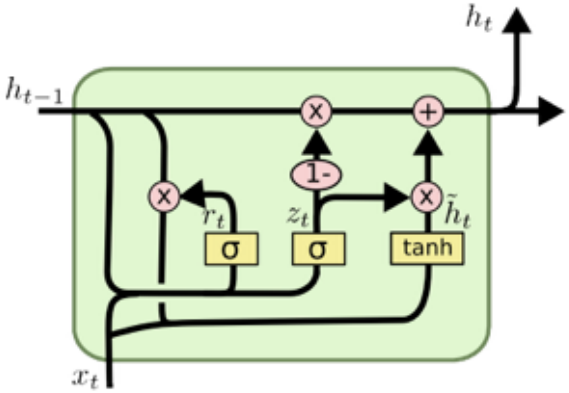


Figure 2.3.5: GRU architecture [38].

GRU uses less training parameters and therefore less memory, executes and trains faster than LSTM's [10]. However, LSTM is known to perform better on datasets using longer sequences.

Features for Recurrent Neural Networks GRU and LSTM can extract and memorize characteristics of time-series automatically during training. However, one can help the model by adding extra features that highlight certain known characteristics. Features that can be encoded into input variables include:

- Lags
- Trend
- Seasonality

- Dummy variables

Since the range of values of the features can vary widely, it is a common practice to use normalization. Another motivation is that neural networks tend to converge faster. Two popular choices of rescaling features are MinMaxScaler (2.7) and StandardScaler (2.8).

$$\tilde{x} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{2.7}$$

$$\tilde{x} = \frac{x - \text{mean}(x)}{\text{sd}(x)} \tag{2.8}$$

Optimizers

A machine learning algorithm is built by specifying an optimization procedure, a cost function, and a model family. The optimization algorithm is used to change the attribute of the neural network such as weights and learning rate to reduce losses. Some of the common optimizers and their specifics are listed below:

- Gradient Descent (GD)
  GD is the regular gradient descent optimization, which works to find the minima of a cost function and where the weights are updated as follows:

$$w_i^{(\text{next})} = w_i - \eta \frac{\partial J(\mathbf{w})}{\partial w_1}$$

- Stochastic Gradient Descent (SGD)
  SGD is one of the most used optimization algorithms for machine learning. The advantage of using SGD as opposed to GD is that the weights updates are done as each sample is processed. Thus, the subsequent calculations already use improved weights.

- Momentum
  Momentum optimizer increases the training speed by giving to the update process a tendency to keep moving in the same direction. At each iteration, it adds the local gradient to the momentum $\mathbf{m}$, where β is the momentum and it takes values between 0 and 1.

$$m_1 = \beta m_1 + \eta \frac{\partial \mathrm{J}(\mathbf{w})}{\partial \mathrm{w}_1}$$
$$w_1^{(next)} = w_1 - m_1$$

- Nesterov momentum
  Nesterov Momentum is a faster variant of the Momentum optimization.

- AdaGrad
  AdaGrad is an optimizer that keeps track of a learning rate for each parameter and it adapts the learning rate over time. It decays the learning rate faster for steep dimensions than for dimensions with gentler slopes. Each feature w_i is updated using the following formula:

$$s_1 = s_1 + \left(\frac{\partial J(\mathbf{w})}{\partial w_1}\right)^2$$
$$w_1^{(next)} = w_1 - \frac{\eta}{\sqrt{s_1 + \epsilon}} \frac{\partial J(\mathbf{w})}{\partial w_1}$$

- RMSProp
  RMSProp is solving AdaGrad problem of stopping before the global minimum is reached. The main difference consists of accumulating only the gradients from the most recent iterations (not from the beginning of training).

$$s_1 = \beta s_1 + (1 - \beta) \left(\frac{\partial J(w)}{\partial w_1}\right)^2$$
$$w_1^{(next)} = w_1 - \frac{\eta}{\sqrt{s_1 + \epsilon}} \frac{\partial J(w)}{\partial w_1}$$

- Adam optimization
  Adam optimizer combines the ideas of Momentum optimization and RMSProp.

2.4 Related Work

The common schools of thought on studying energy consumption prediction involves stochastic models, regression models and modern machine learning methods such as neural networks. One of the limitation of stochastic models is the increase of complexity when modelling multi-variate time-series. Commonly, their usability is explored on short-term predictions use cases. Their performance on long-term tasks is not well highlighted in the literature. The following category, namely regression models greatly reduced the complexity in modelling multivariate inputs. MLR is an

important method in multivariate statistical analysis which makes possible to predict the outcome of a response variable using several explanatory variables. MLR is widely used in the industry and in many research fields. In , the author uses MLR and ANN to predict the electricity consumption of Thailand. In Tuaimah's research [47], the MLR method is used to present a short-term load forecast for Iraq's power system requirements. However, one limitation of these models is that the correlation between the variables changes with time and space [4]. A second limitation lays in the fact that the predictors need to be manually selected by a domain expert or a thorough exploratory analysis.

Neural networks tackled the tedious feature selection process by automatically learning features of time series from historical data using recurrent units and thus increasing model's accuracy.

Common machine learning methods explored in the literature for long-term forecasting include Multilayer Perceptron (MLP), K-Nearest Neighbors Algorithm (KNN) and other FNN variations. An overview of several ANN techniques used in forecasting long and medium-term have been summarized in [27]. In the same paper [27], the author proposes a Multiplicative Error Model aimed for a forecasting window of 48 months, showing good results for periods of recession and high volatility. In a study from 2019 [39], the author is comparing a simple MLP model architecture with a KNN model to predict electricity load for 1 year ahead horizon using temperature as an additional explanatory variable. However, the training of MLP was a FNN without backpropagation. In a study from 2008 [24], the authors use a backpropagation model to forecast daily maximum electricity load and reach comparable percentage errors in a 2 months test. A limitation of FNN is that they cannot capture sequential information in the input data which is required for dealing with sequence data.

Recently, RNN [22] proved to be a popular choice of model class for time-series forecasting ([54], [42]). These networks tackle the limitations of FNN by taking advantage of the sequential information. Several success stories from recent forecasting competitions placed sequence models amongst the winning solutions [50].

RNNs are employed for modeling various types of time related data. Especially LSTM-RNNs have received notable performance in several tasks such as music generation, natural language translation [46], speech recognition [15] and traffic

prediction. In [51], the authors evaluate a LSTM architecture versus a hybrid LSTM on short-term traffic flow prediction concluding that the hybrid LSTM model is closer to the accuracy and real-time requirements of the task.

LSTM-RNNs have shown interest in the energy sector research field. In [29], the authors propose a LSTM recurrent neural network-based framework to predict short-term residential load forecasting. The results proved that LSTM has a strong ability for time series data prediction, outperforming state-of-the-art in the field of load forecasting. The authors emphasize the capability of LSTM to deal with the volatility of predicting individual smart-meters. The work of Song Xuany et al. [45] proposes a LSTM based model for time-series oil rate prediction and an optimization method for the model. The LSTM model outperformed ARIMA and RNN vanilla models providing a promising method for predicting the time-series oil rate.

However, the application of LSTM architecture on long-term applications is limited. One mention includes the work of Hartwig [19] who shows good accuracy of LSTMs on large time scale of mid-to-long term prediction (with 12 steps ahead, monthly resolution) for photovoltaic power generation.

GRU-RNNs have also received attention in the realm of sequence learning. Effective applications of GRU based models include text recommendations [5] and short-term photovoltaic forecasting [49]. Moreover, in [32], GRU has shown superiority over MLR and SVM for the processing of time series data and forecasting of primary Chinese energy consumption. In a study on 27 state-of-the-art methods for predicting electricity prices [31], deep neural networks, LSTM and the GRU models are shown to obtain a predictive accuracy that is statistically significantly better than all other models. In his work, Jozefowicz [26] compared the GRU and LSTM models and found that the GRU model was able to achieve comparable results to the LSTM model on multiple issues. However, applications on industrial electricity consumption for long-term forecasting of these architectures are not existent to my knowledge.

In addition to architectural choices, research work has been done in the direction of improving neural network models performance by enhancing the process of selecting input features for the model. In their research, Flores et al. show good results for the usage of the autocorrelation function and partial autocorrelation function as tools to help and improve the construction of the input layer for univariate time-series artificial neural network (ANN) models as already used in classical time-series analysis [12].

When forecasting time-series with machine learning, even though the methods allow for learning and automatic extraction of data characteristics, the selection of features is important as it enhances the learning. The current research experiments with the tools such as PACF and ACF which could be used for feature selection.

Considering these deep learning approaches, my work relates to [45], [31], [29], [32] which use GRU and LSTM architectures. Inspired by good results of LSTM on forecasting tasks, my work relates to [29] in the sense I am applying a similar LSTM architecture and an exploratory analysis as tools to conduct my research. However, the model's configuration and tools in the exploratory analysis differ. Nevertheless, I have been inspired by [32] who highlights the potential of GRU simpler architecture and competitive performance to LSTMs. The two RNNs are compared against a MLR benchmark. In addition, my work has been influenced by [12] to use PACF and ACF tools as a method to enhance the feature selection process of MLR and improve the input features selection for the RNN.

To my knowledge, the energy sector has not established yet an effective approach for long term energy consumption predictions. My work is unique as I evaluate the effect of the two architectures with a different configuration on the industrial energy consumption datasets from various industrial players in Sweden. In addition, the application requirements are different where the forecasting has a long-term horizon (1 year) and the used dataset spans over a period of 5 years, larger than in any of the research papers mentioned above. The dataset is unique as each industrial entity is characterised by a multi-variate time series thus the LSTM and GRU based model parameters can increase significantly. The performance of RNN forecasting models was assessed against a MLR via RMSE.

Chapter 3

Use case

From a high-level perspective, this thesis work aims to design a long-term forecasting model for electricity consumption able to generate low accuracy errors that can easily scale up and fit a large corpus of 230 industrial entities. The forecasting time window is long-term as industries hedge for the next one, two, or three years window. In this research, the forecasting model is meant to project electricity consumption values for a yearly horizon.

This section describes the entire collection of the electricity consumption time-series, for which the results and findings of the experiments are targeted. The section also describes the sampled dataset from the entire corpus used for experimentation purposes.

3.1 Electricity Consumption Corpus

The electricity consumption corpus refers to the collection of electricity consumption datasets of 230 industry entities. The proposed solution in this report was designed to be applied to the entire corpus.

Given its characteristics, the dataset can be included in the spatio-temporal data type category. A subcategory of spatio-temporal data type that represents well the current data-set is the *point reference data*. According to [48], point reference data consist of measurements of a continuous spatio-temporal field such as temperature or energy consumption over a set of moving reference points in space and time. The data can be represented as a set of tuples $\{(r_1, l_1, t_1), (r_2, l_2, t_2) \ldots (r_n, l_n, t_n)\}$. Each tuple (r_i, l_i, t_i)

denotes the measurement of a sensor r_i at the location l_i of the spatio temporal filed at time t_i.

The dataset includes the following attributes: meters id, customer id, location of meters. Each of the industrial customers has a various number of smart meters spanning from few to tens of smart-meters, installed at different locations, thus the dataset of each industrial customer comprises multiple time-series. Each smart meter logs energy consumption readings every hour and the recordings are measured in MWh. Individual time-series may differ in terms of the available time window, depending on the contract.

3.2 Sampled Dataset

As the dataset is rather large with over 230 companies reporting energy consumption, a selection of a couple of companies was used for the analysis, experimentation, and evaluation phase. A random selection was used, as opposed to conditional sampling (based on the location of the business customers).

Descriptor	**Count/Values**
Industrial entities:	6
Resolution:	hourly data-points
Number of entries:	1 289 861
Mean hourly energy consumption:	1.8 Mwh
Standard deviation:	4.2
Min-max:	0- 4.3 Mwh
Time window:	2015-01-01 to 2020-03-12

Table 3.2.1: Statistics overview on the sampled dataset

The sample dataset has been randomly selected from the database, comprising the energy consumption data from the period January 2015 to March 2020. Missing data and data comprehensiveness challenges motivated the choice of selecting the datasets where at least 60% of the values of each time-series are available. For the missing values up to 40%, I used interpolation as a data augmentation method.

As a result, the experimental data was comprised of datasets from three different industrial entities. Each of the three datasets consists of a different number of aligned time-series of 1892 time steps each, representing daily energy consumption for an industrial player in Sweden. An overview of the dataset can be seen in Table 3.2.1 .

Chapter 4

Methods

This chapter covers the strategy for this research and it describes the reasoning behind various research choices. First, the chapter introduces the direction undertaken in approaching the forecasting task. Secondly, it presents the data cleaning and preparation choices of the experimental datasets. Thirdly, the chapter covers an exploratory data analysis to visualize characteristics of the corpora and gain insights that motivate further choices (Section 4.3), a review on pre-processing methods (Section 4.2), an assessment of time-series forecasting using traditional methods (Section 4.4.2) versus learning methods (Section 4.4.3).

4.1 Directional Approach in Forecasting time-series

From the methodologies used to approach load forecasting mentioned in Section 2.3, my research is mainly focusing on model selection by developing and comparing two models and evaluating their applicability in the given case scenario. Model selection was chosen to highlight the applicability of modern approaches as opposed to traditional ones.

In the same time, the work considers the feature selection approach and it includes experiments with various structural and time-related features; however it disregards other features such as the exogenous ones. The choice of focusing on time-related features is based on the exploratory data analysis insights, which indicate structure in the series where structural features could lead to good performance of the model.

A second consideration was the complexity of the model. The motivation of the work was to experiment with a simple model that can later be compared with more complex versions. As no prior research on RNNs performance trained exclusively with time-related features were found on a industrial energy dataset, it was natural to first explore this area, before increasing models complexity with exogenous features. Moreover, the following work also considers the hyper-parameter tuning by including experiments with various values for the batch size and sequence size.

Using time and structural features as a central component, two systems for forecasting industrial energy consumption were developed and compared. The first implementation uses basic time related features for training a MLR model. The second implementation compare 3 variants of the RNNs. The first variant uses RNN for feature learning from the individual smart meter data. This setup is compared against RNNs where the same features as the ones provided by the Linear Regression are provided. The third variant adds additional features discovered in EDA.

4.2 Data Cleaning and Preparation

To make a fair comparison of the forecasting methods, the same data cleaning process has been applied to the raw dataset before all the experiments. The data cleaning and preparation operations applied to the input dataset are listed:

- Remove time-series that have more than 60% missing values (missing values are: 0, NaN or empty spaces)
- Apply interpolation for the missing values in the time-series which have at least 40% values recorded

As any other step, the cleaning step was no error-free process and relies on several choices that can impact the training set quality. One of the assumptions made was that the energy consumption per hour can never drop to 0 MWh. Some other observations worth mentioning is that, in the worse case scenario 400 datapoints may be interpolated. A backwards interpolation can lead to a section in the time-series where all the values are equal. As a result, this method disregards the underlying time-series patterns (e.g. seasonal pattern). Table 4.2.1 shows the number of remaining time-series for each of the three experimental datasets, after cleaning was performed.

Table 4.2.1: Remaining number of time-series after cleaning

Dataset	# Remaining time-series
Industrial Entity A	17 out of 31
Industrial Entity B	4 out of 4
Industrial Entity C	6 out of 10

Another pre-processing choice was the aggregation resolution. The raw dataset contained hourly energy consumption. However, the experimental dataset was aggregated to a daily granularity because an hourly input was not serving the goal of 1 year forecast. A monthly granularity was also neglected because it would have had reduced the training dataset to only 60 datapoints (12 months x 5 years). The advantage of using daily aggregation was that it maintained a large dataset for fitting the model and preserves the weekly patterns.

As the total energy consumption differs greatly between industrial entities, scaling the input data before applying any of the models has been adopted. Specifically, min-max scaler was used. This way, the RMSE values between industrial entities could be fairy compared.

4.3 Exploratory Data Analysis

Of special interest was to identify dataset similarities between the industrial consumers, as it affects the model training choices. To explore the dataset, couple of descriptive techniques have been applied such as: distributions, time-plots and autocorrelations. EDA aims to highlight the main properties, sudden changes, outliers and missing values of time-series.

4.3.1 Distributions

A profile for each industrial entity was created by analysing their distribution shape, variance and operational range. The analysis was performed both at an industrial entity level but also at a smart-meter level.

Distributions of unique industrial entities

Distributions (histograms) are constructed using the daily aggregation of consumption per industrial entity ($y_{n,m} = \sum_{n,m}(x_{n,m})$, where $y_{ie,date}$ is the total daily consumption for

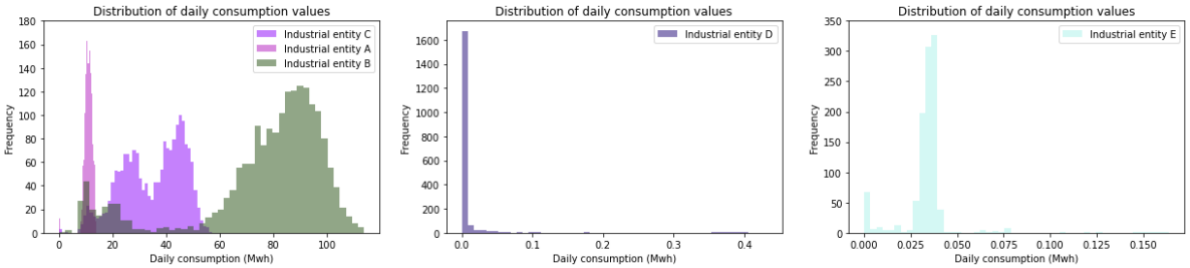


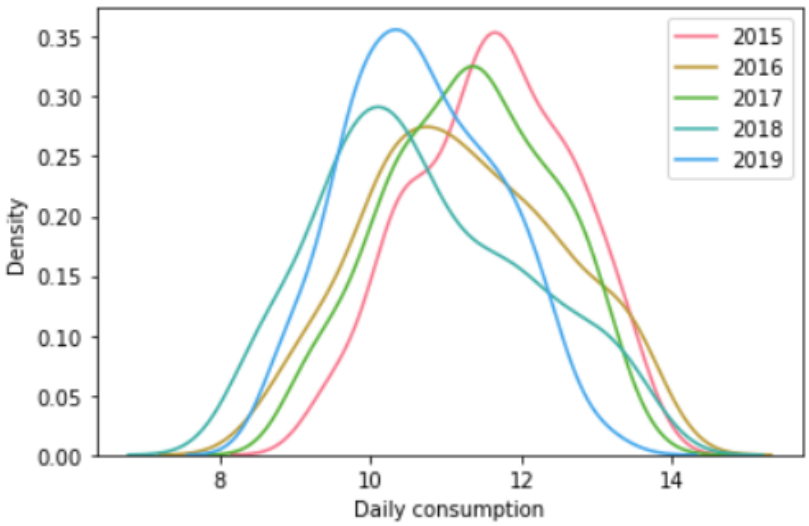
Figure 4.3.1: Distribution of daily total consumption per industrial entity on the experimental dataset; *x axis*: Daily Consumption (MWh), *y axis*: Density Probability; Different operation regimes for each industrial entity

industrial entity n for day m and $(x_{n,m})$ is the hourly consumption). The visualizations motivated the following choices:

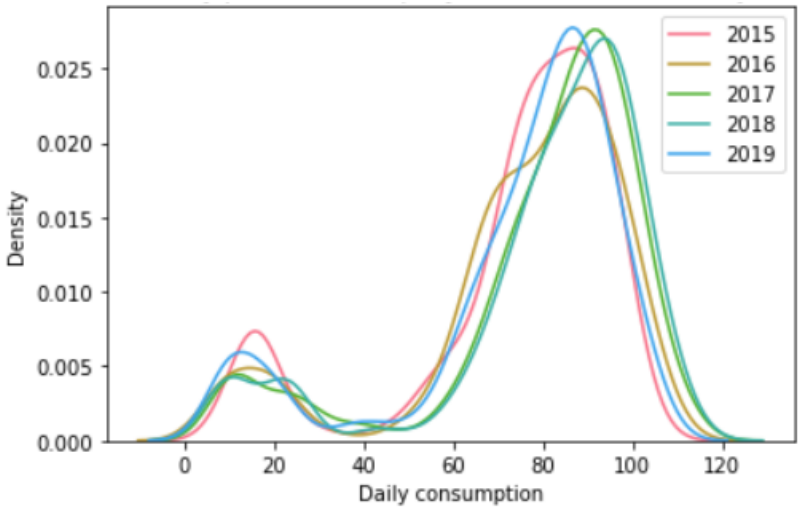
(a) Individual model training for each industrial entity: As shown in Figure 4.3.1, the distribution shape, variance and operational range differ between industrial entities. The histogram shows that the daily consumption of Entity A has a compact shape, with a clear operational profile. Entities B and C display a bimodal distribution indicated by the 2 distinct peaks. Entity D presents a right skewned distribution, where the large amount of 0 values hint towards data quality issues. While Company A has a small variance, Company B and C have a large variance, spanning from 0 to 56 MWh, 130 MWh respectively. Entities D and E have aggregated consumption under 1 MWh thus operating on a lower consumption regime. All of the above reinforce the choice of training a model for each industrial entity as opposed to a general model since there can not be no model fit for all. Also, the distributions guarantee that the validation set (last year in the time-sequence) was sampled from the same distribution as the training set.

(b) Data pre-processing: The variance of daily consumption spans over a large range (i.e. Entity B), which may indicate missing entries or issues with the smart-meters. Another signal of a bad quality was the large amount of values indicating 0 or close to 0 consumption (i.e. Entities D and E in Figure 4.3.1). Thus, interpolation and identification of smart-meters with a majority of missing values for future elimination was required.

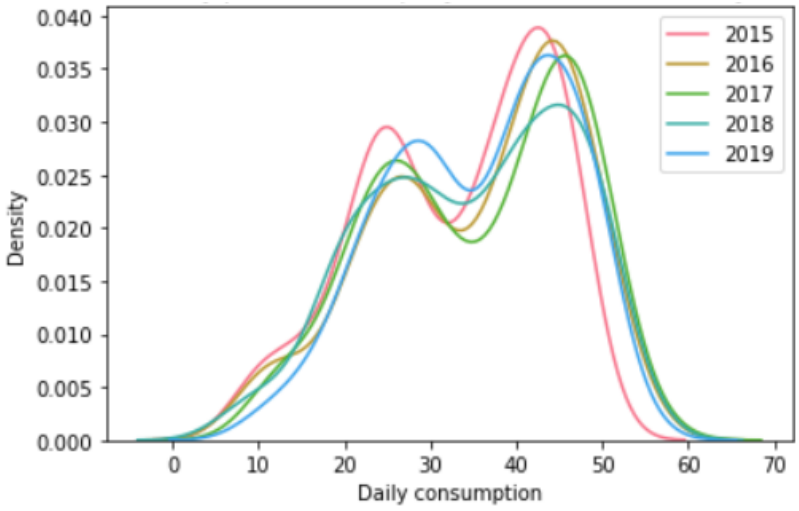
(c) Implementation considerations: In learning, the purpose is to find a hypothesis (function) which approximates a unique target function. This task is carried



(a) Industrial Entity A



(b) Industrial Entity B



(c) Industrial Entity C

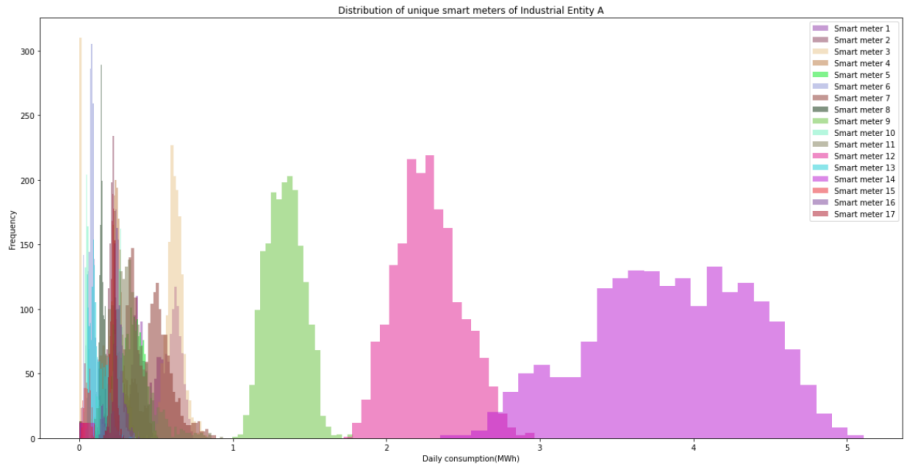
Figure 4.3.2: Density probability of daily consumption time-series (in MWh) for Industrial Entity A, B and C for period 2015-2019.

out based on a set of realizations of the target functions. It is assumed that the realizations of the function are from the same distributions (training dataset, predicted dataset). While in image classification problems, the visual characteristics of an object has the same feature distribution, with time-series it is not the case, as the distribution of inputs might change.

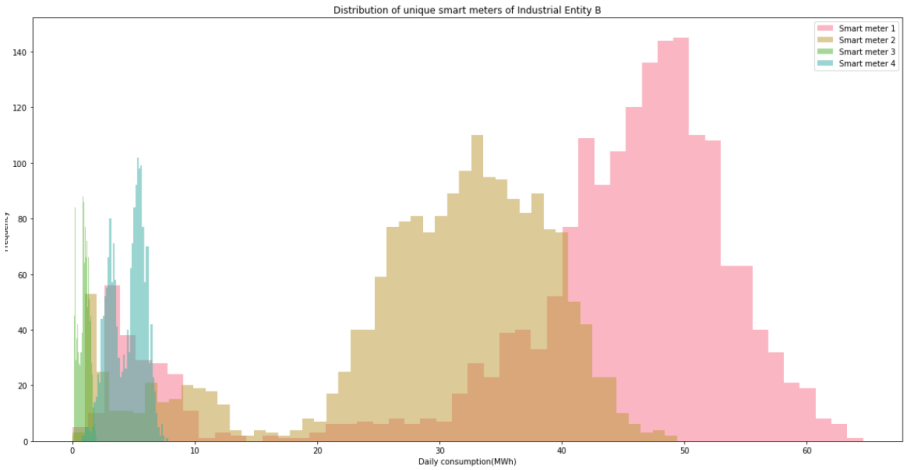
However, with time-series data, the output distribution is not guaranteed as the processes can easily change over time, transforming the distribution. When the development dataset and the production dataset are not from the same distribution, data is not stationary, and the statistical properties will keep shifting as new values come in. In real applications, the addition and removal of smart-meters is dynamic. Whether a model should be retrained depends on the operational range of the newly introduced smart-meters and their impact on the overall distribution. However, Figure 4.3.2 shows that the probability density for five consecutive years is overlapping. The distribution of yearly lagged values is similar (Figure 4.3.2b and Figure 4.3.2c). Thus, as no major distribution shift is observed, one can infer that it is not mandatory to retrain the model yearly.

Distributions of unique smart-meters

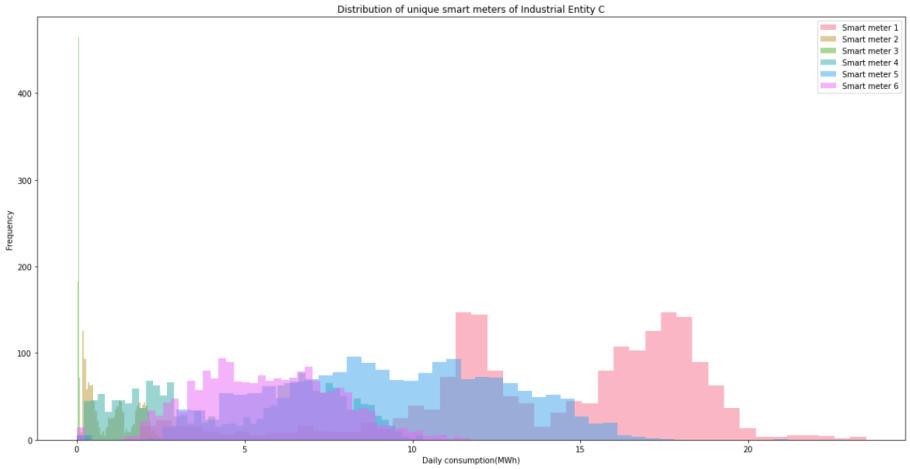
The histograms in this section are plotted after data-cleaning and removal of empty time-series. As shown in Figure 4.3.3, the smart-meters of every industry entity display various distributions, different variance, and operational range. However, there are groups of similar distributions. For example, the smart meters of Industry Entity A operating under 1MWh display small variance and pointy heads. The same applies to *Smart meters 5 and 6* of Industrial Entity C. Smart meters with similar distribution characteristics can be considered as smart meters that operate under similar regimes. This indicates potential in forming clusters of categories with specific operational range. Additionally, we can see that one single smart meter can also operate under two different working regimes reflected by the bimodal distribution(i.e. Industry Entity C: *Smart meter 1*, Industry Entity C: *Smart meters 1, 2*). Bimodal distribution applies to smart meters that operate in the higher end of energy consumption and it can be explained by maintenance consumption times versus production times. Much information is contained at a smart meter level, information that can be exploited by including individual smart meter time-series as input to the forecasting model.



(a) Industrial Entity A



(b) Industrial Entity B



(c) Industrial Entity C

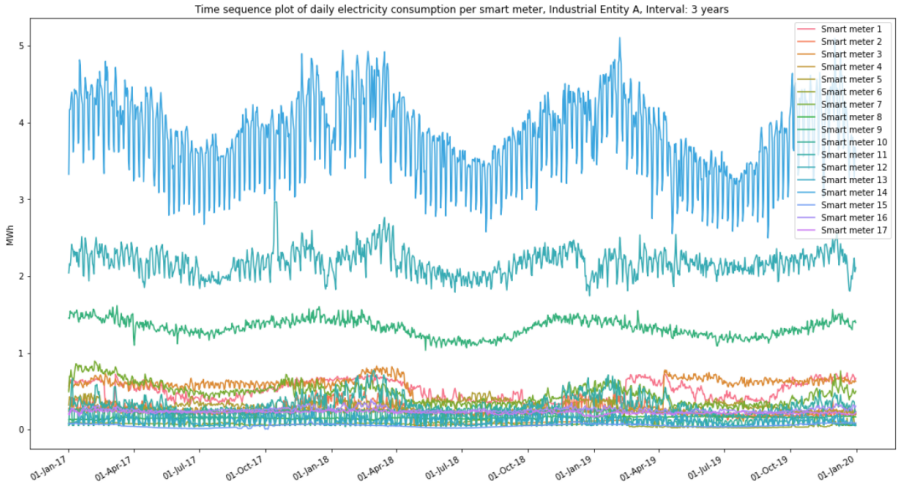
Figure 4.3.3: Distribution of daily consumption of unique smart-meters for Industrial Entity A, B, C; each color represents the frequency of daily consumption data-points of a smart-meter; smart-meters show different working regimes as multiple peaks and lows are present; Few smart-meters operate with high energy consumption while the rest operate under 1 MWh on a daily base.

4.3.2 Time Sequence Plots of Energy Consumption

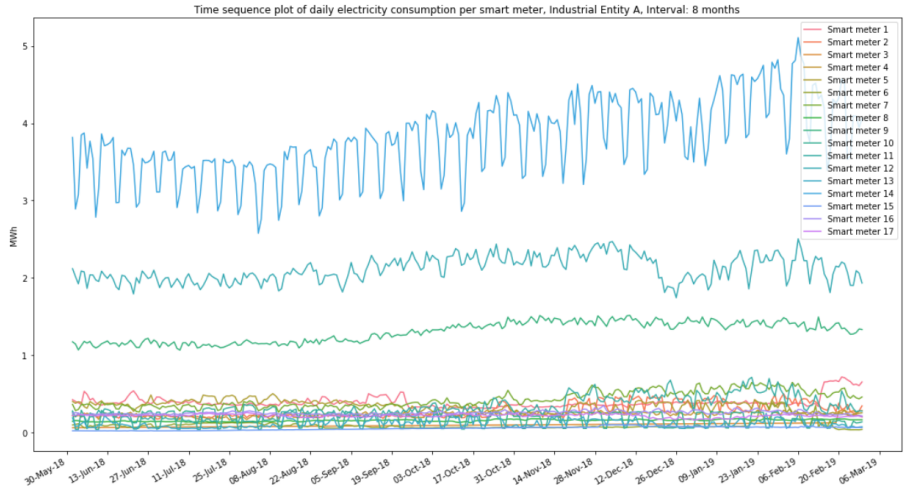
From the time sequence diagram (Figure 4.3.4 and Figure 4.3.5), we can identify at least two categories of time-series data. The first category includes stationary time-series with irregular variation (i.e. Entity A: Smart meters 3, 4, 5, 9; Entity B: Smart meter 3; Entity C: 3) or cyclic variation (i.e. Entity A: smart meters 3, 10; Entity B: smart meters 3, 4). These time-series are recorded by smart meters operating at lower ranges (i.e. below 0.5MWh for Industrial Entity A and C and below 5MWh for Entity B). In a second category, we can see the non-stationary time-series which are characterized by pronounced seasonality and slight trend for some (i.e. Entity A: smart meters 9, 12, 14). These time-series display higher operational ranges. The seasonal pattern can be observed at a yearly level with regular troughs for August and January months (i.e. Entity B: smart meters 1, 2), but also at a weekly level (i.e. Entity C: smart meter 4). Weekly seasonality is signaling the difference between working week versus weekend consumption. However, weekly seasonality is not present for all smart meters. Slight downward trends can be observed for smart meters 1 and 2 of Industrial Entity B. Another information that can be inferred from both the distributions and the time plots is the repeating patterns over the years 2017, 2018, 2019.

4.3.3 Autocorrelation and Partial Autocorrelation Plots

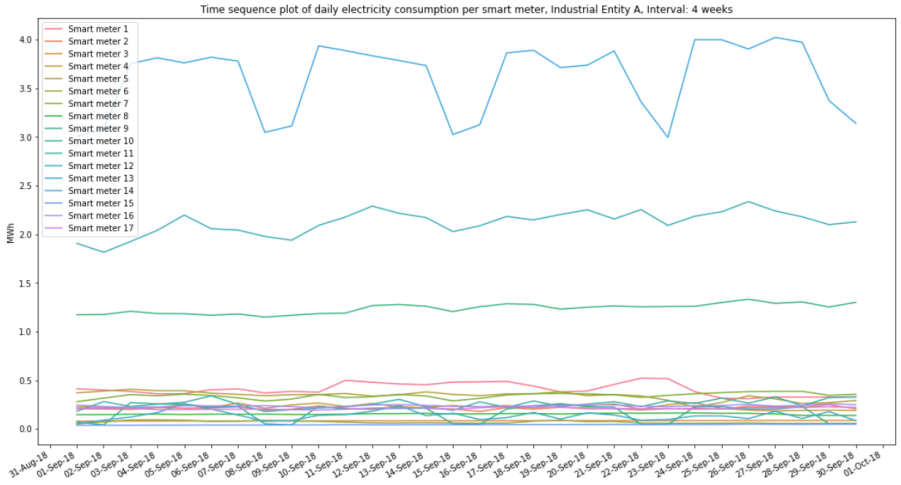
How has the tool been applied and why using it: Autocorrelation and partial autocorrelation plots are used to confirm time-series features such as seasonality and trend which have been visually identified in the smart-meters time-plots. However, the ACF and PACF were applied to the univariate time-series which record the total daily amount of energy consumption per company. This comes in opposition to constructing the plots for each smart-meter time-series individually. Also, ACF and PACF plots are used at the beginning of the modeling, to obtain the input representative lag window, or the representative time-delays needed, to improve neural network models on univariate time-series forecasting. To compute the confidence intervals for the autocorrelation scores, the defaults of python's library *statsmodels* have been used. The difference between other implementation methods is the way the variance is calculated. In python, the variance is calculated using Bartlett's formula, where $Var(r_k) = 1/N(1 + 2(r_1^2 + r_2^2 + ... + r_{k-1}^2))$, which results in the first increasing, then flattening confidence level.



(a) Industrial Entity A; Time-horizon: 3 years.



(b) Industrial Entity A; Time-horizon: 8 months.



(c) Industrial Entity A; Time-horizon: 4 weeks.

Figure 4.3.4: Time-series plots of daily consumption (MWh) recorded by each smart meter of Industrial Entity A. Time-horizon: 3 years, 8 months, 4 weeks. Description: mostly cyclic time-series, with few time-series with weekly seasonality.

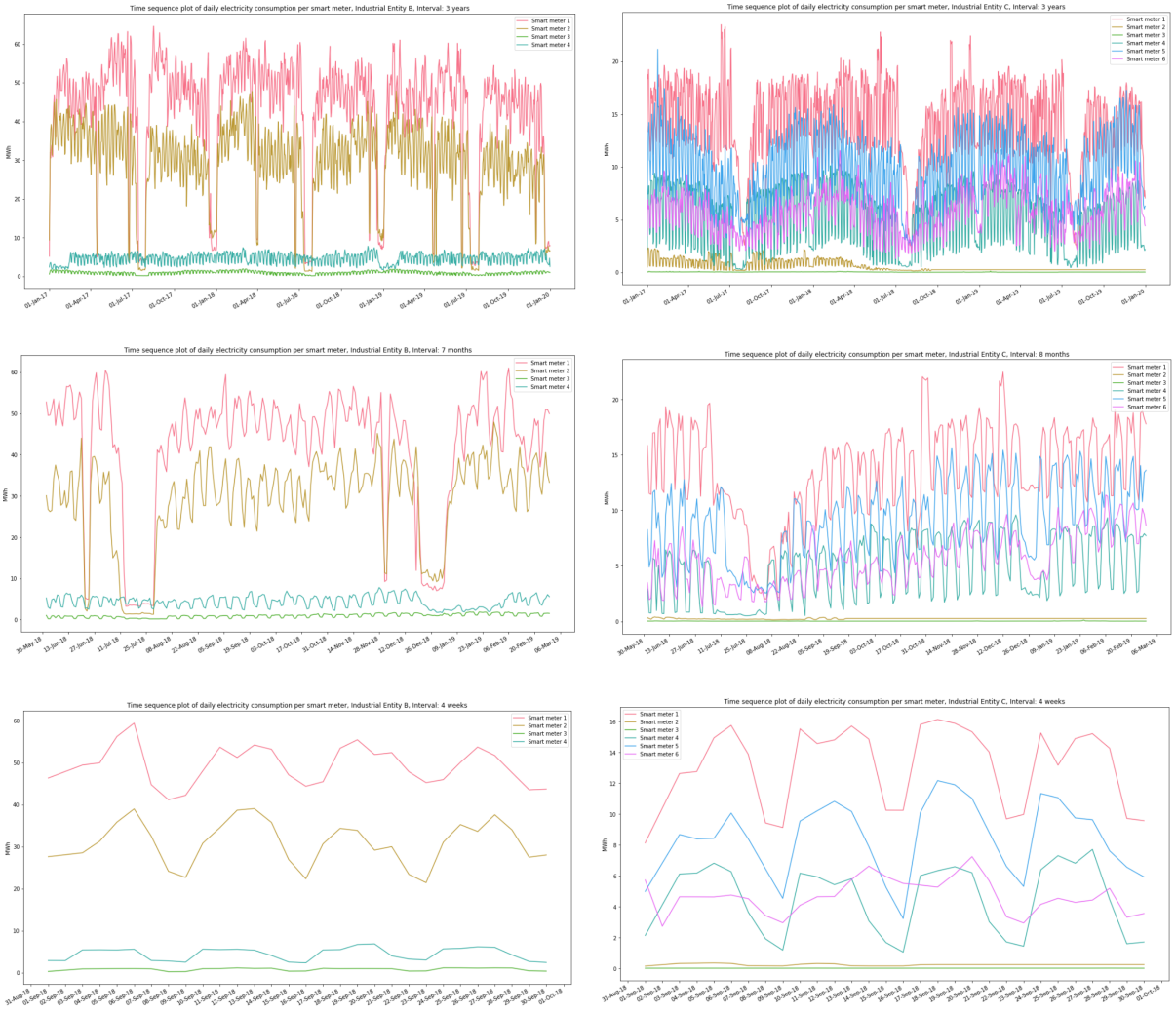


Figure 4.3.5: Time-series sequence of daily consumption (MWh) recorded by each smart meter of Industrial Entity B (left column) and Industrial Entity C (right column). Rows 1-3: 3 years, 8 months, 4 weeks sequence. Description: weekly seasonality with regular peaks and troughs and yearly seasonality with regular troughs in August and January month.

Interpretation of ACF and PACF plots Figure 4.3.8 displays the results of the autocorrelation and partial autocorrelation plot for each of the three industrial entities: A, B and C, on the cleaned dataset, before any differencing. The ACF and PACF are calculated for the entire time-series data, as opposed to only one section. Both plots show whether there are significant correlations between the same variable at different times, where the difference in time is determined by k - the lag. The ACF is a powerful tool in time-series analysis for identifying important features in the data.

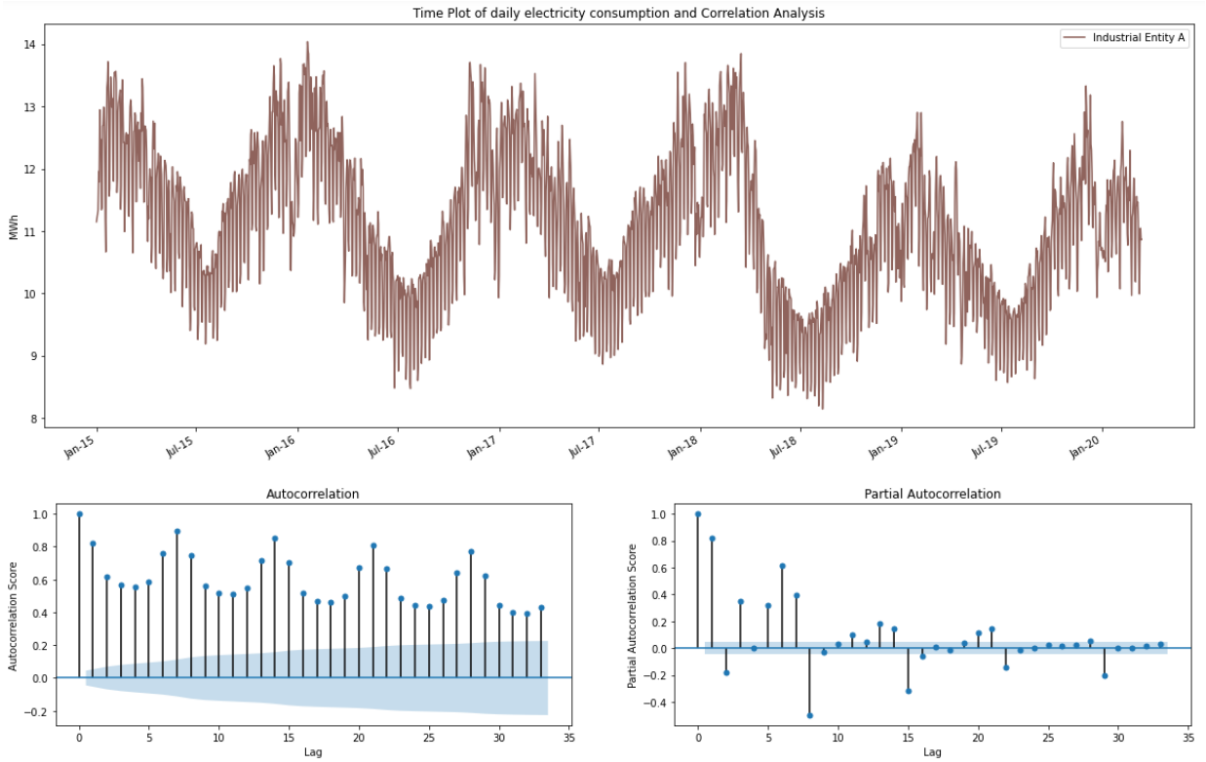


Figure 4.3.6: Time plot of total energy consumption per day for Industrial Entity A shows yearly seasonality; ACF shows weekly seasonality, but no trend; PACF plot shows significant correlations for the first 8 lags, except the 4th lag. The furthest significant correlation can be seen at 29th lag.

The ACF plot of Industrial Entity A shows a large spike at lag 1 that decreases after a few lags, displaying a mixture of exponential decay and damped sinusoid expressions. The sine waves are symmetrical, whose amplitude decreases linearly over time. The center of the sine stays constant indicating no trend. Also, the fact that there is no clear ”cut-off” after lag k indicates there is no moving average of the time-series. The sinusoidal peaks appear at 0 and every multiple of 7, which indicates weekly seasonality. The autocorrelations are significant for a large number of lags, but it could be that the autocorrelations at lags 7, 14, 21 are merely due to the propagation of the autocorrelation at lag 7. Thus, by consulting PACF, we can confirm that the significance

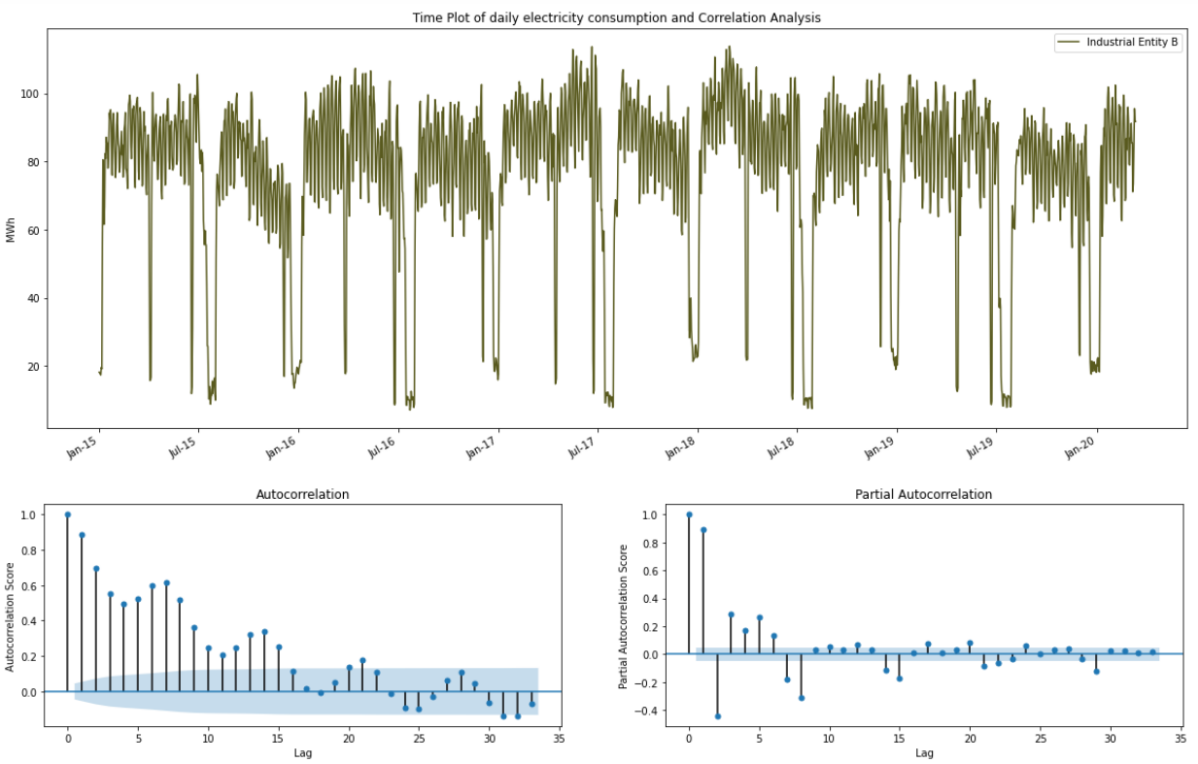


Figure 4.3.7: Time plot of total energy consumption per day for Industrial Entity B shows yearly seasonality; ACF plot shows weekly seasonality, but no trend; PACF plot shows significant correlations with lags from 1 to 8, but also 14th and 15th lag

persists at values with a shift of 7 units in time. It can be also observed that the furthest significant correlation is at the 29th lag. However, significant correlations are present in the first 8 lags, where the highest correlation can be observed at lag 2, 6, 8.

The ACF plot of Industrial Entity B shows a large spike at lag 1 followed by a decreasing wave that alternates between positive and negative correlations. The weekly seasonality is confirmed by the sinusoidal function with peaks at 7 units in time. The ACF shows that all first 15 lags are significant. The PACF confirms the significance of lags 1 to 8 and 14 to 15. However, lag 1, 3, 4, 5, 6 are positively correlated while lag 2, 7, 8 are negatively correlated.

The weekly seasonality is also present in the time-series from Industrial Entity C. As the sinusoidal patterns are centered and there is no clear ”cut-off” in the ACF, we can confirm there is no moving average term. The PACF shows a significant lag for the first 10 lags, with a higher correlation for 1st, 6th and 8th lag. The correlation score stays above the significance level for the next multiple of 7/8.

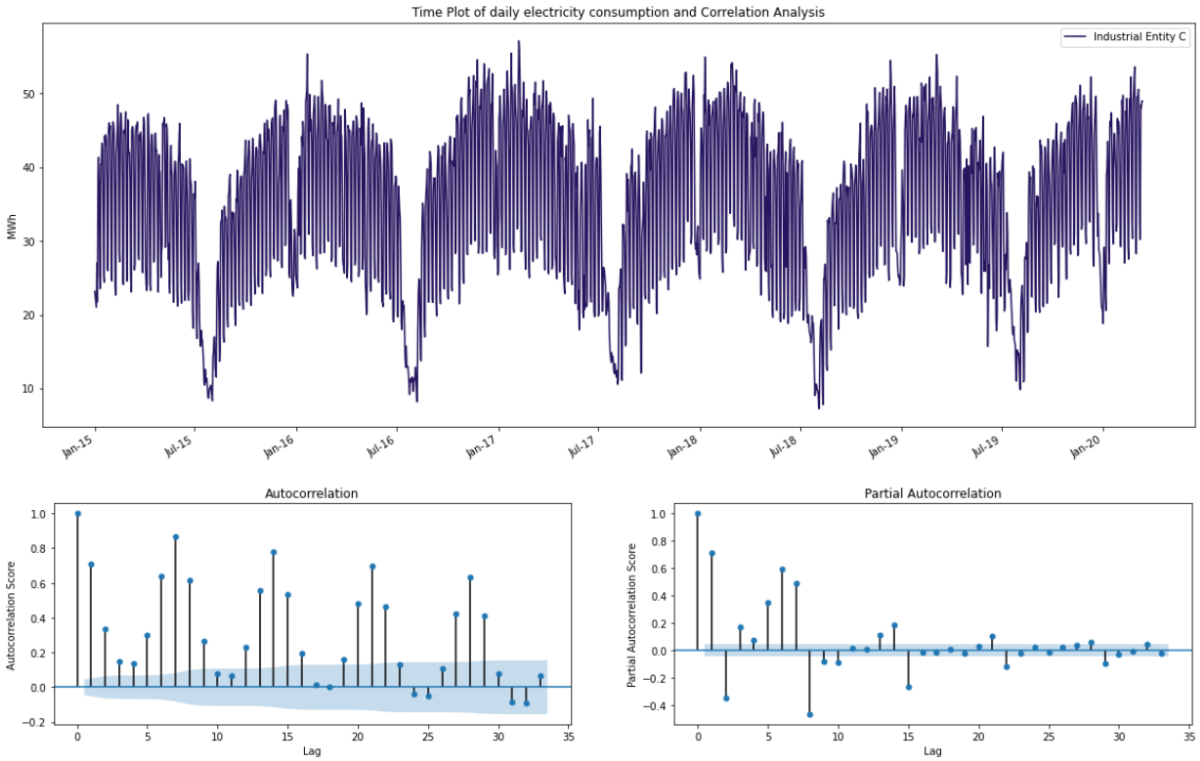


Figure 4.3.8: Time plot of total energy consumption per day for Industrial Entity C shows yearly seasonality; ACF plot shows weekly seasonality, but no trend; PACF plot shows significant correlations for the first 10 lags, but also at 22 and 28th lag.

Summary: Overall, all the three studied cases show significant correlations for the fist 8 lags, and thus have explanatory power over the time-series. However, some lags have more explanatory value than others and differ from one industrial entity to another. Thus, it worth paying attention which lags to select per industrial entity. A customized selection for each entity will ensure a better selection of features. However, this method will trade of the model generalization power. Since scalability is one of the requirements, the yearly seasonality (the lag of 1 year, 2 years, 3 years) was used as a predictor for each one of the industrial entities.

4.4 Forecasting Methods on the Industrial Dataset

4.4.1 ARIMA Variants

Prior to assessing modern forecasting techniques, some of the classical methods have been evaluated. Due to their low error performance and week power in persisting the characteristics of the time-series for the out of sample predictions, these models have not been further explored nor included in the results section.

However, for future reference, it worth mentioning their forecasting errors and the limitations of these methods. Due to low performance of the ARIMA models, only the results for one industrial entity are reported. The models mentioned below have been fitted to the univariate time-series representing the total daily electricity consumption of Industrial Entity A. To configure the parameters of a $SARIMA(p,d,q)(P,D,Q)s$ model *auto.sarima* command was applied initially, followed by empirical testings. Also, prior to model fitting, the univariate series was tested and its stationarity confirmed using Dickey–Fuller test. As mentioned in Section 2.3.2, The general SARIMA model is noted as $SARIMA(p,d,q)(P,D,Q)s$, where P is the number of seasonal autoregressive (SAR) terms, D is the order of seasonal differencing and Q is the number of seasonal moving average (SMA) terms, respectively. In the seasonal part of the model, these three parameters operate across multiples of lag s. Importantly, the s parameter influences the P, D, and Q parameters. The trend elements (p,d,q) are the same as in ARIMA models.

Three model configurations which achieved the lowest errors during the empirical study are mentioned below, together with the forecasting error for out of sample prediction.

- $SARIMA(1,0,2)x(0,1,1)_7$: the number of periods in each season has been set to 7, as daily seasonal patterns have been observed earlier in the EDA; the model daily average RMSE for the out of sample test (365 days ahead) recorded a value of 0.1977
- $SARIMA(2,0,3)x(1,1,0)_{12}$: a secondary model with 12 steps for a single seasonal period has been fitted; daily average RMSE for the out of sample test displayed an error of 1.0968
- $ARIMA(8,0,3)$: the order of AR was set to 8, and a MA to 3; the results showed a RMSE value of 0.2165

The input data has been previously tested for stationary using the Dickey-Fuller test. The statistic of the test (-2.59) was lower than the critical value (at 10%, 2.56) which indicates stationary. One of the limitations of the model is the inability to follow double seasonality and maintaining the time-series characteristics for a large out-of-sample forecast. Another drawback is the fact that the forecast is greatly affected by the starting time of the out of sample dataset. To achieve better results, one needs to align the out of sample dataset with the start of a season. Another limitation imposing computational

difficulties was to set a yearly seasonality of 365 data points. The linearity of the experimental dataset was tested by fitting a linear line and computing the resulted r-value. The r-value was very close to 0 (-0.25) which indicates almost no linearity. This characteristic can be also noticed visually in the time-series plots. The design of ARIMA models and its variations is intended for linear data which can explain the low performance given the non-linearity of our dataset. While the long-term forecasting application is not excluded in the literature from the possible utilizations of ARIMA models, the most important application is the short-term forecast.

4.4.2 Multiple Regression Experiment

There are two main types of multiple regression analyses which are the standard multiple regression (also known as the entry method) and the selection method. There are four selection procedures used to yield the most appropriate regression equation: forward selection, backward elimination, stepwise selection, and block-wise selection. The selection methods aim to reduce the set of predictor variables to those that are the most significant in an automatic or manual way. The most commonly used multiple regression analysis is the standard multiple regression where all independent variables are entered into the equation at the same time. This research uses the entry standard method as a baseline model for forecasting. The predictors are selected following an exploratory data analysis.

The chosen implementation of linear regression makes it a deterministic model. The linear regression analysis is not performed using the artificial neural networks method, thus the fitted linear regression model has no hidden layers.

MLR model was applied to forecast consumption for 1 year ahead, with a monthly resolution. Thus the regression model must capture yearly and monthly seasonality. The experimental dataset followed the data cleaning process (4.2). Once the daily forecasts have been computed, the forecasts were aggregated at a monthly level.

Feature extraction: The features have been selected following the EDA. Despite the correlations found at various lags (see Figures 4.3.6, 4.3.7, 4.3.8), the choice of model's predictors was motivated by a general approach rather than customization of the model per entity, due to the large corpus of industrial entities in the real-life dataset.

Three explanatory variables were identified in the exploratory data analysis: lag of 1, 2 and 3 years and the average of the lags (Figure 4.3.2). The time plots displayed in Figure 4.3.6, Figure 4.3.6 and Figure 4.3.6 show that the seasonality persists over the there years with no major change in mean. As a result, the three lags have been considered to represent the yearly seasonality. The lags were produced by shifting the dataset by 365 time points in the future. To repopulate the missing values due to shifting, linear interpolation was applied.

Model Formulation: The multiple regression model used can be formally written as:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_{i3} x_{i3} + \beta_{i4} x_{i4} + e \tag{4.1}$$

where, y_i = daily energy consumption at time i, where i=1, ..., N
β_0 = the intercept
β_i = the slope of the regression surface (The β_i
represents the regression coefficient associated with each x_i)
x_{i1} = energy consumption with 1 year lag
x_{i2} = energy consumption with 2 years lag
x_{i3} = energy consumption with 3 years lag
x_{i4} = mean of the energy consumption in the past 3 year
e_i = an error term, normally distributed about a mean of 0

Assumption for the errors are that they are independently identical distributed (i.i.d.) with distribution $\varepsilon \sim N\,(0, \sigma^2)$.

Model coefficients: To find the values for the coefficients that minimize the cost function, a closed-form solution has been used. Normal Equation with a least square cost function has been used for computing coefficients for the MLR model.

Experimental Setup: The dataset was apportioned into training and test sets, with an approximated 80-20 split. The linear regression model was fitted on 4 years of electricity consumption daily data-points, while the testing was done in the last year of the time sequence. The model has been constructed using 4 features: 1, 2, 3 year lags and the average of the lags. Features have been transformed using $MinMaxScaler()$. To evaluate the model, the daily average RMSE was computed.

Even though the forecast aims to produce monthly predictions, using monthly error as an evaluation metric would reduce the number of errors to 12 data-points, reducing the error accuracy. If the model is unbiased, the mean of the residuals will be zero or close to zero, and therefore the sum of the residuals will be close to zero. Thus, adding the daily forecasts into monthly forecasts will not propagate the error to the monthly total. Also, the model fit is evaluated using R^2.

Time complexity: To fit the linear regression model, sci-learn uses the least squares solution which is computed using the singular value decomposition of X input matrix. This method has a cost of $O\left(n_{\text{samples}}\, n_{\text{features}}^2\right)$. If we double the number of features, we multiply the computation time by 4.

Experimental Pipeline: The MLR has been implemented according to the following pipeline.

Algorithm 1: Linear regression model pipeline with 1 year prediction

Input: Time-series $\{x_{c_1}[t]\, x_{c_2}[t]\, ...x_{c_i}[t]\}$ where $c \in$ {Industrial Entity A, Industrial Entity B, Industrial Entity C}, number of smart meters N, i $\in$ {1, ..., N} and t $\in$ [2015-01-01, 2020-03-08]

Output: Predictions for 1 year ahead $\{x_c[t - pi : t]\},$ for $pi = 365$

Data: Raw dataset $\{x_{c_i}[t]\}$

1 $input \leftarrow \{x_{c_i}[t]\}$
2 Preprocess, clean and aggregate all time-series in the $input$
3 Create two dataframes: $trainSet,\ testSet = SPLIT(x_c[t])$
4 Generate and populate with features both dataframes: $trainSet$ and $testSet$
5 Interpolate empty spaces due to shifting on $trinSet$
6 Transform $trainSet$ using $MinMaxScaler()$
7 Fit $LinearRegression()$ model on $trainSet$
8 Transform $testSet$ using $MinMaxScaler()$
9 Run prediction on $testSet$
10 Return predictions to original values
11 Compute RMSE between predictions and target

4.4.3 Recurrent Neural Networks Experiment

In regression models, feature engineering is one of the pre-processing steps required prior to fitting the model. In opposition, neural networks automate this process. Neural networks learn an internal representation of the data. This representation contains latent features which can not be observed directly. However, it is interesting to study the impact of feature engineering and selection used as an additional enhancer to the model. Once the model configuration is selected, the proposed RNN model is tested with three different input vector sets. The same model configuration is used on datasets from three different companies.

Choice of layer architecture GRU/LSTM: There are two different cell types to improve RNN regarding long-term dependencies in sequential input data: LSTM and GRU.

One ***similarity*** consists of including the concept of gates in cell architecture. The gates decide which information is removed or added to the network. The gates learn what information is relevant to keep or forget during training. This is achieved through a sigmoid function. The sigmoid squeezes the passing values between 0 and 1. This is helpful to forget or update the values. Because any number multiplied by 0 will make it forgotten [41].

One of their ***differences*** consists in the number of gates. Every gate looks at the input x_t and h_{t-1} and outputs a number between 0 and 1. GRU has 2 gates, while LSTM has an additional one called the output gate. As a result, they differ in the information each of them passes over to the network. LSTM outputs the new cell state $C^{(t)}$ and the hidden state of the cell $h^{(t)}$ (see Figure 2.3.4). In oposition, GRU passes information further using only one parameter, the so called hidden state h_t. The hidden state of GRU is a merged version of the cell state and hidden state found in LSTM. Given they have 2 gates instead of three, one can infer that less information is passed over by the GRU than LSTM cells.

Both GRU and LSTM saw good performance in long-term dependencies task. In 2015, a study by Greff, et al.[17] over a set of popular variants concludes that they are all similar. Some studies show that GRU outperform LSTMs [18] in certain scenarios. The ***choice*** of using GRU for the initial phase of the experiment was motivated by its simpler architecture, thus less parameters to learn and higher potential of

lower training times. Also, it came natural to compare the newer GRU architecture (2014) with its predecessor LSTMs (1997), which have been well recognized in the literature.

Choice of number of units in a LSTM/GRU layer: The number of hidden units is a direct representation of the learning capacity of a neural network. It reflects the number of learned parameters. Using more units could lead to perfectly memorize the complete training set and increase the risk of over-fitting.

The number of units selected in a LSTM/GRU layer is not equal to the vector, nor the timesteps. The num_units (parameter in Keras) in an LSTM cell refers to the dimension of the cell-state vector C_t which is the same as the cell output vector h_t (Figure 2.3.4). To my understanding, this means that the num_units decides how much of the past information should be carried over the network. For example, by setting $num\_units = 20$, it means that only the previous 20 timesteps have an influence over the output. The number of LSTM recurrent cells is automatically selected and should not be cofused with the number of units. The number of cells is equal to the number of fixed time steps. For example, if we take a sequence with a total of 1168 timesteps, we can divide them in batches of size 3 and have a sequence of an approximate length of 389 timesteps per batch. The 389 is the number of LSTM cells in the layer because each cell ingests a new timestep.

The number of 512 units was chosen arbitrary. The large number of units was motivated by providing the model with a considerable learning capacity.The number of 20 units showed good results in a short-term prediction task [29] and was considered as a candidate for a long-term task. Also, the aim was to evaluate the effect of a reduction in parameters in model's performance.

Experimental Setup: A sequence-to-sequence design pattern has been adopted (see Section 2.3.4). This particular architecture fits the use case allowing for a sequence of inputs and a sequence of outputs. The dataset was split into a training set (4 years of daily energy consumption) and a testing set (1 year), with an approximated split of 80-20. The first set of experiments compares the RNN model to the standard Multiple Regression. To make a fair comparison between the RNN proposed model with the baseline model, the first experiment is designed to train a RNN model with the same input vectors as the Multiple Regression Model. The next step is

to compare the manual selection of features with RNNs capability to automatically extract feature representations from the original time-series. The third experiment comprises introducing engineered featured to the training set, in addition to the original time-series, and test whether the performance is improved. The experiments and their different inputs have been summarised in Table 4.4.1.

Experiment	*Input vector*
RNN-v1	lag 1 year, lag 2 years, lag 3 years, mean of energy consumption
RNN-v2	all time-series of a Industrial Entity, total sum of electricity consumption per day
RNN-v3	all time-series of a Industrial Entity, total sum of electricity consumption per day, day of the week, day of the year
RNN-v4	total electricity consumption per day

Table 4.4.1: Description of the RNN experiments based on their input vector

Implementation details: The experiments have been conducted using the following RNN model configuration:

Parameter	*Value*
Optimizer	RMSprop
Learning rate	1e-3
Loss function	MSE
Layers	GRU, dense
Activation function Dense Layer	sigmoid
Output units GRU	E1=20, E2=512
Output units Dense layer	1
Callbacks variables (EarlyStopping, ReduceLROnPlateau)	(patience = 5, patience = 0)

Table 4.4.2: Default parameters used for training RNNs

For implementation, Keras with Tensorflow as backend were used. The model was built using the Sequential API, which allows creating models layer by layer in a step by step fashion.

The first layer of the model is a Gated Recurrent Unit, while the second layer is a fully-connected layer. The number of GRU units have been chosen empirically. Two experiments followed. First, a GRU with 512 outputs for each time-step in the sequence and secondly a GRU containing 20 output units. The weights of the GRU have been initialized with the *glorot_uniform* kernel initializer.

The goal is to predict 1 output signal (daily energy consumption), thus a fully-connected layer was used to map the 512/20 values down to 1 value. As the output-signals in the

dataset have been previously limited to be between 0 and 1 using a scaler-object, the activation function for the dense layer could be set to $sigmoid$. Thus, the output of the neural network has been limited to be between 0 and 1. A possible limitation with using the $sigmoid$ activation function is that the output values can not be outside of the range established by the training set.

The $RMSprop$ optimizer is used to improve the training speed by updating the weights and learning rate in order to reduce the losses. RMSprop was selected instead of $AdaGrad$ as the later often stops too early when training neural networks. But also because using AdaGrad, the learning rate gets scaled-down that the algorithm ends up stopping entirely before reaching the global optimum. The initial learning rate for the RMSprop has been empirically set to *1e-3*.

The MSE is one of the common loss functions used in regression predictive modeling type of problems which involve predicting a real-valued quantity.

The choice of parameters such as batch size and training sequence size has been motivated empirically based on the results in Table 4.4.3. It was decided to use one time training instead of cross-validation and division by sequences as it resulted in the lowest error.

Training sequence size	*Batch size*	*RMSE on test set*
42	32	0.7864
365	4	0.7350
365	3	0.6806
1167	1	0.6597

Table 4.4.3: Evaluation of batch and training sequence parameters; for Medium Company A.

Evaluation metric: There are various ways to measure the error of a point forecasting method (see Section 2.3.1). RMSE has been chosen as the main evaluation criteria as it gives a relatively high weight to large errors as compared with MAE. Given the currently studied use case, the large errors are particularly undesirable. However, the RMSE is a good measure of accuracy, but only to compare prediction errors of different models or model configurations for a particular dataset, as it is scale-dependent. To allow comparison between the prediction models across datasets (eg Industrial entity A, B and C), a normalized version of the RMSE was applied. While the RMSE values have been computed on the original dataset, the normalized version

of RMSE has been computed on a transformed dataset using the min-max scaler. Thus, the NRMSE (Normalized RMSE) takes values between 0 to 1, while the RMSE values are in MWh and are dependent on the scale of each dataset. To report the overall 1 year ahead prediction of a model, the daily average RMSE was preferred instead of monthly average RMSE.

Experimental Pipeline

Algorithm 2: RNN pipeline for 1 year prediction

Input: time-series $\{x_{c_1}[t]\, x_{c_2}[t]\, ...x_{c_i}[t]\}$ where $c \in$ {Industrial Entity A, Industrial Entity B, Industrial Entity C}, number of smart meters N, i $\in$ {1, ..., N} and t $\in$ [2015-01-01, 2020-03-08], f activation function

Output: Predictions for 1 year ahead $\{x_c[t - pi : t]\}$, for $pi = 365$

Data: Raw dataset $\{x_{c_i}[t]\}$

$input \leftarrow \{x_{c_i}[t]\}$
Preprocess and clean $input$
Depending on the experiment, generate lagged and/or deterministic features
Split dataset into $trainSet, testSet$
Transform $trainSet, testSet$ with $MinMaxScaler()$
$model \leftarrow sequencial()$
Add layers: GRU, Dense
Declare loss MSE worm-up and optimizer
Declare callbacks: EarlyStopping, ReduceLROnPlateau
for $epoch = 1..E$ **do**
    for $batch = 1..B$ **do**
        $model.fit()$
Evaluate model on $testSet$
Return predictions to original values
Compute RMSE, average of daily error

Chapter 5

Results

This chapter describes collected results from analyzing an industrial electricity consumption dataset and experimenting with traditional and machine learning methods for forecasting. Finally, the chapter describes a solution for long-term forecasting using Recurrent Neural Networks (Section 4.4.3).

5.1 The Forecasting Task

The forecasting task of interest in this research has been the long-term prediction of point estimates, in this case, 1 year ahead. A daily forecast resolution was selected for research purposes. However, the real case application required a monthly prediction. The forecasting task has been formulated as a regression problem where the output variable of interest is a real and continuous value, namely the electricity consumption of industrial entities. Thus, the models chosen had to allow for a sequence-to-sequence architecture design.

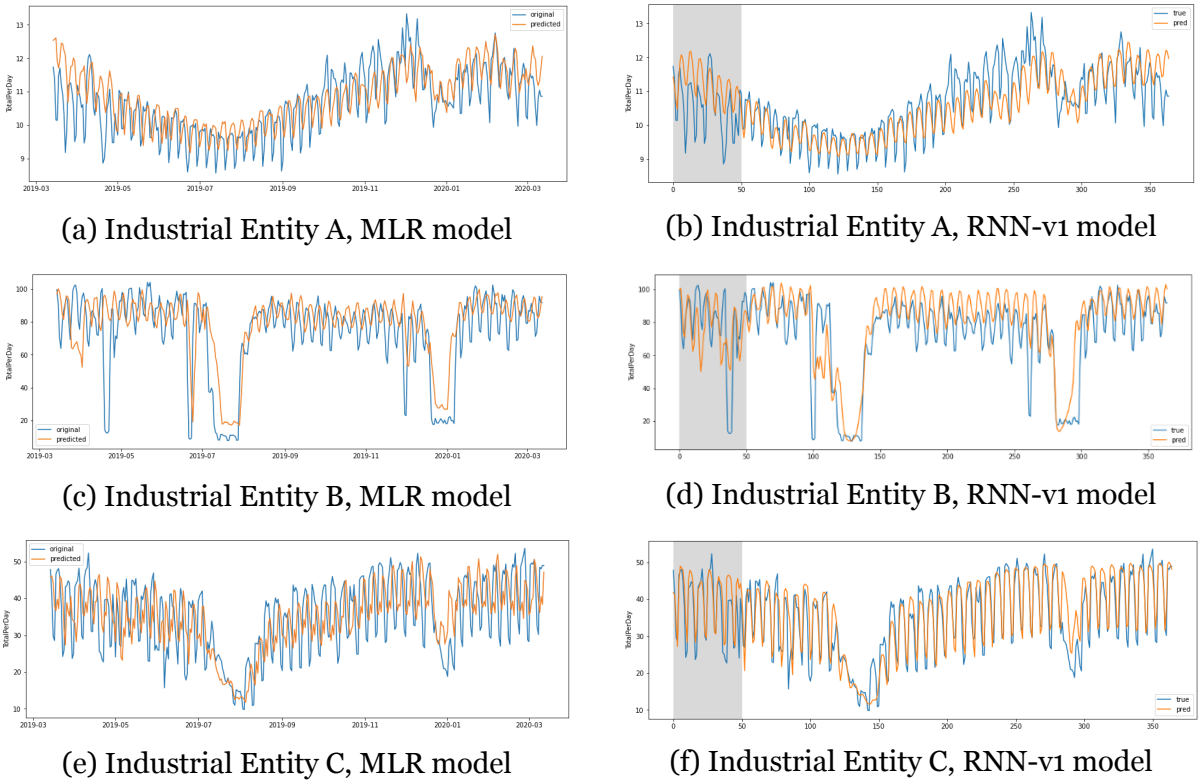
5.2 Results of Long-Term Forecasting on Industrial Datasets

This section presents the results obtained by using variations of RNN specifically designed to handle time-series data implemented in TensorFlow. The experiments have been applied to thee different industrial datasets. The RNN experiments vary in the model configurations and the input selection of features. The RNN model's results

are compared to a Multiple Regression model. One of the drivers of this research was to propose a solution to a real industry scenario. The research was initiated by a pre-study. The purpose of the pre-study was to better understand industrial electricity consumption datasets. The new insights motivated the choice of training individual models per company. The pre-study was followed by an exploratory data analysis which led to feature selection for the models' customization.

5.2.1 Long-term Forecasting with Multiple Regression

The performance of the MLR model is recorded in Table 5.2.2. The prediction RMSE metric is accompanied by time prediction plots for visual inspection presented in the figure below. The RNN model improved the prediction accuracy by 5% to 10% depending on the dataset, when compared to the MLR model.



(a) Industrial Entity A, MLR model

(b) Industrial Entity A, RNN-v1 model

(c) Industrial Entity B, MLR model

(d) Industrial Entity B, RNN-v1 model

(e) Industrial Entity C, MLR model

(f) Industrial Entity C, RNN-v1 model

Figure 5.2.1: One year ahead prediction time plots. Left column: MLR. Right column: RNN-v1.

When we analyze the predictions for Entity A, both models are missing the picks in September and December months. We can also notice that between May and September, the MLR model overestimates the maximum energy consumption, while the RNN model anticipated the picks right. However, none of them can predict the

troughs for dataset A or B.

On the other hand, the RNN model captures well the troughs and peaks of Entity C, while the MLR does not. The RNN's good model fit is reflected in the RMSE value of 0.10 compared to 0.20 for MLR.

5.2.2 Long-term Forecasting with RNN

The RNN experiments vary in terms of (1) input features and (2) model configuration. First, the importance of different features selection methods have been tested by feeding the model with three different input vectors. The specific parameters used for the RNN model are listed in the following Table 5.2.1.

	Industrial Entity A	Industrial Entity B	Industrial Entity C
Number of time-series	17	4	6
Input signal shape	(1533, 18)	(1533, 5)	(1533, 7)
Time granularity	daily	daily	daily
Batch size	1	1	1
Learning rate	1e-3	1e-3	1e-3
Number of layers	2	2	2
Memory unit type	GRU/LSTM	GRU/LSTM	GRU/LSTM
Number of units	20/512	20/512	20/512
Epochs	20	20	20
Train-Test split	80-20%	80-20%	80-20%

Table 5.2.1: Model configuration for RNN-v2 with for all three datasets.

The proposed RNN model performance has been compared against a baseline model. A non-learning model, namely a Multiple Regression has been chosen and designed as a baseline model for this research. To make a fair comparison, the inputs of the Multiple Regression model and RNN-v1 are kept the same. The later versions (RNN-v1, RNN-v2) aim to improve the model performance by including additional structural features (see Section 4.4.3 for details).

The results presented in Table 5.2.2 show that RNN outperforms the baseline model MLR on all 3 datasets, regardless of the input vector. The results of the proposed model configuration show less error than other models used for similar forecasting tasks in the available literature as shown in Table 5.2.3).

Results of RNN experiments with different inputs: Adding deterministic effects to the input vector (namely the day of the week and the day of the year) for the

	Description	Method	RMSE
Medium Industrial Entity A Energy consumption (MWh)	1 year forecast, daily resolution	MLR	0.1592
		RNN-v1	0.1077
		RNN-v2	0.1119
		RNN-v3	**0.0965**
		RNN-v4	0.1178
Large Industrial Entity B Energy consumption (MWh)	1 year forecast, daily resolution	MLR	0.1937
		RNN-v1	0.1499
		RNN-v2	0.1351
		RNN-v3	0.1318
		RNN-v4	**0.1199**
Large Industrial Entity C Energy consumption (MWh)	1 year forecast, daily resolution	MLR	0.2039
		RNN-v1	0.1003
		RNN-v2	0.1337
		RNN-v3	0.1167
		RNN-v4	**0.0991**

Table 5.2.2: Prediction performance of various models proposed in the current research. The RMSE values are normalised and the error unit is MWh. Models performance is compared across three different datasets reported based on the scaled RMSE score.

	Description	Method	RMSE
Aggregated active power (kW)	1 year forecast, 1 week resolution	ANN	0.246
		SVM	0.188
		SVN	0.457
		CRBN	0.182
		FCRBM	0.170
		RF + LR	0.145

Table 5.2.3: Performance of different forecasting procedures available in the literature. Artificial neural network (ANN), support vector machine (SVM), recurrent neural network (RNN), conditional restricted Boltzmann machine (CRBM) and factored conditional restricted Boltzmann machine (FCRBM) results are from [37]; Random forest (RF) and linear regression (LR) results are from [34].

Industrial Entity A dataset had effects in reducing the RMSE. However, for the other two datasets, Entity B and C, the smallest daily prediction error was achieved with the RNN-v4, where only the total energy consumption per day was used as an input. This indicates that RNN performs well when the input is represented by a univariate time-series to be forecasted. Feeding the RNN network with multivariate time-series coming from individual smart meters did not increase the model's accuracy for the Industrial Entity B and Entity C.

For the Industrial Entity C, the model with the best fit was RNN-v4, closely followed

by RNN-v1 (where the input vector included the lag of 1, 2 and 3 years and the mean of the lags), which shows that the manual feature selection had a comparable effect with the capability of RNN-v4 to automatically extract time-series characteristics from the original time-series.

For all three datasets, the RNN-v3 showed less error than RNN-v2 and RNN-v1. This shows that augmenting the input with deterministic effects leads to a reduction in the prediction error. However, the amplitude of the error improvement varies depending on the size and the characteristic of the dataset.

The time prediction plots of the model version that achieved the least RMSE for each of the datasets are presented in Figure 5.2.2.

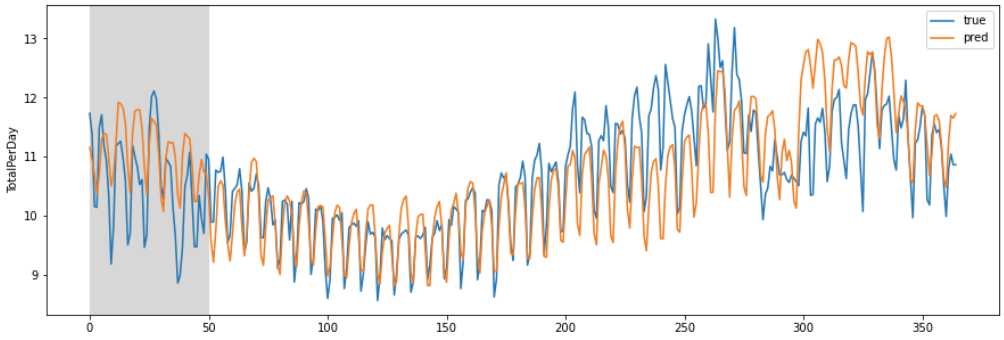
Results of RNN with GRU vs LSTM layer: Two hidden layer architectures, namely GRU and LSTM have been tested for identifying their effects on the forecasting task. Models performance has been quantified based on the three available datasets, where the input is represented by the individual time-series and the total electricity consumption per day (see Section 4.4.3 for details).

Dataset	GRU units	RMSE RNN-v2	LSTM units	RMSE RNN-v2
Industrial Entity A	20	**0.1242**	20	0.1505
	512	0.1119	512	**0.0882**
Industrial Entity B	20	**0.1377**	20	0.1728
	512	**0.1351**	512	0.1467
Industrial Entity C	20	**0.1763**	20	0.1859
	512	0.1337	512	**0.1239**

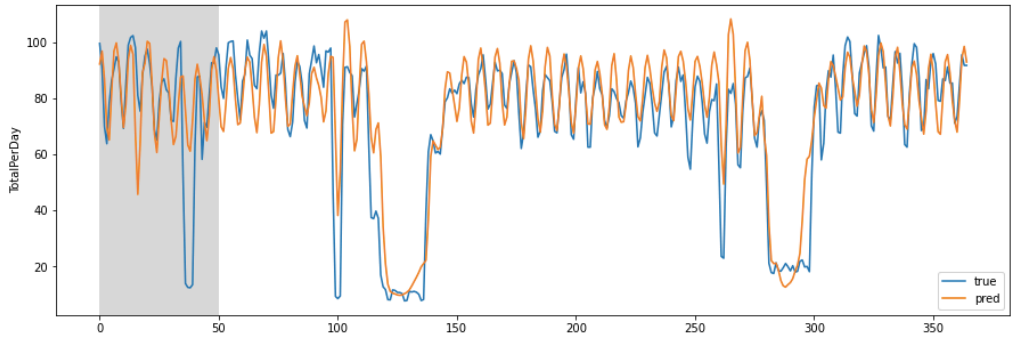
Table 5.2.4: Prediction performance of RNN with GRU layers compared to RNN with LSTM layers on three different datasets. The RMSE values are normalised and the error unit is MWh.

We can see in the Table 6.1 that the GRU-RNNs model acquired a lower prediction error than LSTM-RNNs model on 4 out of the 6 scenarios. In particular, GRU-RNN outperformed LSTM-RNN when the number of units was set to 20, for all of the 3 industrial datasets.

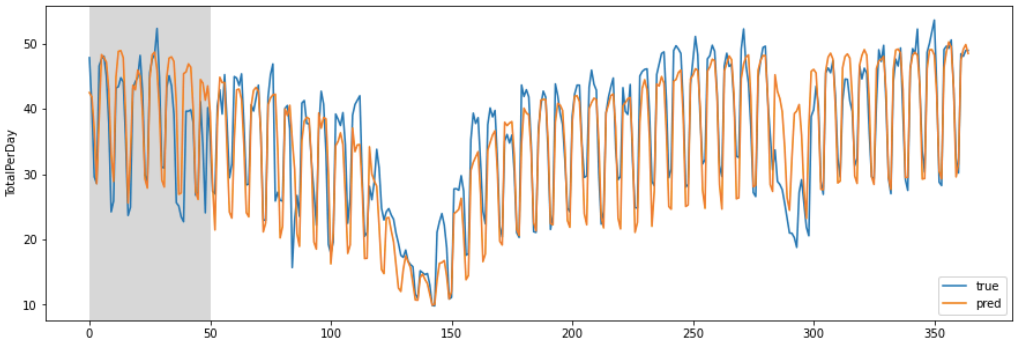
Increasing the number of GRU units by approximately 25 times did not improve significantly the prediction error for Entities A and B. The error improvement for the mentioned datasets is between 0.02-1% which does not justify the resulted increase in training time and the number of parameters. However, one can see an improvement



(a) Industrial Entity A, model: RNN-v3 (RMSE = 0.0965)



(b) Industrial Entity B, model: RNN-v4 (RMSE = 0.1199)



(c) Industrial Entity C, model: RNN-v4 (RMSE = 0.0991)

Figure 5.2.2: Time plots predictions one year ahead of model versions that achieve the least RMSE.

for Entity C of approximately 4% (from RMSE = 0.1763 to RMSE = 0.1337). The experiments also showed that for LSTM architecture, the reduction in forecasting error decreased in a larger amount with the increase in the number of LSTM units.

There is a decrease in error prediction associated with the increase in the number of units of GRU or LSTM which differs from dataset to dataset. The maximum RMSE reduction by increasing the number of GRU units by 25 fold for was 4% (Entity C), while LSTM saw a maximum error reduction of 6.2% (Entity A).

The following paragraph explores the experimental results by looking into the specifics of LSTM versus GRU and by looking at the characteristics of the datasets. Research studies found GRU to be comparable to LSTM. In theory, LSTMs remember longer sequences than GRUs and outperform them in tasks requiring modeling long-distance relations. As mentioned in Section 2.3.4, LSTMs have a long term memory kept along the network using the cell state and a short-term view of the situation kept by the hidden layer. In opposition, GRU has no output gate and the full state vector is outputed at every time step. Given all of these mentions and the insights from the exploratory data analysis, the following hypothesis have been formulated:

- Very long-distance dependencies are not required for datasets with a certain profile. For datasets with pronounced yearly pattern which maintain a similar amplitude over the years, GRU's simpler architecture can contain and transfer the important information through its single output, the hidden state.
- We can infer that the existence of the output gate (and the transfer of long term memory through the cell state) limits the learning capability of LSTMs, when not enough units are selected. The hypothesis was formulated as LSTM with 512 units achieved lower prediction error for Industrial Entities A and C.
- GRUs seem to acquired a lower prediction error for datasets with lower prevalent content, regardless of choice of hidden units. The cyclic patterns of Industrial entity B span over a shorter time period than for Industrial Entity A and C (refer to Figure 4.3.6, 4.3.7, 4.3.8). Also, Industrial Entity B is not characterised by double seasonality as for the other 2 datasets. Another parameter that could have influence the model's results (of Entity B) is the number of inputs of RNN-v2. Industrial Entity B has fewer smart-meters and thus less predictors influencing the response variable (see Table 5.2.1).

It is important to mention that these are just hypothesis. Further research is necessary to properly explain the impact of specific dataset characteristics on GRU performance.

Comparison: Multiple Linear Regression versus Recurrent Neural Networks The recurrent neural network methods proved to be superior to the statistical methods alternative, regardless of the selection of training input. Their strength lays in the automatic feature extraction from complex data with a combination

of linear and non-linear transformation in the neural nets. Statistical models do not have memory so these can not project past patterns into the future. Also, the linear models perform best on purely linear data. Between the two implemented classical statistical models, the Multiple Linear Regression showed a better fit and lower forecasting error than SARIMA. The application of statistical methods required cumbersome pre-processing and inefficient feature engineering. Of course, one can argue that the forecasting results using Multiple Linear Regression have been limited by the user knowledge or domain expertise and that the results could have been improved if better explanatory features were discovered and implemented.

However, the experimental results on three different datasets showed a RMSE reduction of a minimum of 0.05 and a maximum of 0.1 when RNN model was used as opposed to Multiple Linear Regression, for the same set of inputs (Table 5.2.2). Changing the inputs to RNN models resulted in even lower RMSE values. Additionally, looking from a time and system perspective, the RNNs methods reduce the complexity of the forecasting pipeline and time required to find a reasonable model.

5.2.3 Hardware and Tools

Azure Machine Learning (Azure ML) [36] is the computational platform used to run the experiments. It provides a pre-configured Python virtual environment and computational resources. The experiments have been implemented using Python 3. The main software used for Data Analysis is pandas [35]. The main software libraries used to implement the models are TensorFlow [33] for the RNNs, scikit-learn [40] library for MRL models, while for ARIMA variants statsmodels [44] library was used. As well, the well-known numpy, math and matplotlib Python libraries have been used.

Development Environment

Platform: Azure ML

Name Virtual Machine: Standard D12

vCores: 4

Memory: 28 GB

Local SSD: 200 GB

Main software libraries

Tensor Flow: version 2.1.0

Pandas: version 0.23.4

Sklearn: version 0.20.3

Chapter 6

Conclusions

6.1 Discussion

This section outlines the conclusions drawn from the current research It aims to answer the research question presented in Section 1.3.

The main research question has been divided into sub-questions as follows:

Amongst the existing approaches to forecasting tasks available in the literature, which methods are best suited for predicting long-term electricity consumption of industrial entities? Which RNN architectures are suitable for the current case study? And how do the results of traditional methods compare to learning methods for electricity consumption datasets?

A review and comparison of strategies for multi-step ahead time-series forecasting indicate that RNNs are the most suitable prediction method for long-term forecasting for this particular application domain. The methods overcome the challenges of nonlinear variation of energy consumption. As known from previous research, the RNN models lack interpretability. In this research, the first task was to design an interpretable baseline model (MRL) by running a thorough exploratory analysis and later to design a learning model (RNN) and to compare them given the same inputs. The model performance of RNN when compare to MLR showed a RMSE improvement between 5% and 10%.

Further on, different approaches to improve RNNs performance and their effects have been compared. RNNs showed improved prediction performance in long-term

forecasting tasks over the classical statistical methods such as SARIMA and Multiple Linear Regression tested in the current research but also compared to other approaches available in the literature. This was true for all of the training inputs (RNN-v1, RNN-2, RNN-v3, RNN-v4) it was tested with. The strength of RNNs lays in the automatic feature extraction from complex data with a combination of linear and non-linear transformation in the neural nets. The reduction in the forecasting error between statistical and learning methods varies between datasets. The maximum forecasting error (RMSE) reduction was approximately 10% for the Industrial Entity C datasets.

One of the MLR disadvantages is that in order to increase the model's performance, one needs to customize individual features for each entity, thus compromising the scalability feature.

In addition to providing better forecast accuracy than previous methods, the proposed RNN approach has a number of key advantages compared to classical approaches and other global methods: (1) the model learns seasonal behavior and other time characteristics automatically (2) reduction in complexity of the forecasting pipeline reduce and therefore increase maintainability (3) no need to specify a particular model form or to make any a priori assumption about the statistical distribution of the data (4) the desired model is adaptively formed based on the features presented from the data.

How can EDA be used in the feature selection step to improve the performance of learning models?

It was not the goal of this thesis to find the best feature selection for modeling the time chosen time-series, but to show that following a thorough Exploratory Data Analysis, and using ACF and PACF function could be used to improve the model's performance.

For the large industrial datasets (B and C), the best prediction performance was achieved by the RNN models trained with the univariate time-series (namely RNN-v4), representing the aggregated energy consumption per day. This excludes the application of the EDA, ACF and PACF findings. Feeding the network with all the time-series available per Industrial Entity resulted in a decrease in the model performance. This effect highlights the importance of aggregation choice in forecasting

tasks.

However, for the medium Industrial Entity A dataset, the best RNN performance was achieved by including deterministic features, namely the day of the week and day of the year (RNN-v3).

It is also worth mentioning that, for all three datasets the RNN-v3 trained with deterministic features showed less error than RNN-v2 and RNN-v1, but not RNN-v4.

To conclude, EDA, ACF and PACF tools proved to be plagued by subjective choices and even though for one of the datasets the experiments showed better model performance by including findings acquired through these methods, we cannot conclude that these can always improve the model prediction.

How does the selection of GRU or LSTM units influence the prediction performance in the long-term forecasting task?

The experiments show that the GRU architecture outperforms LSTM architecture in long-term forecasting tasks in certain scenarios: (1) when the two architectures are compared against a lower number of units, such as 20 units per layer; (2) for certain datasets, regardless of the choice of number of units. As GRU units include fewer training parameters in their structure, this architecture has a secondary advantage of achieving faster training times, besides the decrease in model's error. The effect of increasing the number of GRU units by 25 fold was minimal with an error reduction varying from 0.02% to 4% depending on the dataset. The RNN built with LSTM architecture showed a higher improvement rate with an increase in the number of units. The effects of increasing the LSTM by 25 fold led to a maximum error reduction of 7% for the long-term forecasting task.

Why LSTM cells fail to outperform GRU?

One known limitation of deep learning models is the lack of interpretability, failing to provide explanations on their predictions. As one of the effects, there is no agreement in the literature on which of the LSTMs or GRUs are better. In an attempt to explain the experimental results of Table , the architectural differences and the specifics of the datasets have been addressed. It is hard to tell, at a glance, which part of the GRU is essential for its functioning.

However, one of the main difference in comparison to LSTM is the absence of the output gate (see 4.4.3 for a detailed comparison). In LSTM, output gate decides what the next hidden state should be. In addition, LSTM transfers information not only through the hidden state but also through the cell state, which is attributed to the long term memory. A hypothesis is that very long-term dependencies are not required for industrial electricity consumption datasets. These datasets are characterised by strong yearly and weekly patterns with a similar amplitude over the years.

It is interesting worthy to discuss the characteristics of Industrial Entity B dataset on which GRU outperformed LSTM, regardless of the choice of recurrent units. Two differences of the Industrial Entity B dataset have been identified: (1) shorter seasonal pattern of 6 months instead of a year (2) dataset is not characterised by double seasonality as the other two.

The following hypothesis have been formulated:

- The output gate and the long-term memory of LSTMs contained by the cell state are limiting the learning for electricity consumption datasets with enhanced seasonality.
- GRUs are able to abstract the patterns of industrial electricity consumption datasets better than LSTMs. A lower number of hidden units in the GRU layer leads to better abstraction of time series characteristics of industrial datasets. Preserving a longer sequence of information is not necessary in a dataset with high yearly seasonality.

Previous literature suggests that LSTMs should in theory remember longer sequences than GRUs and outperform them in tasks requiring modeling long-distance relations, while GRU architecture is suitable for short datasets. Thus, it reinforces the idea that GRU makes a suitable architecture for forecasting tasks of energy consumption as time-series datasets with a daily or monthly resolution are limited to thousands of time data points. This comes in opposition to natural language processing tasks, where the input is not directly limited by time.

6.2 Future Work

The proposed long-term forecasting method consists of a deep learning model with GRU units. The most promising input to the model was the single time-series representing the daily aggregated energy consumption. Using all individual smart-meter time-series as network inputs did not enhance the learning of the model. In opposition to the initial hypothesis, including individual time-series as input vectors increased the overall forecasting error for two of the three Industrial Entities. The experiments showed that the increase in the number of units did not impact the GRU model performance significantly. As future work, it will be interesting to continue experiments with GRU and LSTM for various numbers of hidden units (when the network input is represented by a single time-series: the total daily aggregated consumption from all the smart-meters). Thus, a far more conclusive result can be achieved regarding the impact of hidden units on the model's performance.

When it comes to the network input, this thesis has focused on the time-related features. Currently, for two of the industrial datasets, the best performing model was achieved by learning the time patters and features just from the univariate total electricity consumption series, without additional features. For the last datasets, the best model performance was achieved by training with deterministic features as well as the individual time-series originated from smart-meter at different locations. It comes naturally to proceed with experiments with exogenous features such as the location of smart-meters and outside temperature. The activation function used in the dense layer was $sigmoid$, which limits the output of the neural network to be between 0 and 1. This bounds the predicted electricity consumption values to the maximum and minimum values of the training dataset. Thus it will be interesting to study whether different activation functions in the dense layer such as $ReLU$, which allows for the output to take on arbitrary values, could improve the prediction performance.

The time prediction plots showed that the RNNs prediction of picks and troughs has limitations and thus further work and research could be done in this direction. Moreover, the accuracy of the models used in this thesis can potentially be improved by doing large-scale hyperparameter tuning of the models' hyperparameters such as learning rates and different strategies for weight initialization.

In addition to input selection and hyperparameter tuning strategies of improving the forecasting performance, it will be interesting to experiment with other models

architecture such as CNN. For example one study [25] proposes a CNN network for long-term forecasting of events. Furthermore, hybridization is reported to be effective in the advancement of prediction models. For example, the CNN-LSTM neural network, which combines convolutional neural network (CNN) and long short-term memory (LSTM) showed good performance for residential energy consumption and it records the smallest value of root mean square error compared to the conventional forecasting [28]. In the current work, the time-series were represented as sequential data. As future work, it will be interesting to represent time-series as graph data structures, where the location of smart meters can be encoded in the graph structure and the edges can encode the temperature difference between locations or physical distance.

Another relevant problem in the industry alongside forecasting for existing industrial entities/customers is the monthly prediction of energy consumption for new industrial customers. As opposed to existing ones, when historical time-series are available for analysis, for the new customers the amount of data is limited if not non-existent. Thus, a natural question unfolds: *How can available time-series from existing industrial entities be used to extrapolate predictions for new customers with a limited amount of data?*

As mentioned before, an industrial entity usually has tens of time-series originating from different smart-meters. One suggestion for solving this problem is to cluster different time-series into similar groups based on the available training data. As indicated by EDA, the time-series from all the three industrial entities could form 3 to 5 clusters. The next step would be to collect a short amount of data (one/two weeks) from the new customer and assign it to a cluster, extrapolating from the cluster's information to the specific time-series.

Bibliography

[1] Adhikari, Ratnadip and Agrawal, R. K. *An Introductory Study on Time Series Modeling and Forecasting*. 2013. arXiv: 1302.6613 [cs.LG].

[2] Adya, Monica and Collopy, Fred. "How effective are neural networks at forecasting and prediction? A review and evaluation". eng. In: *Journal of Forecasting* 17.5□6 (1998), pp. 481–495. ISSN: 0277-6693.

[3] Adya, Monica, Collopy, Fred, Armstrong, J.Scott, and Kennedy, Miles. "Automatic identification of time series features for rule-based forecasting". eng. In: 17.2 (2001), pp. 143–157. ISSN: 0169-2070.

[4] Armstrong, J. Scott. "Illusions in regression analysis". eng. In: *International journal of forecasting* 28.3 (2012), pp. 689–694. ISSN: 0169-2070.

[5] Bansal, Trapit, Belanger, David, and McCallum, Andrew. "Ask the GRU: Multi-Task Learning for Deep Text Recommendations". eng. In: *Ask the <h>GRU</h>: Multi-Task Learning for Deep Text Recommendations*. 2016.

[6] Bingham, N. H and Fry, John M. *Regression: Linear Models in Statistics*. eng. Springer undergraduate mathematics series. London: Springer London, Limited, 2010. ISBN: 9781848829688.

[7] Box, George E. P. *Time series analysis : forecasting and control*. eng. Rev. ed.. Holden-Day series in time series analysis. San Francisco: Holden-Day, 1976. ISBN: 0-8162-1104-3.

[8] Brockwell, Peter J and Davis, Richard A. *Introduction to Time Series and Forecasting*. eng. Springer Texts in Statistics. New York: Springer, 2002. ISBN: 0387953515.

[9] Chatfield, Christopher. *The Analysis of Time Series: Theory and Practice*. eng. 1st ed. 1975.. 1975. ISBN: 1-4899-2925-8.

[10] Cho, Kyunghyun, Merrienboer, Bart van, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. eng. In: (2014).

[11] Faloutsos, Christos, Gasthaus, Jan, Januschowski, Tim, and Wang, Yuyang. *Forecasting Big Time Series: Old and New; Tutorial for VLDB, 2018*. 2018. URL: `https://www.dropbox.com/s/tx9rja6uz06rvja/fcst-tutorial-vldb-draft.pdf?dl=0`.

[12] Flores, J. H. F, Engel, P. M, and Pinto, R. C. “Autocorrelation and partial autocorrelation functions to improve neural networks models on univariate time series forecasting”. eng. In: IEEE, 2012, pp. 1–8. ISBN: 9781467314886.

[13] Géron, Aurélien. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O’Reilly Media, 2017. ISBN: 978-1491962299.

[14] Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. The MIT Press, 2016. ISBN: 0262035618.

[15] Graves, A. and Jaitly, Navdeep. “Towards End-To-End Speech Recognition with Recurrent Neural Networks”. In: *ICML*. 2014.

[16] Graves, Alex. “Generating Sequences With Recurrent Neural Networks”. In: *CoRR* abs/1308.0850 (2013). arXiv: `1308.0850`. URL: `http://arxiv.org/abs/1308.0850`.

[17] Greff, Klaus, Srivastava, Rupesh K, Koutnik, Jan, Steunebrink, Bas R, and Schmidhuber, Jurgen. “LSTM: A Search Space Odyssey”. eng. In: *IEEE transaction on neural networks and learning systems* 28.10 (2017), pp. 2222–2232. ISSN: 2162-237X.

[18] Gruber, Nicole and Jockisch, Alfred. “Are GRU Cells More Specific and LSTM Cells More Sensitive in Motive Classification of Text?” eng. In: *Frontiers in Artificial Intelligence* 3 (2020).

[19] Han, Shuang, Qiao, Yan-hui, Yan, Jie, Liu, Yong-qian, Li, Li, and Wang, Zheng. “Mid-to-long term wind and photovoltaic power generation prediction based on copula function and long short term memory network”. eng. In: *Applied energy* 239 (2019), pp. 181–191. ISSN: 0306-2619.

[20] Hartwig, Frederick. *Exploratory data analysis*. eng. Quantitative applications in the social sciences ; no. 07-016. Newbury Park, [Calif.] ; London: SAGE, 1979. ISBN: 9781412984232.

[21] Hochreiter, Sepp and Schmidhuber, Jürgen. "Long Short-Term Memory". eng. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[22] Hochreiter, Sepp and Schmidhuber, Jürgen. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://doi.org/10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[23] Hyndman, R.J. and Athanasopoulos, G. *Forecasting: Principles and Practice*. OTexts, 2014. ISBN: 9780987507105. URL: https://books.google.dk/books?id=nmTQwAEACAAJ.

[24] Ismail, Zuhaimy and Jamaluddin, Faridatul. "A Backpropagation Method for Forecasting Electricity Load Demand". In: *Journal of Applied Sciences* 8 (Dec. 2008). DOI: 10.3923/jas.2008.2428.2434.

[25] Jain, Aishvarya Kumar, Grumber, Christian, Gelhausen, Patrick, Häring, Ivo, and Stolz, Alexander. "A Toy Model Study for Long-Term Terror Event Time Series Prediction with CNN". eng. In: *European journal for security research* (2019). ISSN: 2365-1695.

[26] Jozefowicz, Rafal, Zaremba, Wojciech, and Sutskever, Ilya. "An Empirical Exploration of Recurrent Network Architectures". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 2342–2350.

[27] Khuntia, Swasti and Rueda, Jose. "Long-Term Electricity Load Forecasting Considering Volatility Using Multiplicative Error Model". eng. In: *Energies* 11.12 (2018). ISSN: 19961073. URL: http://search.proquest.com/docview/2316421456/.

[28] Kim, Tae-Young and Cho, Sung-Bae. "Predicting residential energy consumption using CNN-LSTM neural networks". eng. In: 182 (2019), pp. 72–81. ISSN: 0360-5442.

[29] Kong, W., Dong, Z. Y., Jia, Y., Hill, D. J., Xu, Y., and Zhang, Y. “Short-Term Residential Load Forecasting Based on LSTM Recurrent Neural Network”. In: *IEEE Transactions on Smart Grid* 10.1 (2019), pp. 841–851. DOI: 10.1109/TSG.2017.2753802.

[30] Kuznetsov, Vitaly and Mariet, Zelda. “Foundations of Sequence-to-Sequence Modeling for Time Series”. In: (2018).

[31] Lago, Jesus, De Ridder, Fjo, and De Schutter, Bart. “Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms”. eng. In: *Applied energy* 221 (2018), pp. 386–405. ISSN: 0306-2619.

[32] Liu, Bingchun, Fu, Chuanchuan, Bielefield, Arlene, and Liu, Yan. “Forecasting of Chinese Primary Energy Consumption in 2021 with GRU Artificial Neural Network”. eng. In: *Energies (Basel)* 10.10 (2017), p. 1453. ISSN: 1996-1073.

[33] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: https://www.tensorflow.org/.

[34] Massidda, Luca and Marrocu, Marino. “Smart Meter Forecasting from One Minute to One Year Horizons”. eng. In: *Energies (Basel)* 11.12 (2018), p. 3520. ISSN: 1996-1073.

[35] McKinney, Wes. “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[36] Microsoft. *What is the Azure Machine Learning SDK for Python?* 2020. URL: https://docs.microsoft.com/en-us/python/api/overview/azure/ml/?view=azure-ml-py.

[37] Mocanu, Elena, Nguyen, Phuong H, Gibescu, Madeleine, and Kling, Wil L. “Deep learning for estimating building energy consumption”. eng. In: *Sustainable Energy, Grids and Networks* 6 (2016), pp. 91–99. ISSN: 2352-4677.

[38] Olah, Christopher. *Understanding LSTM Networks*. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[39] Patel, Mihir, Dabhi, Dharmesh, Patel, Ravi, and Patel, Jignesh. *Long Term Electrical Load Forecasting considering temperature effect using Multi-Layer Perceptron Neural Network and k-Nearest Neighbor algorithms*. May 2019. DOI: 10.13140/RG.2.2.29592.65288.

[40] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[41] Phi, Michael. *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. 2020. URL: https://www.michaelphi.com/illustrated-guide-to-lstms-and-grus-a-step-by-step-explanation/.

[42] Salinas, David, Flunkert, Valentin, and Gasthaus, Jan. "DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks". In: (2017).

[43] Schmidhuber, J. "Deep Learning in Neural Networks: An Overview". In: *Neural Networks* 61 (2015). Published online 2014; based on TR arXiv:1404.7828 [cs.NE], pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003.

[44] Seabold, Skipper and Perktold, Josef. "statsmodels: Econometric and statistical modeling with python". In: *9th Python in Science Conference*. 2010.

[45] Song, Xuanyi, Liu, Yuetian, Xue, Liang, Wang, Jun, Zhang, Jingzhe, Wang, Junqiang, Jiang, Long, and Cheng, Ziyan. "Time-series well performance prediction based on Long Short-Term Memory (LSTM) neural network model". eng. In: *Journal of petroleum science engineering* 186 (2020), p. 106682. ISSN: 0920-4105.

[46] Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. "Sequence to Sequence Learning with Neural Networks". In: (seqtoseq).

[47] Tuaimah, Firas and Abass, Huda. "Short-Term Electrical Load Forecasting for Iraqi Power System based on Multiple Linear Regression Method". In: *International Journal of Computer Applications* 100 (Aug. 2014), pp. 41–45. DOI: 10.5120/17492-8011.

[48] Wang, Senzhang, Cao, Jiannong, and Yu, Philip S. "Deep Learning for Spatio-Temporal Data Mining: A Survey". eng. In: (2019).

[49] Wang, Yusen, Liao, Wenlong, and Chang, Yuqing. “Gated Recurrent Unit Network-Based Short-Term Photovoltaic Forecasting”. eng. In: *Energies (Basel)* 11.8 (2018), p. 2163. ISSN: 1996-1073.

[50] *Web Traffic Time Series Forecasting*. https://www.kaggle.com/c/web-traffic-time-series-forecasting. Accessed: 2010-09-30.

[51] Xiao, Yuelei and Yin, Yang. “Hybrid LSTM Neural Network for Short-Term Traffic Flow Prediction”. eng. In: *Information (Basel)* 10.3 (2019), p. 105. ISSN: 2078-2489.

[52] Zhang, Guoqiang Peter. “Time series forecasting using a hybrid ARIMA and neural network model.” In: *Neurocomputing* 50 (June 2, 2003), pp. 159–175. URL: http://dblp.uni-trier.de/db/journals/ijon/ijon50.html#Zhang03.

[53] Zhang, Guoqiang, Eddy Patuwo, B, and Y. Hu, Michael. “Forecasting with artificial neural networks:: The state of the art”. eng. In: *International Journal of Forecasting* 14.1 (1998), pp. 35–62. ISSN: 0169-2070.

[54] Zhu, L. and Laptev, N. “Deep and Confident Prediction for Time Series at Uber”. In: *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. 2017, pp. 103–110.