The School of Mathematics

THE UNIVERSITY
of EDINBURGH

Dynamic and Unconstrained Demand Forecast

by

Abzal Yessengazy

Dissertation Presented for the Degree of
MSc in Operational Research with Data Science

August 2020

Supervised by
Dr Miguel Anjos

Abstract

In the railway industry, the passenger demand forecasting plays an important role in a big picture of the Revenue Management System (RMS). An accurate forecast helps companies to make right strategic and planning decisions at the right time. Moreover it is used to set up ticket pricing policies. Traditional Single Task Algorithms treat the target passenger booking numbers as one dependent variable that has to be predicted for all types of products (economy, business, premium etc.). However, in this dissertation, we propose a novel Multi-Task Learning (MTL) model that separates demand into multiple subtargets (products) and simultaneously provides predictions on each of them. The proposed model turns out to be flexible in managing hyperparameters, losses and their respective weights. In addition, it is also less sensitive to overfitting to specific target since adjusts hyperparameters so that commonly beneficial for all targets. The proposed MTL Deep Neural Networks (DNN) model has been implemented using Functional API of Keras library and resulting predictions were evaluated versus two other Single Task Learning (STL) DNN models with similar architecture. Results are promising, out of 14 evaluation criterion, MTL model showed the best performance on 5, whereas in others it lost with minimal difference in percentage.

Keywords: Railway Passenger Demand Forecasting, Multi-Task Learning, Deep Neural Networks, Revenue Management

Acknowledgments

I would like to express my deep appreciation to my academic supervisor, Prof. Miguel Anjos, for his continuous, enthusiastic support and detailed feedback on my work throughout the dissertation. I am also grateful to my industrial supervisors' team at Expretio, led by Thibault Barbier and his incredible data science team members, Sabine Akchiche, Aliou Sarr and Marc Tetreau for their work in preparing the data and their many helpful inputs and directions related to both the technicalities of working with the data, modelling and the interpretation of the results. I appreciate the entire team's enormous efforts to supply me with the ready data set that was central to the project. Thanks also to Dr Chris Dent, my personal tutor, for continuous support in these tough times due to Covid-19 outbreak. Finally, many thanks to the University of Edinburgh for the opportunity to be part of this exciting real-world project carried out in collaboration with respectful Company.

Own Work Declaration

I declare that this dissertation was written by myself and that the work contained here is my own, except where explicitly stated otherwise in the text.

(Abzal Yessengazy)

Contents

List of Tables

List of Figures

Acronyms

RMS Revenue Management System

DBD Day Before Departure

CRISP-DM CRoss-Industry Standard Process-Data Mining

ARIMA Autoregressive Integrated Moving Average

SARIMA Seasonal ARIMA

SVM Support Vector Machines

ML Machine Learning

MTL Multi-Task Learning

STL Single Task Learning

MAE Mean Absolute Error

MSE Mean Squared Error

TS Time Series

ACF autocorrelation function

PACF partial autocorrelation function

ANN Artificial Neural Networks

DNN Deep Neural Networks

lasso least absolute shrinkage and selection operator

MAPE Mean Absolute Percentage Error

WAPE Weighted Absolute Percent Error

MLP Multi-Layer Perceptron

MDA Mean Decrease Accuracy

1 Introduction

1.1 The Problem Statement

Forecasting a product demand in the railway industry, namely for the train tickets at various fares offered by the operator, plays a crucial role in their Revenue Management System (RMS). The term *demand* here corresponds to potential passengers that arrive throughout the active reservation period of several months to eventually buy offered products. Knowledge of the number of passengers boarding the train is required for several reasons. Firstly, *strategic planning* purposes; a capacity of carriages, a number of required staff, catering and other services offered to passengers. Meantime, a total forecasted number of potential passengers can be utilized to plan the load to the train stations on certain days. Secondly, *price adjustment* purpose; in a complex RMS, the forecasted values are fed to fare optimisation model as an input. This means the dynamic fare adjustment to products are directly affected by the results of forecast. The development of a model that could achieve a high accuracy in predictions may increase chances of the company to make right decisions at the right time. Concluding, the usage of the smart allocation of available resources can minimise operational costs, whereas properly managed product prices maximise profits.

The aim of the dissertation is to explore new approaches in a railway passenger demand forecasting that are able to solve two main challenges of existing models. First of all, the proposed approach has to address the issue of modelling the **Dynamic** nature of demand curve. To put it simply, it is interested in figuring out how the demand for the specific product has been evolving over time at various points of booking horizon (see Figure 1). For instance, 1 day before the departure of train the cumulative booking number for product 1 was 75, but for the actual day of departure it was only 3. Knowing the velocity by which demand is progressing is important.

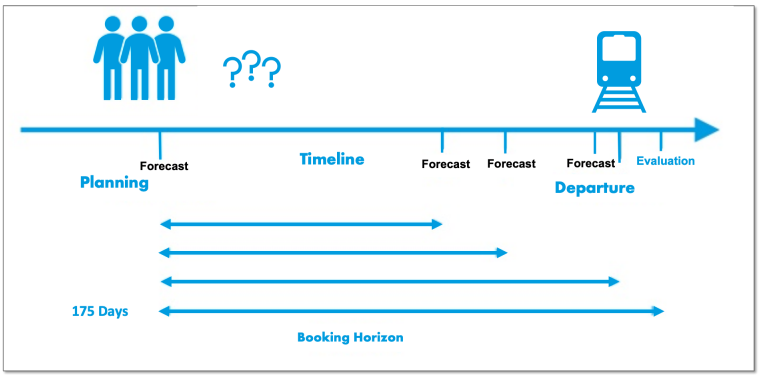

Figure 1: Active Booking Horizon of 175 DBD when customers can book tickets for the train. Forecasting is done at various points of booking horizon.

Whereas, the **Unconstrained** part asks the question: *"Is the historical number of boarded passengers a good representation of actual interest in a specific product?"*. Traditional forecasting approaches deploy machine learning or statistical algorithms to analyse and predict future demand based on historical passengers' number. These algorithms often lead to acceptable results, nonetheless have significant discrepancy - they are constrained on the i) capacity of a carrier that accepted a certain number of passengers ii) the offer amount that company decides to allocate at certain DBD. For instance, the train with a full capacity of 100 seats requires accepting only 100 passengers, however, it might not necessarily be the case that only 100 customers were interested in purchasing the tickets. In fact, passengers might cancel their bookings or might not show up and many other reasons that force passengers to not to board the train. Awareness of real market interest is required to adopt strategic plans of the railway operator, e.g. by expanding the capacity of carriages, adding extra carriages or new itineraries relying on the analysis. This leads us to explore the unconstrained demand forecasting algorithms to better understand the real demand.

In order to efficiently construct such algorithms, we considered the overall number of bookings that railway operator has received. Thereby using the historical bookings data we attempt to model

an unconstrained demand forecast. Expretio has been working on different ways of dealing with the unconstrained demand forecast. One of the methods that had been used was to utilise elasticity and demand weighting using the historical data. However, the quality of the results of this approach was highly dependent on assumptions of demand provided by a not automated analysis.

For this project, Expretio supplied us with the real data collected by one of its clients, a major railway operator in the United Kingdom. The railway operator sells train tickets through their website. Hence using web analytics they are able to provide the data on ticket bookings made online by potential customers. These historical data was used to determine the level of interest in a certain product. Expretio's data science and operational research teams are convinced that the analysis of historical bookings data might be sufficient to build an unconstrained demand forecasting model and improve precision. Thus, this project's aim is to conduct a detailed research of such a hypothesis and report on any findings that as a consequence worthwhile investing time and efforts.

1.2 Key Focus Areas

The purpose of the dissertation is to investigate novel approaches that will both capture the dynamic and unconstrained nature of the passenger demand forecast as well as properly fit the operational requirements of Expretio. Therefore, a high-level task was to come up with the demand forecasting machine learning or statistical approach based on historical data and analyze the performance on the railway operator's real booking data.

Additional issues to address whilst handling the tasks include, but not limited to:

- Inspect existing benchmark models to identify potential algorithms that best fit the task.
- Conduct full-stack time series analysis and identify statistical properties of data such as trend, seasonality and effects of additional factors.
- Identify proper levels of aggregation to build a time series model and investigate its capabilities.
- Identify which time series aspects are crucial while modelling the traditional machine learning algorithm and come up with the ways of their incorporation.
- Properly handle the dynamic part in the modelling stage, also obtain a specific demand prediction given an origin & destination, a departure date, a departure time, a day before departure and a product.
- Incorporate booking velocity and cumulative bookings.

1.3 Dissertation Plan

The dissertation is organized as follows:

Chapter 1 - Introduction: Provides information on the general setup of a business task, the main motivation for conducting the research and explains the role of the demand forecasting process in the big picture of revenue management. In addition, it lays out the main challenges of the RMS whilst using the traditional approaches such as dynamic and unconstrained aspects. The key areas of research are presented in the final section of the chapter.

Chapter 2 - Literature Review: Describes the up to date research papers closely related to the topic and briefly provides information on their strengths. Explains how the forecasting has become a complex task since the boom of the internet, allowing customers to choose between a variety of options. On top of that, an increase in data sizes added another complexity. However, it has inspired a number of novel approaches to be born, and the chapter presents its applications in passenger demand forecasting task in several cross-sectional industries. The major drawbacks of the previous methods with respect to a specific task conclude this chapter.

Chapter 3 - Methodology: In order to tackle any machine learning task the specific methodology has to be applied depending on the nature of the task. It is a common practice to make use of CRISP-DM methodology in cases where one needs to prove the applicability of a proposed method. The Chapter explains in detail how the CRISP methodology can be utilized in our case study.

Chapter 4 - Data Analysis & Preparation: addresses the full-stack time series analysis and methods for pre-processing the data, along with a detailed description on data structure, statistical properties such as trend, seasonality, external factors affect before jumping into a modelling part. In addition, the section describes the essential motivations behind choosing modelling decisions.

Chapter 5 - Modelling: Two modelling approaches have been investigated in detail including time-series and mathematical-programming methods. As a multi-variate time series model Facebook Prophet model had been tested and the results of the experiments were presented. For the latter, the main focus was to showcase a strong potential and capability of Multi-Task Learning as part of transfer learning artificial neural networks. The detailed technical formulation as well as comprehensive comparison with two other models have been illustrated. In addition, results and limitations of each model have been discussed.

Chapter 6 - Conclusions: Discussion of the results of data analysis and modelling stages. In addition, several ideas on further improvements of the model such as new features, using special tools for hyperparameter tuning and interesting hybrid model's architecture were presented.

2 Literature Review

The history of building the passenger demand forecasting systems goes back to early 1950s, when major airline companies first faced the challenge of passenger allocation and catering planning issues. Ever since numerous researches have been carried out suggesting various methodologies, especially focused on the airline industry rather than a railway. However, considering the fact that these industries have mostly identical problem formulations with few exceptions, it is worth mentioning the developments in both areas.

As previously mentioned, the forecasting demand is one of the fundamental elements of RMS. Knowledge of the amount and the volume of demand for seats on any given flight or a train trip are crucial in order to provide input for a successful model for optimisation. The results of the research in Pölt [21] clearly confirms the fact that any increase in forecast precision results in increases of profit.

After internet and online sales boom in the 2000s, when buyers were given an opportunity to match and compare ticket prices with no restrictions on fares Boyd [6], forecasting has become more complex task as opposed to earlier years. Meantime, more automated methods and systems for RMS have been introduced to address the huge amounts of data and customers' choice behaviour that requires precise models Nason [19].

In the first decade of the 2000s, researchers in the field have studied different aspects of the problem including the RMS theory by Talluri [26] and applied RMS by Chiang [12]. Whereas, papers such as Bitran [4] focuses on specific areas of RMS such as the development of a typology, pricing and the implications of electronic-commerce. However, as stated in the paper by Cleophas [9], none of them addresses the RMS in a systematic way, but does touch different aspects of it.

When it comes to passenger demand forecasting, Cleophas [9] outlines four major aspects that RMS takes into account:

1. **Demand arrival:** the distribution of passenger requests for tickets in the diverse booking classes over the course of the sales period for specific flights or itineraries (travel paths between origin and destination).

2. **Demand volume:** the calculation of the development of overall demand over the course of months and years.

3. **Demand behaviour:** the reaction of customers requesting tickets to the alternatives that they are offered by the railway/airline and its competitors.

4. **Demand detruncation:** the task of deducting the actual demand from observed sales data. This aspect of forecasting influences the previously mentioned ones as it provides the data basis from which estimates about trends or sensitivities are drawn.

Demand arrival corresponds the cumulative number of bookings throughout the reservation phase or horizon for particular class (business, economy), product (fare) and the itinerary (route). Demand arrival distribution for the first time was discussed in Beckmann and Bobkowski [3]. The resulting demand can be seen as booking curve that tends upwards. In addition to positive accepted bookings, there is an effect of cancelled bookings that negatively impacts the booking curve. The stated paper suggests methods for usage of the demand curve in forecasting purposes. However, this method is highly depended on non-controllable outside circumstances that may leave the curve assumptions no longer valid. To deal with complex and long-term booking horizons Lee [17] suggested to divide it into more manageable time slices or data collection points. If the bookings have already taken place to the particular trip it is considered as an advanced booking method.

Another common way of dealing with the demand arrival is to model it with a Poisson process. It is a special case of Markovian processes. The approach is based on the assumption that the present event is influenced by the latest or previous events. The number of steps back in time determines the level of the Markovian process to be used. A good illustration of the application of Poisson process can be found in Talluri [26] and Walczak [28]. They used the Poisson arrival rates as parameters that require to be estimated using time series methods.

Precise estimation of demand arrival rates may not be sufficient without the knowledge of the specific amount of demand. The demand volume can fluctuate and mainly correlated with overall economic trends, holidays, and from seasons of year down to weekly or daily patterns. Two main methods have had in-depth researches using time series forecasting, one being *ad hoc* method which takes into account the data patterns and emulations. On the other hand, the second method called *associative* method focuses on possible causal relationships between the booking data and influence factors. However, the latter can become more challenging considering the main assumptions that include the causal factors' independence.

Optimal seat allocation for any given point of time within the booking horizon has always been a challenge. Chen [11] proposed a method of statistical learning that estimates a market value for tickets being purchased at a specific time. This paper thereby extends a model based on a discrete-time Markov decision problem in Lautenbacher and Stidham [16] to the network level. Instead of explicitly forecasting demand, value functions for remaining seats are estimated and updated. This concept offers the opportunity of already considering a certain degree of demand behaviour when estimating demand arrival. However, it does not allow customers to choose between various itineraries based on the current available situation.

2.1 Application of Time Series Models

Time series models have been a common choice for passenger demand forecasting task. One of the most popular time series models is Autoregressive Integrated Moving Average (ARIMA) model. This method uses Moving Average (MA) and Autoregressive (AR) models in combination. MA uses the average values of subsets of original data to predict the future values, whereas AR derives predictions using a weighted sum of previous time step values. Integration of these models' predictions results in final forecasted values. Additional complexities in the model capture the non-stationary aspects of time series. One of them is Seasonal ARIMA (SARIMA) presented by Box and Jenkins [5]. ARIMA does not allow users to directly model the seasonal components of time series, hence latter extends its capabilities. This ability of capturing seasonality aspect makes SARIMA model relevant for passenger demand forecasting tasks. So, in Fildes [22] the researchers obtained good results by applying the model in airline and railway passenger forecasting tasks. Another interesting example of using the model in the railway industry is in Milenković [32], who experimented with SARIMA model to forecast Serbian railway passenger flow based on historical monthly counts and obtained outcome was reasonable as well. An interesting study conducted by Wei Ming [29] introduces a hybrid model of ARIMA-Support Vector Machines (SVM) prediction models applied to airline passenger flow. Moreover, authors showed

that taking advantage of the unique strength of ARIMA and combining the strength of SVM using nonlinear modelling yields promising results.

In summary, ARIMA models with various extensions have been widely utilized and proved its applicability in ranges of tasks compared to other time series models. However, the obvious limitations on the levels of aggregations that these models yield leads us to rely more on flexible machine learning models.

2.2 Application of Machine Learning Models

The aforementioned methods fall into the category of statistical-based forecasting techniques. Throughout the last decade, especially observing the huge computational enhancements and machine learning gaining popularity, there is a huge shift in using the so-called mathematical programming-based methods. One of the papers that approaches the railway demand forecasting problem with machine learning algorithm was Sharif [24]. The paper suggests a comprehensive theoretical methodology to approach the demand forecasting with Artificial Neural Networks (ANN). In the meantime, they tested their approach on real data obtained from European railway company. The authors compared the performance of traditional Multi-Layer Perceptron (MLP) with their novel approach and obtained the affirmative outcomes for their hypothesis. On top of that, Mohie [18] applied genetic algorithm (GA) to enhance backpropagation ANN to forecast the number of airline passengers in the region of Egypt. The GA adapted the weights in the ANN model which resulted in better model performance in this case. The paper suggests to include features such population size, employed population, per capita income(PCI), Gross domestic product (GDP), gross national product(GNP), economic growth rate an foreign exchange rate, which is quite unusual, but interesting.

While discussed approaches are efficient and applicable in various fields, they might not scale well to the operational requirements of Expretio, where time series models would involve a number of models for the different product types, itineraries and booking horizons. Even basic regression models per itinerary would still require separate models training on each of five product types. Hence adds complexity to deployment, maintenance and after-deployment parameter amendments if necessary.

Nonetheless, throughout the dissertation, we will be describing both time series and mathematical programming approaches applied in two different levels of aggregation (hourly-daily for former and no aggregation for latter) in order to overcome described complexities.

3 Methodology

3.1 CRISP-Data Mining

In order to execute any project, one has to first split it into smaller tasks and come up with the detailed plan. In Machine Learning (ML) context, the common way to do it is to use methodologies. The CRoss-Industry Standard Process-Data Mining (CRISP-DM) is a well-known methodology that is both flexible and efficient. The CRISP breaks down the DM project into six phases, defines their respective tasks, and the relationships between these tasks. The relationships may exist between any of the tasks and mostly dependent on the data in hand. The life cycle of DM project according to CRISP approach is shown in the Figure 2. Moving back and forth between tasks is always the case following the iterative nature of the process. The arrows represent the general flow of the cycle, hence lessons learned in one phase is deployed to feed to the subsequent phases and so on. As a note, since our main goal is to test the scalability of proposed models, we follow the *proof of concept approach* which in turn does not require the immediate deployment of the model. As a result, the *deployment phase* in the diagram is out of scope of the dissertation and thus removed.

In the following part, we will briefly summarize each block of the CRISP-DM chain applied to our project:

Business Understanding: The ability to accurately forecast the passenger demand for specific itinerary at any point of booking horizon is crucial in overall RMS of railway operator. It helps with proper planning of catering, infrastructure and helps to save costs. In the meantime, precise demand

values further fed to price optimization model, so, it directly impacts pricing policy. The details are given in the Introduction Section of the dissertation.

Data Understanding: Expretio has provided the ready historical bookings data that is central to our project. This is a structured time series data set collected by railway operator in the period from June 2018 up to March 2020. The data set is used to build models that could be efficiently deployed to forecast *dynamic* and *unconstrained* demand. The detailed analysis of the data set are presented in the Data Analysis & Preparation Section of the dissertation.

Data Preparation: Cleaning, filtering and feature engineering of input data to time series and traditional ML models vary, so have to be treated accordingly. Data preparation may also include steps such as data normalization, handling the missing data and outliers. Methods used during the project are described in the Data Analysis & Preparation Section of the dissertation.

Modelling & Evaluation: Amongst time series models we consider the novel Facebook's Prophet model. The performance of Prophet model on hourly and daily aggregation levels was analyzed to showcase its capabilities. However, the main idea was to propose MTL since it best fits the operational necessity of Expretio. The comparison of MTL model with 2 types of DNN models was presented as part of the exploration of mathematical programming methods. The first being Single Output STL as the baseline model as well as Multi-Output STL. The proposed MTL model compared to stated DNN models with the similar architecture to obtain a complete and fair evaluation. Details are presented in Modelling Section.

The overall idea was to start with simple Prophet and DNN models and subsequently increase complexity, meantime tracking the changes in performance. For instance, the Prophet model without any regressors and added regressors (multivariate features). For the MTL model, first to test it with 1 layer and a small number of nodes, and then increase the layers structure and their respective nodes.

In case of DNN we have to note the complexity of the model with lots of parameters and options to explore. Thus, hyperparameters such as learning rate, batch size, drop out rate, regularization rates, epochs, activation functions and weight initialization are tested using Keras Tuner, but will not be extensively discussed. The reason is to keep the focus on what is important for this dissertation, namely the comparison of MTL with single-task models.

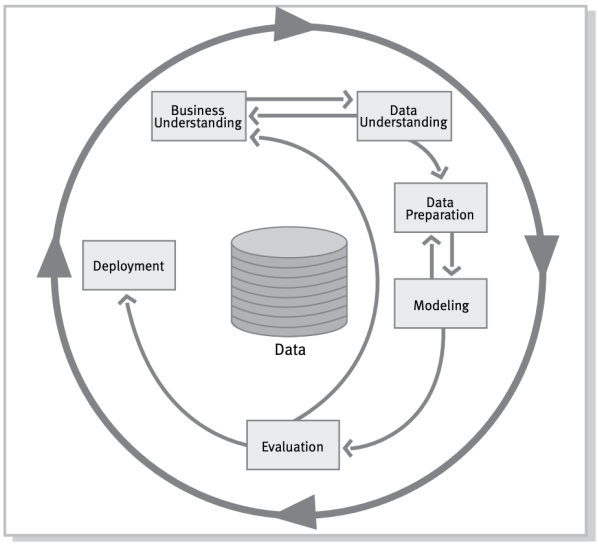

Figure 2: Phases of the CRISP-DM reference model. *Source: CRISP-DM formal documentation*

4 Data Analysis & Preparation

As part of the project, Expretio supplied us with the historical bookings data set which in turn had been obtained from the railway operator. It consists of structured time series data collected from June 2018 up to March 2020.

This section describes the details of the data in hand including its structure, understanding of both original and additional predictors. Feature standardization, methods for handling the missing data and the analysis of target variable, also its change over the time will be extensively discussed.

Note: *Taking into account the confidentiality agreement between the University of Edinburgh and Expretio, hereinafter the specific names of the features and resulting numbers will be renamed and hidden respectively.*

4.1 Data Understanding

The data contains 126557 and 1809 unique records for training and hold-out testing respectively. The original predictors (features) are shown in Table 1.

Predictor(Feature)	Data Type	Information	Example
Product	Categorical	Type of Product	Product 1
OD	Categorical	Origin-Destination of the itinerary	A-C
DBD	Integer	Days Before the departure	45
Capacity	Integer	Capacity allocated to product	165
Duration Hours	Float	Duration of the trip Origin-Destination	2.2
Train No	Integer	Train Identification Number	4567
Departure Date	DateTime	Departure Date of train in Y-M-D format	2018-12-01
Departure Time	DateTime	Departure Time of train in H:m:S format	06:30:01
Bookings(Target)	Integer	Number of Bookings	3

Table 1: Historical Bookings Data Set predictors, target and their descriptions

In total, we had 8 independent features that were used to predict the target regression variable. Whereof 2 categorical features, 2 date-time, and others numerical. Categorical features were encoded using one-hot encoding and simple replacement with numerical codes, e.g. Origin-Destination A-B = (1), A-C = (2), B-C = (3). Generally, we observed a performance improvement by approx. 1% whilst using one-hot encoding.

DNN models are sensitive to the scale of input data, therefore we first have to normalize it. For data standardization 2 methods were considered, (i) Standard Scaler that centralizes features by removing the mean and scaling to unit variance, (ii) MinMax Scaler that scales features using its highest and lowest values, and transforms it to have values between 0 and 1. Practically, the former yields better results and further used for all models.

Further, we will discuss time series analysis to better understand the target variable and variables that are used for predictions.

4.2 Time Series Analysis

Considering the time series structure of the data we in purpose chose to use the time series analysis instead of traditional EDA. The main reason is to explore how various statistical factors such as trend, seasonality, lags and holiday impacts the demand curve.

Typically, a time series is a sequence taken at successive **equally** spaced points in time, hence can be interpreted as a sequence of discrete-time data. However, there is a special case of time series called an unevenly (or unequally or irregularly) spaced time series. In our case, there is a sequence of observation time and value pairs (t_n, X_n) with strictly increasing observation times, but as opposed to equally spaced time series, the spacing of observation times is not consistent. Almost all statistical time series forecasting models like ARIMA, SARIMAX, including Facebook Prophet algorithm, require time-series sequence to have evenly distributed time steps. This means we have to aggregate the observations

into 30 minute, hourly or even daily levels. The aggregation helps to smooth out outliers and missing periods. However, once we aggregate, it cannot be re-sampled to obtain predictions given specific Departure Date/Time/DBD/Itinerary/Product. It is one of the weaknesses of time series models.

On the other hand, time series analysis is a combination of statistical methods for analyzing time series data in order to extract meaningful insights and the characteristics of the data. Namely, it provides statistical tools to learn how the current state of (t_n) is affected by previous steps ($t_{1...n-1}$) or their dependencies in different points in time. Considering our data structure with uneven time steps, the common way is to deal with is to apply various levels of aggregation, e.g. 30 minutes, hourly, and daily, in order to implement the analysis.

Let's have a look at Figure 3 that represents an average hourly aggregated booking number throughout the sample period of 23 months. We immediately observe how the Covid-19 pandemic affected the ticket sales looking at numbers from February to March of 2020, so there is a dramatic decline. On the other hand, there are two other periods that stand out. This can be clearly seen in the Figure 4.

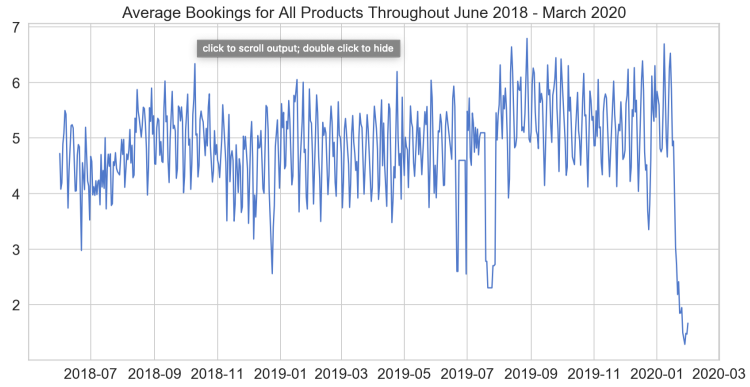

Figure 3: Average Number of bookings throughout entire period

Typically, the train schedules vary from time to time. During weekdays, trains run from 6:00 am in the morning up until 10:00 pm in the evening. Moreover, departure time is not consistent, e.g. this Monday trains might kick off at 6:00 am, but following week it might change to say 6:15. This adds up another complexity, but handled using aggregation that smooths out the periods.

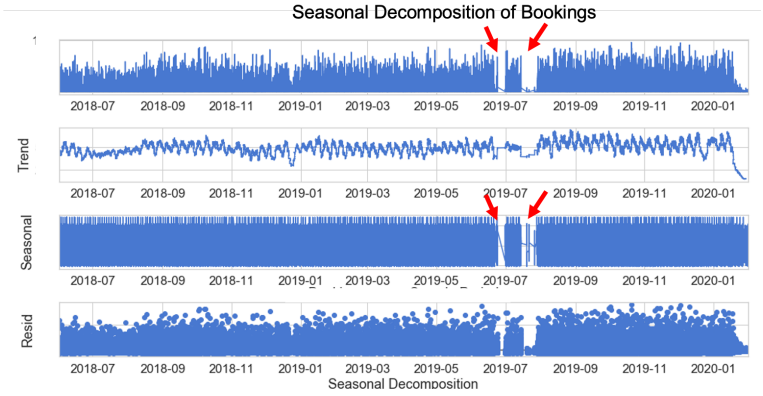

Figure 4: Seasonal Decomposition of Bookings Data

A few highlights of time series sample data analysis:

- Flat trend line: No strong evidence on an up or down trend, possibly taking a wider sample range would be helpful.
- Seasonality signs can be observed, better illustration in Figure 5. This means that customers buying behaviour tends to change in a similar way depending on what season or month the purchase is made, e.g in January people consistently tend to travel less than say in August.

- Also, there is a weekday effect on ticket sales, Monday to Wednesday we noticed uptrend whereas Saturday and Sunday are the days when people travel a lot less.
- No trains were running in period between 22nd and 30th of June 2019 inclusive.
- On days between 22nd and 25th of July inclusive no trains were running.

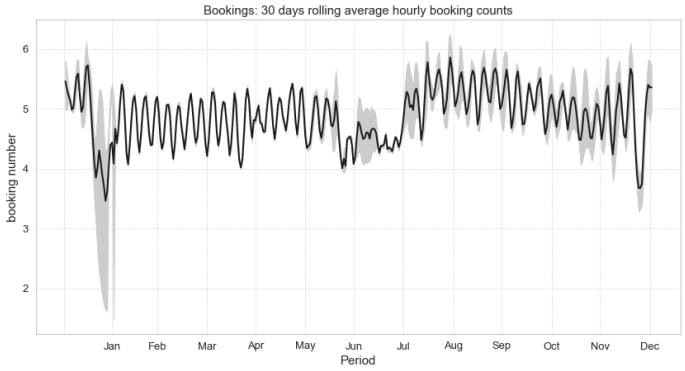

Figure 5: Rolling Average of Bookings Data. Clear signs of seasonality effect on sales.

On top of that, there were days when trains run only late trips.

There are many probable reasons for trains to not to run as scheduled. This might range from political, social decisions or business decisions, capital maintenance reasons, down to simple lack of bookings. But indeed, more important thing for us is in handling the missing data and outliers. The time series model's performance is highly affected by presence of missing data or outliers in data. Ignoring them might result in poor performance.

There are many ways of handling missing data, we tested a few of them to ensure minimal effect on models. As a note, simple removing them from our sample would not give anything, because whilst data aggregation, e.g. using either hourly or daily levels it will appear as missing anyways. Therefore, first method we used was **Linear Interpolation**, which simply fills in missing data with its estimated value using previous values in a sequence. However, it did not yield proper results, since the sequence of missing data was too big so forward filling failed. Another way was to use a time series model to train on past data to forecast missing data sample. We used FB prophet itself to fill in the missing days. As a result, we obtained a smoother data set to test our models.

Furthermore, having an unusual decline due to Covid-19 Outbreak has not had any effect on our results, since we used data from June 2018 to January 17th, 2020 for training and validation and, data from January 18th to January 31st, 2020 for hold-out testing. Including February and March of 2020 would possibly harm the models' performance, because of its unusual and strong decline.

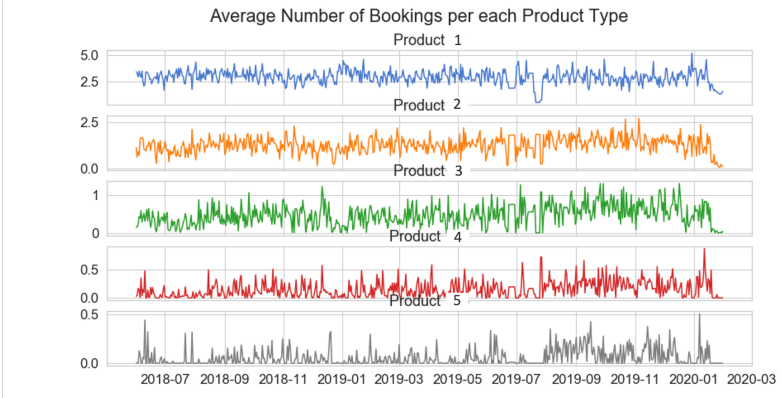

Figure 6: Average Number of bookings per Product type

By inspecting bookings patterns for each product type we observe that generally they share similar seasonality and flat trend, except variance in mean bookings. This is understandable, the majority of customers prefer to buy low-fare tickets if available. This phenomenon is well-documented in Castelli [8]. Briefly, the paper states that the price sensitivity(low or high) is the most determining factor of demand volume(number of customers).

Figure 7 illustrates how sales tend to change throughout the booking horizon or at different points of DBD. Interestingly, we notice that customers prefer to purchase tickets either too early, between 100-140 DBD or too close to the departure date. Whereas days between 75-12 show flat rates, not taking into account higher variance in former. In summary, the average bookings are not so high up to almost 12 DBD, before it jumps to 6 times higher average value. This might be a slight evidence that differentiates train customers from airline customers, since not too many people plan their train trip long before the trip.

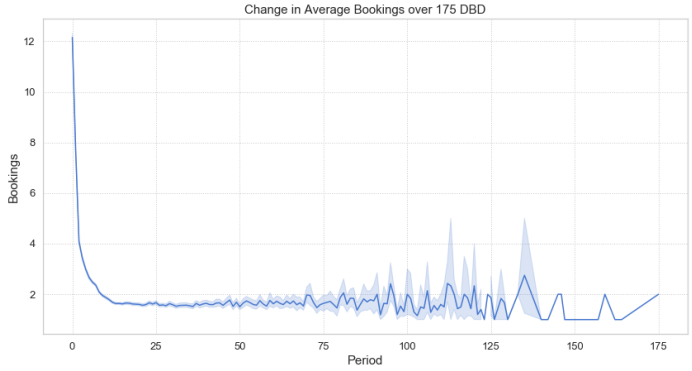

Figure 7: Change in average bookings over 175 DBD of booking horizon. Closer to the actual departure days, the numbers go up with high booking velocity.

4.3 Feature Engineering

In order to capture the dynamic nature of the problem, the original features shown in Table 1 are not sufficient. To train DNN models, we have to add features that better represent the time lags, speed of booking arrival and volume. This gives models more dynamic training. Below, we will describe what they are, how they derived and motivation behind them.

4.3.1 Statistical Features

In order to capture the dynamic nature of booking curve, data scientists from Expretio suggested to utilize *booking velocity*. The booking velocity is simply a cumulative sum of bookings for a specific product and itinerary collected on days before the actual DBD. For instance, to predict the number of bookings for Product 1 at point 3 DBD, we used the cumulative sum of bookings from 175 to 3, and fed it to the model. This added booking dynamics to the model, so we trained it to take into consideration how the number has been evolving over time. In practical context, this was done by adding another feature that calculates the sum of bookings over given DBD.

Another common method in deriving features from dependent variable in time series models is to inspect it using the autocorrelation function (ACF) and partial autocorrelation function (PACF) plots. These functions represent the correlation coefficients of time series with its lags. In other words, we want to understand which duplicate or lag of our current time step represents or maximally correlated with itself. In ACF, we consider steps and compare it to its lags independently. This might give misleading results since we are unsure if another set of variable explains their causal relationship. Whereas, the PACF calculates the degree of relationship between two random variables, but also with the effect of a set of controlling random variables removed. In other words, the "partial" correlation between two variables is the amount of correlation between them which is not explained by their mutual

correlations with a specified set of other variables. These plots helped us to identify what time lag to use in rolling and expanding mean features.

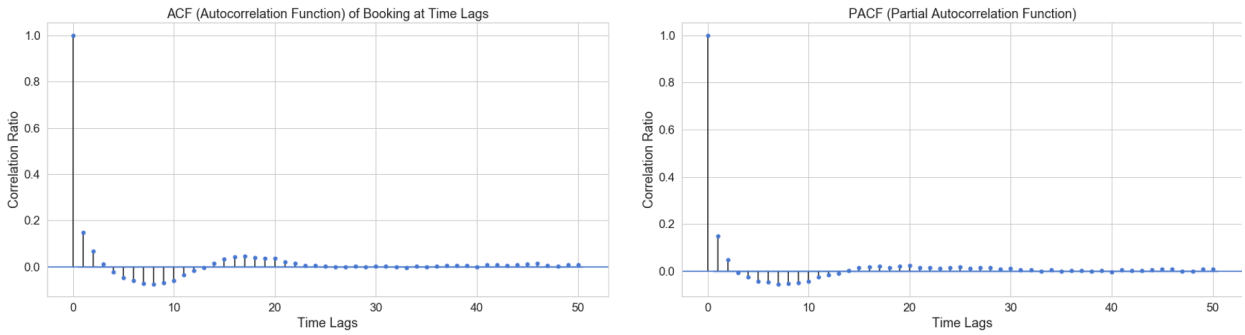

Figure 8: ACF and PACF analysis of time lags.

Both ACF and PACF plots in Figure 8 illustrate a small spike in correlation only at lag 1, others being even less, thus not statistically significant, meaning there is no effective lag that is able to explain the higher-order auto-correlations. However, a Moving Average and Expanding Mean features with lag 1 were included and further tested their importance in predictions generated by the model.

4.3.2 Date & Time Features

Whilst modelling time series task using mathematical-programming methods, such as DNN, the best practice is to extract additional features from Departure Date and Departure Time indices. These are year, month, day, hour and minute as feature columns. Moreover, as mentioned in the data analysis section, there is a clear signs of seasonality effect on sales, so therefore boolean weekday-weekend and quarterly features were added accordingly.

On top of that, by subtracting DBD from departure date we obtained another date time column to derive features. Although, majority of predictors extracted from DBD were similar to departure date-time, features such as Day of Booking was not correlated.

The final list of all new features along with data types are given in Table 2.

Feature	**Data Type**	**Feature**	**Data Type**
Cumulative Sum	Integer	expanding mean	Float
Rolling mean	Float	DepartureYear	Integer
DepartureQuarterOfYear	Integer	DepartureMonthOfYear	Integer
DepartureWeekOfYear	Integer	DepartureDayOfYear	Integer
DepartureDayOfMonth	Integer	DepartureIsWeekend	Boolean
DepartureDayOfWeek	Integer	DepartureIsWeekday	Boolean
DepartureMinuteOfDay	Integer	DepartureHourOfDay	Integer
y1 (Product 1)	Integer	**y2 (Product 2)**	Integer
y3 (Product 3)	Integer	**y4 (Product 4)**	Integer
y5 (Product 5)	Integer	**y6 (Total Bookings)**	Integer

Table 2: List of new features designed for DNN models and new tasks for MTL and multi-output STL.

For the sake of clarification, the last 3 rows in Table 2 are not independent features, but new dependent tasks (targets) $(y_1 ... y_6)$ designed for MTL and multi-output STL models. The detailed description of how described models will be using these tasks will be presented in the following Modelling Section 5.

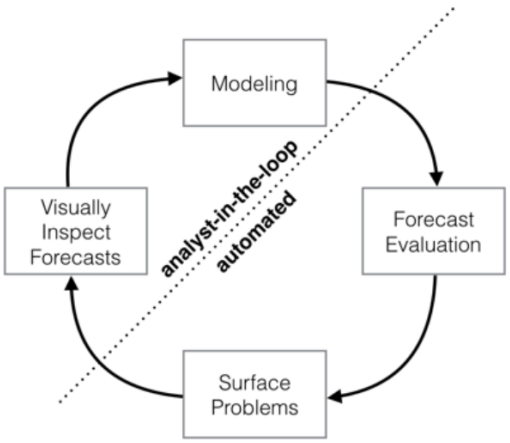

Figure 9: Schematic view of the analyst-in-the-loop approach to forecasting at scale, which best makes use of human and automated tasks. [27]

5 Modelling

This chapter expands the details of the proposed model to predict demand for a different type of products simultaneously. As previously stated two approaches were investigated, first being statistical Time Series (TS) approach, namely FB Prophet model with hourly and daily aggregation levels, second being MTL with DNN model.

5.1 Facebook Prophet Model

Recently announced by Facebook an open-source TS Prophet model has gained popularity amongst many business organizations helping with their strategic planning. As their formal paper states [27], the purpose was to ease the forecasting process and minimize human expert's intervention which is often expensive considering the shortage of time series analysts in the market. *Intuitive*: the authors are convinced any person having domain knowledge in specific business is able to adjust the model as of their specific needs. Implementation is available as open-source software in Python and R .

FB Prophet implements a Generalized Additive Model that models a time series as the sum of different components (non-linear trend, periodic components and holidays or special events) and allows to incorporate extra-regressors (categorical or continuous).

The main advantages of these model from other time series models:

- Parameters are easily interpretable;
- The measurements are allowed to be irregularly spaced, e.g. we can train the model to include only specific timeframes;
- Fitting is very fast - convenient to interactively explore and experiment with other model specifications, e.g. a Shiny application Chang [10].

5.1.1 Trend and Changepoints

There are 2 trend models that capture changes in TS: a saturating growth model, and a piecewise linear model.

For growth forecasting, the core component of the data generating process is to model how the number of bookings has grown and how it is expected to continue growing. Typical growth depends on some capacity constraints, in our case there is a population constraint. The typical logistic growth model's formulation is as follows:

$$g(t) = \frac{C}{1 + exp(-k(t-m))}, \tag{5.1}$$

where C the carrying capacity, k the growth rate, and m an offset parameter.

Two important aspects of growth that are not captured in the above equation. First, the carrying capacity is not constant – the number of population increases due to immigration or tourists, so does the growth ceiling. Thus in the Prophet model the capacity C is not fixed, but replaced with a time-varying capacity *C(t)*. Second, the growth rate is not constant. New products or itineraries can profoundly alter the railway operator's growth rate, and yet, Prophet allows to incorporate a varying rate in order to fit historical data by explicitly defining so-called *Change Points*.

Suppose there are S change points at times s_j, $j = 1..S$. Let's define a vector of rate adjustments δ, where δ_j is a change in rate that occurs at time s_j. The rate at any time t is then the base rate k, plus all of the adjustments up to that point: $k + \sum_{j:t\, s_j} \delta_j$, such that:

$$a_j(t) = \begin{cases} 1, \ if \ t \geq s_j; 0, \ otherwise, \end{cases} \tag{5.2}$$

where a_j is a vector $a_j(t) \subseteq (0,1)^S$.

The rate at time t is then $k + a(t)^T\delta$. When the rate k is adjusted, the offset parameter m must also be adjusted to connect the endpoints of the segments. The correct adjustment at change point j is easily computed as:

$$\gamma_j = (s_j - m - \sum_{l \leq j} \gamma_l)(1 - \frac{k + \sum_{l<j} \delta}{k + \sum_{l \leq j} \delta}) \tag{5.3}$$

Therefore, the piece-wise linear function takes the form of:

$$g(t) = \frac{C(t)}{1 + exp(-(k + a(t)^T\delta)(t - (m + a(t)^T\gamma))} \tag{5.4}$$

In our case, the capacity parameter in constrained modelling part was chosen as *C(t)* = 200 as a result of historical data analysis. The equation (5.4) allow us to manually add change points sparse priors δ. However, taking into account the lack of domain knowledge on, for instance, dates of new itineraries introduction or other growth-altering events in railway operator's history, the automatic change points control s_j parameter was chosen.

Importantly, a sparse prior on the adjustments δ has no impact on the primary growth rate k, so as δ goes to 0 the fit reduces to standard(not-piecewise) logistic or linear growth.

5.1.2 Seasonality

Business time series often have multi-period seasonality as a result of the human behaviours they represent. For instance, a 5-day workweek can produce an effect on a time series that repeat each week, while vacation schedules and school breaks can produce effects that repeat each year. To fit and forecast these effects we must specify seasonality models that are periodic functions. Prophet makes use of Fourier series to provide more flexibility to the model of periodic effects Harvey & Shephard [14]. Assume P to be the regular period we expect the time series to have (e.g. $P = 365.25$ for yearly data or $P = 7$ for weekly data, when we scale our time variable into days). We can approximate arbitrary smooth seasonal effects with:

$$s(t) = \sum_{n=1}^{N} (a_n \cos(\frac{2\pi nt}{P}) + b_n \sin(\frac{2\pi nt}{P})) \tag{5.5}$$

a standard Fourier series. To fit seasonality we have to estimate $2N$ parameters $\beta = [a_1, b_1, .., a_n, b_n]^T$. Prophet does the estimation by a matrix of seasonality vectors for each value of t in our historical and future data, for example with yearly seasonality and $N = 5$,

$$X(t) = \left[\cos(\frac{2\pi(1)t}{365.25}), ..., \sin(\frac{2\pi(5)t}{365.25}) \right] \tag{5.6}$$

The seasonal component is then:

$$s(t) = X(t)\beta \tag{5.7}$$

Since our model is a generative model $\beta \sim Normal(0, \sigma^2)$ is assumed to have Gaussian distribution.

5.1.3 Holiday Effect

Another major advantage of the Prophet is the ability to incorporate holiday dates in the model. Some holidays do not follow the exact periodic frequency, for instance, Thanksgiving on the fourth Thursday of November, or large sports events. For our case, we have incorporated the periodic holidays in the United Kingdom that have already taken place, see the Table 10.

Date	Holiday	Date	Holiday
01/01/2018	New Year's Day	17/03/2019	St. Patrick's Day [Northern Ireland]
02/01/2018	New Year Holiday [Scotland]	18/03/2019	St. Patrick's Day [Northern Ireland]
17/03/2018	St. Patrick's Day [Northern Ireland]	19/04/2019	Good Friday
19/03/2018	St. Patrick's Day [Northern Ireland]	22/04/2019	Easter Monday [England, Wales, Northern Ireland]
30/03/2018	Good Friday	06/05/2019	May Day
02/04/2018	Easter Monday [England, Wales, Northern Ireland]	27/05/2019	Spring Bank Holiday
07/05/2018	May Day	12/07/2019	Battle of the Boyne [Northern Ireland]
28/05/2018	Spring Bank Holiday	05/08/2019	Summer Bank Holiday [Scotland]
12/07/2018	Battle of the Boyne [Northern Ireland]	26/08/2019	Late Summer Bank Holiday [England, Wales, Northern Ireland]
06/08/2018	Summer Bank Holiday [Scotland]	30/11/2019	St. Andrew's Day [Scotland]
27/08/2018	Late Summer Bank Holiday [England, Wales, Northern Ireland]	25/12/2019	Christmas Day
30/11/2018	St. Andrew's Day [Scotland]	26/12/2019	Boxing Day
25/12/2018	Christmas Day	01/01/2020	New Year's Day
26/12/2018	Boxing Day	02/01/2020	New Year Holiday [Scotland]
01/01/2019	New Year's Day	17/03/2020	St. Patrick's Day [Northern Ireland]
02/01/2019	New Year Holiday [Scotland]		

Figure 10: The list of Holidays in the United Kingdom during sample period.

Incorporating this list of holidays into the model is made straightforward by assuming that the effects of holidays are independent. For each holiday i, let D_i be the set of past and future dates for that holiday. We add an indicator function representing whether time t is during holiday i, and assign each holiday a parameter k_i which is the corresponding change in the forecast.

$$Z(t) = [\mathbf{1}(t \in D_1), ..., \mathbf{1}(t \in D_1)] \tag{5.8}$$

and taking

$$H(t) = Z(t)k \tag{5.9}$$

So, this was done in a similar way as seasonality by generating a matrix of regressor. Similar to seasonality, we assume the prior k to have Gaussian distribution.Taylor [27]

5.1.4 Main Model

Combining all the previous additive terms, FB Prophet then uses decomposable time series model similar to proposed in Harvey [14] with the following equation:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \tag{5.10}$$

where *g(t)* is the trend function which models non-periodic changes in the value of the time series, *s(t)* represents periodic changes (e.g., weekly and yearly seasonality), and *h(t)* represents the effects

of holidays which occur on potentially irregular schedules over one or more days. The error term ϵ_t represents any idiosyncratic changes which are not accommodated by the model; as before the parametric assumption that ϵ_t is normally distributed Taylor [27].

5.1.5 Extra Regressors

As mentioned, Prophet makes it easy to incorporate extra regressors as an additive or multiplicative term to the linear part of the model. Using this characteristic, the original features discussed in Section Data Analysis & Preparation were fed into a model since they directly affect how the trend curve changes in time. The list of extra regressors are as follows:

- Duration Hours
- DBD
- Origin-Destination
- Capacity
- Product

Since we had to re-sample the data into hourly and daily time steps, the median/mean/mode of categorical and numerical features were taken. The main requirement that Prophet requires regressors to be present in both training and testing sets, in other words, we must know in advance the future values of regressors.

5.1.6 Results & Limitations

As a baseline model we used FB Prophet without any parameters' tuning, subsequently, adding holiday effects and as a last step combining with extra regressors. The results are given in the Table 3 below.

Metric	Baseline Model	Tuned Model	Regressor Model
RMSE	35.45	27.36	22.16
MAE	25.44	18.49	14.79
MAPE (%)	69.63	54.25	26.32

Table 3: FB Prophet comparison of model perfromance in different settings.

Moreover, looking at the Figure 11 we confirm that transforming the model to receive additional features as regressors can be easily adjusted and achieves more precise generalization.

As we observe, FB Prophet model can be well suited for most of the time series tasks and does not depend on neither univariate nor multivariate nature of the task. By using hourly or daily aggregation of uneven time steps in model, we obtained prediction values. This was helpful to identify how trends, seasonality and holiday influences the bookings. Also it allows us to understand which regressors store useful information and correlated with the dependent variable. This was further utilized to build a traditional machine learning model that is on the same level of aggregation. i.e. predicts values given date/time/OD/DBD features.

Concluding, FB Prophet's can be utilized to predict future bookings in regular spaced time series, but cannot handle the uneven time series. Another drawback is that unless we built separate models per product or itinerary, which makes maintenance and deployment cost-inefficient, it cannot be incorporated in the similar level of aggregation as of Expretio's operational requirements, thus cannot be compared to existing benchmark models in their usage. These are the main issues leading us to investigate ML models that allow to implement forecasting in a necessary format.

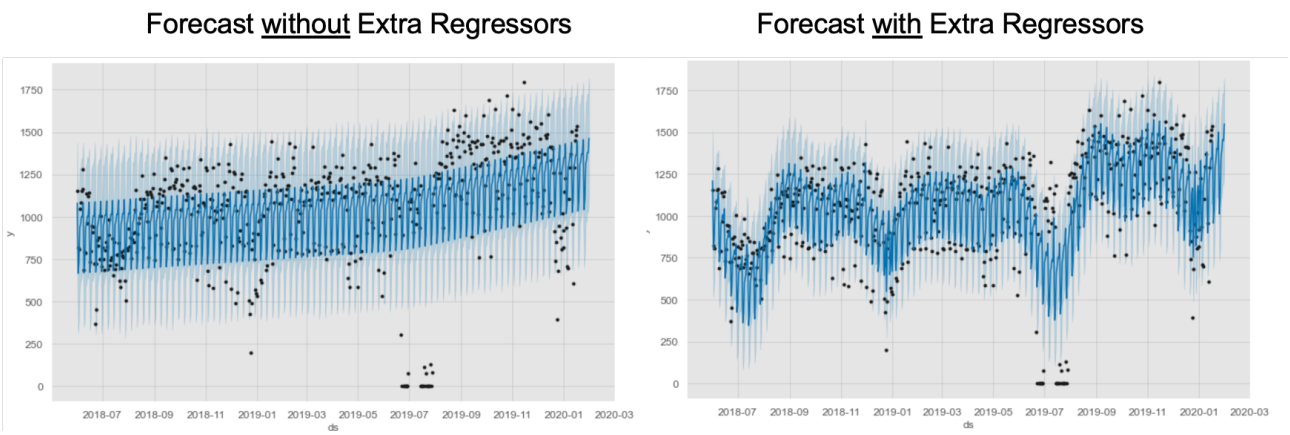

Figure 11: Illustration of FB Prophet model results *with* and *without* extra regressors.

5.2 Multi-Task Learning and Why It Can Work?

In ML tasks the objective is to optimize model using a particular metrics, i.e. minimize loss or improve accuracy. Normally it depends on business needs since any shift in percentage in metrics may lead to different levels of impact on profit or costs. Also, there is a performance benchmark that usually have taken into account while testing an approach or hypothesis. In our case study, the goal is to minimize the time series regression model's loss with respect to each passenger booking number using Mean Absolute Error (MAE) or Mean Squared Error (MSE). Mostly, it is a single model that used for one particular task and therefore has a single objective. So, there is a sequential process that includes fine-tuning the model parameters to obtain the minimum loss. The single output approaches usually yield good results and suitable for the majority of tasks, however sometimes, by focusing on one task we may disregard impacts on other tasks and potentially relevant information on them might be ignored. In Ruder [23], the author provides evidence on how multiple related tasks with shared representations or features might lead to better results and generalize well rather than a single task model. This information sharing and combining multiple outputs in a single model is defined as MTL.

MTL is in the family of transfer learning approaches. Sinno [15] described MTL as inductive transfer learning which aims to find a good feature representation to minimize domain divergence and classification or regression model error. MTL accomplishes this by using the domain information from one task and using it as an inductive bias to another. In his work, Ruder [23] explains the parameter sharing between multiple tasks as follows, *"MTL effectively increases the sample size that we are using for training our model. As all tasks are at least somewhat noisy, when training a model on some task A, our aim is to learn a good representation for task A that ideally ignores the data-dependent noise and generalizes well".* So, normally different tasks have their specific noise patterns, a model that learns two tasks simultaneously is able to learn a more general representation. Learning just task A causes the risk of overfitting to task A, while learning A and B jointly enables the model to obtain a better feature representation through averaging the noise patterns.

Formalized definition of MTL is given in Zhang [30], "Given m learning tasks $i, i = 1, ..., m$, where all the tasks or a subset of them are related, multi-task learning aims to help improve the learning of a model for task i by using the knowledge contained in the m tasks.

MTL has been successfully applied in Natural Language Processing (NLP), speech recognition, computer vision and drug discovery. However, in this dissertation, we extend its application to multi-target time series regression problem with structured data.

There are several perspectives on the reasoning behind the success of using MTL. From biological cognitive standpoint, it is somehow similar to how the human brain works and the way humans learn. Mostly, while learning a new thing, we keep using the knowledge learned from previous things and apply it to solve another task. For instance, learning to do one sport might help to quickly adapt to another related sport.

From a technical standpoint, the Inductive transfer used in MTL improves the model performance. For example, Ridge regularization results in inductive bias, where the preference is for sparse solutions

Ruder [23]. In terms of MTL, the inductive bias is created by auxiliary outputs. This might result in that the model will prefer hypotheses that explain multiple outputs instead of one output and therefore will generalize better.

Beyond aforementioned intuitive and theoretical reasons, there are some underlying mechanisms that were firstly discovered by Caruana [7]. Caruana was the pioneer in the field of multi-task learning, and he was the one, who introduced this technique. He presents four different mechanisms to provide evidence behind why MTL might work.

In practice, we often have training data that is not perfectly balanced in sizes. For instance, the target A might have a lot of data to train on, whereas B has a lot less. The MTL approach increases the training data for task B by transferring information from task A. This might result in an absolute advantage if we work with 2 closely related tasks.

5.2.1 Methods for Multi-Task Learning in Deep Neural Networks

By now, we have been emphasizing the theoretical reasons for deploying MTL and have not touched the actual ways of implementation, specifically in DNN context. In fact, DNN is simply MLP which has many hidden layers, so the standard approach to perform MTL in DNN is to use either hard or soft parameter sharing between hidden layers.

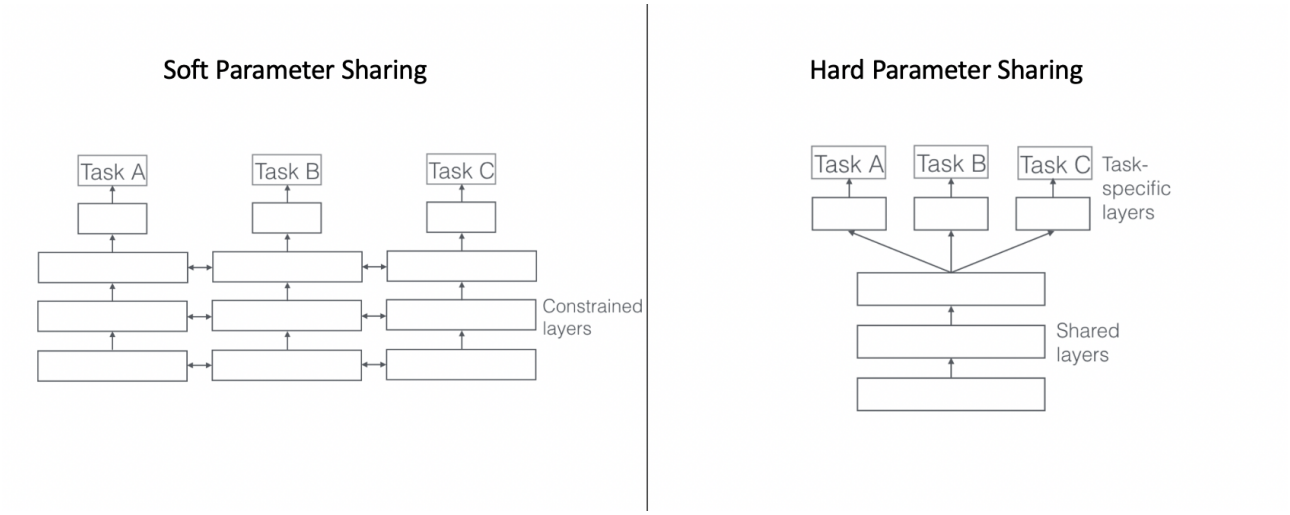

Figure 12: Soft and Hard Parameter Sharing in MTL DNN according to Caruana [7]

In case of **Soft Parameter Sharing**, each task has its own model with particular parameters. We add a constraint to encourage similarities among related parameters between the tasks. More specifically, we learn a model for each task and penalize the distance between the different models' parameters using l_2 norm or trace norm regularization for instance. This approach gives more flexibility for the tasks by only loosely coupling the shared space representations unlike hard parameter sharing that we will introduce.

Hard Parameter Sharing on the other hand, is the most commonly used type of MTL and was firstly discussed in Caruana's early works back in 1993. It is applied by sharing the hidden layers between each and every task, meantime keeping several task-specific output layers as shown in Figure 12.

Moreover, in hard parameter sharing one can specify weights to each task's loss. It helps to regularize the priorities, say, if obtaining a more precise model for Task A is more important than Task B. It has a drawback though, if the weight is chosen too large, other task's losses might lose relevance, and thus will perform poorly. Correct weight parameter selection is a hard task, so default equal weights normally are a good start. In this project, the default loss weights have been used.

MTL uses so-called **Attention Focusing** technique. For instance, if we have a data for a particular task which is very noisy, limited or high-dimensional, for STL it might be hard to distinguish among features whether it is important or not, whereas MTL can easily overcome it, since other tasks will provide additional evidence on the relevance of those features.

MTL's **Representation Bias** helps model to prefer representations that are commonly important for all other tasks. This will also force the model to generalize to new tasks in the future as a hypothesis space that performs well for a sufficiently large number of training tasks, it will also perform well for learning novel tasks as long as they are from the same environment Baxter [2].

5.2.2 Block-sparse regularization

In order to describe technicalities we will define it with mathematical notations. Suppose we have T tasks, where t represents one of them, and a model m_t with parameters a_t of dimensionality d. Then parameters can be written as a vector $\underline{a}_t$:

$$\underline{a}_t = \begin{bmatrix} a_{1,t} \\ . \\ . \\ . \\ a_{d,t} \end{bmatrix}^T \tag{5.11}$$

By stacking these column vectors $a_1, ..., a_t$ to form a matrix $A \in \mathbb{R}^{d \times t}$. The i-th row of A then contains parameter a corresponding to the i-th feature of the model for each task, while the j-th column of A contains the parameters $a_{.,j}$ corresponding to j-th model. Caruana [7].

There is a sparsity assumptions for parameters of models, so Argyriou [1] provides evidence that all models share a small set of features. In terms of our matrix A, it means that only a few rows are used as features whereas others being equal to 0, hence not used for all tasks. This is accomplished by enforcing l_1 regularization norm to the MTL setting. Considering the fact that l_1 norm constraints the some of the parameters, so only a few parameters left as exactly 0. This method is also known as least absolute shrinkage and selection operator (lasso).

In contrast to single task model where l_1 norm is computed based on vector $\underline{a}_t$ of the respective task t, for MTL we do it over all tasks, i.e. on all parameter matrix A. To compute it, we first compute the l_q norm for each row $\underline{a}_i$ that contains parameter for feature i across all tasks, so that the result is a vector $b = [\| a_1 \|_q ... \| a_d \|_q] \in \mathbb{R}^d$. Afterwards, we compute l_1 norm of this vector, forcing only a few rows of matrix A to be equal to 0.

Thus, depending on what constraints we choose to use for each row, we may want to assign different l_q. Generally, the term is known as mixed norm constraints of l_1/l_q norms, i.e. block-sparse regularization.Yuan [31]

Whilst block-sparse regularization is considered as plausible for many cases, it is very dependent on the extent to which the features are shared across tasks. [20] has shown that if features do not overlap by much, l_1/l_q norm regularization might actually be worse than element-wise l_1 regularization.

5.2.3 Multi-Task Model Architecture

The limited number of experiments have been done using a regular laptop with 3 CPU's and 8 GB memory. The best performed architecture in terms of both Mean Absolute Error (MAE) and Mean Squared Error (MSE) is presented in Figure 13. Neural networks were implemented using Functional API of Keras Library.

In total there are 22 inputs features to the model, whereof 2 categorical features were encoded using one-hot encoding, and others being numerical features. After input layers, the fully-connected dense layer of 256 neurons is placed. All dense layers have a 'ReLu' activation function and 'HeNormal' weight initializations. The relu activation uses the following formulate:

$$f(x) = max(0, x) \tag{5.12}$$

Where x denotes the input of a neuron. The 'HeNormal' initialization draws samples from a uniform distribution within [-limit, limit] where limit is sqrt(2/fan_in) and where fan_in is the number of input units in the weight tensor. For regularization a Ridge l2 regularizer and dropout is added to network. Then follows another fully-connected dense layer with 240 nodes. The network follows a diverge

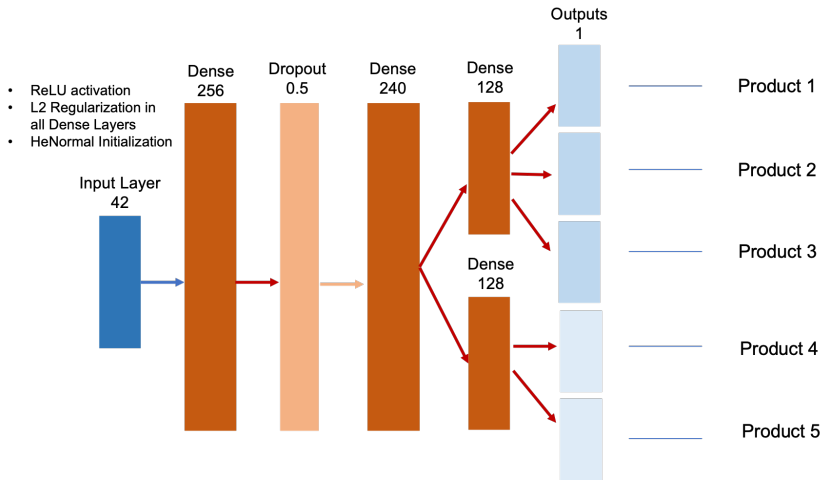

Figure 13: Multi-Task Learning Final Architecture.

structure and at the end, before each single output layer, a separate 2 dense layers of 128 neurons are placed.

5.2.4 Loss Function and Parameters

Losses: As aforementioned, choosing a specific metric for model evaluation is task-specific. In our case, the choice of proper loss function largely impacts the model performance. Typically for regression models, there are 2 popular loss functions used: MAE and MSE or more often Root MSE. Since loss is simply a difference between true and predicted values, MSE and MAE are then defined as follows:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$
$$MSE = \frac{\sum_{i=1}^{n} (y_i - x_i)^2}{n}$$

where y_i is a true observed value, x_i predicted value, n is a number of observations or data points that gives arithmetic average of error term.

While experimenting with losses, we noticed that training using MAE results in very low MAE, however, the actual predicted values are way lower than true values. As of analysis of historical data, we figured out the that the target Bookings vary from a minimum of 1 up to a maximum of 100. Since the difference is large and MSE has square term in the equation the difference gets more apparent. Whereas residuals in MAE contributes proportionally to the total error, the error grows quadratically in MSE. This ultimately means higher scale in booking values in our data will contribute greatly to total error in the MSE than it does for MAE. On the other hand in MSE case, our model will be penalized more for making predictions that largely differ from the corresponding actual value. In a nutshell, choosing MSE as an objective function provided more robust estimation and it was decided to utilize it in all DNN models.

In addition, Expretio has their own evaluation metric with additional Mean Absolute Percentage Error (MAPE) and Weighted Absolute Percent Error (WAPE) that also divides target into levels such as DBD, Origin-Destination, Departure Date and per Product. This gives more precise and complex evaluation of the models.

Regularization: In order to avoid model overfitting, *l2* regularization as well as dropout layer after the first embedding layer were included. The dropout layer randomly ignores or *drops* layers' output. Consequently, each update to a layer, while training, gives a different view of the configured layer. This makes the training process more noisy and forces nodes that are not dropped out to learn more of the inputs. The choice of former was greatly influenced by Caruana [7], whereas adding a dropout layer practically slightly improved holdout testing loss.

5.2.5 Feature Importance

Feature selection is considered as a good practice when we have a large data set and high feature dimensions. Feature importance using statistical and machine learning methods can help to automatically identify those features in our data that contribute the most to the dependent variable. There are several major benefits of performing feature importance testing prior modelling:

- Avoid Overfitting: Less redundant data provides a boost in performance of the model and decreases chances of making wrong decisions based on noise in data.
- Importantly, it greatly reduces training time since less data results algorithm to train faster.

The Chi-Square feature selection is a typical choice while selecting features. The χ^2 test is used in statistics to test the independence of two events or variables. Figure 14

	Feature	Score
17	cumsum	1.129514e+06
15	DaysBeforeDeparture	3.474432e+05
11	DepartureMinuteOfDay	5.364928e+04
16	PricePoint	3.535846e+04
21	DayOfYearBooking	7.804138e+03
1	DepartureDayOfYear	6.810958e+03
2	Capacity	6.582659e+03
13	DepartureWeekOfYear	9.845815e+02
6	DepartureHourOfDay	9.152558e+02
4	DepartureDayOfWeek	7.380401e+02
0	DepartureDayOfMonth	6.207931e+02
8	DepartureIsWeekend	5.431679e+02
9	DepartureIsWeekend	5.431679e+02
20	DayOfWeekBooking	2.170352e+02
12	DepartureMonthOfYear	1.994727e+02
7	DepartureIsWeekday	1.864425e+02
5	DurationHours	1.434566e+02
10	DepartureMinuteOfHour	1.074670e+02
3	TrainNumberString	6.331022e+01
14	DepartureQuarterOfYear	5.416191e+01

Figure 14: Feature importance by Chi Squared Test.

Secondly, sklearn provides built-in feature importance testing using ExtraTreeRegressor model, which implements a meta estimator that fits a number of stochastic decision trees on various subsamples of the data set. Table 15 shows the results of testing.

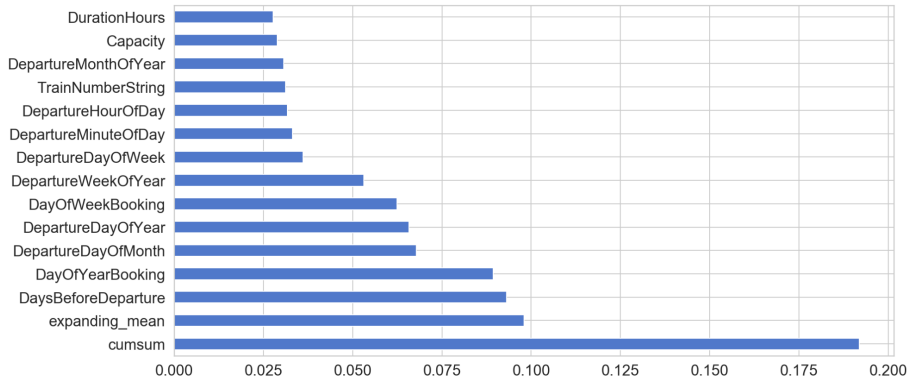

Figure 15: Feature importance by ExtraTreeRegressor.

Lastly, the python package Eli5 provides an excellent tool to understand black box algorithms such as DNN's. In case of feature selection, it measures how the score reduces when any feature is removed from the model; the method is also known as *"permutation importance"* or *"Mean Decrease Accuracy (MDA)"*.

Implementation is straightforward, it starts sequentially removing features from the data set, retrains the model and checks how the loss changes. It does it for every feature, so iteration is computationally intensive and also depends on the complexity of the model. We used original Multi-Output

Weight	Feature
0.4619 ± 0.0014	cumsum
0.2775 ± 0.0458	DepartureMinuteOfDay
0.1253 ± 0.0029	DayOfYearBooking
0.0592 ± 0.0168	DepartureDayOfYear
0.0149 ± 0.0008	DepartureWeekOfYear
0.0133 ± 0.0006	DaysBeforeDeparture
0.0020 ± 0.0002	DepartureDayOfWeek
0.0015 ± 0.0001	DepartureQuarterOfYear
0.0012 ± 0.0026	DepartureDayOfMonth
0.0004 ± 0.0001	DayOfWeekBooking
0.0002 ± 0.0000	DepartureIsWeekend
0.0002 ± 0.0000	DurationHours
0.0000 ± 0.0006	DepartureMinuteOfHour
-0.0000 ± 0.0000	expanding_mean
-0.0001 ± 0.0000	rolling_mean
-0.0003 ± 0.0000	DepartureIsWeekend
-0.0004 ± 0.0000	DepartureIsWeekday
-0.0049 ± 0.0007	DepartureMonthOfYear
-0.0123 ± 0.0018	TrainNumberString
-0.0207 ± 0.0007	DepartureHourOfDay

Figure 16: Feature importance by Eli5 permutation test using original Multi-Output STL model.

STL model which is in case the most relevant model to our task. At the end, it provides the permutation test results with an ascending level of feature importance (Figure 16)

Features importance order suggested by tests slightly vary, except Cumulative sum bookings, which consistently stay on top. DBD is also seen to be included in the list of most correlated features, whereas rolling and expanding mean features seems less important for DNN than for ExtraTreeRegressor.

Overall, feature importance tests provide powerful insights on how to proceed with modelling, thus had been taken into account.

5.2.6 Multi-Task Learning versus Single-Task Learning

In order to evaluate MTL's performance we have chosen two baseline models: 1) Single-Output and 2) Multi-Output STL. By 'Output' we here refer to Target Bookings that we have to predict. On the other hand, as mentioned, we have 5 different types of products. For simplicity, we can think of them as 5 fares or classes such as Economy Class 1, 2 and Business Class 1, 2 and say Premium Class. The traditional way of applying ML model is to have 1 single categorical feature with these 5 product types and the target variable Bookings that would be matched with each product.

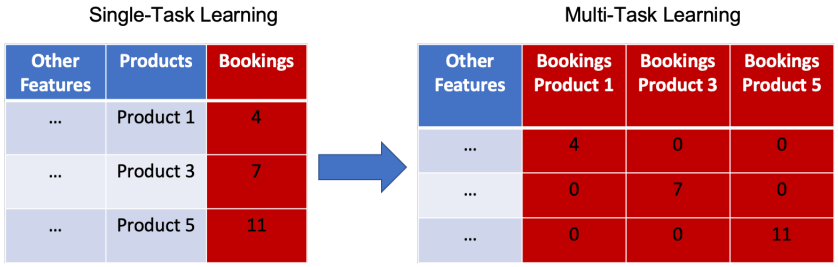

Figure 17: Transforming STL into MTL illustrated in matrix form.

Instead of using combined 1 Output, MTL model splits it into 5 different targets. (Figure 17). So that we have 5 tasks with 5 different loss-objectives to train and optimize the model parameters on.

The last days of Booking Horizon are the most important ones since it entirely inspects the model capabilities. As previously mentioned MTL does a great job in capturing periodical changes in targets, therefore the true and predicted booking curves almost lie on top of each other, except slight fluctuation while the Multi-Output STL showed the worst result from this standpoint.

As already mentioned, Expretio has its complex metrics that measure the model predictions from different levels. The full results that we obtained from all 3 models are presented in Table 4.

Analysing the results we observe that MTL model has beaten 2 STL models in 5 cases, showing outstanding precision in terms of handling the Departure Date and DBD. This means MTL adjusted

Category	Metric	Single-Output Model	Multi-Output Model	MTL Model
All Categories	RMSE	**6.44**	10.31	10.64
	MAE	**3.40**	5.99	6.05
	MAPE (%)	**53.63**	139.48	158.13
	WAPE (%)	**42.37**	74.51	75.41
Departure Date & DBD	MAPE (%)	**31.99**	35.62	34.10
	WAPE (%)	28.45	27.51	**17.55**
Origin-Destination	MAPE (%)	**22.10**	26.22	30.95
	WAPE (%)	27.41	**22.98**	34.82
DBD	MAPE (%)	21.19	25.58	**17.20**
	WAPE (%)	27.54	18.91	**10.91**
Departure Date	MAPE (%)	29.06	15.85	**11.14**
	WAPE (%)	27.41	16.09	**10.60**
Product	MAPE (%)	23.40	**10.40**	23.82
	WAPE (%)	27.41	**5.27**	19.21
Total Wins		**6**	**3**	**5**

Table 4: Models' comparison by target categories.

weights in such a way, so that it understands how the booking curve changes over time. Also, in terms of predicting values for specific product it did a good job as well, however way worse than Multi-Output model.

Figure 18: MTL and 2 STL models performance comparison, True versus Predicted values by models in terms of i) Days Before Departure, ii) Departure Date iii) Product Types

In terms of predictions by-product, we observe that Multi-Output STL performed the best. It can be explained considering the main aim of separating single Bookings Target into a combination of all products that are to obtain better differentiation from one another. Meanwhile, we observe that MTL resulted in good estimation as well.

6 Conclusions

The current dissertation aimed to explore new ways of overcoming several challenges in railway passenger demand forecasting task in cooperation with Expretio, a privately-owned corporation based in Montreal, Canada. Namely, two key issues have been raised: incorporating dynamic and unconstrained properties of demand forecasting task into a model. As suggested by Expretio, to address the dynamic nature of the task, it was decided to utilize so-called booking velocity, which simply calculates the cumulative sum of bookings at and before the certain DBD. Meanwhile, for unconstrained part, we have utilized the historical booking numbers, excluding the cancellation and no-show information. Overall, by analyzing the results we confirmed that it is sufficient and improves models' general precision.

Likewise, Expretio has had a good estimator in place, however along the way they were interested in discovering and experimenting with new approaches. Thus, the task was to sequentially get familiarized with existing benchmark models used in related industries, especially the airline industry. Upon discovering the most promising and applicable approaches, we had to research in detail the way it can be adapted into our specific case. Finally, the models performing the best according to testing on Expretio's complex metrics had to be inspected and analyzed.

Typically two approaches are used in tackling time series regression problems, the one is being statistical time series approaches such as ARIMA and its extended versions like SARIMA or SARIMAX, the second is traditional ML approaches including state-of-art Random Forest and XGBoost algorithms. The dissertation presented a comprehensive review of up to date models and their applications in various use-cases and described motivations behind choosing FB Prophet model as novel time series model and MTL model from the family of transfer learning ML models. FB Prophet has significant advantages in terms of simplicity in modelling (uses additive model), incorporating multi-variate nature in a model, computational cost as well as easiness in interpreting the obtained results. However, the main drawback was its make-future-function failed in handling irregular time series data. Thus required aggregating the data into evenly distributed either hourly or daily aggregation levels, which was not quite acceptable by Expretio's operational requirements.

On the other hand, despite numerous cases when Multi-Task Learning has been successfully applied to NLP and computer vision tasks, very few cases have been considered to apply MTL in tasks with structured time series data. Instead of combining all bookings for all product types and itineraries as it is done in STL, the MTL model partitions the task into multiple separate tasks. This helps to avoid model to overfitting to one particular task by introducing additional regularizations, also provides great flexibility in modelling hence benefits in a computational perspective. For instance, MTL in DNN context has separate output tasks for each of the product, and by some reasons, if we want to make changes in one particular product or task, because of its poor performance, MTL makes it easy to only change layers and nodes connected to that specific task. It is not required to re-train and re-assign the entire model and weights as a result it is quite efficient.

Moreover, the computational experiments carried out in python jupyter notebook confirm that there is a huge potential and scalability in deploying MTL in Expretio's demand forecasting tasks. Especially, MTL has shown excellent performance in adequately capturing date and time components of the task, beating 2 other STL models by in average 5% and 18% respectively. Tensorflow's Keras library provides Functional API module that helps to build MTL architecture, whereas Keras parameter tuning tools Hyperband and Random Search can greatly ease the automation process of model learning.

7 Further Work

For future research, given sufficient resources, it is recommended to include a rolling window validation to be more confident on the results of the tests. Rolling window validation refers to using various time windows such as, for instance using first 6 months of 2019 for training and testing on the following 3 months, then moving to the subsequent months and so on.

In addition, to further improve the proposed MTL model's performance, it is recommended to run Keras Hyperband or Random Search hyperparameter tuners on a larger search spaces. The target hyperparameters may include automatic detection of the number of optimal layers, nodes, activation functions ('normal', 'tanh', 'relu'), drop out rates and learning rates. Because of limited computational power on a macbook pro with 3 CPUs, we had to restrict the search space and still run learning for hours.

Functional API of Keras library is a powerful tool that allows to model any sort of DNN architectures. It is interesting to model a hybrid architecture, namely combining LSTM Sequential and Dense models. The former is well-known for handling structured time series tasks, whereas using an idea of MTL with separate tasks, it possibly can be enhanced.

In terms of feature engineering, examples of new features that has potential to add extra information are as follows: last time the train trip occurred, how many times a week does the itinerary takes place or how many of the similar destination are carried out on the same day.

On top of that, throughout the project, we were focusing solely on analysing time series on the actual train departure date. However, the dates when customers tend to book their trips have not been explored. Spending some time on understanding the behaviour of customers and how it changes over time would be an interesting idea.

References

[1] Argyriou and Pontil. Multi-task feature learning. *Advances in Neural Information Processing Systems*, 1(1):7–39, 2007.

[2] Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39, 2000.

[3] M. Beckmann and F. Bobkowski. Airline demand: an analysis of some frequency distributions. *Research Logistics Quarterly*, 5(1):43–51, 1958.

[4] G. Bitran and R. Caldentey. Commissioned paper: an overview of pricing models for revenue management. *Journal of Revenue and Pricing Management,*, 5(1):203–229, 2003.

[5] G. Box, G.E.P. Jenkins. Time series analysis: Forecasting and control. *Holden Day*, 27(1):2–32, 1970.

[6] E. Boyd and R. Kallesen. The science of revenue management when passengers purchase the lowest available fare. *Journal of Revenue and Pricing Management*, 3(1):171–177, 2004.

[7] R. Caruana. Multitask learning. *JOUR*, 28(1):41–75, 1997.

[8] P. R. Castelli L. and U. W. An airline-based multilevel analysis of airfare elasticity for passenger demand. *Proceeding of the 7th ATRS Conference*, 1(1), 2003.

[9] N. K. Catherine Cleophas, Michael Frank. Recent developments in demand forecasting for airline revenue management. *J. Revenue Management,*, 3(1), 2009.

[10] I. Chang, G. Tiao, and C. Chen. Estimation of time series parameters in the presence of outliers. *Technometrics*, 30:193–204, 05 1988.

[11] G. D. Chen V. and J. E. Solving for an optimal airline yield management policy via statistical learning. *Journal of the Royal Statistical Society Series C (Applied Statistics)*, 52(1):19–30, 2003.

[12] C. J. Chiang W. and X. X. An overview of research on revenue management: Current issues and future research. *Journal of Revenue and Pricing Management,*, 1(1):97–128, 2007.

[13] A. C. Harvey and N. Shephard. *Structural time series models*, volume Vol. 11:Econometrics, pages 261–302. North Holland, Amsterdam, (edited by g.s. maddala, c.r. rao and h.d. vinod) edition, 1993.

[14] S. Harvey, A. Peters. Estimation procedures for structural time series models. *Journal of Forecasting*, 1(1):89–100, 1990.

[15] S. Jialin. Transfer learning. *Nanyang Technological University*, 1(1), 2010.

[16] C. Lautenbacher and S. Stidham Jr. The underlying markov decision process in the single-leg airline yield-management problem. *Transportation Science,*, 33(1):136–146, 1999.

[17] A. Lee. Airline reservations forecasting probabilistic and statistical models of the booking process. *Massachusetts Institute of Technology*, 5(1):43–51, 1990.

[18] M. S. F. M. M. Mohie El-Din and A. A. Abouzeid. Airline passenger forecasting in egypt (domestic and international). *International Journal of Computer Applications*, 1(1):5–32, 2017.

[19] S. Nason. Forecasting the future of airline revenue management. *Journal of Revenue and Pricing Management,*, 6(1):64–66, 2007.

[20] Negahban and Wainwright. Model selection and estimation in regression with grouped variables. *Joint support recovery under high-dimensional scaling: Benefits and perils of $l_{1,\infty}$-regularization.*, 1(1):7–39, 2006.

[21] S. Pölt. Forecasting is difficult – especially if it refers to the future. *AGIFORS*, 33(1):171–177, 1998.

[22] F. R. Evaluating the forecasting performance of econometric models of air passenger traffic flows using multiple error measures. *International Journal of Foreacsting*, 27, 2011.

[23] S. Ruder. An overview of multi-task learning in deep neural networks. *Cornell University CS Journal*, 1(1):5–32, 2017.

[24] S. Sharif Azadeh, P. Marcotte, and G. Savard. Railway demand forecasting in revenue management using neural networks. *International Journal of Revenue Management*, 7:18–36, 01 2013.

[25] T. Shiina and J. R. Birge. Stochastic unit commitment problem. *International Transactions in Operational Research*, 11(1):19–32, 2004.

[26] K. Talluri and G. van Ryzin. Theory and practice of revenue management. *Journal of Revenue and Pricing Management,*, 6(1):64–66, 2004b.

[27] S. J. Taylor and B. Letham. Forecasting at scale. *Peer Journal*, 1(1):5–32, 2017.

[28] D. Walczak and S. Brumelle. Semi-markov information model for revenue management and dynamic pricing. *OR Spectrum*, 29(1):61–83, 2007.

[29] Z. H. Wei Ming, Yukun Bao and T. Xiong. Multistep-ahead air passengers traffic prediction with hybrid arima-svms models. *The Scientific World Journal*, 14, 2014.

[30] Q. Y. Yu Zhang. An overview of multi-task learning. *National Science Review*, 5(1):30–43, 2017.

[31] Yuan and Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 1(1):7–39, 2006.

[32] M. M. Švadlenka L. Sarima modelling approach for railway passenger flow forecasting. *Transport*, 33, 2018.