PROBABILISTIC FORECASTING OF MULTIPLE TIME SERIES WITH SINGLE RECURRENT NEURAL NETWORK

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SARP TUĞBERK TOPALLAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
SCIENTIFIC COMPUTING

SEPTEMBER 2022

Approval of the thesis:

PROBABILISTIC FORECASTING OF MULTIPLE TIME SERIES WITH SINGLE RECURRENT NEURAL NETWORK

submitted by **SARP TUĞBERK TOPALLAR** in partial fulfillment of the requirements for the degree of **Master of Science in Scientific Computing Department, Middle East Technical University** by,

Prof. Dr. A. Sevtap Selçuk Kestel _____
Dean, Graduate School of **Applied Mathematics**

Assoc. Prof. Dr. Önder Türk _____
Head of Department, **Scientific Computing**

Prof. Dr. Ceylan Yozgatlıgil _____
Supervisor, **Statistics, METU**

Examining Committee Members:

Prof. Dr. Ömür Uğur
Scientific Computing, METU _____

Prof. Dr. Ceylan Yozgatlıgil
Statistics, METU _____

Prof. Dr. Seher Nur Sülkü
Econometrics, Hacı Bayram Veli University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: SARP TUĞBERK TOPALLAR

Signature :

ABSTRACT

PROBABILISTIC FORECASTING OF MULTIPLE TIME SERIES WITH SINGLE RECURRENT NEURAL NETWORK

TOPALLAR, SARP TUĞBERK
M.S., Department of Scientific Computing
Supervisor : Prof. Dr. Ceylan Yozgatlıgil

September 2022, 71 pages

Time series forecasting can be summarized as predicting the future values of a sequence indexed by timestamps based on the past records of that sequence. Optimal or near-optimal resource allocation requires accurate predictions into the future. The study presents investigation performed on both classical methods and more contemporary methods from the literature. The classical methods studied are Autoregressive Integrated Moving Average (ARIMA), Exponential Smoothing (ETS) and Seasonal-Trend decomposition using LOESS (STL). One of the more contemporary time series models is a deep learning method called DeepAR, which is a Recurrent Neural Network (RNN) comprised of Long Short Term Memory (LSTM) cells. Both novel and classical approaches pose challenges unique to their own methodologies. Classical methods for example require the data to satisfy restricting modeling assumptions and each series to be preprocessed and modeled one by one. On the other hand, machine learning methods require a great amount of quality data for training, where it would be a challenge where the abundance of data may not be available. The proposed solution is to model many similar series with a single common model at the same time. The similarity in the context of this research refers to the fact that each series has the same recorded metric. The data set investigated in the study holds records of demand for many physical stores throughout Türkiye, where each individual series corresponds to a single physical location. There are a total of 120 individual series spanning between 2018 and 2022.

The probabilistic forecasts are obtained by training the DeepAR model in order to learn a

probability distribution and producing the point forecasts from sampling the learned probabilistic function. The probabilistic forecasts of different quantiles provide practicality such as forecasting in different quantiles for each series individually and can be tuned for different sensitivity to forecasting errors per series.

At the end of the study, the provided error metrics for accuracy measurement are MAE (Mean Absolute Error) in order to show the actual value of the forecasted demand and MAPE (Mean Absolute Percentage Error) in order to compare the results with other models independent of the scale. The DeepAR model provided more accurate results compared to ETS and ARIMA on average for the whole data set in terms of both accuracy metrics.

Keywords: Time series, Demand forecasting, Deep neural networks, Long short-term memory neurons, Recurrent neural networks, Autoregressive recurrent networks

ÖZ

BİR ÖZYİNELİ SİNİR HÜCRESİ AĞI İLE ÇOKLU ZAMAN SERİLERİNİN OLASILIKSAL TAHMİNLENMESİ

TOPALLAR, SARP TUĞBERK
Yüksek Lisans, Bilimsel Hesaplama Bölümü
Tez Yöneticisi : Prof. Dr. Ceylan Yozgatlıgil

Eylül 2022, 71 sayfa

Zaman serisi tahminleme zaman ile endekslenmiş bir dizi değerin geçmişte almış olduğu değerlere bakılarak gelecekteki değerlerin tahminlenmesi olarak özetlenebilir. Optimal veya optimale en yakın kaynak planlanması geleceğe dair yüksek doğruluklu tahminlimeler gerektirir. Bu çalışma, literatürden hem klasik yöntemler hem de daha çağdaş yöntemler üzerinde yapılan incelemeleri sunmaktadır. İncelenen klasik yöntemler, Otoregresif Entegre Hareketli Ortalama (ARIMA), Üstel Düzeltme (ETS) ve LOESS kullanılarak Mevsimsel-Trend (STL) ayrıştırmasıdır. Öte yandan daha çağdaş zaman serisi modellerinden biri, Uzun Kısa Süreli Bellek (LSTM) hücrelerinden oluşan bir Tekrarlayan Sinir Ağı (RNN) olan DeepAR adlı bir derin öğrenme yöntemidir. Hem çağdaş hem de klasik yaklaşımlar, kendi metodolojilerine özgü zorluklar ortaya çıkarır. Örneğin klasik yöntemler, verilerin kısıtlayıcı modelleme varsayımlarını karşılamasını ve her serinin birer birer ön işleme tabi tutulmasını ve modellenmesini gerektirir. Diğer yandan, makine öğrenimi yöntemleri, eğitim için büyük miktarda kaliteli veri gereksinimi duyarlar veri bolluğunun mevcut olmayabileceği durumlarda bu durum büyük bir zorluk yaratır. Bu araştırmada önerilen çözüm, birçok benzer seriyi aynı anda tek bir ortak modelle modellemektir. Bu araştırma bağlamındaki benzerlik, aynı metriğin ölçüldüğü bir çok ayrık serinin bir arada incelenmesini ifade eder. Araştırma için incelenen veri seti, Türkiye genelinde her bir serinin tek bir konuma karşılık geldiği birçok fiziksel mağazaya olan talep kayıtlarından oluşmaktadır. 2018 ile 2022 yıllarını kapsayan toplam 120 ayrı seri bulunmaktadır.

Araştırmada kullanılan DeepAR modeli veri setinden ortak bir olasılık dağılımını öğrenir. Öğrenilen bu olasılık dağılımından yapılan farklı çeyrekliklerdeki örneklemler ile olasılıksal tahminler elde edilir. Seçilen farklı çeyreklikler, farklı hata hassasiyetlerine sahip olan her bir ayrık seri için yapılan tahminler için özel olarak ayarlanabilir ve her seri için farklı bir değere sahip olabilir.

Çalışmanın sonunda, doğruluk ölçümü için sağlanan hata metrikleri, öngörülen talebin gerçek değerini gösterirken Mutlak Ortalama Hata (MAE) ve sonuçları ölçekten bağımsız olarak diğer modellerle karşılaştırırken Mutlak Ortalama Yüzdesel Hata (MAPE) metrikleridir. DeepAR modeli, her iki doğruluk metriği açısından da tüm veri seti için ortalama olarak ETS ve ARIMA modellerine kıyasla daha doğru sonuçlar üretmiştir.

Anahtar Kelimeler: Zaman serileri, Talep tahminleme, Derin sinir ağları, Uzun kısa-süreli bellek sinir hücreleri, Özyineli sinir ağları, Özbağlanımlı özyineli ağlar

To my family

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my thesis supervisor Prof. Dr. Ceylan Yozgatlıgil for her support and patient guidance throughout my studies. The time she spared to share her experiences and give advice to make this study possible is truly appreciated and will remain an inspiration to me.

I would also like to thank the members of my thesis defense committee for their insightful comments and discussions.

Last but not least, I would like to thank my family for their continual support for me throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACF	Auto Correlation Function
ADF	Augmented Dickey-Fuller
AIC	Akaike Information Criterion
ANN	Artificial Neural Network
ARIMA	Auto Regressive Integrated Moving Average
ARMA	Auto Regressive Moving Average
BIC	Bayesian Information Criterion
BPTT	Back Propagation Through Time
CNN	Convolutional Neural Network
DBA	DTW Barycenter Averaging
DNN	Deep Neural Network
DTW	Dynamic Time Warping
ECG	Electrocardiogram
ETS	Exponential Smoothing
EWM	Exponentially Weighted Mean
FFT	Fast Fourier Transform
GAM	Generalized Additive Models
GAN	Generative Adversarial Networks
GRU	Gated Recurrent Unit
IQR	Interquartile Range
KPSS	The Kwiatkowski–Phillips–Schmidt–Shin Test
L-BFGS	Limited Memory Broyden–Fletcher–Goldfarb–Shanno
LOESS	Locally Estimated Scatter-plot Smoothing
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MSE	Mean Squared Error

N-BEATS	Neural Basis Expansion Analysis for Interpretable Time Series Forecasting
NLP	Natural Language Processing
PACF	Partial Auto correlation Function
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SSE	Sum of Squared Error
STL	Seasonal-Trend decomposition using LOESS
TBATS	Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components

CHAPTER 1

INTRODUCTION

Time series data is a collection of regularly collected data where each data point is indexed by a time stamp. Stock market data in finance, temperature records in climatology or electrocardiogram (ECG) records in medicine are all examples of time series data. Aim of forecasting is to predict the future values of a given series with high accuracy by analyzing the data collected in the past. Predicting a single value into the future is called as one step ahead forecasting and predicting N values into the future is called N-step ahead forecasting. For both cases, the forecasts can be generated for a single value which is called univariate forecasting (e.g., temperature records for a location) or the prediction task can be performed for multiple variables changing simultaneously over time, in this case, such a process is called multivariate forecasting (e.g., each axis of a vector in $\mathbb{R}^n$).

Improving the forecast accuracy enables more optimal resource provisioning which in turn brings efficiency to any kind of planning effort. In order to have more accurate forecasts, there have been numerous approaches studied with different advantages and disadvantages. It is possible to examine the available forecasting methods in two main categories, such as classical methods and more contemporary methods. Classical methods include Autoregressive Integrated Moving Average (ARIMA), Exponential Smoothing (ETS) and Seasonal-Trend decomposition using LOESS (STL). In univariate or low dimensional datasets the classical methods proved their applicability but as time progressed, the volume of recorded information grew rapidly as well as the complexity of the recorded information. The advent of technology, more specifically the increase of computational power, allowed much more complex analysis methodologies to be utilized. This increase in computational power has been especially beneficial for machine learning algorithms to be created for analysis of time series data to obtain even more accurate forecasts. Mainstream adoption of machine learning methods, most notably deep learning methods, have shown their applicability for modeling both large and high dimensional datasets [42]. Deep learning is the study of Deep Neural Networks (DNNs) which are multi layer Artificial Neural Network (ANN) architectures. Even though the idea of a neural network is coined more than sixty years ago [78], research activity in the deep learning field reached historical peaks in the recent years [114].

Neural networks can have different names depending on their architecture or the specific types of neurons comprising the network. Recurrent connections in the network architecture

allow the cells of the network to pass the current hidden state to itself as an input for the next step during both training and inference. These types of neurons are called recurrent neurons, and long-short-term memory (LSTM) [45] cells and Gated Recurrent Unit (GRU) [20] cells are examples of recurrent neurons. Networks that include recurrent neurons are called Recurrent Neural Networks (RNNs). RNNs showed their applicability in sequence to sequence modeling especially in the Natural Language Processing (NLP) domain [16]. RNNs learn the training data as an ordered sequence and this enables the ability to learn the past behavior of any sequence of data (e.g., time series data, sentences in a text) for the model. Since the network learns the behavior of the series sequentially, during inference time the predictions made by the model are also generated as an ordered sequence for multi step forecasting. On the other hand there are networks passing the hidden states only to the downstream layers and these are called as feed forward neural networks. Named after their convolutional layers, the Convolutional Neural Networks (CNNs) are examples of feed forward neural networks and they have shown success for learning the spatial properties of the data which makes them very successful for tasks like image classification or object detection.

In addition to the required computational complexity, deep learning methods require huge amounts of data in order to generate highly accurate predictions. When increasing the size of the training data regularly is not straightforward or not feasible, it is possible to increase the size of the training data set artificially by introducing various modifications to the training samples and this approach is called data augmentation. In the image classification setting, some examples for data augmentation methods are obfuscation, translation, color modification, cropping, rotation and noise injection. Data augmentation, in addition to the increase in the number of training samples, has also shown [97] regularizing effects to remedy the overfitting problem.

In time series forecasting settings, augmenting the data is not as straightforward or as diversified as in the image setting. Some studied methods [53, 54, 57, 108, 27] to augment time series data require the usage of very complex and resource intensive methods such as training deep neural networks for augmentation. Although there are findings [58] suggesting to increase the size (number of parameters) of the model in order to increase the capacity of the model, this suggestion should be taken with a grain of salt since there are also findings [46] that suggests without increasing the size of the training data, increasing the number of parameters in the model can result with under-trained models. This thesis suggests an approach to obtain an accurate forecasting model without the need for synthetic data generation for training if there are many similar series present in the training data set by creating a global model for all series together instead of training a separate model per each individual series in the training set.

1.1 Motivation and Research Question

In order to obtain accurate time series forecasts with neural networks, the required training data size is very large. Accurate forecasts are still needed when the sufficiently large training

data set is not available. One solution to this requirement is to augment the training data in order to have a more accurate model. This study proposes an alternative method to need for synthetic data generation by computational resource intensive data augmentation methods. In settings where there are many series measuring the same type of metric, instead of creating a single model for each of the recorded series like ARIMA, STL or ETS models, proposed solution is to train a common global model featuring all off the available series in the training set. Aim of this approach is to saturate the neural network model with enough training data without needing synthetically generated data and obtain all of the forecasts for different series in the data set from the same model. By not needing complex data augmentation methods and decreasing the model count to exactly one, even if the abundance of training data is not present, this approach aims to optimize sample efficiency. Experiments are conducted on algorithms that allow the training of a universal common model and generate forecasts for many different series from the same model. The specific data set selected for this study is the proprietary physical demand records for a retail chain from different locations throughout Türkiye. Detailed descriptions of the data set and the model implemented will be thoroughly discussed in the following chapters.

The data set used is collected via physical sensors that do image processing so sensor error is a factor that is worthy of consideration in this setting. Because each sensor or lens have different amount of intrinsic imperfections, error distribution of different series is presumed to be distinct per series (sensor) basis. On top of that, the demand profile for each different location is expected to be unique according to their own characteristic. These factors dictated the decision for performing the preprocessing step for each series to be completed individually. In order to increase the data quality for the model, the pre-processing step which includes both the imputation method for missing values and cleaning method for detecting and then correcting the outliers is developed during this study for this research specifically. The preprocessing methods developed for this research are introduced and explained in detail in Chapter 4. The idea of expanding the training set horizontally by using multiple similar series to train one common model required a strategy for determining which series are appropriate to be clustered together or confirm that clustering is possible or not. This decision is confirmed by using the elbow curve method and silhouette score [88] with Dynamic Time Warping (DTW) [89] is used as the distance metric. This step is done to confirm that the selected series are inherently suitable for to be clustered together after pre-processing is complete and before modeling started. Performance of the pre-processing and the actual model are both presented in the Chapter 5 as results. This study aims to create a forecasting system capable of pre-processing and modeling many series at once with accurate predictions and presents the mathematical background for the used methods.

This study lays out a holistic approach from exploratory data analysis, data preprocessing, to modeling and evaluation of the generated model. Preprocessing steps are discussed with detail including the numerical calculation methods as well as the algorithms subject to experiment. In addition, the reasoning behind the selection of the algorithms used is explained. The evaluation of the model is presented with selected error metrics for performance as well

as the durability (i.e. degradation after training) and the potential for generalization of the model are all discussed. The practical applicability of the model and ways to improve the presented work is described at the conclusion part of the study. Since the selected data set covers the demand for physical shops it is important to acknowledge the effects of the COVID-19 pandemic. In order to isolate the study from the effects of the pandemic the data set spans from January of 2018 to January of 2020. This scope is used for both training and testing the produced model in the calculation of the accuracy metrics. In general, time series forecasting methods assume that the training data and the test data come from the same distribution but in real world there are trend shifts or sudden behavior-altering events can occur. These external factors can be modeled to some degree with additional features accompanying the actual series during the modeling stage but a world changing event like COVID-19 pandemic is a near unprecedented occasion. Thus, the effects of it are completely left out of this research, instead of experiments on modeling the effects of the pandemic.

1.2 Thesis Structure

Structure of the thesis is outlined and presented with brief information about each chapter as follows.

Chapter 1 - Introduction is the current chapter, presents the introduction, the research question and motivation as well as the structure of the thesis.

Chapter 2 - Literature Survey related research and methods to problem definition and different approaches.

Chapter 3 - Methodologies Used description and detailed explanation of the methods used in this investigation.

Chapter 4 - Data Analysis exploratory data analysis, data preprocessing, cluster analysis, model training hyperparameter tuning and performance benchmark of each step.

Chapter 5 - Results And Future Work summary of achieved results and the possible approaches to be considered in order to increase the achieved performance even more.

Chapter 6 - Conclusion the final discussion about the study and whether the findings of the experiments are consistent with the theorized framework.

CHAPTER 2

LITERATURE SURVEY

In this chapter the relevant research is presented, the presented material covers both the well known research as well as the more contemporary work published regarding the subject. Main aspects of the literature review is to present the methodologies about time series forecasting in general, available methodologies for demand forecasting specifically, deep learning methodologies and available data sets. Time series forecasting is an active area of research and deep learning methods utilized by this field are evolving as more research is published on the matter. Some of the work focus on the architecture of the network as a whole and the mechanism implemented between each layer. Examples of research on the network architecture between layers are presented in Section 2.2. On the other hand, there is also research conducted on how changes in internal architecture of each of the neurons can be beneficial to improve overall accuracy, which is explained in detail in Section 3.1.4. More specific research aimed at demand forecasting is explained in Section 2.3 with examples from different areas, especially from the retail domain.

There are many methods which does not incorporate deep learning methods such as Auto Regressive Moving Average (ARIMA) [8], Exponential Smoothing (ETS) [12, 47, 110], Seasonal and Trend decomposition using Loess (STL) [21], Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components (TBATS) [72], Neural basis expansion analysis for interpretable time series forecasting (N-BEATS) [83] and a multi component additive regression model called Prophet [103]. Some of these algorithms are inherently unusable for the prediction task at hand because of the failed modeling assumptions, as explained in the Section 2.1, and some of them cannot process the external information except the measurements in the series and lastly many of them does not support the idea of one common model for multiple series and requires each series to be modeled separately which contradicts the core motivation of this research so not all of the methods mentioned here were subjected to the experiments for this study.

There are multiple deep learning methods introduced in the Section 2.2 including the well known LSTM type networks and more novel research in order to explain the area of deep learning applied to time series data is headed to. The LSTM type networks are very crucial for this research since the core component of the main algorithm examined in this thesis is an LSTM network. Depending on the difference in the layer architecture some RNN networks

can be named as Elman Networks, Jordan Networks of Hopefield Networks furthermore depending on their cell architecture they can be called as LSTM or GRU networks as well depending on the cells comprising the network.

2.1 Methods For Time Series Analysis

The methods well known in time series forecasting domain well before deep learning methods have gained ground. The main drawback of these models is that they are all univariate models, which cannot be utilized for the common model scheme of this research in the forecasting aspect but their performance is measured for the imputation studies where the pre-processing step is done individually per series. The Python implementation in the $statsmodels$ package [94] is used for the experiments of the methods presented in this chapter. These algorithms are not able to obtain a global model from many series but require each series to be modeled individually thus the applicability of these are considered for the imputation study and the actual forecasting is not done on these models.

2.1.1 Exponential Smoothing (ETS)

Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations age toward the beginning of the series [12, 47, 110]. This mechanism allows the more recent data to be more prominent when generating forecasts which would allow recent changes of behaviour of the data to be captured. The components of the method are (E, T, S) which are **e**rror, **t**rend and **s**easonality respectively. Either of these can be one of additive, multiplicative or none (i.e. not going to be included in the model). This model does not require the removal of seasonality or trend from the input data, since it models those components in particular. It is a well studied [35] approach and aims to estimate its parameters by minimizing the sum of squared errors (SSE) for each $e_t = y_t - \hat{y}_{t|t-1}$ in the series where $t \in [1, T]$ and y is the actual value and $\hat{y}$ is the predicted value in a series of length T, minimization is carried out one step at a time of which the error can be formulated as follows

$$\text{SSE} = \sum_{t=1}^{T} (y_t - \hat{y}_{t|t-1})^2 = \sum_{t=1}^{T} e_t^2. \tag{2.1}$$

This minimization is non straightforward since it includes a non linear optimization scheme and implementation in the $statsmodels$ package [94] uses numerical methods such as complex-step differentiation for calculating the covariance matrix which is used for maximum likelihood estimation. Another implementation of ETS is present in the R language and is one of the most well studied forecasting algorithms [50]. The performance of the ETS algorithm will be compared in more detail in the following chapters with other candidate algorithms in the imputation phase.

2.1.2 Auto Regressive Integrated Moving Average (ARIMA)

Auto Regressive Moving Average (ARIMA) model is a very well known time series model in the family of Box-Jenkins models but it works under certain conditions [8]. These conditions can be summarized as the modeling assumptions and the ARIMA model requires a stationary time series, a stationary time series is defined as follows

- mean of the series, μ, is constant and does not change depending on the time
- variance of the series, σ, is constant and does not change depending on the time
- no seasonality observed in the series
- the autocovariance function (i.e. $\mathrm{COV}(Z_t, Z_{t+h})$) of the process Z_t depends only on h and not t

stationarity examination can be done visually by using the plot of the series itself combined with auto correlation and partial auto correlation plots of the series [51]. These are special plots called correlograms. These correlograms are used to determine the parameters $\mathrm{AR}(p)$ and $\mathrm{MA}(q)$ of the ARIMA model for a stationary series. In addition to graphical perusal of correlograms and the series plot itself, there are formal statistical methods for confirming that if a series is stationary or not such as The Kwiatkowski–Phillips–Schmidt–Shin Test (KPSS) and Augmented Dickey-Fuller (ADF) tests. ADF test has the null hypothesis that a unit root is present in an autoregressive model of a given time series which means the inspected series is not stationary [31]. Similarly, the KPSS test has the null hypothesis which given series is stationary around a deterministic trend [61]. In case of non stationary data it is possible to achieve stationarity by detrending or differencing methods depending on the data and procedures may be repeated until stationarity is achieved (if it can be achieved) for more than one time. Since the remedy for non stationarity will change depending on the series itself, and the number of applications, achieving seasonality requires that each series should be inspected separately even sourced from similar measurements. In addition, seasonal behavior can be present in the data which can be modeled with SARIMA model or can be removed by deseasoning since ARIMA does not model the seasonality. After the proper modeling preparations, ARIMA methods should have the process of diagnostic checking. In order to confirm that no correlation is observed in the final model's data which can be done by statistical tests like Ljung-Box [73] or Box-Pierce [9]. Auto-correlation of the residuals (errors) should also be investigated for the model which can be done with the help of the Breusch-Godfrey [10, 37] test and in a healthy model there should not be a strong indication of auto-correlation of residuals. The very complex nature of this approach diminished the practicality of this solution for imputation stage since parameters of the model should be adjusted for each series in order to maximize the accuracy.

2.1.3 Seasonal-Trend decomposition using LOESS (STL)

Seasonal and Trend decomposition using LOESS (STL) [21] approach aims to obtain the decomposed components of the series as seasonal component, trend component and the residuals. This model does not require the removal of seasonality or trend from the input data and optimizes the model parameters according to the given data without any preventing assumptions. Seasonal component is obtained by using locally estimated scatterplot smoothing (LOESS) as a smoothing method. This is done by applying locally weighted polynomial regressions at each point in the data set in order to build the additive regression model $x_i = t_i + s_i + e_i$ where x_i is a measurement from the series, t_i is trend, s_i is seasonality and e_i is the error or residuals of that measurement. The goal is to obtain the best fitting curve to the presented data using regressions. The periodicity of the seasonality of the data like daily, weekly or yearly is given as a required parameter and window size (lag count) for the seasonal smoother and trend smoother can be provided as parameters can be used to fine tune the results better which requires special attention per series in order to use this algorithm as the imputation method of choice. The main attraction of this algorithm is that it is robust to outliers.

2.1.4 Prophet

The Prophet algorithm [103] is an additive regression model $y(t) = g(t) + s(t) + h(t) + \epsilon_t$ where g denotes the trend function, s denotes the periodic changes (seasonality), h represents special days like holidays and ϵ_t error terms assumed to be normally distributed. The fit of the model is performed similar to Generalized Additive Models (GAM) [44] by using the L-BFGS [14] algorithm. The fit for the trend is performed with a saturating growth model or a piecewise linear model, the seasonality is modeled with Fourier series and multiple seasonalities can be modeled automatically and the special days or holidays must be provided to the algorithm by the user of the algorithm. There is another implementation called neural prophet which is a hybrid approach that combines the original prophet and a covariate module which can be trained as a neural network [104]. The algorithm is also experimented on in the imputation phase and results are presented in Section 4.3.

2.2 Deep Learning Methods For Time Series Analysis

This section is mainly interested in the architecture of deep learning methods and how they are utilized to increase the accuracy of forecasting. The different approaches presented here are brief and prepared in order to convey the core idea of the different approaches rather than to provide an in-depth analysis of each different approach.

2.2.1 Hybrid Approach

Hybrid approach is referred to as combining a classical algorithm like ARIMA with a machine learning algorithm like gradient boosting [33] with the help of libraries like XGBoost or LightGBM, extensive research is available [41, 115] on comparisons of different combinations with different variations of component methods. The core idea about these approaches is to detect where each type of algorithm is best at and combine it to have a result which would potentially have the best of both worlds by combining the advantageous aspects of each.

2.2.2 Convolutional Neural Network (CNN) Approach

The convolutional Neural Networks which are known to be successful for the image processing domain also have their applications in time series forecasting as temporal convolutional networks [107] or combined with recurrent structure [109]. There are comparative studies that inspect the potential of CNNs for image and sequence modeling such as time series and speech [66]. Similarly, for the classification problem for images CNNs are also proposed [116] for time series as well.

2.2.3 Transformers Approach

Transformers are the types of networks that make use of the encoder decoder architecture [105]. In the natural language processing domain, especially in the machine translation problem which is a very active area of research for sequence to sequence modeling, the attention mechanism applied by the transformers have shown great success [29, 13] and are capable of generating very large language models successfully [19] as well as smaller ones [46]. Attention is the mechanism applied in the transformer networks to mimic the cognitive attention of humans when they are reading or translating text. It has some applications in time series forecasting [87, 70, 117, 111] but compared to NLP the applications are very limited in number.

2.2.4 Generative Adversarial Networks (GAN) Approach

Generative Adversarial Networks (GANs) [39] feature two main components a generator and a discriminator. The generator network is the network that generates the output and it is presented to the discriminator which is already capable network. In the context of the image generation discriminator can correctly classifies images and the task of the generator is to create new images that are good enough to convince the discriminator that the generated images are real. GANs are very successful for tasks like image generation or style transfer and in the time series domain there are data generation [113, 55], time series classification [28] and outlier detection [68, 36]. Although forecasting implementations exist [60] this is

another novel approach and compared to RNN based solutions not yet widely adopted.

2.2.5 Recurrent Neural Network (RNN) Approach

Most of the literature presented in this survey is either using the LSTM or GRU cells in the RNN architecture or combining the formerly mentioned approaches on top of the RNNs. By the volume of the published research it is possible to say that this is the most dominantly used type of neural network for time series forecasting. The core model (DeepAR) is also another example of this architecture, and details of this architecture are presented in more detail in Section 3.1.2. There are many versions of recurrent architecture with different variations applied to the time series forecasting problem this is a well studied area of research with extensive surveys available [43, 7].

2.3 Deep Learning Methods For Demand Forecasting

Focus of this chapter is narrowed down to the more similar research with similar data and underlying algorithms instead of more generally available algorithms. There are well known datasets where research is continuing like CIF2016 [100], M5 data set [75] and Online Retail I, Online Retail II datasets provided by UCI ML Repository [32] which are publicly available. On the other hand, there are proprietary datasets that are not publicly available as the one used for this thesis. Studies of this type provides a research opportunity that would allow the time series forecasting methods to be experimented on data sets not generally available. On the literature there are similar research conducted specifically in the retail domain which are introduced in this chapter.

The initial procedure for the data analysis is the collection of the data itself and methods in the retail domain focuses on two main aspects of data collection. The first aspect is the potential demand which can be measured by the number of visitors in actual physical locations or virtual stores online which can be named as the footfall. The second aspect is the realized demand which is the number of people actually committed a transaction like a purchase which can be generally called as sales data. Collection of sales data is straightforward, since transactions are recorded electronically in both physical and online stores. On the other hand the measurement of the number of people which have the potential to convert to a sales transaction requires additional infrastructure for measurements. This is especially prominent in physical locations where logging the web requests in order to keep track of the visitor is not possible like online stores. The infrastructure requirement in the retail domain can be addressed with vision based solutions processing information from camera sensors. This is a common practice [96, 81, 79] in order to achieve automatic and accurate record for the footfall measurements and the data collected for this thesis is generated with a similar strategy. Although there have been previous research [59] done on the retail market in Türkiye the focus of the aforementioned studies were directed towards to the sales perspective rather than

the demand perspective. In addition, globally there are many research done about forecasting the demand in different industries like transportation [118, 62], tourism [65, 56] as well as retail [90, 69, 52] which is the main research domain for this study. In this domain, there are findings that show that LSTM networks can produce very accurate results and compared to CNN more accurate results [63, 119]. The probabilistic approach of forecasting in retail were already adopted in order to have more optimal inventory management, better staff scheduling or efficient supply chain operations [64].

There is no inherit limitation for the methodologies mentioned in Section 2.2 for the retail domain specifically but it is necessary for the implemented method to have the capability to process or model all of the given series together in order to scale the available training data horizontally and make data abundant for each model respectively. Even though methodologies mentioned in the Chapter 2 as literature survey are all viable time series forecasting strategies, in order to realize the horizontal scaling approach the work reviewed here cannot be all candidates for the continuation of the research because of the underlying modeling restrictions. In Chapter 3 the methodologies determined are appropriate for this research are introduced with detailed descriptions and their working principles are discussed as well as the reasoning behind the decision to be used.

CHAPTER 3

METHODOLOGIES USED

The used methodologies are explained in detail in this chapter. In order to provide more clarity, the preliminaries to the subject are presented in Section 3.1 which would provide the foundation of the methods used in this research and some possible problems which can occur and possible ways to prevent them. Not all of the methods from the literature can be applied to the given problem since some of them failed to fulfill the assumptions of the model. The application of the modeling is completed at two folds. The first step is to individual modeling for the imputation and outlier cleaning. For this first stage the Exponential Smoothing approach is used and details of the approach are presented in Section 3.2.3 and the actual forecasting is performed with DeepAR, which is presented in Section 3.2.4 in detail which is the core model of this research.

3.1 Preliminaries For The Subject

This chapter is dedicated to present some preliminary notions in order to make reasoning behind trade-offs and the points made in the following chapters more clear. Some historic information, variations of the used components and generic formulae about the networks are discussed. The equations described here are generic equations for exemplary purposes, actual derivations can be different with different activation functions or cell architectures. For example, the gradient needs to be calculated for the optimization in order to minimize the error function or train the network through back propagation. In order to calculate this gradient the partial derivatives of the following are chained together; the equations of the cell architecture, activation functions and the loss function. It is impractical to calculate and present each of the different calculation from the numerous combinations but this is a well studied topic and there are abundant resources with derivations of backpropagation for different activation/loss functions and for different architectures. The main aspect of this research is focused on the implementation of the derived methods, in order to make that implementation built on solid foundations, minimal amount of equations for related subjects are presented here accompanied by visual explanations as supplementary materials.

3.1.1 Representational Power of Neural Networks

The Universal Approximation Theorem for neural networks means that there exists a network that can approximately represent a function f independent from the definition of the function itself hence the universality. In the context of the universal approximation theorem, the term *width* is used for the number of the neurons layer wise and the term *depth* is used to denote the number of layers in the network. It is shown [23] that for the sigmoid function the arbitrary width can be a universal approximator. This result is further extended [67] since any non-polynomial activation function still preserves the approximation capability. Instead of a specific activation function, it is shown [48] that the representational power of the network still exists as long as the multilayer forward structure is maintained. On the assumption that there exists a function representing the future values of a series, universal approximation theorem shows that neural networks can learn (approximate) the function by learning from the data and will be able to generate accurate forecasts to some degree. This is shown for recurrent neural networks [93] as well as the feed forward neural networks.

3.1.2 Recurrent Neural Network (RNN)

Recurrent neurons pass the current output (i.e. hidden state) to themselves as another input combined with the output of the previous layer for the next step. This approach enables them to remember the encoded information from the previous steps thus it can be considered as a "memory" mechanism. Whereas in the feedforward neural networks, each neuron's calculated result (i.e. hidden state) is passed only onto the downstream layers and the new hidden state is calculated without combining the output of the same neuron in the previous step but using only the output from the previous layer in the current step. Figure 3.1, visualizes the difference between these two types of networks. The left side of both diagrams represents the fully connected inputs to each neuron from a fully connected layer and single arrows on the right side of each neuron represents the individual output of that neuron.

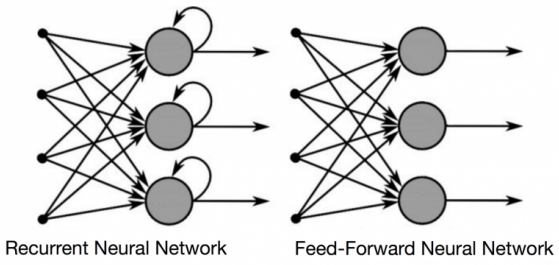


Figure 3.1: Comparison of different types of neurons [26]

The difference in the network architecture introduces a difference in the network behaviour as well. During training the weights in each neuron is determined with back propagation in feed forward neural networks. On the other hand, recurrent networks apply the same principle

through time and this is called Back Propagation Through Time (BPTT) which is explained further in Section 3.1.3. Back propagation aims to optimize weight parameters of each neuron in order to minimize the error function or cost function which is the core idea of training a network. Although the principle is the same for both types of networks, BPTT has its own unique problems such as exploding or vanishing gradients because how the gradients are calculated this is explored in detail in Section 3.1.4.

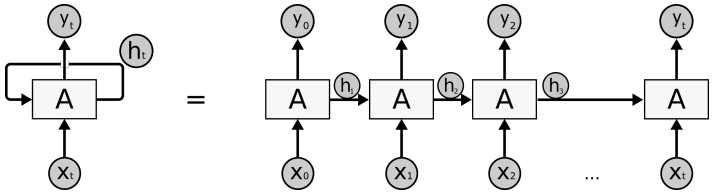


Figure 3.2: Unrolled representation of a recurrent neuron [82]

Figure 3.2, is the unrolled representation of a recurrent neuron. In the loop notation (left) x_t denotes the t^{th} input from the input sequence and h_t denotes the t^{th} hidden state which will be multiplied with W_h, which is the weights to multiply the hidden state before using it as an input for the next time step. y_t is output from the neuron A that presumably will be passed to the next layer. Since the input is a sequence with length t the unrolled notation helps to visualize how the sequence is consumed by the neuron A and the intermediate steps. All of these steps can have a bias term as well in addition to the weights but for the visual simplicity they are omitted in this diagram.

3.1.3 Back Propagation Through Time (BPTT)

BPTT calculation is needed for changing the weights according to the gradient of the error term in order to minimize the error (i.e. value of the error function). For a sequence with length T the total error can be denoted as follows where E is the error function that calculates the difference between the actual value y and the predicted value $\hat{y}$. W is the weight matrix. If optimization is performed with the gradient descent algorithm, the right side will be used to update the weights with learning rate α which is a scalar hyper-parameter.

$$\frac{\partial E}{\partial W} = \sum_{n=1}^{T} \frac{\partial E_n}{\partial W}, \quad W \leftarrow W - \alpha \frac{\partial E}{\partial W}. \tag{3.1}$$

For the neuron in Figure 3.2, for each time step t, the hidden state can be represented like $h_t = f(h_{t-1}, x_t)$. For example, f can be a nonlinear activation function such as $\tanh$, W_h would represent the weights of the hidden state input from the previous time step, W_x would represent the weights for the input sequence and b_h would represent the bias for the hidden state. Putting those together, the hidden state at the time step t can be written as $h_t = \tanh(W_h h_{t-1} + W_x x_t + b_h)$. Details like these depend on the internal architecture of each individual cell and different applications of each of these will yield a different equation for the back propagation calculation. Similarly for the output to the next layer is represented

by $y_t = f(h_t)$, in this case f can be a different nonlinear activation function like $\mathrm{softmax}$ which will yield $y_t = \mathrm{softmax}(W_y h_t + b_y)$, where W_y represents weights for the output and b_y is the bias for the output. In order to train the network (minimize the total error) by adjusting the weights of the network, it is required to "learn" the direction which the weights will be updated. This direction information can be obtained by using the gradient matrix. This gradient is calculated from the output layer to the input layer throughout the network hence the name back propagation. In the given example, the gradient between layers can be calculated with the following generic equation at the k^{th} step with the help of the chain rule as follows,

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial y_k}\frac{\partial y_k}{\partial h_k}\cdots\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W} \tag{3.2}$$

as k grows larger the multiplicative operation gets more terms and gradient calculation is going to have the potential to vanish (converge to zero) since the derivative of $\tanh$ is always smaller than the value 1.

$$y = \frac{d}{dx}\tanh(x) = 1 - \tanh^2(x), \quad y \in [0, 1]. \tag{3.3}$$

On the other hand, a more subtle problem can also arise, and the W_h can potentially overpower the derivative of the $\tanh$ which is always bounded. This problem is named as the exploding gradient problem. Possible solutions to the vanishing or exploding gradient problems are explored in the Section 3.1.4.

3.1.4 Vanishing or Exploding Gradient Problem

When the calculation requires multiplications even for a scalar x, the following number will grow rapidly x^n when $x > 1$ as n gets larger and larger conversely the same number will get close to 0 when $|x| < 1$. This is also true for individual scalars in the matrices used in the optimization (training) of the neural networks like weight matrices and gradient matrices. It is shown [6] that exploding and vanishing gradients pose a serious problem for RNNs. There are solution methods proposed in order to remedy the problem. For the exploding gradient problem, clipping and limiting the maximum allowed gradient magnitude to a constant, can be a straightforward solution [84]. Another approach is to apply the BPTT algorithm to a subset of time steps instead of all the time steps which is called as truncated back propagation through time and there are findings [80] suggesting that the truncated version is also capable of performing as accurately.

For the problem of vanishing gradients, proposed solutions are to use the Long Short Term Memory (LSTM) [45] cells or Gated Recurrent Unit (GRU) [20] cells in the recurrent network. Changing the cell architecture allows the partial derivatives in the gradient matrices to be calculated additively instead of multiplicatively. In addition, both of these specific cell architectures allow to remember the previous cell state or forget it with specific gates (internal connections) and these connections are the mechanism allowing to remember long term

dependencies or forget the changing (no longer valid) trends in the context of the sequential input. Either type of cell has shown usability for time series forecasting [34]. In addition, there are different variants of LSTM cells like Time Aware LSTM [5] or Bidirectional LSTM [4]. Variants of the GRU cell are also available [30, 112].

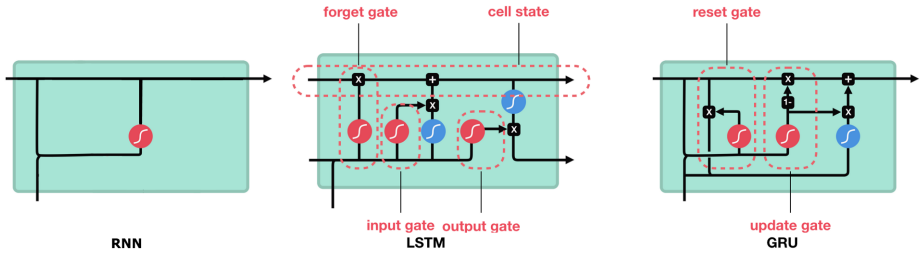


Figure 3.3: Different internal cell architectures with point-wise addition, multiplication and sigmoid (red), tanh (blue) activations [18]

Another approach is to modify the optimization steps instead of the architecture of the network by introducing hessian-free approach and there are findings [76, 77] showing this approach is also capable of producing accurate results. Optimization problems for neural networks are not straightforward and in addition to the problems described in this section there are still additional research requiring to improve the optimization stage such as handling the local optimal points which require high amounts of computational resources [22].

3.1.5 Cold Start Problem

Cold start problem in the context of this research is that the generated model is expected to generalize for newly obtained data without retraining. This generalization requires that when similar series are clustered and modeled together when another unseen series is added to the inference set without introducing it in the training set, the monolithic model should be able to generate fairly accurate forecasts for that series as well. Performance here can be measured by leave one out validation method that is if the number of training series is n, train the series with $n - 1$ series and obtain forecasts for the one left out and repeat this as many times as required in order to determine the network's performance for the cold start problem. This study aims to show results that information sharing between multiple series during training will alleviate the cold start problem as further series is added to the data set when working on the forecasts which would create a great practicality for this approach to be adopted.

3.2 Algorithms used

The progression of the study required the adoption of different modeling approaches for different stages of the research. For example for imputation the more classical methods are experimented on like ARIMA, ETS and STL. This is because compared to neural networks

the required computational power is significantly lower. More explainable and parsimonious models are preferred for this stage, it is expected that modeling the series individually will capture the behaviour of each series better since each series can have different average magnitudes and different seasonal behaviour. Capturing the original for imputation is critical overall quality of the generated model will be directly impacted by the training data set and the imputation step aims to improve the quality of the training data. In addition classical models are well understood and well studied in the literature and results obtained from them are more interpretable. For the actual forecasting the deep learning methods are utilized and introduction of all of the methods from data preparation to modeling experiments are introduced in this chapter with necessary computational background.

3.2.1 Dynamic Time Warping (DTW)

Dynamic time warping is a distance metric between two series $x = (x_0, \ldots, x_{n-1})$ and $y = (y_0, \ldots, y_{m-1})$. In order to calculate the DTW score the first step is to align (warp) the series. Aim of this warping is to match the similar behavior between two series and calculate the similarity accordingly. The naive approach to calculate the difference between two series can be achieved by the Euclidean distance between two series. This can be formulated as follows

$$ED(x, y) = \sqrt{\sum (x_i - y_j)^2} \tag{3.4}$$

where ED denotes the Euclidean distance between two series x and y where $0 \leq i < n$ and $0 \leq j < m$. In case of the Euclidean distance metric the two series are aligned by their timestamps and the score of dissimilarity E is calculated accordingly. This distance metric can work very well if the compared series are behaving synchronously. This assumption may not hold even if the series come from similar origins like both are the records of demand for retail stores in the physical world of the same brand. Even though both series would have some periodic peaks, these peaks might not occur at the same time in the both series simultaneously. In order to have a more robust distance metric that accounts for these differences, instead of alignment of the time stamps it is possible to align the behaviour of the series before the distance between two series is computed. DTW as a distance metric solves this exact problem with two stages, first the correct alignment is found by solving the optimization problem [89] as follows

$$DTW(x, y) = \min_{\pi} \sqrt{\sum_{(i,j) \in \pi} ED(x_i, y_j)^2} \tag{3.5}$$

the minimum value or the optimal warping path can be found with the following simple steps to find the optimal warping path π

1. calculate the Euclidean distance between x_0 and all points $y_0, \ldots, y_m$ and obtain the minimum

2. repeat 1 for each point remaining in the series x (each x_i where $i \in [1, n]$)

3. repeat 1 and 2 for the series y

4. the path π is generated by selecting the minimum error generated in 1 froam each possible i and j indices between of two series x and y

the steps described above would require exponentially growing number of calculations when the number of the values in the series increases, in order to make the search for the optimal solution there are some constraints introduced for limiting the search space. Calculation of the DTW becomes solution of equation 3.5 such that each path $\pi = [\pi_0, \ldots, \pi_K]$ satisfies the following constraints:

- $\pi_k = (i_k, j_k)$ where $0 \leq i_k < n$ and $0 \leq i_j < m$
- $\pi_0 = (0, 0)$ and $\pi_K = (n-1, m-1)$
- $\pi_k = (i_k, j_k)$ and $\pi_{k-1} = (i_{k-1}, j_{k-1})$ follows for all $k > 0$
    * $i_{k-1} \leq i_k \leq i_{k+1}$
    * $j_{k-1} \leq j_k \leq j_{k+1}$

the reason behind the constraints for the optimization is as follows:

- $i_{k-1} \leq i_k$ and $j_{k-1} \leq j_k$: guarantees the monotonicity by restricting the alignment path to not to go back in time stamps or the redo the calculation the previously visited items again
- $i_k \leq i_{k+1}+1$ and $j_k \leq j_{k+1}+1$: guarantees the continuity by restricting the alignment path to not to skip elements in order to not to save the important characteristics of the series
- $\pi_0 = (0, 0)$ and $\pi_K = (n-1, m-1)$: guarantees that all of the both series is included in the calculation by setting the boundary conditions of each series
- although omitted in this study another constraint can be defined as the warping window (Sakoe-Chiba band) as $|i_k - j_k| < r$ where $r > 0$ and r denotes the window length in order to make sure that the warping does not wander off and match two points of the series with too much time index difference between them. There are different variants of this same strategy such as the Itakura parallelogram.

Combining all the information presented, the DTW is calculated as a similarity value between two series that are temporally aligned. Temporal alignment tries to align the movement (behaviour) of the series, this can be considered of minimizing the Euclidean distance between resampled series for a given pair of time series with cross-similarity matrix. Implementation is available in the $tslearn$ package [102] in Python and this implementation is used for this research. Additionally, there are different computational variations [1, 98] of the DTW algorithm to make computations faster, further decreasing the time complexity of the computations even in sub-quadratic time [38].

3.2.2 Silhouette Score

The silhouette score [88] is used to measure the fitness of the clusters for a clustering work setting. It is formulated by Equation 3.6 for a single sample s and for samples in the same cluster it can be calculated as the mean value of each member of the cluster. The value is bounded between -1 for unsuccessful clustering and $+1$ for ideal clustering. The silhouette score around 0 means that the clusters created are not sufficiently distinct and there is a great amount of overlap between the clusters.

$$s = \frac{b - a}{\max(a, b)} \tag{3.6}$$

where

a: The mean distance between a sample and all other points in the same class.

b: The mean distance between a sample and all other points in the next nearest cluster.

Silhouette score is applicable when the ground truth labels for each sample and class is not known and there are different algorithms applicable for the similar cases such as Calinski-Harabasz Index [15], Davies-Bouldin Index [25] for evaluating the performance of the clustering.

3.2.3 Exponential Smoothing (ETS)

The method of exponential smoothing aims to model the series by using the weighted averages of the past values in order to determine next values. The weights in this averaging process decay exponentially in order to give the least importance to the oldest observation whereas the most recent observation has the most amount of weight [47, 110]. The same idea can also be implemented as a state space method [50] in order to generate forecasts. The state space method includes all of the components (i.e. Error, Trend, Seasonality) together [51]. It can be formulated by two different parameterizations as follows:

For additive seasonality `ETS(A,A,A)`

$$
\begin{aligned}
y_t &= \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t \\
\ell_t &= \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t \\
b_t &= b_{t-1} + \beta\varepsilon_t \\
s_t &= s_{t-m} + \gamma\varepsilon_t,
\end{aligned}
$$

and for multiplicative seasonality `ETS(A,A,M)`

$$
\begin{aligned}
y_t &= (\ell_{t-1} + b_{t-1})s_{t-m} + \varepsilon_t \\
\ell_t &= \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t/s_{t-m} \\
b_t &= b_{t-1} + \beta\varepsilon_t/s_{t-m} \\
s_t &= s_{t-m} + \gamma\varepsilon_t/(\ell_{t-1} + b_{t-1})
\end{aligned}
$$

where

- ℓ_t : denotes the level component and α its smoothing parameter
- b_t : denotes the trend component and β its smoothing parameter
- s_t : denotes the seasonal component and γ its smoothing parameter
- m : is the frequency (e.g. 4 for quarter data, 12 for monthly data)
- $\varepsilon_t \sim \text{NID}(0, \sigma^2)$: denotes the errors are normally and independently distributed with mean 0 and variance σ^2

The ETS model family is experimented on with the available data set in order to determine the best fitting model components for error, trend and seasonal components and the summary of the fitted models the Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC) are used as well as the visual examination of the plots and best performing method is observed to be the one that includes all of the terms. Among all parameters included models, the one with multiplicative seasonality by the measure of AIC and BIC. Additive seasonality for this method means that seasonality is fairly constant throughout the series, but the multiplicative seasonal component is able to capture seasonal behaviour changing together with the magnitudes of the values in the series. In this study all of the measurements are strictly positive integers since they denote the number of people thus multiplicative seasonality does not cause a computational problem.

3.2.4 DeepAR

Unlike conventional methods, the DeepAR [91] algorithm does not require the modelled data to uphold certain assumptions such as stationarity of the series, residuals to have Gaussian

distribution or homoscedasticity. Moreover, the model is capable of generating probabilistic forecasts which would provide another layer of practicality. This is because even though generated from the same measurements each individual series can have their differences in magnitude or different sensitivity to errors in prediction. If the demand is going to be predicted for a location where possibility of being under-stocked or understaffed is not affordable by the business then the predictions created for that series should accommodate this need for real life applications. This requirement can be addressed by the probabilistic forecasts generated by DeepAR. Upon completion of the model training, the model can generate different probabilistic predictions from different quantiles called PX where X denotes the probabilistic quantile. For example, $P90$ predictions obtained from the model imply that 90% of the time the actual value will be less than the forecasted value. By using different quantile values it is possible to obtain different forecasts with varying levels of sensitivity depending on the application. In addition to the actual measurements recorded in the series which are called as *target* data set, this algorithm is also capable of ingesting information from *related* and *metadata* data when provided for each individual series in the data set. The *target* data set holds the records for the actual time series with unique identifier for the series, the time stamp and the actual measurement value to be learnt and forcasted. The *metadata* data set describes the additional features for the uniquely identified series for example if the training set contains demand records throughout Türkiye, which province that the specific series is collected from can be provided to the model via this data set. The *related* data set holds the information about the time period for the uniquely identified series in the set, for example special days or holidays, a series in the data set can have different holidays depending on the location. The three data sets used for this study are further detailed in Section 4.4 with actual examples from the data set used. The $GluonTS$ package [2] of Python provides the implementation of DeepAR.

Different from vanilla LSTM networks that produces point forecasts for the given time series instead what DeepAR does is for given inputs of the series the network outputs the parameters of the probability distribution. For example, the Gaussian function in this application is parameterized by its mean μ and its standard deviation σ as follows

$$\begin{aligned}
\ell_G(z|\mu,\sigma) &= (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(-(z-\mu)^2/(2\sigma^2)\right) \\
\mu(\mathbf{h}_{i,t}) &= \mathbf{w}_\mu^T \mathbf{h}_{i,t} + b_\mu \\
\sigma(\mathbf{h}_{i,t}) &= \log\left(1 + \exp\left(\mathbf{w}_\sigma^T \mathbf{h}_{i,t} + b_\sigma\right)\right)
\end{aligned}$$

where mean is calculated with an affine function using the output of the network and the standard deviation is obtained by applying an affine transformation followed by a $\mathrm{softplus}$ activation on the calculation of the $\sigma(\mathbf{h}_{i,t})$. The function is defined as $\mathrm{softplus}(x) = \log(1 + e^x)$ in order to ensure $\sigma > 0$. The parameters used are the output $\mathbf{h}_{i,t}$, the weights to be learnt for the mean and standard deviation w_μ and w_σ respectively. Similarly, the biases b_μ and b_σ are the biases for the mean and standard deviation respectively.

Additionally another example is the negative binomial distribution which is a popular choice when the forecasted measure is positive integers [99, 17] similar to this research. The discrete

random variables such as the number of people are best predicted with a negative binomial distribution. This distribution is parameterized by its mean $\mu \in \mathbb{R}^+$ and a shape parameter $\alpha \in \mathbb{R}^+$ as follows

$$\ell_{NB}(z|\mu,\sigma) = \frac{\Gamma(z+\frac{1}{\alpha})}{\Gamma(z+1)\Gamma(\frac{1}{\alpha})}\left(\frac{1}{1+\alpha\mu}\right)^{\frac{1}{\alpha}}\left(\frac{\alpha\mu}{1+\alpha\mu}\right)^{z} \tag{3.7}$$
$$\mu(\mathbf{h}_{i,t}) = \log\left(1+\exp\left(\mathbf{w}_{\mu}^{T}\mathbf{h}_{i,t}+b_{\mu}\right)\right)$$
$$\alpha(\mathbf{h}_{i,t}) = \log\left(1+\exp\left(\mathbf{w}_{\alpha}^{T}\mathbf{h}_{i,t}+b_{\alpha}\right)\right)$$

in order to ensure that both parameters are greater than zero, the outputs from the network's fully connected layer are passed to the softplus function. Here α is called as the shape parameter for negative binomial distribution and it adjusts the variance relative to the mean as follows $\text{Var}[z] = \mu + \mu^2\alpha$. In this research, the negative binomial distribution is used as the probability distribution for the likelihood estimation since all of the series in this study are from a distribution where each measurement is a positive integer. These likelihood functions are shown in Figure 3.4 between $\mathbf{h}_{i,t-1}$ and $z_{i,t-1}$ on the left side during training and similarly at the same level for the inference shown on the right.

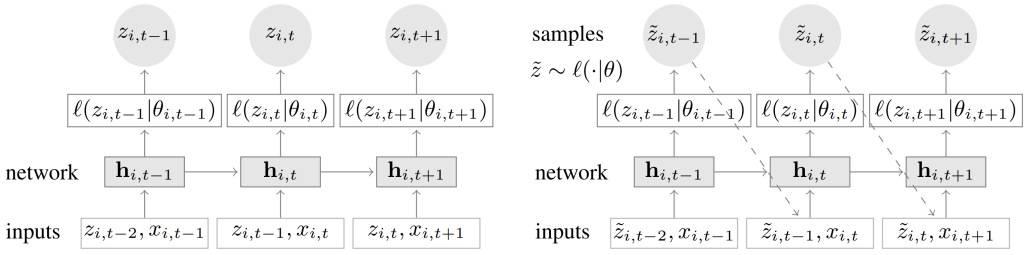


Figure 3.4: Architecture of DeepAR [91]

In Figure 3.4 the left side represents the training part of the model. At each time step t, the covariates $x_{i,t}$, the value of interest of the series of previous step $z_{i,t-1}$ and the output from the network at the previous time step $\mathbf{h}_{i,t-1}$ are provided to the network as inputs (bottom line of the left part). From these inputs the network creates the output $\mathbf{h}_{i,t} = h(\mathbf{h}_{i,t-1}, z_{i,t-1}, x_{i,t}, \Theta)$ and this output is used to compute the parameters $\theta_{i,t} = \theta(\mathbf{h}_{i,t}, \Theta)$ of the likelihood $\ell(z|\theta)$ which is used for training the model parameters. In this architecture h is a function implemented by a multi-layer recurrent neural network with LSTM cells, i is the index or the unique identifier of the series, z is the actual measurements we want to predict, x is the covariates (a matrix with all of the supplementary data to the series i.e. `related` and `metadata` data), the likelihood is the negative binomial for this research denoted in Equation 3.7 and the parameters of this likelihood is given by the function θ and lastly Θ is the model parameters. DeepAR follows the autoregressive recurrent network architecture [40, 101] in order to model the conditional probability of $P(\mathbf{z}_{i,t_{0:T}}|\mathbf{z}_{i,1:t_0-1}, \mathbf{x}_{i,1:T})$. This is a sequence to sequence model left side representing the encoder and the right side representing the decoder in the Figure 3.4. The features of the model include the standardized covariates with zero mean and unit variance and it is possible to include the day-of-the-week feature by adding an additional feature

with values ranging from 1 to 7 for each of the weekdays. This is expected to help the model to capture the weekly seasonality in the daily data provided for this research and it is possible to increase the number of features like week-of-the-year or day-of-the-month for different expected seasonality periods.

Training of the model, optimization of the model parameters Θ which consists of h and θ, is achieved by maximizing the log-likelihood in the Equation 3.8 as follows for N different time series and T time steps

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{t=t_0}^{T} \log \ell(z_{i,t}|\theta(\mathbf{h}_{i,t})) \tag{3.8}$$

and in order to handle the different of magnitude problem between different series the model uses a scale factor v_i for each individual series as follows

$$v_i = 1 + \frac{1}{t_0} \sum_{1}^{t_0} z_{i,t} \tag{3.9}$$

this scaling factor is utilised in the training of the network since the activation functions used in the network have a finite range and values with different magnitudes would reach the possible maximum or the minimum of that range depending on their magnitude in order to prevent such a case model uses the following

$$\mu = v_i \log\left(1 + \exp(o_\mu)\right)$$
$$\alpha = \log\left(1 + \exp(o_\alpha)\right)/\sqrt{v_i}$$

where o_μ and o_α are the outputs from the network for the mean and variance for each individual series and v_i is the average value of the series i. For the specific data set used in this research excluding the outliers had gained special importance because of this scale factor since the data set is discovered to have outliers causing very heavy skew problems as described in Section 4.1.1

For prediction, the decoder part on the right side of Figure 3.4 the dashed line represents the generated sample $\tilde{z}_{i,t-1}$ to be passed as an input for the next step in order to obtain $\tilde{z}_{i,t}$ by generating one step forecasts many times creating a sequential output of the desired length immediately starting from the end of the training data set. The samples $\tilde{z}_{i,t}$ are obtained from the θ distribution which in this case is a negative binomial distribution thus $\tilde{z} \sim \ell(\cdot|\theta)$ and the mechanism providing the `PX` predictor mentioned early in this chapter is realized via this mechanism since each $\tilde{z}_{i,t}$ can be sampled form the distribution multiple times. So the z values are the actual measurements existing in the training data set whereas $\tilde{z}$ values are sampled from the likelihood function ℓ learnt by the network and used as one step information for multi step forecasting scenarios hence the auto-regression aspect of this model.

CHAPTER 4

DATA ANALYSIS

The data contains individually recorded series for different locations of an apparel brand throughout Türkiye where each individual series correspond to a specific store. There are 122 different stores in the data set and outlier analysis and imputation studies are performed for each series individually. The collected data does contain possible sensor errors and over-counts (outliers) as realized in the exploratory data analysis step. Since the data set required to be cleaned from the outliers and there is no indicator for each sample is a correct measurement or an outlier an unsupervised cleaning algorithm is required. Besides from detecting the possible outliers this cleaning algorithm is expected to replace the possible outlier sample with a normal sample. Performance of the algorithm is measured by a control data set. In this control data set some values are replaced with artificially generated outliers and these are filtered by the algorithm. After that filtered result is compared with the clean version of the control data set and similarity of these two series is used to measure the success of the cleaning algorithm and presented in the relevant chapter. Another requirement for the cleaning algorithm is to identify and replace values in the series considering that different series behave differently. The difference in behaviour of the series can be the magnitude of the recorded values, seasonal repetition patterns or different tendencies in the level. The intended result is to develop a common cleaning algorithm to clean the entire data set without requiring the effort of individually fine-tuning the cleaning per series.

Besides from the erroneous measurements the data set is also observed to have missing values. The reason for the missing values can be the sensor fault, communication fault or the actual physical location not to be accessible in the time of the measurement. These missing values are imputed in order to increase the model performance and to have a data set with as many samples as possible. Performance of the imputation is measured by a control data set by deleting random data points in the series for imputation and calculating the difference between the imputed value and the actual value. This step is completed for both single point imputation and consecutive imputation and results are presented. Confirmation of the model performance improvement is also presented by comparing a model with imputation and without imputation.

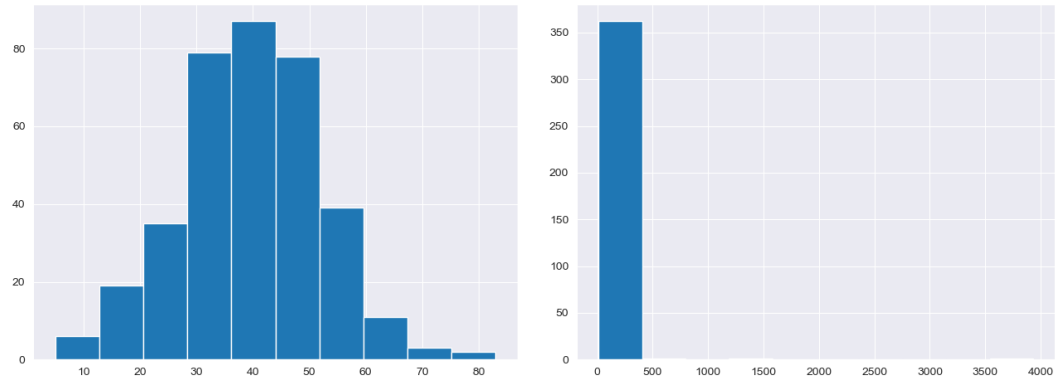
Cleaning and imputation operations complete the preprocessing step for this study and in order not to introduce too much artificially generated data trickle into the modeling stage, the amount of cleaned/imputed data allowed per series is limited to 25% of the series.

4.1 Exploratory Data Analysis

The exploration of the training data set is explained in this chapter. The importance of clustering and the need for treatments for outliers and missing data is detected via the analysis performed here and the next chapter presents the methodology for those steps. The aim of this analysis is to detect where the quality of the data can be improved in order to achieve better overall accuracy for the forecasting system. The descriptive statistics of the set are very helpful in understanding the values in the series such as the magnitude of the different series, the clustering analysis is performed in Section 4.1.2 to observe that the nature of the series in the set can be clustered together appropriately and finally the seasonality analysis is performed in order to see that a repetitive nature in the series is present and common for the training data set.

4.1.1 Descriptive Statistics

The descriptive statistics of the series are inspected for each of the series in order to obtain a better idea about the data before modeling. The distribution of the values in the series have a slight skew with fairly amount of clean data can be seen in the left side of the Figure 4.1 the presence of an outlier even with one sample can dramatically change that and prevents the observation of the values in the series. Observations like this example should be eliminated by data cleaning which is explained in detail in Section 4.2.



(a) Without an outlier (b) With an outlier

Figure 4.1: Effect of the outlier presence on the histograms

After outliers are cleaned from the data set using the method in the Section 4.2 in order to observe the magnitude difference among different series in the data set the box-plot is used a random subset of the training set is shown in Figure 4.2. It can be clearly seen that the series in the data set varies a great deal in terms of magnitude for the recorded values and this observation is key when the series are compared by their movement in Section 4.1.2 and this comparison is made after the values are normalized. Additionally the scale factor explained in the Section 3.2.4 for DeepAR algorithm show suitability for a set containing many series

with different magnitudes and pre-processing steps did not include a normalization step for the values to be passed on to the model before training because of this.

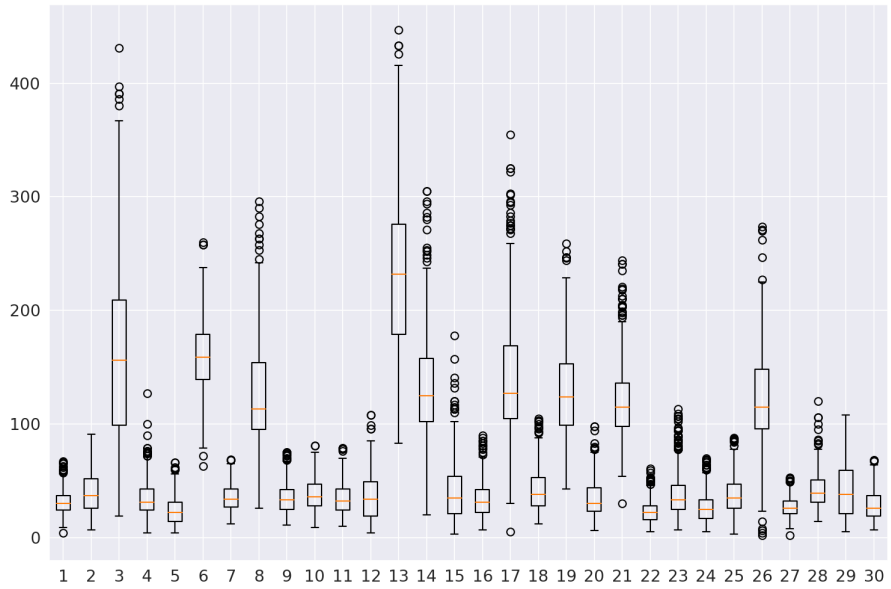


Figure 4.2: Box plot of randomly selected thirty series

4.1.2 Clustering Analysis

To remove the magnitude difference between different series, all the values of the series are normalized to zero mean and unit variance first, this step is especially important in order to measure the similarity of behaviour between series, the clustering step is performed with the $TimeSeriesKMeans$ [102] algorithm with Dynamic Time Warping (DTW) Section 3.2.1 as distance metric. Different numbers of clusters are tried and in order to find the optimum cluster number the inertia of the series (calculated again with DTW) and silhouette score of the different number of clusters are examined. Available in Python, $TimeSeriesKMeans$ [102] algorithm is used in this research and implementation follows the idea of Majorize-Minimize Mean Algorithm [92]. This implementation can also work with missing values by putting `NaN` values as placeholders during calculation which makes the algorithm very convenient to use before any imputation is performed. The principle of the algorithm is as follows

Let $\mathcal{D}$ be a set of sequences, in order to find the barycenter (Fréchet mean), μ, DTW Barycenter Averaging (DBA) algorithm tries to solve the following optimization problem

$$\min_{\mu} \sum_{x \in \mathcal{D}} DTW(\mu, x)^2. \tag{4.1}$$

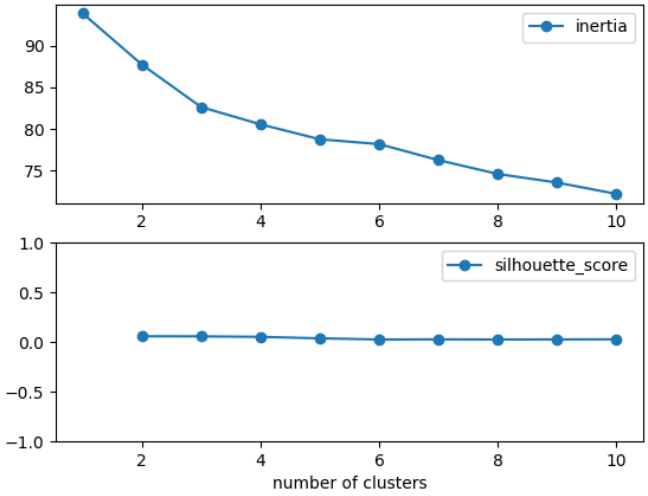


Figure 4.3: Inertia and Silhouette score with different values of K

After the barycenter is calculated by Equation 4.1 [86] method minimizes sum of squared DTW distance between the barycenter (the average of the cluster) and the other series in the cluster. This is achieved iteratively, an arbitrary average series is calculated and in each iteration, algorithm computes the DTW between each individual series and the temporarily averaged series and improving the performance of the clusters by calculating and selecting the minimum DTW for the given alternative clusters for the series [24]. DBA is a costly algorithm, in terms of iteration number (number of input series) it has linear time complexity but time complexity is quadratic in terms of the length of the each sequence [24]. It is shown [71] that the average sequence generated by DBA does not depend on the order of the input series, the initial selection at the beginning of the clustering is dependent on the selected sequence to be clustered on.

Number of clusters, which is K, is determined by examination of the inertia and silhouette score metrics displayed on Figure 4.4. The elbow method is generally used in the examination of the inertia plot, this method requires a significant change of slope but that is not present in the inertia score and the decrease of the score is near linear as can be seen in Figure 4.3. As per the silhouette score, higher is preferable but it is maximized at the minimum allowed K, which is 2, and the silhouette score decreases as the number of clusters grows this means a lower amount of clusters is preferable.

In addition to examination of inertia and silhouette score a visual analysis can be performed on the normalized series in order to observe how much of the behaviour of the series are overlapping. In Figure 4.4 the series are plotted on top of each other with a gradient colouring scheme so the darker colour means a higher overlap between different series.

Further examination of Figure 4.5 shows that when the series is attempted to be clustered into two groups the resulting clustered groups are very similar in nature but divided on the premise that K is selected as two.

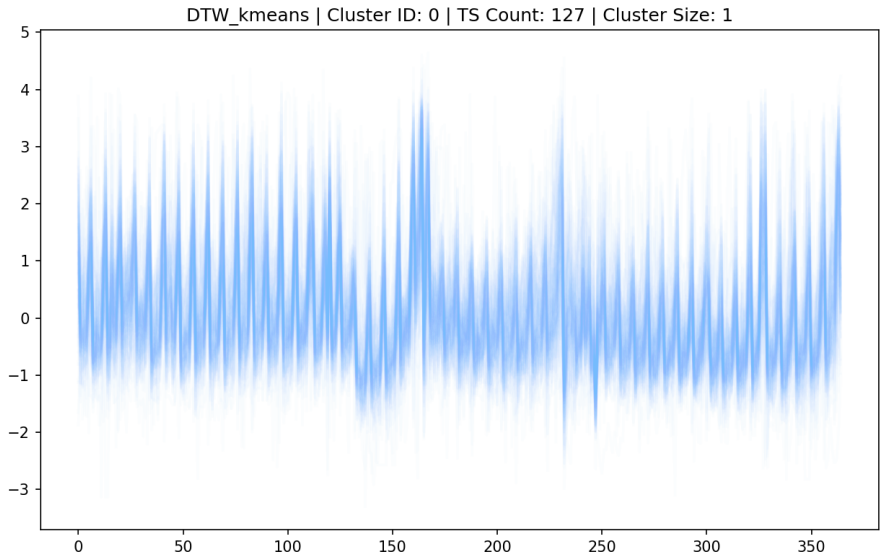


Figure 4.4: Series clustered when $K = 1$

Analysis made in this section is to confirm the assumption of generating a common model by providing the series together can increase the accuracy of the forecasting model because the movement of the series is similar in nature. The findings described in this chapter provide a solid ground for this assumption since they show that the series is extremely prone to be clustered as a singular group and dividing the number of clusters more does not improve the inertia or silhouette score and this is one of the key observations confirming that the data are appropriately grouped together for the global model from which the forecasts will be obtained. The result of this examination is that there is no need to cluster the given data set with more than one group.

Additionally the averaged series is considered as a good candidate as the imputation source for the missing values in the series. This would be achieved by filling in the missing values with the averaged series from the result of the barycenter of the cluster. Further exploration of this idea is determined to be not applicable and the reason can be seen in Figure 4.6, especially from the x_{75} to x_{100} indices in the series. It can be seen that there are sudden constant sections in the barycenter, this plateau type of points are result of the noise around the darker points of the series. These noisy points are result of slight phase shifts between different series and average of this phase shifted signals converges to their middle. This is similar to the destructive interference of sine and cosine waves when they are summed. This observation showed that idea of imputation from the barycenter would not be a well generalizing application for the data set of this research and alternative imputation methods are investigated for the study and presented in Section 4.3.

To conclude, the behavioural resemblance of the series assigned to a single cluster can be verified with the silhouette scores for different cluster sizes. The silhouette scores of different

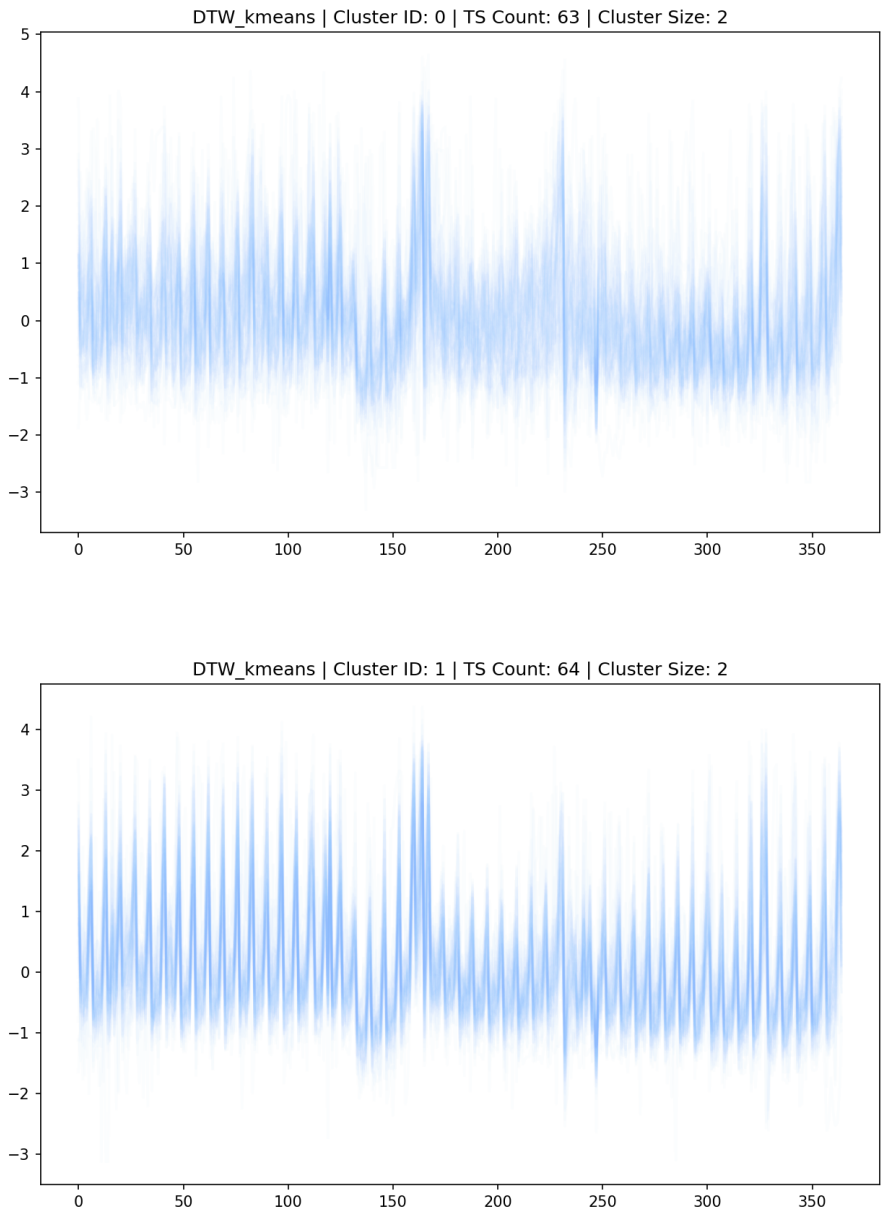


Figure 4.5: Series clustered when $K = 2$

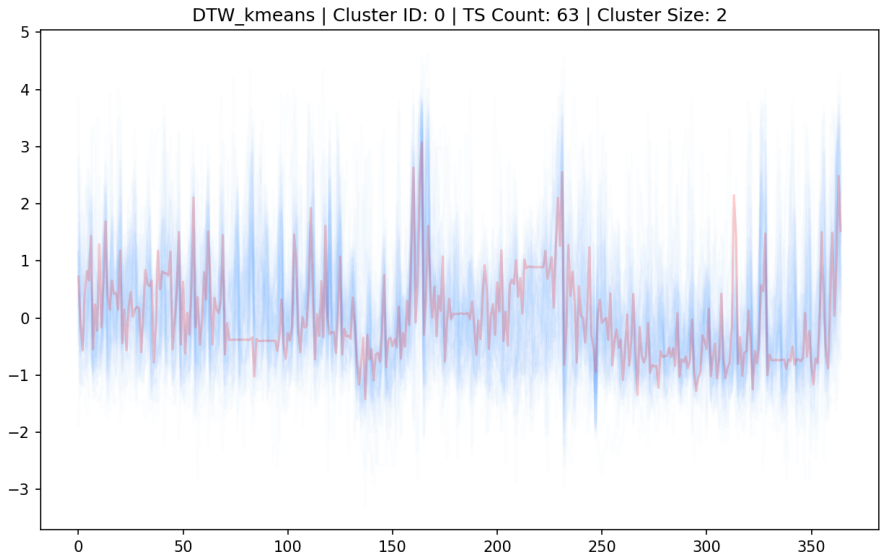


Figure 4.6: Series clustered when $K = 2$, barycenter (red)

clusters are near zero as can be seen in Figure 4.3 indicated that when the given data set is clustered the generated clusters have significant overlap. This suggests that in data sets where silhouette score is near zero may not benefit from multiple clusters. The silhouette score obtained from the experiment with different cluster sizes yielded near 0 values between 0.03 and 0.06 thus the study is continued with a single cluster for the modeling phase.

4.1.3 Seasonality Analysis

The seasonal behaviour of the series is a very important factor in order to determine the future values for the series when forecasting is performed also one of the hyperparameters for the DeepAR model is `context_length` which should be selected as a multiple of a seasonal period in the series. When the series are expected to have seasonal behaviour this can be observed by the examination of the ACF/PACF plots as well as the plot of the series themselves. This proposition can be further confirmed by Fourier's series analysis which is explained in this chapter as well.

Figure 4.7 shows examples from the cleaned series and their ACF/PACF plots. It can be observed that some series have indications of very strong seasonality (a) there exists also series that a clear seasonality cannot be observed (b) it can be concluded that the series observed here is not stationary. The conclusion is that the series in the data set are not homogeneous in terms of magnitude of the data, skewness of the data and seasonal behaviour and the model is expected to learn this and produce the forecasts accordingly.

The STL decomposition $statsmodels$ package [94] is used to obtain the seasonal component of each of the series, and these seasonal components are passed through to the $scipy$ package's

(a) Weekly seasonality observed

(b) Seasonality cannot be observed

Figure 4.7: ACF PACF correlograms for different series

[106] implementation of Fast Fourier Transform (FFT) [11] in order to obtain decomposition of the temporal frequency of the series individually. The most dominant frequencies is inspected for each of the series and the observed frequencies is converted to days by applying $1/\text{freq}$ and the obtained results were in the neighbourhood of the $7, 3.5, 2.3$ which are roughly equal to the $7/1, 7/2, 7/3$ respectively. This observation combined with the correlograms of the series suggests that there is a weekly seasonality present in the series.

4.2 Outlier Detection and Cleaning

What is considered normal for a specific series may be an outlier for a different location and the developed methods are expected to adjust accordingly when per series pre-processing is performed. In particular for outlier detection, treating the measurements in the series as a random sample and trying to find the outliers by only inspecting the magnitude of the recorded values is not a correct strategy because whether a data point is an outlier depends also on the position of the data point in the series. A contextual outlier means the position of the outlier is as important as the magnitude of the data point within the series. An example for this case is that a data point with magnitude X can be detected as an outlier depending on the location in the series. For a series with an increasing trend X can be an outlier towards the beginning of the series and it may not be flagged as an outlier towards the end of the series. So instead of checking the values of the series as a random bag of samples the movement of the series is tried to be captured here and the idea is to determine a measurement is an outlier or not using this information. The methods shown in Figure 4.8 are rolling window for mean, median and exponentially weighted mean where the window size is set to 14 or double the seasonal period of the series. Clearly closest matching method to the original series is clearly the exponentially weighted mean.

This information is combined with the assumption that values that deviated enormously from the exponentially weighted mean are potential outliers. The threshold selected for data points are allowed to deviate from this mean is determined with the Interquartile Range (IQR) of the rolling window. If a data point deviates above the 3*IQR, then that sample is deemed to be an outlier. When outliers are detected, the treatment has to be performed in such a way that there should not be a disturbance with the movement of the series. In order to address this concern, the outliers and the missing data are imputed together with the ETS modeling explained in the following chapter.

In Figure 4.9 it can be clearly observed that since the outliers in the sets are unbounded above there is a potential that the existence of the outlier skews the exponentially weighted moving average too much. This problem is prevented by calculating the IQR in a rolling window of 14 samples and the detected outlier is imputed with other missing values after the cleaning procedure is completed. 14 samples are selected in order to include at least two seasonal periods of the series. Weekly seasonality is observed in Section 4.1.3.

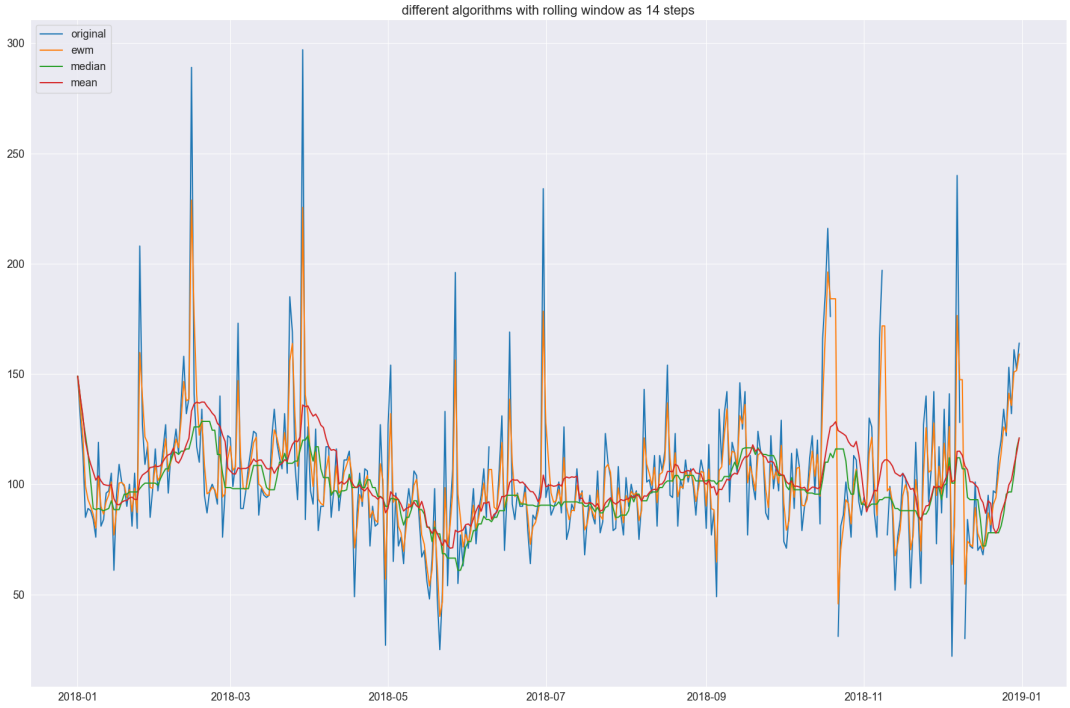


Figure 4.8: Different rolling windows for capturing the movement of the series

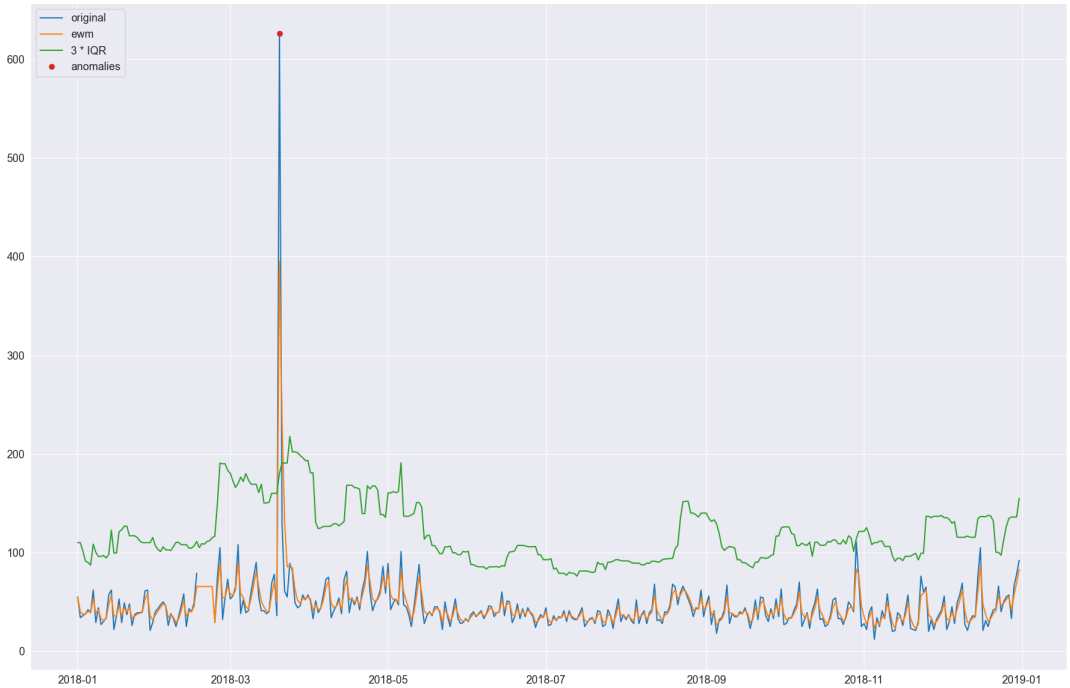


Figure 4.9: green: contextual outlier threshold, red: detected outlier

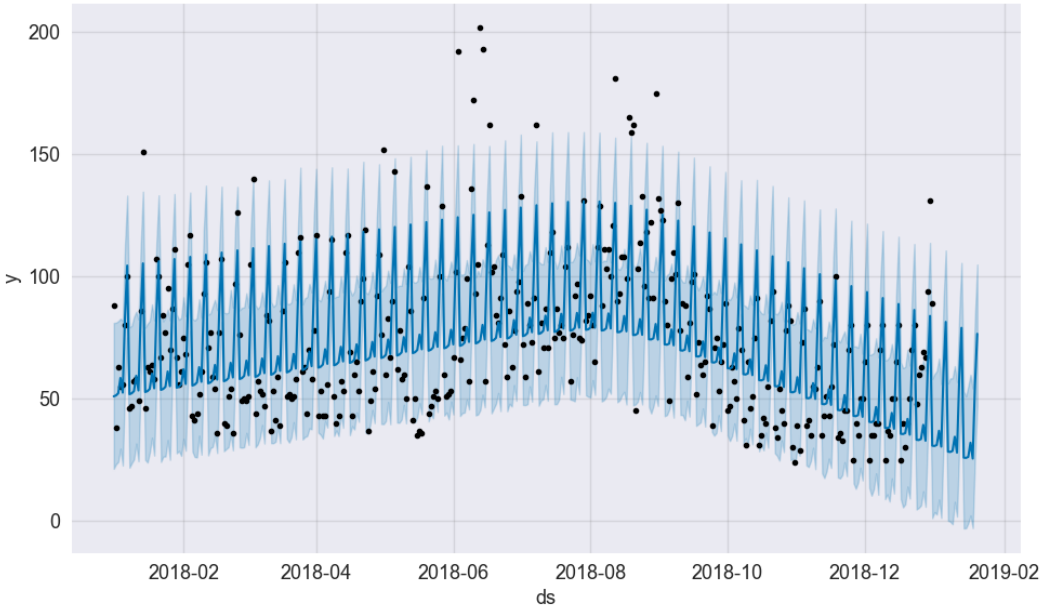


Figure 4.10: Prophet model fitted (blue) on an example series (black dots)

4.3 Missing Data Imputation

This imputation strategy is developed specifically for this study. The challenge with this specific data set is that simple imputation strategies cannot be used with this data set. The reason is that simple imputation strategies like interpolation, backfilling (last observation carried forward) or frontfilling (next observation carried backward) are most accurate when the series is fairly stationary around a trend (ups and downs are small in magnitude) and the missing values are points where immediate neighbours are available. In this data set there are multiple consecutive missing values present for multiple days and the simple imputation algorithms fail to capture the up and down movement of the series. For this reason, the special imputation method is developed for this study and is described in detail.

Since the imputation is going to be performed on each series individually the classical forecasting methodologies are candidates for this imputation except ARIMA since the series in the data set does not satisfy the modeling assumptions. On the other hand, Exponential Smoothing and Seasonal-Trend decomposition (STL) methods are the candidates for this imputation and Prophet model is also experimented on. Among these three models the best performer was the Exponential Smoothing method, in Figure 4.10 the Prophet model is fitted to the data and it can be observed that the model fails the capture the exact movement of the data for the majority of the series, and a similar lack of performance is observed with STL model as well. Additionally, after modeling is done with STL the errors (residuals) are tested for normality with the Shapiro-Wilk [95] test and the assumption of normality of the residuals is observed to be not fulfilled.

The exponential smoothing method applicable to the series is determined as the ETS(A,A,M)

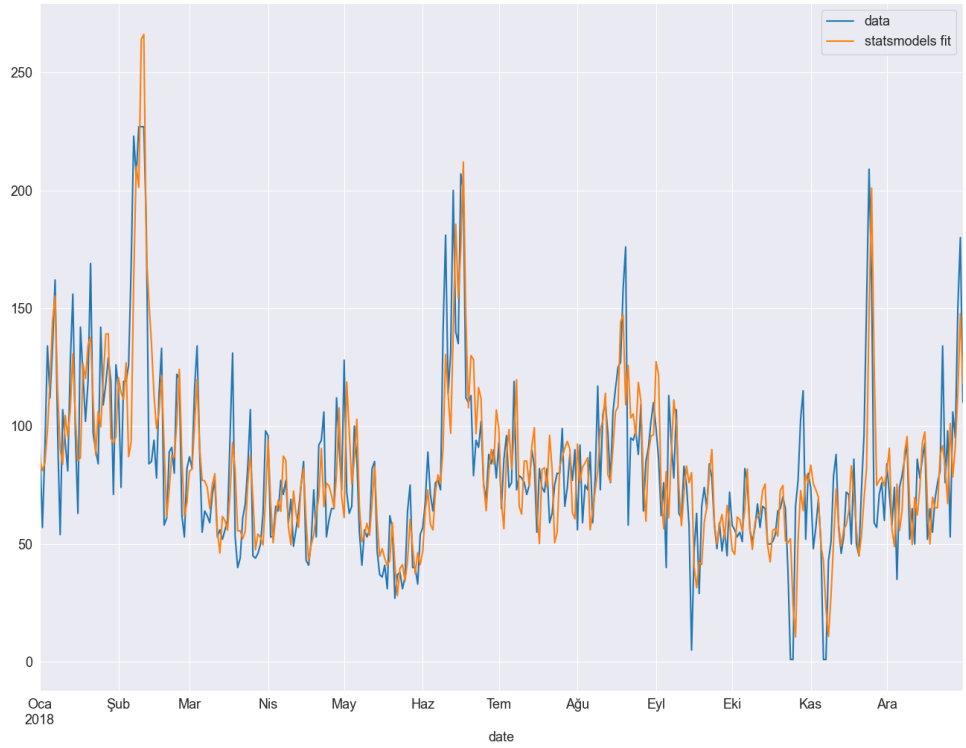


Figure 4.11: Example ETS(AAM) fit

model as explained in Section 3.2.3. An example ETS fit is presented in Figure 4.11 which shows that the model can capture the movement of the series successfully.

the imputation presented in Figure 4.12 is obtained with an iterative approach. The first two seasonal periods of the series (first fourteen days) are imputed with seasonal averages. The seasonal average is calculated with the non-missing measurements in the series and seven different averages are calculated with each week-of-day. This step is provided for the ETS method to capture the seasonal behaviour. After the initial step is completed the iterative approach is applied for each missing value where the previous data up to the missing value index is used to model the series and the missing values are imputed with the forecasts of the ETS model until the end of the series is reached.

In order to quantitatively measure the performance of the imputation method the data set is subjected to the controlled imputation experiment. The setup of the experiment is simple, randomly deleting 25% of the data from the series in the data set and imputeing the values with the ETS algorithm and calculating the error metrics. In this setting the proposed imputation method have yielded 30% Mean Absolute Percentage Error (MAPE) on average throughout all series in the data set. The magnitude difference amongst the series is to be considered when evaluating this metric. Another metric to be considered for this evaluation is the mean absolute error (MAE), which on average was 13 actual people for the entire data set.

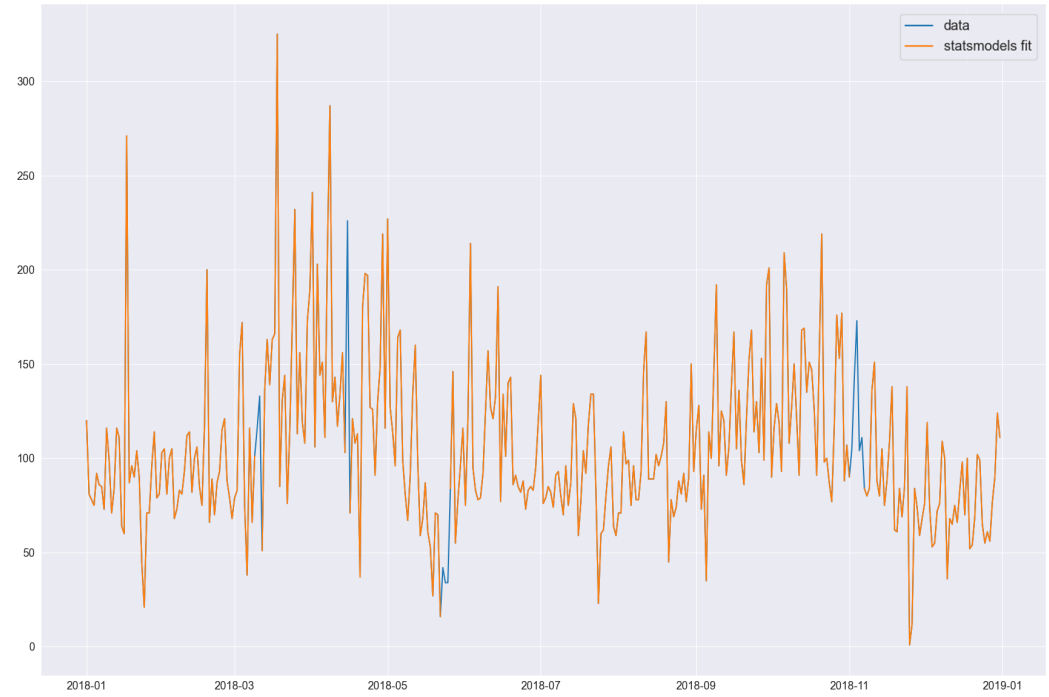


Figure 4.12: Multiple consecutive imputation with ETS(AAM)

4.4 Final Form and Example Data

Data preprocessing is concluded at this point of the research. From the entire data set there are only 122 series that fulfilled the visual inspection and 25% modification rule determined previously which states that no series is allowed if the amount of missing data and outliers is greater than 25% of the whole series which spans two full years would not be allowed into the modeling or the forecasting stage. These 122 series are distributed among Türkiye and collected from more than 30 provinces. The number of series per province for the five most crowded provinces of Türkiye is tabulated below in Table 4.1.

Table 4.1: Number of series collected from 5 most populated provinces in Türkiye

Istanbul	27
Ankara	12
Izmir	9
Bursa	7
Adana	3

The final format of the series is exemplified in the following table, the data is considered as three main groups which are called target, related and metadata but for the DeepAR model the data is leaned with all of them in the same line as columns of covariates just as presented. The naming scheme for these three main data groups are to increase the human readability and presented as is from the original paper of the DeepAR algorithm [91].

Table 4.2: Example rows of the training data set

Target			Related		Metadata		
series_id	timestamp	target_value	is_weekend	is_holiday	is_mall	city	avg_demand
0	2018-01-01	101	0	1	1	Istanbul	136
0	2018-01-02	142	0	0	1	Istanbul	136
			⋮				
121	2019-12-31	109	0	0	0	Izmir	96

It is possible to extend these features even more for even wider practical applications. For example this data set consists of just one brand's stores from different location throughout Türkiye but if the brand has multiple different types of stores like clothing stores or utility stores that can be a new feature in the Metadata category. Features named as Metadata are the features specific to the series to be forecasted on in this case they are the properties of the physical store. Related series is used to denote the properties of the time weather the measurement is collected in a holiday date or during the weekend or not it can also be extended with the weather forecast for that time if weather information is available for that date. Lastly, the main data is in the target data set, on which the forecasts will be generated. Further explanations of the example data provided in Table 4.2.

Table 4.3: Explanations of the columns in the data set

name	definition	range	type
series_id	the unique identifier for the series	$[0, 121]$	integer
timestamp	date in the format `YYYY-MM-DD`	$[2018, 2020)$	date
target_value	the demand value to be forecasted on	$[0, \infty)$	integer
is_weekend	a flag denoting the day is at the weekend or not	$0, 1$	bool
is_holiday	a flag denoting the day is at a holiday or not	$0, 1$	bool
is_mall	a flag denoting the store is inside a mall or not	$0, 1$	bool
city	the city where the store is located in	–	string
avg_demand	average demand for the specific store	$[0, \infty)$	integer

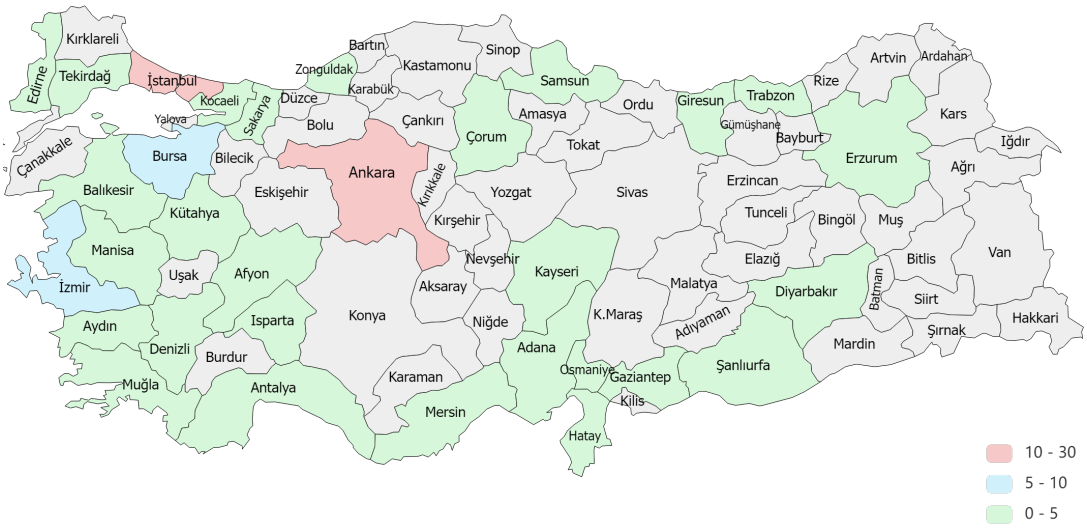


Figure 4.13: Provinces of Türkiye Included in the Study According to the Number of Series (Locations)

CHAPTER 5

RESULTS AND FUTURE WORK

The obtained results and proposals to improve the study in the future are presented in this chapter with the discussion of the obtained results. Different algorithms and their performance are also presented to show that information sharing among series via a singular model indeed contributed to the improved accuracy for the given data set. The comparison with the results of the original DeepAR study can be found in the Appendix E. DeepAR algorithm is capable of producing probabilistic forecasts such as P10, P50, P90 and these three probabilistic forecasts are obtained from the model. Through optimization, the DeepAR algorithm learns a probabilistic function and the PX type of probabilistic forecasts determine where the point forecast will be sampled from this probabilistic function as shown in Figure 5.1.

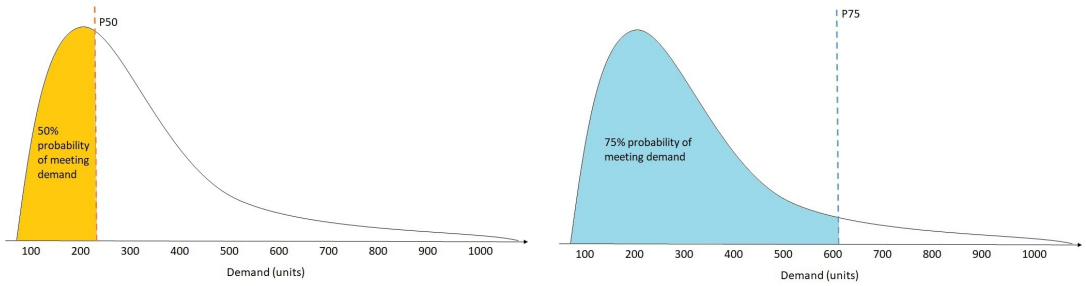


Figure 5.1: Probabilistic quantiles [3]

The point forecasts used for hyperparameter optimization and MAPE calculation is the P50 predictor which implies that 50% of the time the actual value will be less than the predicted value. Although the P50 is used to determine the model parameters, the other predictors are obtained from the model and inspected and presented. The hyperparameters of the model are tuned with trial and error in an iterative search manner where the search values and the increments are presented in Table 5.1. These values are selected through the test set and the error metric to be optimized is selected as MAPE.

Table 5.1: Hyperparameter values used to select the best

hyperparameter name	lower bound	upper bound	increment size	best value
LSTM layers	2	4	1	2
LSTM cells	20	50	10	40
Epoch	100	1000	100	500
learning rate	1e-4	1e-1	1e-1	1e-3

5.1 Results

The obtained results are presented together with the conclusion derived from the research is discussed in this chapter. The aim of this research is to show that in the presence of many similar series a global forecasting model can be trained in order to obtain accurate forecasts. The accuracy of the proposed system would benefit from the information sharing between series and this scheme would successfully train a neural network without the need for synthetic data generation in order to increase the amount of the training samples. Additionally instead of generating (training and optimizing) a model per series where the number of series is large would pose a challenge for the practical ability. The proposed method has shown that it is possible to train a global model for all of the series at the same time generate forecasts from the single model for all series with different probabilistic quantiles. This would allow even more configurable inference even from the same model and it is shown that the produced model is durable provided with the assumption that the behaviour of the series does not drastically change for the test data compared to the training data.

5.1.1 Error Metrics

Table 5.2: Formulae of the error metrics

mean absolute error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}\lvert y_t - \hat{y}_t\rvert$
mean absolute percentage error	$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\left\lvert\frac{y_t - \hat{y}_t}{\hat{y}_t}\right\rvert$

In Table 5.2 the actual value of the series at time t is denoted by y_t and the respective forecast value is denoted by $\hat{y}_t$ and the results presented are the average of these metrics for all series in the data set.

5.1.2 Model Evaluation

The accuracy of the model is presented in Figure 5.2. here it can be clearly seen that the model fulfills the promise of different predictors and their sensitivity compared to the result.

The plot which starts from 2019-12-04 and ends at 2020-01-23 is produced with four consecutive inference iterations of 14 days which is the forecast horizon and there are 56 total test samples presented in order to cover 8 consecutive seasonal windows since the series has weekly seasonality. Average daily value for the presented series is 156 and on average the P50 predictor forecasts has the error of 26 people for the example series.

Figure 5.2: Forecasts from different quantiles

On the other hand, the average error metrics for the model for all series in the same testing window are presented as follows in Table 5.3 with different models. Average target value which is the potential demand for all series is around 85 people per day. Multiple algorithms are tried and presented in Table 5.3, and it is possible to conclude that DeepAR outperforms the ARIMA and ETS models with the same data set.

Table 5.3: Complete error metrics of the model

model name	MAE (avg)	MAPE (avg)	MAPE (min)	MAPE (max)
DeepAR	**19**	**0.26**	0.13	0.37
ARIMA	44	0.39	0.26	0.49
ETS	46	0.40	0.28	0.56

This method provides practical applicability for a very large data set and this is more prominent if the data set is expected to grew larger in time when new series are added to this set. The implications of this problem is explained in the Section 3.1.5 and with 5 series which are completely withheld from the training similar accuracy results are obtained from the same model. The error metrics for the cold start data set are for MAE of 22 and for MAPE of 0.29.

The P10 predictor can be considered as a consistent underestimator, and P90 can be consid-

ered as a consistent overestimator. In a situation where demand has to be met against all costs, for example, for an electrical grid the higher the probabilistic quantile the safer the produced forecasts are to meet the demand. Similarly for non-critical series, P10 can be a candidate for cost saving measures. Depending on the application it is possible to produce a vale for the probabilistic quantile in the interval $[1, 99]$ with DeepAR [91]. This probabilistic assumption is also tested with the testing set and the exceed amounts for each predictor is tabulated in the Table 5.4 where it can be observed that the probabilistic assumptions hold with a margin of error and the closest performer for the probabilistic aspect is the P90 which is convenient since that predictor is a candidate for more critical practical use cases.

Table 5.4: Predictor error measurement for different quantiles

Probabilistic quantile	Forecasted value greater than test value (%)	Expected value(%)
p90	11.37	10
p50	40.13	50
p10	82.40	90

The robustness of the model is tested via observation of the error metrics when the forecast horizon is largely extended. The produced forecasts are recorded for four times the forecast horizon which is 56 days and each forecast horizon's (forecasted values of 14 consecutive days) error metrics are compared. As time progressed, the accuracy of the model remained similar; thus, it is observed that the performance of the model does not degrade frequently. This observation is critical for the model monitoring aspect of this research. The recurrent architecture of the model allows that using the generated forecast as the consecutive observation the forecasts can be generated very far into the future. On the other hand, as time progresses, the actual observations will be recorded too. The decision for the retraining of the model can be made upon observing this degradation periodically and the new training set can have the same with shifting the start and end timestamps of the data into the future or adding the new observations in the end which would increase the training time.

5.2 Future Work

In the future there may be further improvements to the model with a new weather feature. Special calendar days are added to the training set based on the findings of a research showing that specific dates have effects on retail demand [49] and similarly, there are published research on the effects of weather on retail demand [74]. This idea can be tested by comparing two different models with weather data included and excluded side by side and comparing the accuracy results in order to observe whether a significant improvement in the performance is observed or not.

Another practical application can be suggested for the findings of this research is that obtaining forecasts for a long horizon into the pandemic era and comparing the results with the actual recorded data during the pandemic. This analysis can present a snapshot of the world

in order to provide an estimate for a world that pandemic has never happened and the difference between can provide some idea to the world and how much demand is lost during the pandemic in the retail sector in a very narrow scope that is the results may not generalize.

CHAPTER 6

CONCLUSION

This research presented the models used and their place in the literature, as well as the background needed to convey the idea of time series modeling. The data set and the preparation steps before the modelling stage are explained in detail, and the results of each step are discussed with the introduced metrics. The completion of the research showed that if there are many similar series are recorded instead of modeling the series one by one with multiple different models, a model that can process all the series together by allowing information sharing between the series performs better compared to the conventional methods like ETS and ARIMA for the selected data set. The DeepAR model used for this research showed that with Autoregressive Recurrent Neural Networks it is possible for the model to learn long-term dependencies and obtain accurate forecasts for time series data similar to the original article in which the model is presented [91]. The produced model shows practical usability for modeling and forecasting multiple series at once thus in a setting where there are already many series and the number of the series is expected to grow continually this practical advantage is expected to bring further improvements in the performance of the model.

The proposed practical usage envisioned by this research is to predict the possible maximum demand for a series instead of creating the exact point forecasts for the retail applications. The motivation for the suggestion is as follows, for example let the demand in the physical store be a 100 people and on average the P50 predictor of this research has the average MAPE of 0.26 thus the predictions for that time point is expected to be in the interval of $[74, 126]$ when determining the stock amount for a clothing store the demand needs to be met but overstocking by a small margin would be very affordable when compared to being understocked which would result in potential revenue loss since clothes can be sold in a later day or week. On the other hand consumables with narrow expiration dates like food or medicine the retailer would prefer to sell all of the stock before the product becomes unusable and this case can be more suited for predictors smaller than the 50% quantile (e.g. P10 - P50) on the other hand clothing retailer would prefer a higher quantile upper limit. The selection of the preferred quantile would entirely depend upon the beneficiary of this model and how sensitive the specific application is to being overstocked or understocked if the predictions of the possible maximum demand are obtained for stock management.

Study is concluded by confirmation of the idea when there are many similar series present

in a data set which are confirmed to be similar enough to be modeled together, allowing information sharing by globally modeling each series in a single model instead of a model per series can result in accurate forecasts without requiring synthetic data augmentation for the training data. The sensitivity of the quantile can be determined per specific application, and the model produced with DeepAR produced robust forecast even for the series completely out of sample (i.e. cold started series) could find practical applications for planning efforts. The exploratory data analysis was a critical part of this study in order to shed light to the data in order to make the necessary imputation and cleaning efforts in order to improve the quality of the forecasts produced by the model.

REFERENCES

[1] G. Al-Naymat, S. Chawla, and J. Taheri, Sparsedtw: A novel approach to speed up dynamic time warping, in *Proceedings of the Eighth Australasian Data Mining Conference - Volume 101*, AusDM '09, p. 117–127, Australian Computer Society, Inc., AUS, 2009, ISBN 9781920682828.

[2] A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. S. Rangapuram, D. Salinas, J. Schulz, et al., Gluonts: Probabilistic and neural time series modeling in python., J. Mach. Learn. Res., 21(116), pp. 1–6, 2020.

[3] Amazon, Developer guide for aws forecast deepar, `https://docs.aws.amazon.com/forecast/latest/dg/metrics.html`, 2020. Accessed on 1st of April, 2022.

[4] M. Basaldella, E. Antolli, G. Serra, and C. Tasso, *Bidirectional LSTM Recurrent Neural Network for Keyphrase Extraction*, Springer International Publishing, December 2017.

[5] I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou, Patient subtyping via time-aware LSTM networks, in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, August 2017.

[6] Y. Bengio, P. Simard, and P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Transactions on Neural Networks, 5(2), pp. 157–166, 1994.

[7] K. Benidis, S. S. Rangapuram, V. Flunkert, B. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, L. Callot, and T. Januschowski, Neural forecasting: Introduction and literature overview, 2020.

[8] G. E. P. Box and G. M. G. M. Jenkins, Time series analysis; forecasting and control, Time series analysis; forecasting and control., 1970.

[9] G. E. P. Box and D. A. Pierce, Distribution of residual autocorrelations in autoregressive-integrated moving average time series models, Journal of the American Statistical Association, 65(332), pp. 1509–1526, 1970.

[10] T. S. Breusch, Testing for autocorrelation in dynamic linear models, Australian Economic Papers, 17(31), pp. 334–355, December 1978.

[11] E. O. Brigham, *The fast Fourier transform and its applications*, Prentice-Hall, Inc., 1988.

[12] R. G. Brown, *Statistical forecasting for inventory control*, McGraw/Hill, 1959.

[13] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, Language models are few-shot learners, in H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, Curran Associates, Inc., 2020.

[14] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, A limited memory algorithm for bound constrained optimization, SIAM Journal on Scientific Computing, 16(5), pp. 1190–1208, 1995.

[15] T. Caliński and J. Harabasz, A dendrite method for cluster analysis, Communications in Statistics, 3(1), pp. 1–27, 1974.

[16] J. Chai and A. Li, Deep learning in natural language processing: A state-of-the-art survey, in *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, pp. 1–6, 2019.

[17] N. Chapados, Effective bayesian modeling of groups of related count time series, in E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1395–1403, PMLR, Bejing, China, 22–24 Jun 2014.

[18] Y.-Y. Chiang, Recurrent neural networks i, `https://yaoyichi.github.io/spatial-ai.html`, 2022. Accessed on 1st of May, 2022.

[19] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, Palm: Scaling language modeling with pathways, 2022.

[20] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[21] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, Stl: A seasonal-trend decomposition procedure based on loess (with discussion), Journal of Official Statistics, 6, pp. 3–73, 1990.

[22] M. Cuéllar, M. Delgado, and M. Pegalajar, An application of non-linear programming to train recurrent neural networks in time series prediction problems, in C.-S. Chen, J. Filipe, I. Seruca, and J. Cordeiro, editors, *Enterprise Information Systems VII*, pp. 95–102, Springer Netherlands, Dordrecht, 2006, ISBN 978-1-4020-5347-4.

[23] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals, and Systems, 2(4), pp. 303–314, December 1989.

[24] S. Datta, C. K. Karmakar, and M. Palaniswami, Averaging methods using dynamic time warping for time series classification, in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 2794–2798, 2020.

[25] D. L. Davies and D. W. Bouldin, A cluster separation measure, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1(2), pp. 224–227, 1979.

[26] W. De Mulder, S. Bethard, and M.-F. Moens, A survey on the application of recurrent neural networks to statistical language modeling, Computer Speech & Language, 30, 01 2014.

[27] S. Demir, K. Mincev, K. Kok, and N. G. Paterakis, Data augmentation for time series regression: Applying transformations, autoencoders and adversarial networks to electricity price forecasting, Applied Energy, 304, p. 117695, December 2021.

[28] G. Deng, C. Han, T. Dreossi, C. Lee, and D. S. Matteson, *IB-GAN: A Unified Approach for Multivariate Time Series Classification under Class Imbalance*, pp. 217–225, SIAM, 2022.

[29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Association for Computational Linguistics, Minneapolis, Minnesota, June 2019.

[30] R. Dey and F. M. Salem, Gate-variants of gated recurrent unit (gru) neural networks, in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1597–1600, 2017.

[31] D. A. Dickey and W. A. Fuller, Distribution of the estimators for autoregressive time series with a unit root, Journal of the American Statistical Association, 74(366), p. 427, June 1979.

[32] D. Dua and C. Graff, UCI machine learning repository, http://archive.ics.uci.edu/ml, 2017. Accessed on 3rd of June, 2022.

[33] J. H. Friedman, Stochastic gradient boosting, Computational Statistics & Data Analysis, 38(4), pp. 367–378, 2002, nonlinear Methods and Data Mining.

[34] R. Fu, Z. Zhang, and L. Li, Using lstm and gru neural network methods for traffic flow prediction, in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pp. 324–328, 2016.

[35] E. S. Gardner, Exponential smoothing: The state of the art—part ii, International Journal of Forecasting, 22(4), pp. 637–666, 2006.

[36] A. Geiger, D. Liu, S. Alnegheimish, A. Cuesta-Infante, and K. Veeramachaneni, Tadgan: Time series anomaly detection using generative adversarial networks, in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 33–43, 2020.

[37] L. G. Godfrey, Testing against general autoregressive and moving average error models when the regressors include lagged dependent variables, Econometrica, 46(6), p. 1293, November 1978.

[38] O. Gold and M. Sharir, Dynamic time warping and geometric edit distance: Breaking the quadratic barrier, ACM Trans. Algorithms, 14(4), aug 2018.

[39] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, Curran Associates, Inc., 2014.

[40] A. Graves, Generating sequences with recurrent neural networks, CoRR, abs/1308.0850, 2013.

[41] Z. Hajirahimi and M. Khashei, Hybrid structures in time series modeling and forecasting: A review, Engineering Applications of Artificial Intelligence, 86, pp. 83–106, 2019.

[42] J. Han, A. Jentzen, and W. E, Solving high-dimensional partial differential equations using deep learning, Proceedings of the National Academy of Sciences, 115(34), pp. 8505–8510, 2018.

[43] Z. Han, J. Zhao, H. Leung, K. F. Ma, and W. Wang, A review of deep learning models for time series prediction, IEEE Sensors Journal, 21(6), pp. 7833–7848, 2021.

[44] T. Hastie and R. Tibshirani, Generalized additive models: Some applications, Journal of the American Statistical Association, 82(398), pp. 371–386, June 1987.

[45] S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation, 9(8), pp. 1735–1780, November 1997.

[46] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. v. d. Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre, Training compute-optimal large language models, 2022.

[47] C. C. Holt, Forecasting seasonals and trends by exponentially weighted moving averages, International Journal of Forecasting, 20(1), pp. 5–10, January 2004.

[48] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks, 4(2), pp. 251–257, 1991.

[49] J. Huber and H. Stuckenschmidt, Daily retail demand forecasting using machine learning with emphasis on calendric special days, International Journal of Forecasting, 36(4), pp. 1420–1438, 2020.

[50] R. J. Hyndman, *Forecasting with exponential smoothing The State Space Approach*, Springer series in statistics, Springer, Berlin, Germany, December 2008.

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

[52] C. Ingle, D. Bakliwal, J. Jain, P. Singh, P. Kale, and V. Chhajed, Demand forecasting : Literature review on various methodologies, in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 1–7, 2021.

[53] B. K. Iwana and S. Uchida, An empirical survey of data augmentation for time series classification with neural networks, PLOS ONE, 16(7), July 2021.

[54] B. K. Iwana and S. Uchida, Time series data augmentation for neural networks by time warping with a discriminative teacher, in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, January 2021.

[55] P. Jeha, M. Bohlke-Schneider, P. Mercado, S. Kapoor, R. S. Nirwan, V. Flunkert, J. Gasthaus, and T. Januschowski, PSA-GAN: Progressive self attention GANs for synthetic time series, in *International Conference on Learning Representations*, 2022.

[56] V. Joshi, K. Jha, M. Jain, and S. Kulkarni, Tourism footfall forecasting and recommendation system, in *2021 International Conference on Communication information and Computing Technology (ICCICT)*, pp. 1–5, 2021.

[57] K. Kamycki, T. Kapuscinski, and M. Oszust, Data augmentation with suboptimal warping for time-series classification, Sensors, 20(1), p. 98, December 2019.

[58] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, Scaling laws for neural language models, 2020.

[59] Z. H. Kilimci, A. O. Akyuz, M. Uysal, S. Akyokus, M. O. Uysal, B. A. Bulbul, and M. A. Ekmis, An improved demand forecasting model using deep learning approach and proposed decision integration strategy for supply chain, Complexity, 2019, pp. 1–15, March 2019.

[60] A. Koochali, A. Dengel, and S. Ahmed, If you like it, gan it—probabilistic multivariate times series forecast with gan, Engineering Proceedings, 5(1), 2021.

[61] D. Kwiatkowski, P. C. Phillips, P. Schmidt, and Y. Shin, Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?, Journal of Econometrics, 54(1), pp. 159–178, 1992.

[62] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, Time-series extreme event forecasting with neural networks at uber, in *International conference on machine learning*, volume 34, pp. 1–5, sn, 2017.

[63] P. Lara-Benítez, M. Carranza-García, and J. C. Riquelme, An experimental review on deep learning architectures for time series forecasting, CoRR, abs/2103.12057, 2021.

[64] P. D. Larson, D. Simchi-Levi, P. Kaminsky, and E. Simchi-Levi, Designing and managing the supply chain: Concepts, strategies, and case studies, Journal of Business Logistics, 22(1), pp. 259–261, March 2001.

[65] R. Law, G. Li, D. K. C. Fong, and X. Han, Tourism demand forecasting: A deep learning approach, Annals of Tourism Research, 75, pp. 410–423, 2019.

[66] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, The handbook of brain theory and neural networks, 3361(10), p. 1995, 1995.

[67] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Networks, 6(6), pp. 861–867, 1993.

[68] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks, in I. V. Tetko, V. Kůrková, P. Karpov, and F. Theis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*, pp. 703–716, Springer International Publishing, Cham, 2019, ISBN 978-3-030-30490-4.

[69] S. Liao, J. Yin, and W. Rao, Towards accurate retail demand forecasting using deep neural networks, in Y. Nah, B. Cui, S.-W. Lee, J. X. Yu, Y.-S. Moon, and S. E. Whang, editors, *Database Systems for Advanced Applications*, pp. 711–723, Springer International Publishing, Cham, 2020, ISBN 978-3-030-59419-0.

[70] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, Temporal fusion transformers for interpretable multi-horizon time series forecasting, International Journal of Forecasting, 37(4), pp. 1748–1764, 2021.

[71] Y.-T. Liu, Y.-A. Zhang, and M. Zeng, Adaptive global time sequence averaging method using dynamic time warping, IEEE Transactions on Signal Processing, 67(8), pp. 2129–2142, 2019.

[72] A. M. D. Livera, R. J. Hyndman, and R. D. Snyder, Forecasting time series with complex seasonal patterns using exponential smoothing, Journal of the American Statistical Association, 106(496), pp. 1513–1527, 2011.

[73] G. M. LJUNG and G. E. P. BOX, On a measure of lack of fit in time series models, Biometrika, 65(2), pp. 297–303, 08 1978.

[74] G. Makkar, Real-time footfall prediction using weather data: A case on retail analytics, in N. Sharma, A. Chakrabarti, and V. E. Balas, editors, *Data Management, Analytics and Innovation*, pp. 529–542, Springer Singapore, Singapore, 2020, ISBN 978-981-32-9949-8.

[75] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, The m5 competition: Background, organization, and implementation, International Journal of Forecasting, 2021.

[76] J. Martens, Deep learning via hessian-free optimization, in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML10, p. 735–742, Omnipress, Madison, WI, USA, 2010, ISBN 9781605589077.

[77] J. Martens and I. Sutskever, Learning recurrent neural networks with hessian-free optimization, in *ICML*, pp. 1033–1040, 2011.

[78] W. S. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, The Bulletin of Mathematical Biophysics, 5(4), pp. 115–133, December 1943.

[79] Y. Miao, J. Han, Y. Gao, and B. Zhang, St-cnn: Spatial-temporal convolutional neural network for crowd counting in videos, Pattern Recognition Letters, 125, pp. 113–118, 2019.

[80] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, Extensions of recurrent neural network language model, in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5528–5531, 2011.

[81] V. Nogueira, H. Oliveira, J. Augusto Silva, T. Vieira, and K. Oliveira, Retailnet: A deep learning approach for people counting and hot spots detection in retail stores, in *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 155–162, 2019.

[82] C. Olah, Understanding lstm networks, `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, Aug 2015. Accessed on 20th of June, 2022.

[83] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, N-beats: Neural basis expansion analysis for interpretable time series forecasting, in *International Conference on Learning Representations*, 2020.

[84] R. Pascanu, T. Mikolov, and Y. Bengio, On the difficulty of training recurrent neural networks, in S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 1310–1318, PMLR, Atlanta, Georgia, USA, 17–19 Jun 2013.

[85] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison,

A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[86] F. Petitjean, A. Ketterlin, and P. Gançarski, A global averaging method for dynamic time warping, with applications to clustering, Pattern Recognition, 44(3), pp. 678–693, 2011.

[87] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. W. Cottrell, A dual-stage attention-based recurrent neural network for time series prediction, in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, p. 2627–2633, AAAI Press, 2017, ISBN 9780999241103.

[88] P. J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics, 20, pp. 53–65, 1987.

[89] H. Sakoe and S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Transactions on Acoustics, Speech, and Signal Processing, 26(1), pp. 43–49, 1978.

[90] H. Salehinejad and S. Rahnamayan, Customer shopping pattern prediction: A recurrent neural network approach, in *2016 IEEE symposium series on computational intelligence (SSCI)*, pp. 1–6, IEEE, 2016.

[91] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, Deepar: Probabilistic forecasting with autoregressive recurrent networks, International Journal of Forecasting, 36(3), pp. 1181–1191, 2020.

[92] D. Schultz and B. J. Jain, Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces, CoRR, abs/1701.06393, 2017.

[93] A. M. Schäfer and H. G. Zimmermann, Recurrent neural networks are universal approximators, International Journal of Neural Systems, 17(04), pp. 253–263, August 2007.

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

[95] S. S. SHAPIRO and M. B. WILK, An analysis of variance test for normality (complete samples)†, Biometrika, 52(3-4), pp. 591–611, 12 1965.

[96] Z. Shi, L. Zhang, Y. Liu, X. Cao, Y. Ye, M.-M. Cheng, and G. Zheng, Crowd counting with deep negative correlation learning, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[97] C. Shorten and T. M. Khoshgoftaar, A survey on image data augmentation for deep learning, Journal of Big Data, 6(1), p. 60, Jul 2019.

[98] D. F. Silva and G. E. A. P. A. Batista, *Speeding Up All-Pairwise Dynamic Time Warping Matrix Calculation*, SIAM, 2016.

[99] R. D. Snyder, J. K. Ord, and A. Beaumont, Forecasting the intermittent demand for slow-moving inventories: A modelling approach, International Journal of Forecasting, 28(2), pp. 485–496, 2012.

[100] M. Stepnicka and M. Burda, Computational intelligence in forecasting - the results of the time series forecasting competition, in *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, August 2015.

[101] I. Sutskever, O. Vinyals, and Q. V. Le, Sequence to sequence learning with neural networks, CoRR, abs/1409.3215, 2014.

[102] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods, Tslearn, a machine learning toolkit for time series data, Journal of Machine Learning Research, 21(118), pp. 1–6, 2020.

[103] S. J. Taylor and B. Letham, Forecasting at scale, The American Statistician, 72(1), pp. 37–45, 2018.

[104] O. Triebe, H. Hewamalage, P. Pilyugina, N. Laptev, C. Bergmeir, and R. Rajagopal, Neuralprophet: Explainable forecasting at scale, CoRR, abs/2111.15397, 2021.

[105] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, Attention is all you need, in I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, Curran Associates, Inc., 2017.

[106] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, A. Vijaykumar, A. P. Bardelli, A. Rothberg, A. Hilboll, A. Kloeckner, A. Scopatz, A. Lee, A. Rokem, C. N. Woods, C. Fulton, C. Masson, C. Häggström, C. Fitzgerald, D. A. Nicholson, D. R. Hagen, D. V. Pasechnik, E. Olivetti, E. Martin, E. Wieser, F. Silva, F. Lenders, F. Wilhelm, G. Young, G. A. Price, G.-L. Ingold, G. E. Allen, G. R. Lee, H. Audren, I. Probst, J. P. Dietrich, J. Silterra, J. T. Webber, J. Slavič, J. Nothman, J. Buchner, J. Kulick, J. L. Schönberger, J. V. de Miranda Cardoso, J. Reimer, J. Harrington, J. L. C. Rodríguez, J. Nunez-Iglesias, J. Kuczynski, K. Tritz, M. Thoma, M. Newville, M. Kümmerer, M. Bolingbroke, M. Tartre, M. Pak, N. J. Smith, N. Nowaczyk, N. Shebanov, O. Pavlyk, P. A. Brodtkorb, P. Lee, R. T. McGibbon, R. Feldbauer, S. Lewis, S. Tygier, S. Sievert, S. Vigna, S. Peterson, S. More, T. Pudlik, T. Oshima, T. J. Pingel, T. P. Robitaille, T. Spura, T. R. Jones, T. Cera, T. Leslie, T. Zito, T. Krauss, U. Upadhyay, Y. O. Halchenko, and

Y. V.-B. and, SciPy 1.0: fundamental algorithms for scientific computing in python, Nature Methods, 17(3), pp. 261–272, February 2020.

[107] R. Wan, S. Mei, J. Wang, M. Liu, and F. Yang, Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting, Electronics, 8(8), p. 876, 2019.

[108] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, Time series data augmentation for deep learning: A survey, in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, International Joint Conferences on Artificial Intelligence Organization, August 2021.

[109] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, A multi-horizon quantile recurrent forecaster, in *31st Conference on Neural Information Processing Systems (NIPS 2017), Time Series Workshop. Long Beach, CA, USA*, 2017.

[110] P. R. Winters, Forecasting sales by exponentially weighted moving averages, Management Science, 6(3), pp. 324–342, April 1960.

[111] h. wu, J. Xu, J. Wang, and M. Long, Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, in M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pp. 22419–22430, Curran Associates, Inc., 2021.

[112] G. Yiğit and M. F. Amasyali, Simple but effective gru variants, in *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pp. 1–6, 2021.

[113] J. Yoon, D. Jarrett, and M. van der Schaar, Time-series generative adversarial networks, in H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, Curran Associates, Inc., 2019.

[114] D. Zhang, N. Maslej, E. Brynjolfsson, J. Etchemendy, T. Lyons, J. Manyika, H. Ngo, J. C. Niebles, M. Sellitto, E. Sakhaee, Y. Shoham, J. Clark, and R. Perrault, The ai index 2022 annual report, AI Index Steering Committee, Stanford Institute for Human-Centered AI, Stanford University, p. 123, March 2022.

[115] G. Zhang, Time series forecasting using a hybrid arima and neural network model, Neurocomputing, 50, pp. 159–175, 2003.

[116] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, Convolutional neural networks for time series classification, Journal of Systems Engineering and Electronics, 28(1), pp. 162–169, 2017.

[117] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in *Proceedings of AAAI*, 2021.

[118] L. Zhu and N. Laptev, Deep and confident prediction for time series at uber, in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 103–110, IEEE, 2017.

[119] E. Zunic, K. Korjenic, S. Delalic, and Z. Subara, Comparison analysis of facebook's prophet, amazon's deepar+ and CNN-QR algorithms for successful real-world sales forecasting, CoRR, abs/2105.00694, 2021.

APPENDIX A

DTW CLUSTERING

A.1 Example Code

```
# example data inpput.csv format for a daily data with one year
# StoreCode, YYYY-01-01, YYYY-01-02, ... , YYYY-12-31
# <StoreCode>, t_1, t_2, ... , t_365
import pandas as pd
from tslearn.utils import to_time_series_dataset
from tslearn.preprocessing import TimeSeriesScalerMeanVariance
from tslearn.clustering import TimeSeriesKMeans, silhouette_score

cluster_count = # number of clusters to be experimented

# prepare data to laod to tslearn time_series_dataset object
df_pivoted = pd.read_csv('input.csv')
df_pivoted.set_index('StoreCode', inplace=True)

# convert dataframe to time_series_dataset
X = to_time_series_dataset(df_pivoted.values)

# normalize time series to zero mean and unit variance
X_train = TimeSeriesScalerMeanVariance().fit_transform(X)

model = TimeSeriesKMeans(cluster_count, 'dtw')
y_pred = model.fit_predict(X_train)

# Sum of distances of samples to their closest cluster center.
print('inertia', cluster_count, model.inertia_)

# silhouette_score
score = silhouette_score(X_train, y_pred, 'dtw')
print('silhouette', cluster_count, score)

# DBA center (frechet mean) of each cluster
for yi in range(N_CLUSTERS):
    print(model.cluster_centers_[yi])
```

A.2 Algorithm of DTW

```
def dtw(x, y):
    # Initialization
    for i = 1..n
        for j = 1..m
            C[i, j] = inf

    C[0, 0] = 0.

   # Main loop
   for i = 1..n
        for j = 1..m
            dist = d(x_i, y_j) ** 2
            C[i, j] = dist + min(C[i-1, j], C[i, j-1], C[i-1, j-1])

   return sqrt(C[n, m])
```

APPENDIX B

EXAMPLE IMPLEMENTATIONS OF DEEPAR

Example use of DeepAR from the GluonTS documentation [2].

B.1 Example Code of GluonTS

```
%#!pip install gluonts
%#!pip install --upgrade mxnet==1.6.0

from gluonts.dataset import common
from gluonts.model import deepar
from gluonts.mx.trainer import Trainer
# from gluonts.trainer import Trainer

import pandas as pd

url = "https://raw.githubusercontent.com/numenta/
       NAB/master/data/realTweets/Twitter_volume_AMZN.csv"

df = pd.read_csv(url, header=0, index_col=0)
data = common.ListDataset([{
    "start": df.index[0],
    "target": df.value[:"2015-04-05 00:00:00"]
    }],
    freq="5min"
)

trainer = Trainer(epochs=10)
estimator = deepar.DeepAREstimator(
    freq="5min",
    prediction_length=12,
    trainer=trainer
)
predictor = estimator.train(training_data=data)

prediction = next(predictor.predict(data))
print(prediction.mean)
```

```
prediction.plot(output_file='graph.png')
```

B.2 Example Code of PyTorchForecasting

Example use of DeepAR from the PyTorchForecasting documentation which is based on PyTorch [85]

```
from pathlib import Path
import pickle
import warnings

import numpy as np
import pandas as pd
from pandas.core.common import SettingWithCopyWarning
import pytorch_lightning as pl
from pytorch_lightning.callbacks import EarlyStopping, LearningRateMonitor
from pytorch_lightning.loggers import TensorBoardLogger
import torch

from pytorch_forecasting import EncoderNormalizer, GroupNormalizer, TimeSeriesDataSet
from pytorch_forecasting.data import NaNLabelEncoder
from pytorch_forecasting.data.examples import generate_ar_data
from pytorch_forecasting.metrics import NormalDistributionLoss
from pytorch_forecasting.models.deepar import DeepAR
from pytorch_forecasting.utils import profile

warnings.simplefilter("error", category=SettingWithCopyWarning)


data = generate_ar_data(seasonality=10.0, timesteps=400, n_series=100)
data["static"] = "2"
data["date"] = pd.Timestamp("2020-01-01") + pd.to_timedelta(data.time_idx, "D")
validation = data.series.sample(20)

max_encoder_length = 60
max_prediction_length = 20

training_cutoff = data["time_idx"].max() - max_prediction_length

training = TimeSeriesDataSet(
    data[lambda x: ~x.series.isin(validation)],
    time_idx="time_idx",
    target="value",
    categorical_encoders={"series": NaNLabelEncoder().fit(data.series)},
    group_ids=["series"],
    static_categoricals=["static"],
    min_encoder_length=max_encoder_length,
```

```
    max_encoder_length=max_encoder_length,
    min_prediction_length=max_prediction_length,
    max_prediction_length=max_prediction_length,
    time_varying_unknown_reals=["value"],
    time_varying_known_reals=["time_idx"],
    target_normalizer=GroupNormalizer(groups=["series"]),
    add_relative_time_idx=False,
    add_target_scales=True,
    randomize_length=None,
)

validation = TimeSeriesDataSet.from_dataset(
    training,
    data[lambda x: x.series.isin(validation)],
    # predict=True,
    stop_randomization=True,
)
batch_size = 64
train_dataloader = training.to_dataloader(
    train=True,
    batch_size=batch_size,
    num_workers=0
)

val_dataloader = validation.to_dataloader(
    train=False,
    batch_size=batch_size,
    num_workers=0
)

# save datasets
training.save("training.pkl")
validation.save("validation.pkl")

early_stop_callback = EarlyStopping(
    monitor="val_loss",
    min_delta=1e-4,
    patience=5,
    verbose=False,
    mode="min"
)


lr_logger = LearningRateMonitor()

trainer = pl.Trainer(
    max_epochs=10,
    gpus=-1,
    gradient_clip_val=0.1,
    limit_train_batches=30,
```

```
    limit_val_batches=3,
    # fast_dev_run=True,
    # logger=logger,
    # profiler=True,
    callbacks=[lr_logger, early_stop_callback],
)


deepar = DeepAR.from_dataset(
    training,
    learning_rate=0.1,
    hidden_size=32,
    dropout=0.1,
    loss=NormalDistributionLoss(),
    log_interval=10,
    log_val_interval=3,
    # reduce_on_plateau_patience=3,
)
print(f"Number of parameters in network: {deepar.size()/1e3:.1f}k")

# # find optimal learning rate
# deepar.hparams.log_interval = -1
# deepar.hparams.log_val_interval = -1
# trainer.limit_train_batches = 1.0
# res = trainer.tuner.lr_find(
#     deepar,
#     train_dataloaders=train_dataloader,
#     val_dataloaders=val_dataloader, min_lr=1e-5, max_lr=1e2
# )
# print(f"suggested learning rate: {res.suggestion()}")
# fig = res.plot(show=True, suggest=True)
# fig.show()
# deepar.hparams.learning_rate = res.suggestion()

torch.set_num_threads(10)
trainer.fit(
    deepar,
    train_dataloaders=train_dataloader,
    val_dataloaders=val_dataloader,
)

# calcualte mean absolute error on validation set
actuals = torch.cat([y for x, (y, weight) in iter(val_dataloader)])
predictions = deepar.predict(val_dataloader)
print(f"Mean absolute error of model: {(actuals - predictions).abs().mean()}")

# # plot actual vs. predictions
# raw_predictions, x = deepar.predict(val_dataloader, mode="raw", return_x=True)
# for idx in range(10):  # plot 10 examples
#     deepar.plot_prediction(x, raw_predictions, idx=idx, add_loss_to_title=True)
```

APPENDIX C

FOURIER ANALYSIS

C.1 Example Code

```
# example dataframe format for a daily data with
# one year contained in variable called data
# 2018-01-01    149.0
# 2018-01-02    130.0

from scipy.fftpack import rfft, rfftfreq
from statsmodels.tsa.seasonal import STL
from statsmodels.tsa.stattools import adfuller

#STL decomposition
stl = STL(data, robust=True)
res = stl.fit()
fourier = res.seasonal.values

#dickey fuller test
adf_res = adf_test(data)
print(adf_res['p-value'] < 0.05)

#fast fourier transform
nobs = len(fourier)
data_ft = np.abs(rfft(fourier))
data_freq = rfftfreq(nobs)
# this prints the period as unit of days since the data was daily
print(1 / data_freq[2:])
```

APPENDIX D

ERROR METRIC CALCULATION

D.1 Example Code

```
def calculate_mape(y_test, y_predicted):
    y_test, y_predicted = np.array(y_test), np.array(y_predicted)
    mape = np.mean(np.abs((y_test - y_predicted) / y_test))
    return mape


def calculate_mae(y_test, y_predicted):
    y_test, y_predicted = np.array(y_test), np.array(y_predicted)
    mae = np.mean(np.abs(y_test - y_predicted))
    return mae


def calculate_rmse(y_test, y_predicted):
    y_test, y_predicted = np.array(y_test), np.array(y_predicted)
    return math.sqrt(np.square(np.subtract(y_test, y_predicted)).mean())
```

APPENDIX E

BENCHMARK

E.1 Benchmark Data Set

The `electricity` data set which is a publicly available dataset provided by UCI ML Repository [32] contains hourly time series of the electricity consumption of 370 customers and used for performance bench marking in the original study of DeepAR [91] with training window sert as 2014-01-01 and 2014-09-01. The `retail` data set is the proprietary data set used for this research and it spans from 2018-01-01 to 2019-12-31 and details of the datasets are shown at Table E.1

where MAPE is formulated as

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n}|\frac{A_t - F_t}{A_t}| \tag{E.1}$$

A lower value indicates a more accurate model. A_t : actual observation at time t, F_t : forecasted value for time t and n number of observations forecasted in the series. Running time is measured on a hardware with four CPU cores and a single GPU.

Table E.1: Details of `electricity` and `retail` data set and training values

	`electricity`	`retail`
num. of series	370	122
granularity	hourly	daily
domain	$\mathbb{R}^+$	$\mathbb{N}$
num. of training samples	500K	100k
learning rate	1e-3	1e-3
Num. of LSTM layers	3	2
Num. of LSTM nodes	40	40
running time	7h	2h
epochs	500	500
context_length	24	7
likelihood	gaussian	negative binomial
MAPE(avg)	0.30	0.26