FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Zadání bakalářské práce

Název:	Predikce náhlého nárůstu zátěže na webovém serveru
Student:	Jan-Jakub Fleišer
Vedoucí:	Ing. Tomáš Vondra, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Některé webové aplikace zažívají mimo pravidelných změn řízených denní seasonalitou také náhlé špičky v návštěvnosti. Moderní aplikace běží na škálovatelné cloudové platformě, ale běžné reaktivní formy škálování reagují na přetížení se zpožděním a prediktivní škálování umí předpovídat hlavně denní křivky a nikoli špičky.

Proveďte analýzu dat o provozu ze serveru Novinky.cz, která zahrnují anonymizovaná ID jednotlivých zdrojů. Opravte případné anomálie v datech. Proveďte jejich rozklad na jednotlivé časové řady návštěvnosti daných zdrojů a zdroje roztřiďte na takové, které existují dlouhodobě a takové, které představují články s časově omezenou zajímavostí pro čtenáře. U dat představujících články popište způsob náběhu časových řad od začátku do prvního maxima. Navrhněte metodu, jak u nově vzniklého zdroje odhadnout s předstihem v řádu minut jeho maximum, pokud to z dat bude možné. Zjistěte, zda by tato metoda byla přínosná pro prediktivní škálování serverů dané zpravodajské aplikace.

Elektronicky schválil/a Ing. Karel Klouda, Ph.D. dne 6. ledna 2021 v Praze.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Prediction of sudden load increases in a web server

Jan-Jakub Fleišer

Department of Applied Mathematics
Supervisor: Ing. Tomáš Vondra, Ph.D.

May 9, 2021

Acknowledgements

I wish to express my thanks and gratitude to Ing. Tomáš Vondra, Ph.D., for the provided guidance and insight. In addition, I wish to voice my gratefulness for the support I received from my family during the time of writing of this thesis.

Declaration

Citation of this thesis

Fleišer, Jan-Jakub. *Prediction of sudden load increases in a web server.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

Tato práce prozkoumává prediktivní metody pro náhlé špičky pozorované na webových serverech zpravodajských aplikací pro účely autoškálování. Logy webových serveru zpravodajské aplikace jsou analyzovány a jednotlivé řady tvořeny zdroji v logu jsou kategorizovány dle chování. Poznatky získané z analýzy jsou využity jako podklad pro predikce časové řady webových požadavků se zaměřením na náhlé špičky za účely autoškálování v cloudovém prostředí. Prozkoumané modely a jejich chování jsou analyzovány a porovnány. Na závěr je provedena simulace cloudového prostředí pro porovnání autoškálovacích metod. Jsou porovnány metody založené na výsledných modelech a metody s reaktivními přístupy autoškálování. Výsledky jsou vyhodnoceny a je vyznačen potenciální směr pro další práci.

Klíčová slova analýza webových požadavků, autoškálování, hierarchické shlukování, hierarchické časové řady, online zprávy, predikce webových požadavků, predikce časových řad, časové řady

Abstract

This thesis explores the methods for forecasting around sudden peaks in web traffic for news applications and the usability of those for environment autoscaling. Real-world web traffic log data from online news application is analysed, and series formed by resource requests are categorised by behaviour. The insight gained from the analysis is used as a basis for web request time series forecasting approaches focused on sudden peaks for automatic scaling in cloud environments. The explored forecasting models are analysed by behaviour and compared. Finally, a cloud environment simulation is performed to compare the autoscaling method based on the resulting models. Comparisons are made to reactive autoscaling approaches and between each predictive model. The results are evaluated, discussed, and potential future work is outlined.

Keywords autoscaling, hierarchical clustering, hierarchical time series, online news, time series, time series forecasting, web request analysis, web request forecasting

Contents

1 Introduction **1**

2 Theoretical part **3**

2.1 Web-based news platforms . . . 3
2.2 Host environment scaling . . . 4
2.2.1 Reactive scaling . . . 4
2.2.2 Predictive scaling . . . 4
2.3 Web application sections as time series . . . 5
2.4 Time series characteristics . . . 6
2.5 Hierarchical Clustering . . . 6
2.6 Forecasting article popularity . . . 8
2.7 Akaike and Bayesian information criterions . . . 9
2.8 Forecasting methods for Time Series . . . 9
2.8.1 Naive method . . . 10
2.8.2 Average method . . . 10
2.8.3 Seasonal naive method . . . 10
2.8.4 ARIMA and ARIMAX models . . . 10
2.8.5 TBATS model . . . 12
2.9 Regression models . . . 12
2.9.1 Random Forest Regression . . . 13
2.9.2 AdaBoost and Gradient Boosting Regression . . . 13
2.9.3 Linear Regression . . . 14
2.9.4 K-Nearest Neighbours Regression . . . 14
2.10 Model error metrics . . . 15
2.11 Time series Residuals . . . 15
2.12 Time series cross-validation . . . 15
2.13 Bottom up approach to Hierarchical Time Series . . . 16
2.14 Cloud computing Queuing Theory Simulator . . . 16

3 Realisation **19**
3.1 Implementation environment and structure 19
3.2 Data origin 19
3.3 Data format & preparation 20
3.4 Initial analysis 20
3.5 Data correction 23
3.6 Downtime correction methods 23
3.7 Time series labelling, classification 25
3.7.1 Extracting features from the time series for clustering . 26
3.7.2 Clustering, methods and features 27
3.7.3 Clustering review & behaviour labelling 28
3.8 Post-labelling analysis 30
3.8.1 Long-living series analysis 30
3.8.2 Short-living series analysis 30
3.8.3 Growth in long-living series during short-living series peaks 32
3.8.4 Short-living series - Describing peaks 32
3.8.5 Peak-detection method 32
3.8.6 Parameter selection 33
3.8.7 Method shortcomings 34
3.8.8 First peaks in short-living series 34
3.8.9 First peak request drop-offs 36
3.9 Forecasting sudden increases 45
3.9.1 Model evaluation 45
3.9.2 HTS Forecast components 46
3.9.3 Long-living series forecasting 47
3.9.4 Short-living series forecasting 48
3.9.4.1 Regression for tracked series forecasting 48
3.9.4.2 Remainder series forecasting 54
3.9.5 Forecasting methods comparison 55
3.9.6 Local forecast analysis 57
3.10 Environment Scaling simulation 62
3.10.1 Simulation process 62
3.10.2 Simulation results 66
3.11 Discussion 70

4 Conclusion **73**

Bibliography **75**

A Project structure **79**

B Acronyms **81**

C Contents of enclosed CD **83**

List of Figures

3.1 Figure shows mean requests over day-time through working week and weekend. 21
3.2 Figure shows total hourly request sum over dataset lifespan with moving average of 24 hours. 22
3.3 Figure shows lag plot of the sum of all requests with 24 and and 168 hour lags, 1 day and 1 week. 22
3.4 Figure shows the corrected data comparison on two example series, the blue series shows corrected data using the larger mean window and the red using the smaller one. Original data showed zero requests in the corrected interval (Interval of the red series). . 24
3.5 Figure shows an example plot with cluster least and most popular series plots. The figure comes from lifespan features, ward linkage with scaled and normalised features and 12 total clusters; only three are shown in this figure. 29
3.6 Figure shows the share of requests for the 1000 largest peak-series, in descending order. 31
3.7 Figure shows two days, where a single peak-series exceeded its share of requests for more than 10% of the total number of requests. This leads to peak-series spikes being visible in the total sum of all requests, all while the long-living series did not experience growth at the same pace. 37
3.8 Figure shows the number of short-living series published by hour. 38
3.9 Figure shows the sum of all requests and sums of all short and long living series. 38
3.10 Figure explores the relationship of short and long-living series during peaks. The figure on the left compares the short-living series peak size with the requests in the long-living category measured simultaneously. The right figure compares the peak size with the percentual growth of long-living series during said peak over the mean measured at that time over the dataset. 39

3.11 Figure compares total sums of the two series groups (with detected peaks and without). 39
3.12 Figure shows a relationship between the total request size and the slope of the best-fit line until the first peak. The orange line is another best-fit line on the output data. The same is shown for the first peak size and best-fit slope. 40
3.13 Figure shows four examples of the peak-series startup detection method by moving averages. The green series represents the slow-moving average, the red series fast-moving, and the purple vertical line indicates where the start of a peak was detected. 41
3.14 Figure shows histogram with 5 minute interval bins for series startup in minutes. 42
3.15 Figure shows the empirical distribution function for peak occurrence in minutes from beginning of the series. 42
3.16 Figure shows minute interval counts of when a mean window value achieves a perceptual drop from the mean window value measured around the beginning of the peak on individual short-living series. 43
3.17 Figure shows total request size with percentage of requests out of the total series requests from peak until measured drop-off in mean. 44
3.18 Figure shows three sample peaks from the short-living series category with forecasts made by the GB regression model on a 1-minute sample rate. At each minute, a forecast is drawn. As the peak grows, the model attempts to adjust but fails to predict the peak end. The trend of decreasing requests after the peak is captured well. 53
3.19 Figure shows live forecast made by the ARIMAX model. 58
3.20 Figure shows 15-minute sample rate measured RMSE errors model comparison (Lower is better). 59
3.21 Figure shows 1-minute sample rate measured RMSE errors model comparison (Lower is better). 59
3.22 Figure shows two examples of 15-minute HTS Oracle and Real-time model behaviour around peaks. In the upper plot, the Oracle model underestimates the total number of requests, despite knowing the tracked series future values. 60
3.23 Figure shows mean of the first five forecasts made by the 1-minute models with 15-minute intervals. 62
3.24 The top figure shows incoming requests rescaled to match the total volume of data noted by λ. The figures below show selected methods for autoscaling for two simulation scenarios with 80 ms and 25 ms service demand, respectively. In the 80 ms scenario, the Mean HTS RT forecasts mostly lead to the same decisions as with the ARIMAX forecasts but seem more aggressive around peaks. This is a desirable characteristic. 69

List of Tables

3.1 Table lists features used in time series clustering. Features marked with * have been used by Polák in [30] 26
3.2 Measured growth in long-living series requests during peaks. 32
3.3 Table shows comparison of the distributions of the measured values across both considered series groups and remainder after the minimum requests criteria. 35
3.4 Long-living forecasts measured model mean RMSE values per sample rate. 48
3.5 Long-living forecasts measured model mean MSE values per sample rate. 48
3.6 Table shows the considered predictor and dependent variable lengths. 49
3.7 The table shows MSE and RMSE errors for the considered regression models on fifteen minute sample rate. 50
3.8 RMSE and MSE errors measured with each regression model on all testing data and testing data limited to peaks (P. MSE and P. RMSE). Models fitted on reduced peak-only data are marked with (P). 51
3.9 Selected orders for ARIMAX models used in remainder series forecasting. 54
3.10 Total sum of observed mean RMSE values on short-term forecasts. 55
3.11 Total sum of observed mean MSE values on short-term forecasts. . 55
3.12 Total requests forecasts measured model RMSE values per sample rate. 56
3.13 Total requests forecasts measured model mean MSE values per sample rate. 57
3.14 Table shows selected results for different autoscaling approaches grouped by type using 25 ms service demand. 67
3.15 Table shows selected results for different autoscaling approaches grouped by type using 80 ms service demand. 68

Chapter 1

Introduction

A common approach for application deployment is to do so in scalable cloud environments. Public or private cloud environments offer benefits over more traditional static infrastructure allowing for greater efficiency as the workload of applications is frequently not static. Scaling up or down allows the application to handle peaks as well as save resources when underused.

Modern online news platforms are no exception. In this thesis, real-world data is used and analysed. The flaws in the data caused by data collection issues are corrected, and the loss of information from the anonymised data is partially reversed as data is separated and labelled by behaviour. A method for predicting sudden surges in demand is proposed. Those sudden surges or peaks can be thought of as deviations from the expected demand based on long-term historical observation or forecasts.

The specific nature of online news applications and how users interact with them is kept in mind through the thesis, and the suggested methods and models are based on it, seeking to leverage the specific behaviour found in online news applications. The total traffic is decomposed by behaviour into individual time series, and an understanding of their behaviour is developed. Multiple approaches with time series models such as Hierarchical Time-series (HTS), ARIMAX, TBATS and Auto-Regressive models are used and combined with pure regression models. The dynamic nature of news platforms where certain items' impact varies through time, gaining importance or descending into irrelevance playing a significant role in the HTS models. The focus of this thesis is the short-term prediction that could be used in real-time. The method is then evaluated, compared, and selected methods are used in environment scaling simulations. The potential use to drive automatic scaling is then discussed based on the outcome of the simulation.

Briefly, the goals of the thesis are:

- **Analyse the request data -** Gain an understanding of the data and its nature, find patterns in the data and identify flaws that would hinder further efforts.
- **Correct the flaws in the data -** Correct the issues caused by outages that would prevent using the data for forecasting and improve its quality.
- **Label the data according to behaviour -** Identify the individual groups of data by series behaviour and domain context.
- **Analyse the individual labelled groups -** Gain an understanding of the patterns in the individual groups and describe their behaviour and potential consequences for forecasting.
- **Propose models for short-term forecasting -** Leverage the information gained from the labelling and analysis about the individual group patterns to propose a method for short-term forecasting.
- **Evaluate the proposed models -** Analyse the method performance for short-term forecasting and compare it with benchmark and state-of-the-art models for time series forecasting.
- **Discuss the usefulness of the method for news application scaling -** Discuss the potential usefulness of the forecasts for short-term environment scaling.

Chapter 2

Theoretical part

2.1 Web-based news platforms

Web-based news platforms allow users to access published articles about current events. In the case of Novinky.cz, an application from where the data used in this thesis was supplied, the news covers multiple topics, categories where articles similar in nature are grouped. Examples of such groups may be domestic news, world news, sports, weather, financial news and many others. These sections of the application tend to be stable through time, the categories being relatively generic and serving as an aggregation of articles. In contrast, articles published within these sections are more dynamic. Articles tend to be relevant only for a limited amount of time, the placement of an article is also dynamic, a featured article or an article on the homepage is likely to receive more attention which will decrease once it is eventually replaced as new articles are published [1].

Online news platforms are bound to experience sudden surges in demand. This effect is frequently observed with outside events such as political turmoil, celebrity deaths, terror attacks, natural disasters, and others that can draw readers to seek information. An example of this is given by [1] where a peak is observed and attributed to Hurricane Sandy. Such sudden surges may bring issues to the application as the environment gets overwhelmed and cannot provide service smoothly.

Online news platforms are nothing new and have been used since the early days of the internet. Although relatively old, the platforms have not stagnated, and as approaches to hosting and operation for web-based applications evolved, news platforms have been a part of this move. For example, Novinky.cz is hosted in a modern private cloud [2].

2.2 Host environment scaling

With the usage of scalable clouds, naturally, the desire to efficiently use its resources arises. From the client perspective, this is mainly motivated by costs and availability. Applications hosted on scalable clouds can react to demand and scale up or down the resources available; this can save operational costs and prevent the application from running out of resources [3]. In the long-term, this can mean planning ahead for reasonable utilisation through time in a matter of days and weeks, while in the short-term, this can be thought of as reacting to immediate or near-future need in a matter of hours. Failing to scale up may lead to an unsatisfactory experience for users, who frustrated with the given service, will choose competing platforms. Failing to scale down may result in unnecessary costs or cloud environment utilisation as resources could be allocated to others, or disabled [4]. With a focus on preventing, or at least reducing the negative effects of changes in demand, it becomes crucial to scale fast [4]; two scaling approaches are possible reactive and predictive.

2.2.1 Reactive scaling

In reactive scaling approaches, as the name suggests, decisions to scale are based on reactions to the current situation. When overwhelmed, the application and its resources will opt to scale up to relieve its current used resources, while underused applications may decide to scale down to free these resources to others and save costs [4]. Scaling is often done by in-advance prepared resource utilisation thresholds, which drive the reactive scaling when reached. Reactive scaling combined with non-instantaneous provisioning of computational resources may lead to a decrease in service quality until the new resources are available [4]. Furthermore, reactive scaling can be inefficient for web applications due to high fluctuations [3].

2.2.2 Predictive scaling

With predictive scaling approaches (sometimes referred to as proactive [3]), the focus is on predicting the future demand and needs of the environment and attempts scaling at appropriate times to meet this future demand in advance [4].

Predictive scaling attempts to forecast future load to drive in-advance provisioning of resources. Singh, Gupta and Jyoti [3] provide an overview on cloud workload forecasting as well as develop a short-term web-application forecasting approach using ARIMA and SVR models.

A combination of reactive and predictive scaling is desirable as predictive scaling may fail to forecast upcoming spikes. No reaction to these unforeseen events would have an undesirable effect on the quality of service [4].

An example of this approach is presented by Pereira, Araujo and Maciel [5] which introduce a hybrid autoscaler based on utilisation thresholds and model forecasting for future utilisation. The described autoscaler has two stages. First, it verifies if the monitored utilisation does not diverge from selected bounds. If that is the case, the autoscaler acts reactively. If the utilisation is within bounds, the second stage is considered. In the second stage, the multiple models are trained and tested on a collected sample of observations, the best performing model is then selected, and a decision is made to either scale up, down or maintain the number of instances. The proposed method outperformed the reactive alternatives by being more efficient in terms of utilisation [5].

2.3 Web application sections as time series

The load which a web application or, more specifically, the instance running the application is experiencing can be directly observed from hardware utilisation such as RAM or CPU usage through time. Those are metrics that reflect current and past situations and can be used in autoscalers. Additional metrics that can be used are disk usage, network usage, latency, and load balancer queue lengths [4]. Historical observations of those metrics contain information about the application behaviour and can be used to forecast future utilisation. An alternative approach is to observe the end-point nodes of the service and measure those.

An example of this could be the incoming HTTP requests. Such an approach could benefit from additional information or usage context of the application and potentially offer greater forecasting ability. With an understanding of the domain, a more educated approach can be made regarding forecasting and solving the problem.

In the context of web applications, each section, page, or article to be read must be requested by the client. This event is logged into HTTP request logs in the instances of the web servers. The structure of the log used by many web servers is the Common Log Format (CLF), although this can be modified as is the case for the Apache HTTP Server [6]. An example of a log entry in the CLF format is in the 2.1 listing.

Listing 2.1: Example CLF log file entry [6].

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /
    apache_pb.gif HTTP/1.0" 200 2326
```

In the provided example, the entry contains client IP, user-identifier, time, HTTP method, the requested resource, protocol, status code and object size.

Observing the requests over time, specific resources will be requested repeatedly. These repeated requests can represent the readers accessing a published article. A time series for each resource can be constructed by aggregating the requests by resource and summing them over a selected time interval. This time series will then reflect the popularity of the resource and impact over time. By summing the large number of observed per-resource series, the total number of requests through time can be observed.

2.4 Time series characteristics

The time series's nature can be described using settled terms, which give hints on the series's behaviour. Time series can be decomposed into components, each carrying information about a specific pattern from the original series. Decomposition can be additive or multiplicative, and this describes whenever the original data is decomposed as a sum or multiplication of components. The following list describes individual time series components:

- **Seasonal component -** Time series has seasonality when a reoccurring pattern with stable frequency in time can be observed [7]. Seasonal series in this thesis tend to show strong daily patterns.

- **Trend-cycle component -** Increasing or decreasing trend is observed when the time series shows long-term change, growth or decrease in observed values. The cycle pattern describes fluctuations in time without a stable frequency [7].

- **Remainder component -** The reminder is what remains from the original time series after the other components are removed [7].

Time series stationarity is another essential characteristic. A time series is stationary when it fluctuates around constant mean and with constant variance [7]. Some models may force constraints and require the fitted series to be stationary. Series with seasonality or trend is not stationary [7]. Time series can be turned stationary by differentiation, where differences between observations are computed and used as a new differentiated series [7]. Seasonal or multi-order differentiation may be necessary [7]. Time series characteristics are in greater detail described by Hyndman in Forecasting: Principles and Practice [7].

2.5 Hierarchical Clustering

With a large number of non-aligned time series of varying length, clustering based on the series themselves becomes difficult. For such situations, clustering based on features extracted from the time series can be used [8]. Usage of these features reduces dimensionality and allows for less constraining choice of algorithms [8].

The usage of hierarchical clustering for time series as well as comparison with other algorithms has been explored by Wang, Smith and Hyndman [8] which look at clustering of time series using their structural features.

Hierarchical clustering is an algorithm for clustering where each individual item is assigned its own cluster at the bottom-level and where higher-levels are formed by merging a pair of lower-level clusters forming larger clusters up to the desired amount of final clusters, or without limit, a cluster with all considered data at the top level [9, pp. 520–528]. Hierarchical clustering divides into two types, bottom-up (agglomerative) and top-down (divisive). For agglomerative approaches, clusters at lower levels are recursively merged to a single top-level cluster. In the case of divisive approaches, the process starts from a single cluster at the top-level from where at each level a cluster is split into two new clusters [9, pp. 520–528]. In this thesis, only the agglomerative type is used and discussed. For agglomerative clustering, the clusters to be merged G and H are selected by the smallest in-group dissimilarity $d(G, H)$, which is computed from a set of pairwise observations $d_{i,i'}$ with $i \in G$ and $i' \in H$ [9, pp. 520–528]. The considered implementation of bottom-up hierarchical clustering in scikit-learn [10] has multiple linkage methods determining the linkage criteria for merging of clusters [11]:

- **Ward linkage -** Minimises the squared sum of differences within all clusters [11].
- **Single linkage -** Least dissimilar pair (2.1) [9, pp. 520–528].

$$d_{SL}(G, H) = \min_{i \in G, i' \in H} d_{i,i'} \tag{2.1}$$

- **Average linkage -** Average dissimilarity between clusters (2.2, N_G and N_H is the number of items in each cluster) [9, pp. 520–528].

$$d_{GA}(G, H) = \frac{1}{N_G, N_H} \sum_{i \in G} \sum_{i' \in H} d_{i,i'} \tag{2.2}$$

- **Complete linkage -** Most dissimilar pair (2.3) [9, pp. 520–528].

$$d_{CL}(G, H) = \max_{i \in G, i' \in H} d_{i,i'} \tag{2.3}$$

The choice of a linkage method and the number of clusters to find can lead to many combinations to consider. Besides, the input data can be transformed to improve the performance of the algorithm, such transformations may involve scaling the data to a particular range or have the mean removed, and data re-scaled to unit variance [8]. Therefore, features subset, number of clusters, linkage methods and applied data transformations need to be considered.

2.6 Forecasting article popularity

A related topic to this thesis would be predicting the popularity of news articles as the traffic observed in the logs is driven by readership. Therefore, methods developed for predicting article popularity may help short-term prediction of web traffic extensively formed by articles.

Canneyt, Leroux Dhoedt and Demeester [12] have proposed methods for forecasting article popularity on additional features such as category, author, target audience, differentiating the source of incoming views from social media and direct views and others. Their method is not meant to be used for real-time forecasting but rather for getting articles' final popularity. Additional features are unfortunately not available in the data provided for this thesis.

Keneshloo, Wang, Han and Ramakrishnan [13] propose a method based on metadata, contextual, content-based, temporal and social features. Among the observed features were the page views and normalised page views time series, which were the most crucial feature in their (temporal) category, even though the category was not the best performing for total visit prediction.

Time series analysis can be used for articles, and the features acquired from the observed article series are helpful, although they are commonly combined with additional features for total views prediction as seen in [13]. Due to the similarity of the topic of total article view forecasts and article requests through time, the findings in those papers support the intention of forecasting articles based on the time series of request. Article forecasting, however, still poses a challenging problem, especially so as a cold-start problem. Nevertheless, even brief observations on readership shortly after the publishing of the article can lead to significant improvements [12]. These improvements suggest that the very beginning of the article series is of significant importance.

Such approaches to forecasting the total popularity using a large set of user behaviour features, as mentioned above, is a common topic receiving much attention [1]. This topic of final popularity forecasts is not directly usable in the context of short-term time series forecasts but provides insight into important articles' patterns and stages.

Castillo, El-Haddad, Pfeffer and Stempeck [1] observe articles initially receiving attention and shares on social media, something that rapidly deteriorates within a few hours after publishing. With this drop-off, the article visits are primarily explained by internal traffic (visits from within site), and the majority of the articles experience an initial peak followed by a decrease in visits in their first twelve hours. Articles with developing stories can serve as a minority counterexample where the visits are more steady through time.

2.7 Akaike and Bayesian information criterions

The Akaike (AIC) and Bayesian (BIC) information criteria are predictive accuracy measures. Models scoring the lowest values of AIC or BIC are generally preferred over other models [7]. The AIC is defined by the following equation 2.4,

$$AIC = T \log\left(\frac{SSE}{T}\right) + 2(k+2) \tag{2.4}$$

where SSE is the sum of squared errors 2.5, T is the number of observations and k is the number of coefficients for predictors [7].

$$SSE = \sum_{t=1}^{T} {e_t}^2 \tag{2.5}$$

The Bayesian information criterion is given as 2.6 [7].

$$BIC = T \log\left(\frac{SSE}{T}\right) + (k+2)\log T \tag{2.6}$$

Minimising the AIC or BIC information criterion is a good approach for selecting models for forecasting a particular series [7]. Models with minimal AIC values are often best for forecasting [7]. Minimising BIC should lead to a similar selection of the best model. The selected model is either the same as chosen by AIC or one with fewer terms [7].

2.8 Forecasting methods for Time Series

In this section, different forecasting methods and models used at different stages of this thesis are explored. Their primary functionality and usage is explained.

To evaluate and compare the resulting models, the Naive, Average and Seasonal naive models can be used as a benchmark. Their simplicity is not a limiting factor, and they can provide good forecasts for a large number of series, just as they can provide hints on the behaviour of more complicated fitted models [7].

Common notation for time series forecasting includes the indicator of time denoted as t. Forecast values are written as $\hat{y}_t$ and real-observed values as y_t. The usual meaning by forecast $\hat{y}_t$ is the mean of the possible values of y_t given known information $\mathcal{I}$ [7].

In the case of multiple h-steps ahead forecasts the notation is $\hat{y}_{T+h}$. If the information on which the forecast has been made needs to be specified $y_t|\mathcal{I}$ would be used, where $\mathcal{I}$ is the known information, for multiple steps ahead this is $y_{t+h|\mathcal{I}}$ [7].

2.8.1 Naive method

Forecasting using the naive method is trivial. The last observed value forms the new forecast and is repeated for the required number of steps ahead [7]. Equation 2.7 shows the naive method for n-steps ahead.

$$\hat{y}_{t+n} = \ldots = \hat{y}_{t+2} = \hat{y}_{t+1} = y_t \tag{2.7}$$

2.8.2 Average method

Forecasts with the average method are made with the repetition of the average value calculated over a selected number of historical values [7]. The forecasts for n-steps ahead with an average of T historical values is given by the 2.8 equation.

$$\hat{y}_{t+n} = \ldots = \hat{y}_{t+2} = \hat{y}_{t+1} = \frac{\sum_{i=0}^{T} y_{t-i}}{T} \tag{2.8}$$

2.8.3 Seasonal naive method

The naive seasonal method repeats observed historical values similarly to the naive method, but with a seasonal lag introduced so that the repeated data matches the current stage of seasonality [7]. This lag depends on the seasonality length. With daily seasonality, the forecast for midnight is the historical value from midnight of the previous day. The forecast at time $T + h$ is given by the equation 2.9 where m stands for the seasonal period and $k = \lfloor \frac{h-1}{m} \rfloor + 1$.

$$\hat{y}_{T+h} = y_{T+h-km} \tag{2.9}$$

2.8.4 ARIMA and ARIMAX models

The AutoRegressive Integrated Moving Average model is a time series forecasting model with multiple components. The autoregressive part of the model uses a linear combination of previous observations with the given order of p or shortly AR(p) [7]. The equation 2.10 gives an AR model with the order of p, with ε_t being white noise, c the mean of the changes of consecutive observations and ϕ being the model parameters [7].

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t \tag{2.10}$$

The moving average uses weighted observed forecast errors with an order of q, the model can be referred to as MA(q) [7]. Formally, the model is given by the 2.11 equation where θ refers to the model parameters [7].

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q} \tag{2.11}$$

The integration, a reverse of differencing, which is a process where the difference between successive observations are calculated as noted in the 2.12 equation [7].

$$y'_t = y_t - y_{t-1} \tag{2.12}$$

The number of applied first differences gives the order of d. Together, these components form an ARIMA model with an order of (p, d, q) with p order autoregressive part, d degree of first differencing, and q order of the moving average [7]. Altogether, the model can be written as in the 2.13 equation.

$$y'_t = c + \phi_1 y'_{t-1} + \ldots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} \tag{2.13}$$

The ARIMA model requires stationarity of the series [7]. To handle some non-stationary series, the ARIMA model can be extended with exogenous variables to form an ARIMAX model [14]. The exogenous variable can be used with Fourier terms to model the seasonal component of the series as given by 2.14 for $k = 1, 2, 3, \ldots n$ where m is the seasonal period [14].

$$\sin\left(\frac{2\pi kt}{m}\right), \cos\left(\frac{2\pi kt}{m}\right) \tag{2.14}$$

The ARIMAX model is then formally given as 2.15 where α_k and β_k are the exogenous variable coeffiecents [14].

$$y'_t = c + \sum_{k=1}^{K}\left(\alpha_k \sin\left(\frac{2\pi kt}{m}\right) + \beta_k \cos\left(\frac{2\pi kt}{m}\right)\right) + \sum_{i=1}^{p}(\phi_i y'_{t-i}) + \sum_{j=1}^{q}(\theta_j \varepsilon_{t-j}) \tag{2.15}$$

The orders of the ARIMAX models used in this thesis were selected by minimising AIC using the `auto_arima` process from the Pmdarima library [15]. The algorithm in `auto_arima` automatically performs stationarity and seasonality tests to determine the differencing order. A stepwise search is performed to find the orders of p and q, first a preset of models is fit. Out of this preset, the one with the smallest AIC is selected. From this model, additional models are explored where either one or both p and q orders vary by one. When a model with lower AIC is found, it becomes the new selected model. The process is repeated until no better models can be found or the process terminates due to the upper bound threshold for execution [15]. As for selecting the K for the number of Fourier terms, the above process was repeated for values of $k = 1, 2, \ldots, 11$, and the resulting model with minimal AIC was selected.

In the context of seasonal series forecasting, the SARIMA (Seasonal version of ARIMA) models can be used. They, however, do not appear in this thesis due to the large observed seasonal periods with small sample rates data as SARIMA models tend to struggle with such periods [14].

2.8.5 TBATS model

The TBATS model stands for Trigonometric terms, Box-Cox transformations (2.16), ARMA errors, trend and seasonal components. TBATS is a fully automatic model capable of handling long non-integer seasonality [16]. With TBATS models, the seasonality of the series can change through time. This differs from harmonic models where seasonality is static [16].

$$y_t^{(\omega)} = \begin{cases} \frac{y_t^{\omega}-1}{\omega}, & \omega \neq 0 \\ \log y_t, & \omega = 0 \end{cases} \tag{2.16}$$

The TBATS model can be described as in 2.17, where M are the seasonal periods,

$$y_t^{\omega} = l_{t-1} + \phi b_{t-1} + \sum_{i=1}^{M} s_{t-m_i}^{(i)} + d_t \tag{2.17}$$

l_t 2.18 and b_t 2.19 are the local and global trend respectively,

$$l_t = l_{t-1} + \phi b_{t-1} + \alpha d_t \tag{2.18}$$

$$b_t = (1-\phi)b + \phi b_{t-1} + \beta d_t \tag{2.19}$$

d_t is the Auto-Regressive Moving Average ARMA(p, q) process 2.20,

$$d_t = \sum_{i=1}^{p} \phi_i d_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \tag{2.20}$$

$s_t^{(i)}$ 2.21 are the Fourier-like seasonal terms, with $\gamma_1^{(i)}, \gamma_2^{(i)}$ being smoothing parameters and k_i being $\frac{m_i}{2}$ for even values of m_i and $\frac{m_i-1}{2}$for odd values of m_i [17].

$$s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)} \tag{2.21}$$

$$s_{j,t}^{(i)} = s_{j,t-1}^{(i)} \cos\left(\frac{2\pi j}{m_i}\right) + s_{j,t-1}^{*(i)} \sin\left(\frac{2\pi j}{m_i}\right) + \gamma_1^{(i)} d_t \tag{2.22}$$

$$s_{j,t}^{*(i)} = -s_{j,t-1}^{(i)} \sin\left(\frac{2\pi j}{m_i}\right) + s_{j,t-1}^{*(i)} \cos\left(\frac{2\pi j}{m_i}\right) + \gamma_2^{(i)} d_t \tag{2.23}$$

2.9 Regression models

Regression models are a group of models utilised in predicting quantitative outputs [9]. This section covers various regression models appearing in this thesis, relatively simple models such as Linear Regression, which makes an appearance in both analysis and forecasting, more complicated ensemble model of Random Forest and boosting models AdaBoost and Gradient Boosting, all appearing in forecasting, are described.

2.9.1 Random Forest Regression

Random Forest (RF) is an ensemble model consisting of decision trees built from training samples drawn with replacement [18]. First, each decision tree is constructed by best-split of all or a subset of features on each level [18]. Then, the output of the individual trees is combined to form the final output. In the case of regression problems, the resulting dependent variable is the mean of the trees in the ensemble [9, pp. 587–603].

What is a decision tree has not yet been explained. A decision tree is a model with a tree structure where the input sample travels from the root up until one of the leaf nodes [19, pp. 250–256]. Each encountered node has learned information about the best possible split given its set of features and processes the incoming samples accordingly.

The main parameters of RF models are the number of estimators (trees in the forest) and features to be considered for the best-split during tree construction [18].

Random Forests are a popular model able to achieve remarkable results [9, pp. 587–603]. An example related to web application load forecasting in this thesis is the usage of RF regression models in the context of short-term electricity load forecasting as done by Dudek [20] where RF outperform ARIMA and ANN models.

2.9.2 AdaBoost and Gradient Boosting Regression

The Gradient Boosting model, together with AdaBoost, fall into a category of boosting models. The concept of boosting is based on the idea of combining several weaker models, sometimes referred to as weak learners, together in order to create a better final model [19, pp. 130–142]. The weaker model may perform just slightly better than a random guess, and the aggregation of such weak models can result in amplification, and a more complex and competent predictor [19, pp. 130–142]. Such an approach may reassemble the previously mentioned Random Forest model. The critical difference is in how the models are combined.

For the AdaBoost algorithm, a sequence of weak models is given the training set with changing weights. The weights change as the samples pass through the sequential models. A poorly evaluated sample by a model in one stage is assigned a more significant weight, forcing succeeding models to prioritise it [19, pp. 130–142]. The resulting prediction of AdaBoost is a linear combination of simpler hypotheses [19, pp. 130–142].

In Gradient Boosting, the failures of previous models are not identified by weights but rather by the gradient of the loss function from the previous step [9, pp. 337–384]. The initial step is initialisation to the optimal constant model. This is the creation of an initial single terminal node that minimises the loss function on the data. The next step is iterative. A negative gradient of the loss function is computed for each observation based on the previous prediction. Then, a regression tree is fit on the residuals (referred to as pseudo residuals) [9, pp. 337–384]. Next, the values of the terminal nodes of the fitted tree are determined. This value is given as a value that minimises the loss function on observations falling into the terminal node with respect to the previous prediction. In the final stage of the iterative step, the predictions for each sample are updated. This is given by adding the sum of all nodes' output values to where the sample belongs, multiplied by the learning rate to the previous prediction. Once the iteration is finished, the resulting model is a combination of the initial and sequential tree models multiplied by the learning rate [9, pp. 337–384].

2.9.3 Linear Regression

Linear Regression is a linear model with a form as defined by 2.24, where $X^T = (X_1, X_2, \ldots, X_p)$ is the input vector and $\beta_0, \beta_1, \ldots, \beta_j$ are the model coefficients [9, pp. 43–56].

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{2.24}$$

The model coefficients are estimated from the training data. While multiple methods exist, the least-squares method is the most popular method for coefficient estimation [9, pp. 43–56]. In the case of least squares the coefficients $\beta = (\beta_0, \beta_1, \ldots, \beta_j)$ are estimated by minimising the residual sum of squares (2.25) [9, pp. 43–56].

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2 \tag{2.25}$$

2.9.4 K-Nearest Neighbours Regression

The K-Nearest Neighbours (KNN) model finds k-closest observations, neighbours to the provided sample and bases the prediction on them [21]. Commonly used for classification, it can also be used for regression problems. In the case of regression, the prediction is based on a weighted average of the selected k-neighbours [21]. The neighbour weights can be left uniform or scaled by distance from the observation [21].

2.10 Model error metrics

To compare models and evaluate their performance, metrics that indicate the goodness of forecasts are needed. Such metrics can be the Mean Square Error 2.26 or Root Mean Square Error 2.27 [7]. Both RMSE and MSE are scale dependant metrics that can not be compared on series with different scales [7].

$$MSE = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2 \tag{2.26}$$

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2} \tag{2.27}$$

2.11 Time series Residuals

Residuals are the remainder from the observed series after a model is fitted. This can mean the difference between real observations and forecasts $e_t = y_t - \hat{y}_t$ [7]. Observing the residuals after a model is fit can provide information about how well the model fits the data and whenever it can be improved [7].

Looking at the mean of the residuals can reveal bias in the fitted model. Non-zero means suggest that the model is biased [7]. In addition, it is desirable that residuals are normally distributed and have constant variance [7].

2.12 Time series cross-validation

Evaluation of model forecast accuracy in order to be reliable needs to be observed on data the model has not been fit on [7]. One approach is to split the available data into testing and training sets, fit the model on the training set and evaluate the forecast accuracy on the testing set [7].

An alternative approach is cross-validation, where the forecasts evaluation is done with rolling origin. Time series cross-validation is an approach to time series model evaluation which, instead of providing a static split between training and testing sets of data, performs multiple evaluations [7]. The training set consists of all prior observations, and the test set is formed by a single observation ahead, or in a case of multi-step ahead, multiple observations [7]. Each testing set is evaluated, and a step forward in time is taken, and the process is repeated until the end of the available testing data. Measuring MSE and RMSE values and their mean over errors in the rolling testing set offers a suitable method for selecting the best performing forecasting model [7].

2.13 Bottom up approach to Hierarchical Time Series

Hierarchical time series allows forecasting series forming predefined hierarchical relationships described by a tree structure. The parent series are formed as the aggregate of its children series. In this thesis, the bottom-up approach to HTS forecasting is used. With this approach, the bottom level series are forecasted individually, and the forecasts are aggregated to form a top-level forecast according to the defined hierarchy.

Formally, HTS can be described by the following equation,

$$y_t = S * b_t \tag{2.28}$$

where S is the $n{*}m$ summing matrix defining the bottom-level aggregation, y_t is an $n-$dimensional vector including bottom level and aggregated forecasts and b_t is an $m-$dimensional vector with bottom level forecasts.

$$y_t = \begin{bmatrix} y_{total,t} \\ y_{A,t} \\ y_{B,t} \\ y_{AA,t} \\ y_{AB,t} \\ y_{BA,t} \\ y_{BB,t} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} y_{AA,t} \\ y_{AB,t} \\ y_{BA,t} \\ y_{BB,t} \end{bmatrix} \tag{2.29}$$

In the example 2.29 equation, the hierarchy can be described as a symmetrical tree where the root has two children, each with two leaves. Where $y_{A,t}$ indicates forecasts at time t for the left child on the first level and $y_{AB,t}$ the forecast at time t node second from the left [7].

The usage of Hierarchical Time Series is not new in forecasting web traffic. An example can be a case study by Hauskrecht, Liu and Yan [22], which introduce an improved framework for HTS and its usage in web traffic forecasting.

2.14 Cloud computing Queuing Theory Simulator

A simulator would be needed to evaluate the usefulness of the resulting forecasts in this thesis for cloud environment scaling. The simulator of choice was developed by Vondra [4] and is based on Queuing Theory. The simulator, written in R, comes along with a set of auto-scaling functions, approaches which based on the observed utilisation, latency, request-queue length, and a hybrid combination of those provide a decision making logic for scaling the environment at each step in time. Each approach to auto-scaling has to be evaluated on multiple criteria and trade-offs. Among those criteria are machine-hours, starts and stops of the instances and metrics user satisfaction with interactive services.

Machine-hours refers to the total number of hours of running instances [4]. Starts and stops of the instances refer to the number of instances that were stopped or started during the observed time, where a large number may suggest that the used method is unstable. It may also be undesirable as some providers have costs attached to starting and stopping instances [4]. The way user satisfaction is quantified is by using the Apdex formula (2.30) applied to time intervals [4]. Apdex is a response time formula based on splitting the users into groups, satisfied, tolerating and frustrated, where thresholds T and F are set to divide them, T is set by the user and F is defined as $F = 4T$ [4].

$$Apdex = \frac{Requests_{satisfied} + \frac{Requests_{tolerating}}{2}}{Requests_{all}} \tag{2.30}$$

Apdex can split into conservative and strict approaches with the setting of T. In the conservative approach, it is between one to three seconds, while in the strict approach, it is slightly above the average response time [4]. The final metric used in the simulator is the number of time intervals where Apdex (Conservative - 1 second) was below 0.95 (users seeing significant latencies) and the percentage of intervals and where Apdex (Strict) was below 0.7 (the system was overloaded) [4].

The simulator runs in a stepwise fashion where each step is fifteen minutes long. During each step, the scaling logic is consulted for scaling decisions in the environment. If a decision is made to scale up or down, the fifteen-minute step offers substantial time to allow for additional machines to be fully started. A necessary parameter that needs to be specified before the simulation is service demand. This parameter is defined as the lower bound on the service time of a request given by its computational demand [4].

The available simulator methods used for comparison have parameters *up* and *down*, which provide boundaries in the threshold-based implementations. To provide an example, in the utilisation based approach, the *up* parameter refers to the environment utilisation boundary needed to scale up. Once the value is above, a decision to scale is made. The simulator offers a parameter sweep mode which allows finding optimal choice for the threshold parameters. The available autoscalers are listed below:

- **Utilisation based autoscaler -** Threshold-based approach focused on environment utilisation. The utilisation based approach tends to show larger machine-hours during low utilisation and has an increased chance for Service Level Objective (SLO) violations [4].

- **Latency based autoscaler -** Threshold-based approach focused on request latency [4].

- **Queue length based autoscaler -** Threshold-based approach focused on average load balancer queue length. The queue-based autoscaler tends to show more stability with fewer starts but with a larger number of machine-hours [4].
- **Latency-queue hybrid autoscaler -** Threshold-based approach combining latency and queue length. The latency is used for scaling up while queue length is used for scaling down [4].
- **Latency-utilisation hybrid autoscaler** - Threshold-based approach combining latency and utilisation. The latency is used for scaling up, and utilisation for scaling down [4].

Chapter 3

Realisation

3.1 Implementation environment and structure

The analysis, labelling and forecasting in this thesis were conducted in individual Jupyter notebook files, each covering different thesis stages. The notebooks themselves contain all code, models, analysis and forecasts and chronologically walk through the stages. The notebooks and their content is described in the appendix A.

The language of choice in the notebooks is Python 3.7. For data manipulation and analysis the Numpy [23], SciPy [24] and Pandas [25] libraries were used. For data visualisation, the selected library was Matplotlib [26] library enhanced with Seaborn [27] for a more high-level interface allowing for faster creation of visually appealing plots. As for forecasting, the R language was considered as it offers very mature packages for time series forecasting such as the forecast [28] package, but the language and libraries struggled to deal with the relatively large data size. For this reason the Python libraries of Statsmodels [29], Pmdarima [15], Scikit-learn [10] were used for forecasting instead.

The previously mentioned R language played a role later, as the used cloud environment simulator is written in it, a jupyter notebook adaptation of the simulator code was created with minor modifications to fit the template.

3.2 Data origin

The data has been collected by Seznam.cz on the web servers of Novinky.cz site. It accounts for from 6.25% to 12.5% of the requests logged in the environment, varying through time. The dataset spans from 12.06.2019 to 06.08.2019, 55 days in total.

3.3 Data format & preparation

The data comes in a reduced form of the Common Log Format for web servers split into multiple files. The regular columns for client address, user-identifier, HTTP status code and byte size have been removed. The remaining information consists of the request's date and time, the HTTP method and a hash of the requested resource name. The data provider has done this censorship as a means to remove sensitive information from the dataset. Example entries are shown in the 3.1 listing.

Listing 3.1: Example entries in the provided data.

```
[23/Jul/2019:06:30:26+0200] "GET e93fa2ab48a1d7f395028dbb379c406b"
[23/Jul/2019:06:30:26+0200] "GET e93fa2ab48a1d7f395028dbb379c406b"
[23/Jul/2019:06:30:26+0200] "GET 094fb9abe3ee6c1a78c2a6da1801f672"
```

The data has been transformed into CSV (comma-separated value) format and unified into a single file for easy reading and parsing by concatenating the individual files by a shell script.

3.4 Initial analysis

In total, the 64,993,615 requests in the data consist of 1,791,756 unique items (hashes), out of which only 33.32% appear more than once, and only 3.47% of items appear more than ten times. This low recurrence suggests that the number of items with significant traffic is much smaller than the total number of unique items, most receiving little to none. On average, there are 21,719 requests daily belonging to hashes, never to appear again, hashes with less or equal to 10 requests have a daily mean of 53,160 requests and hashes with 100 or less have 76,272 requests.

The requests had to be aggregated and summed over time intervals to work with the data as a time series. Multiple time intervals have been used in the thesis; the most common would be 1, 5, 15 minute and hourly intervals, with an interval being a sum of all specific item (unique hash) requests over the selected period. For example, a 15-minute interval would be formed by the sum of requests from 00:00:00 to 00:14:59, followed by 00:15:00 to 00:29:59 up until the end of the series. The sum of requests over a selected interval then forms a single observation, and the frequency of the observations is referred to as a sample rate. Data summed over one-minute intervals leads to a one-minute sample rate with each observation tied to a specific minute in time.

When looking at the dataset as a whole, either by a sum or by mean over time, clear daily seasonality with no apparent trend can be observed. This seasonality reflects people's behaviour interacting with online news, interest

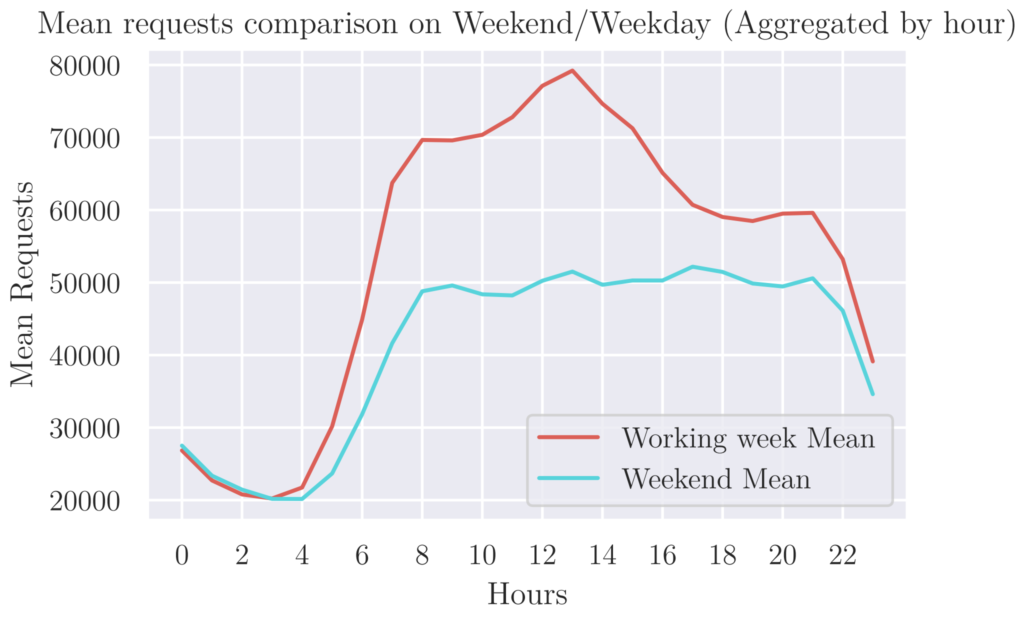

Figure 3.1: Figure shows mean requests over day-time through working week and weekend.

peaking during the afternoon and diminishing at the end of the day. As shown in figure 3.1 apparent scale differences between the working week and weekends have been observed, the latter seeing lower interest. This difference suggests a potential weekly seasonality.

The initial analysis revealed some concerns about the data. First, the scale of the collection as a part of daily traffic differs across dates and, secondly, it contains deployment events in the environment, which leads to gaps in the dataset. These concerns will have to be addressed. Both the gaps and variety in scale can be observed in figure 3.2

The figure 3.3 shows a lag plot analysis with y_{t+24} for correlation with hourly lagged request sums of one day and with y_{t+168} for a lag of one week. In both cases, the figures show correlation and alongside the patterns observed from the plots of the total sum serve as further evidence for daily and weekly seasonality.

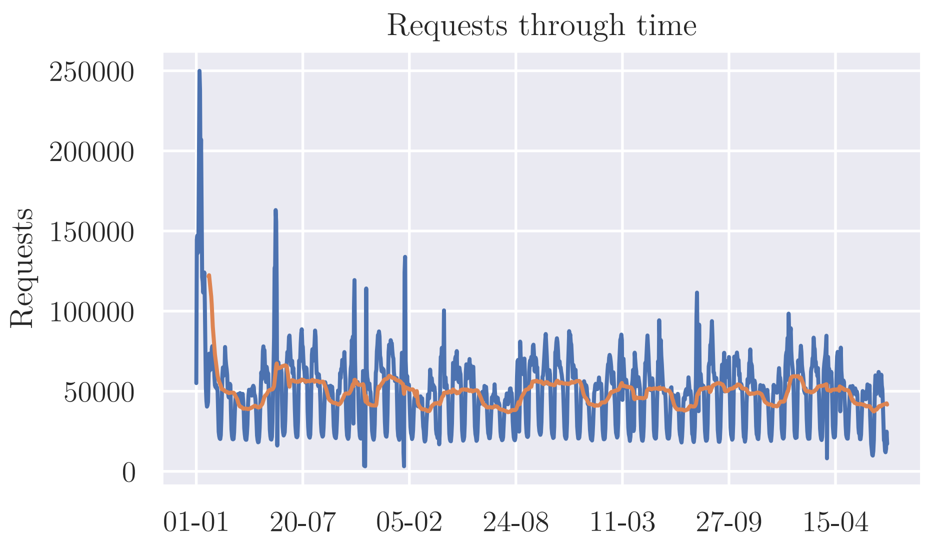

Figure 3.2: Figure shows total hourly request sum over dataset lifespan with moving average of 24 hours.

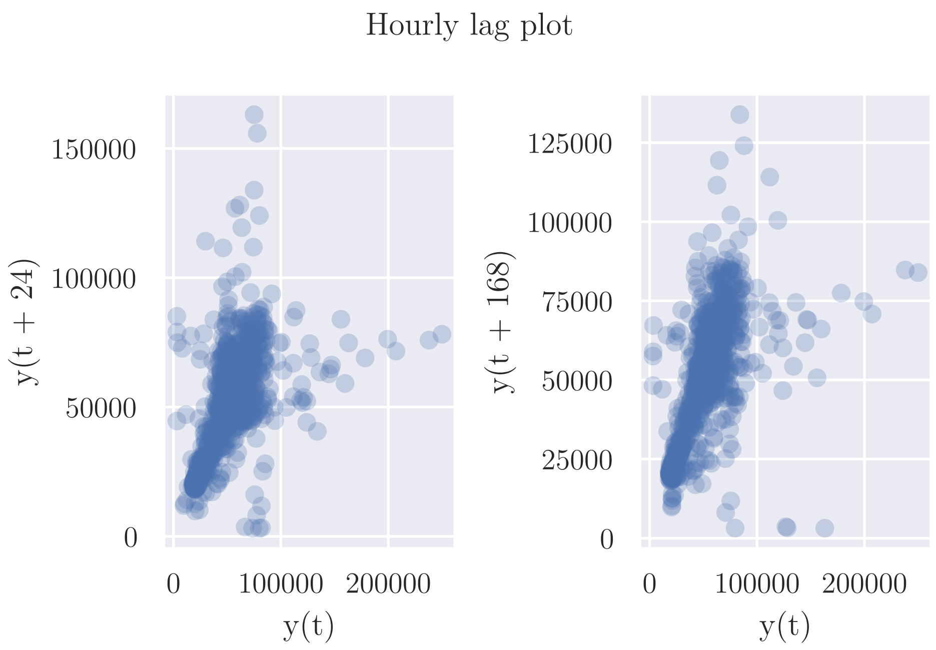

Figure 3.3: Figure shows lag plot of the sum of all requests with 24 and and 168 hour lags, 1 day and 1 week.

3.5 Data correction

In the time span of 12—13 June 2019, the data collected accounted for 12.5% of the entire activity in the environment. This period is at the beginning of the dataset and is the only time where it was so. Other days accounted for 6.25%. To prevent this irregularity from affecting further efforts, the range starting from the earliest request on 12.06 up until 6:30 AM the following day has been removed. As up to that moment, the irregularity could be observed.

In addition, downtime caused by deployment events has affected the environment. These events have happened on 01.07.2019 (Starting from 6:30 AM and lasting for 24 hours), close to the middle of the observed range, and 06.08.2019 at the very end. Both took a day to resolve. This gap leads to disturbance in data continuity. Data at the end, starting from 05.08.2019 at 6 AM, has been removed.

Removing the downtime on 01.07 was not an option as it would break the continuity of the data. Therefore, a different method had to be used.

In addition to outages and known differences in the scale of the data collection, a particular type of anomaly can be observed. These anomalies take the form of a sudden decrease in the volume of incoming requests, frequently preceded or followed by a peak in requests, and they tend to be short-lived (few minutes to roughly forty-five minutes) and have varying length. There is no clear pattern. In the case of outages, no requests were coming in, while in this case, requests did come in, just in a noticeably smaller volume.

3.6 Downtime correction methods

Two methods have been used in this thesis's scope, at different times, one replacing the other once it was found to provide better general results. To reduce the number of series affected by the correction, both methods attempt to modify only those series, where an argument can be made that they have been affected by this downtime. The series had to show activity before and after the downtime on 01.07.2019 to be considered for correction.

Both methods look at intervals of a specified size in time before and after the gap. Then, they compute a mean of those two intervals and replaces the missing values with the newly computed data. To provide an example, missing values for 3:00 PM would be replaced with the mean of 3:00 PM from the previous and following days.

The methods differ in the window size over which the mean is computed. The first one uses seven days before and after the gap, and the mean is computed from fourteen total days. With the second method, the interval size is reduced to a day before and after. The motivation to reduce the interval size in the second method is to improve performance on the series experiencing non-repeating patterns, influencing the mean. When an irregularity,

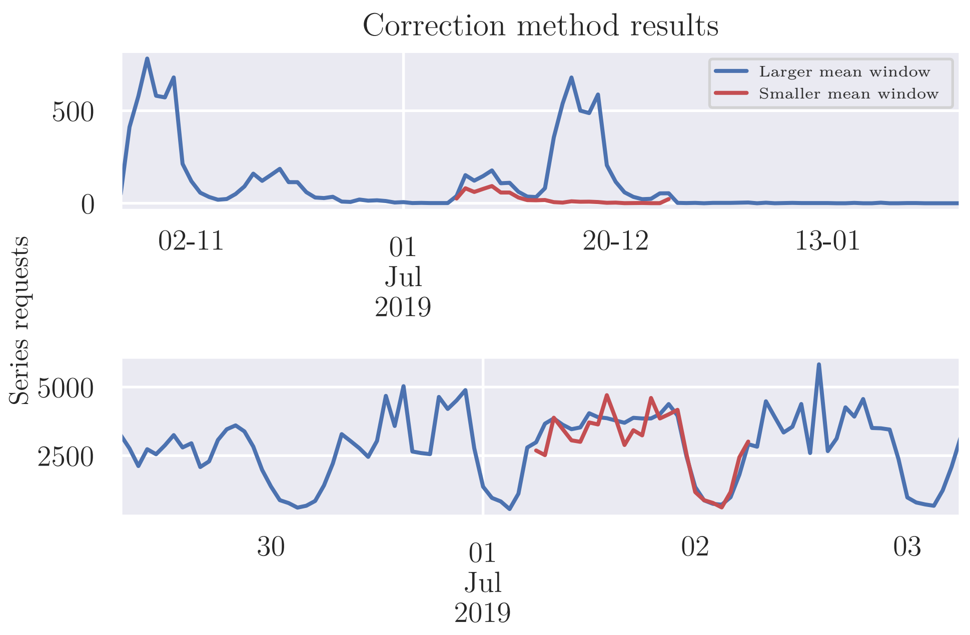

Figure 3.4: Figure shows the corrected data comparison on two example series, the blue series shows corrected data using the larger mean window and the red using the smaller one. Original data showed zero requests in the corrected interval (Interval of the red series).

frequently in the form of a sudden surge of requests, is observed within the correction window, it strongly influences the newly computed corrected data.

The main flaw with the methods is lack of robustness, a series that contained a significant peak over a short period of time, either before or after the downtime, has seen its gap filled by another smaller peak, which is caused by the mean values collected over the entire lifespan being significantly affected by the prominent existing peak, this was despite the likely-hood of such peak in the gap occurring being very slim. On the other hand, seasonal series mostly did not suffer from this phenomenon as irregularities are few and less impactful. This behaviour can be observed in the 3.4 figure.

The effect an observed irregularity has on the series with a smaller window is potentially more significant as the larger mean window moderates the irregularity impact better than a smaller one. Decreasing windows size might seem counterproductive, but with the smaller window, the number of series repeating the irregularity is significantly smaller as the irregularity is less likely to be within the selected window.

It is essential to note at which stages of the thesis the methods were employed. The first method was used initially. The data corrected by this method has been used in the labelling stage of the thesis, which sees as its goal to split the series by their behaviour and provide basic labelling. Further stages used the second method with a smaller mean window. Part of the labelling stage contained efforts to visually inspect the individual series revealing the flaws of using a large correction window. Once this stage has been completed, there was nothing to be gained by re-doing the mostly manual effort using the succeeding method. The series were already labelled, and the methods or the process itself are not the thesis's point.

As for the previously mentioned irregular anomalies, when a correction using means of future and previous observations was attempted on the individual series level, the results were not good, and this approach did not work on these outages. The cause of these anomalies remained unknown. Therefore, the data was not modified until a later stage in the thesis, where a sum of the series was used.

3.7 Time series labelling, classification

The series, stripped of information about their nature, forced a need to spend considerable efforts to provide useful labelling for further work. With the relationship between the series and request addresses gone, the ability to tell which time series was an article, discussion page, news section, or other also disappeared.

Under ideal conditions, the classification method could be built around knowledge of the resource address, its nature being part of a well-defined structure. In the form, as the data was received, no such classification is possible. Only speculations can be made. The most popular series covering the entire span of the dataset and showing consistent daily seasonality are good candidates for the web's individual news sections (such as weather, sports, local news or others). Series displaying peaks in interest over a short span of days disappearing into obscurity seem to match articles' expected behaviour.

The idea of working with articles and section had to be abandoned in favour of more universal labels. For this, two basic labels have been used:

- **Short-living peak dominated series -** Series with a small number of dominating peaks and otherwise uninteresting static behaviour.
- **Long-living seasonal series -** Series that display resolute daily seasonality through the majority of their lifespan.

With the desired labels defined, the series were split into these groups based on their behaviour using hierarchical clustering, the process of which is the topic of the following few sections.

3.7.1 Extracting features from the time series for clustering

To provide data for feature-based clustering of the time series. Features with origin in statistical properties, local extremes, auto-regressive model parameters, lifespan, minute interval counts, and differentiated time-series statistical properties were calculated. Many of those have been identified by Polák in Server Load Clustering [30]. The features are listed in the table 3.1.

Table 3.1: Table lists features used in time series clustering. Features marked with * have been used by Polák in [30]

Feature	Description
Non-zero occurrences	15 Minute sample-rate observations with non-zero requests
Occurrences	Total amount of observations in the series
Non-zero ratio	Ratio of zero to non-zero observations
Mean*	Mean requests over observations
Std*	Standard deviation of the series
Skew*	Time series skew
Sum	Sum of all requests
3 Largest values	Largest observations by amount of requests
95% Confidence interval	Series 95% Confidence interval.
Local maxima above 95% CI*	Number of local maxima observations above the 95% interval.
Akaike information criterion*	Model selection criterion
Bayesian information criterion*	Model selection criterion
Auto-regressive model intercept*	The intercept of a fitted auto-regressive model.
Auto-regressive model lag values* (0, 2, 4, 8, 12, 24, 47 hours)	The parameter values of a fitted auto-regressive model.
Differentiated time series Mean*	Mean of the series after differentiation
Differentiated time series Std*	Standard deviation after differentiation
Differentiated time series Skew*	Time series skew after differentiation
Lifespan	Length of the time series in days

A dataset with the listed features was created, with features being calculated per each time series that met the minimal criteria to be considered worthwhile of investigating. Such criteria have been based on the total number of requests of the specific series. Series with low recurrence of requests were not considered for clustering and further efforts, as the impact such series has on the environment is minimal.

In the analysis, it has been observed that only 3.47% of series have more than ten requests, with the number of needed requests being selected as 100 this number drops to 0.007%, 13034 in total.

This removal seems like a significant loss in the considered data, but this small number of series can explain the vast majority of the total requests. Out of the total 60,964,213 requests (post-correction), 56,982,512 belong to series with more than 100 requests. By percentage, this is 93.4%.

A further advantage of this decision is that the computation of the features is difficult. The total amount of time needed drops from days to within a day (With the possibility of optimisation, this could be reduced).

Another variable that should be mentioned is the sample rate over which the requests form the time series, as the selected interval affects the resulting features. A sample rate of 15 minutes has been selected, although 5-minute sample rate features have been computed as well, usage of these features did not introduce any benefits in the clustering section, and they were not used.

3.7.2 Clustering, methods and features

With the vast amount of features defined in the previous section, in order to find the best possible clusters, a systematic approach was needed. Subsets of features were selected, including the entire set as well, as using only all of the defined features at once was unlikely to provide the best results due to high dimensionality.

Out of all possible combinations, only a subset has been explored; 48 combinations have been explored in total. Out of this large subset, the most promising methods have been selected for a more detailed review, 11 in total. Promising methods offer split into clusters meeting expectations set by the previous analysis. The long being in hundreds compared to thousands of short-living series.

From the clustering process, it has been observed that long-living series mostly end up in one cluster; however, this cluster appears early in the process, and with further merges, as the process continues, it is polluted and results in a larger cluster consisting of mixed series. This outcome is not desirable.

The explored methods and features are specified as:

- **All features -** All computed features, 4 and 12 final clusters. On raw, scaled, normalised and scaled and normalised features.
- **Model based features -** Akaike information criterion, Bayesian information criterion, auto-regressive model intercept, auto-regressive lag values (7 in total). With 12 final clusters. On normalised and scaled features.

- **Lifespan based features -** Non-zero occurrences, occurrences, non-zero ratio, local maxima above 95% CI, Auto-regressive lag values (after 24 and 47 hours) and lifespan with 12 final clusters. On normalised and scaled features.

- **Behaviour based features -** Mean, standard deviation, skew, non-zero ratio, local maxima above 95% CI, Differentiated time-series mean, std with 12 final clusters, on normalised and scaled features.

- **Value based features -** Mean, standard deviation, skew, three largest values, local maxima above 95% CI with 12 final clusters. On normalised and scaled features.

The initial clustering result selection has been based on visualisations and numerical distribution of the resulting clusters. For each of the combinations, box plots showing the feature distribution and graphs of five of the most and least requested series in the clusters have been plotted.

The ideal outcome would be a method providing a clean split of 2 clusters for each series type. No such outcome was observed with the explored methods. Methods providing a split with a smaller cluster with the least and most requested series being long-living have been favoured. Methods providing roughly equally-sized split in terms of numbers or major cluster accompanied by insignificant clusters or clusters formed by individual series have been immediately discarded.

A large number of series and combinations led to difficulties in reviewing and selecting the best results. Cluster plots become unreadable and convoluted with just a small number of series involved. Numerical distribution amongst clusters paired with the knowledge of the ratio of series types is helpful but does not offer a complete understanding of the observed clusters. The feature box plots are numerous and difficult to interpret. Together, they provide an overview, which is good enough to filter out some of the non-satisfactory results but not to decide on without a more detailed review. Figure 3.5 shows a good candidate, the second cluster is small, and the least and most requested series seem to be long-living seasonal series.

3.7.3 Clustering review & behaviour labelling

The promising methods and their final clusters selected in the previous section have been observed in greater detail. All series, in each of the methods per cluster, have been plotted in groups of five. The resulting plots have been manually inspected, and based on the observations, labelling was attributed to each cluster. Some clusters consisted only of a single type, and others were a mix of multiple types. No method could find a clean split where all or the vast majority of the series would be in clusters consisting of one type.

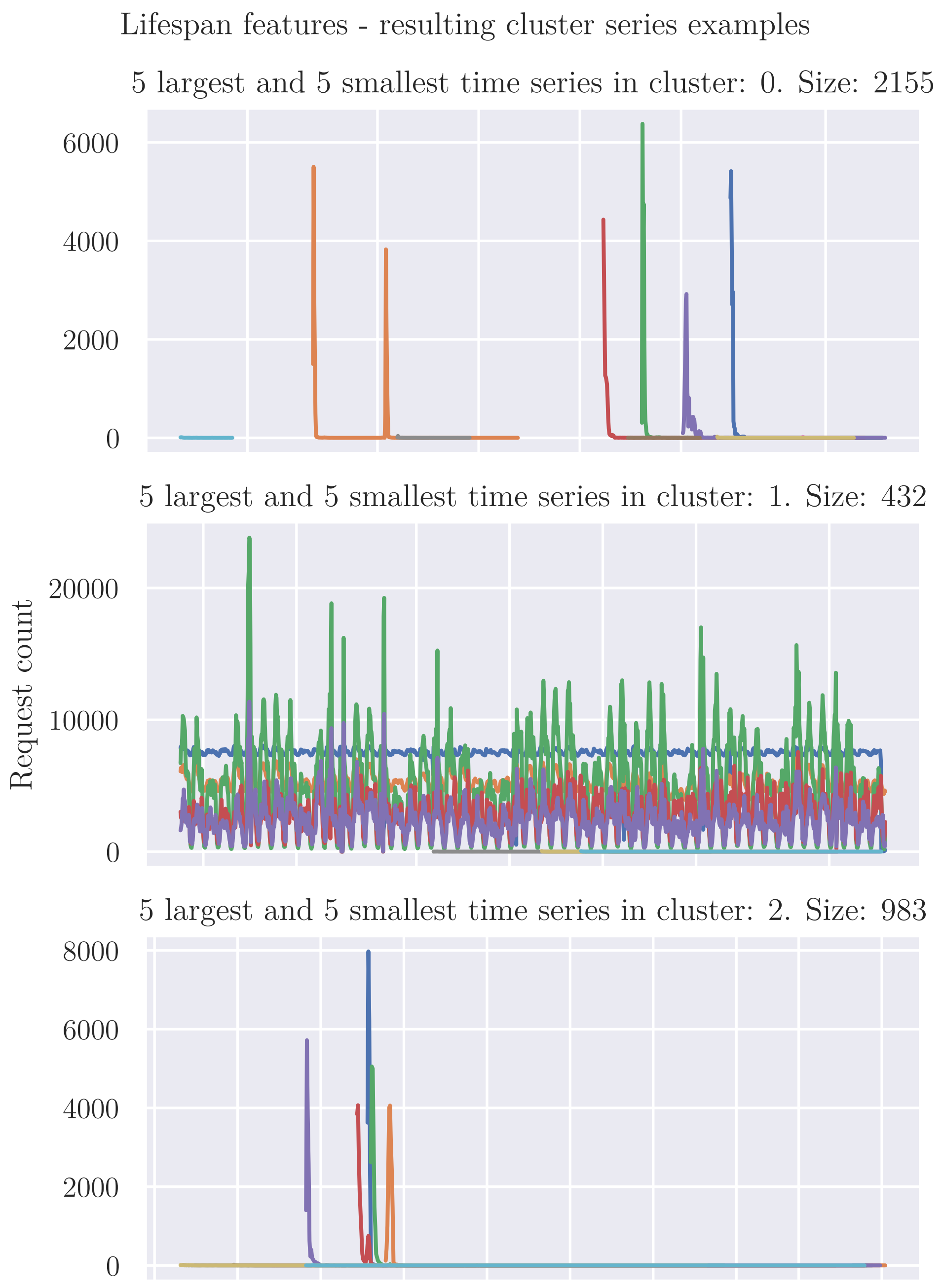

Figure 3.5: Figure shows an example plot with cluster least and most popular series plots. The figure comes from lifespan features, ward linkage with scaled and normalised features and 12 total clusters; only three are shown in this figure.

With the individual clusters marked by the types of series found in them, each method's outcome could be evaluated with more accuracy. The result where the number of series belonging to mixed clusters was smallest has been favoured.

Four best splits have been selected based on these criteria, and each series has been assigned a final type label if in all four best methods it belongs to a cluster with the same label. To provide an example, a series that belongs to short-living clusters in all four methods would be labelled as such. Series belonging to mixed clusters or belonging to clusters with varying labelling had to be manually assigned to a cluster based on per-series observations.

This method reduced the number of series to label from 13034 to 1216, considering the remaining number was still overwhelmingly short-lived, the number of series which had to be manually assigned to the long-living category was small. An additional advantage of this method is the reduction in misclassification caused by manual review of a large number of convoluted plots or by the series passing as either type.

3.8 Post-labelling analysis

A group focused analysis was conducted with the series split and labelled into the predefined long and short living groups. In hindsight, the request cutoff number for series to be considered could have been much larger, although, in small numbers, nonsensical series with few requests were still in the selected data, those were usually also hard to classify as they showed patterns typical for both groups. The short-living peaks account for 18,864,794 requests by 12,780 series, and the long-living seasonal account for 39,003,417 requests by 187 series. Combined, these series account for 93.4% of requests. The missing data consists of series with less than 100 total requests. The total series and the sums of the groups can be observed in the 3.11 figure.

3.8.1 Long-living series analysis

The long-living seasonal series are dominated by few largest series. The largest long-living series accounts for 24% of requests of all long-living series. The five largest together account for 75.4% per cent of all the 39 million requests from their category and for 50.8% of the total considered requests. On average, daily, there are 722,285 requests attributed to this category.

3.8.2 Short-living series analysis

To achieve the same share of requests as in the long-living category (75%), adding the 2,985 largest series together would be needed. This comparison is, of course, unfair. The longer the period observed in the data, the smaller the share a singular peak-series would have, while the long-living series would likely maintain their share. For this reason, in order to correctly assess the impact a peak series had, a reduction in the window size from the entire dataset to a much smaller interval was needed.

The largest of the peak series, on 15-minute intervals, held at its peak a 15% share of the requests in the environment. A much better result, if only short-lived. For the ten largest peak-series, their largest shares average 12.2%. For 100 largest, the number is lower at 8.6%. Figure 3.6 shows the largest shares for the largest 1000 series in descending order. It can be observed that while the share tends to decrease with series size, numerous exceptions exist. The share that a series has out of the total number of requests helped identify periods where peak series played an important role. This can be seen in figure 3.7, where this approach was used.

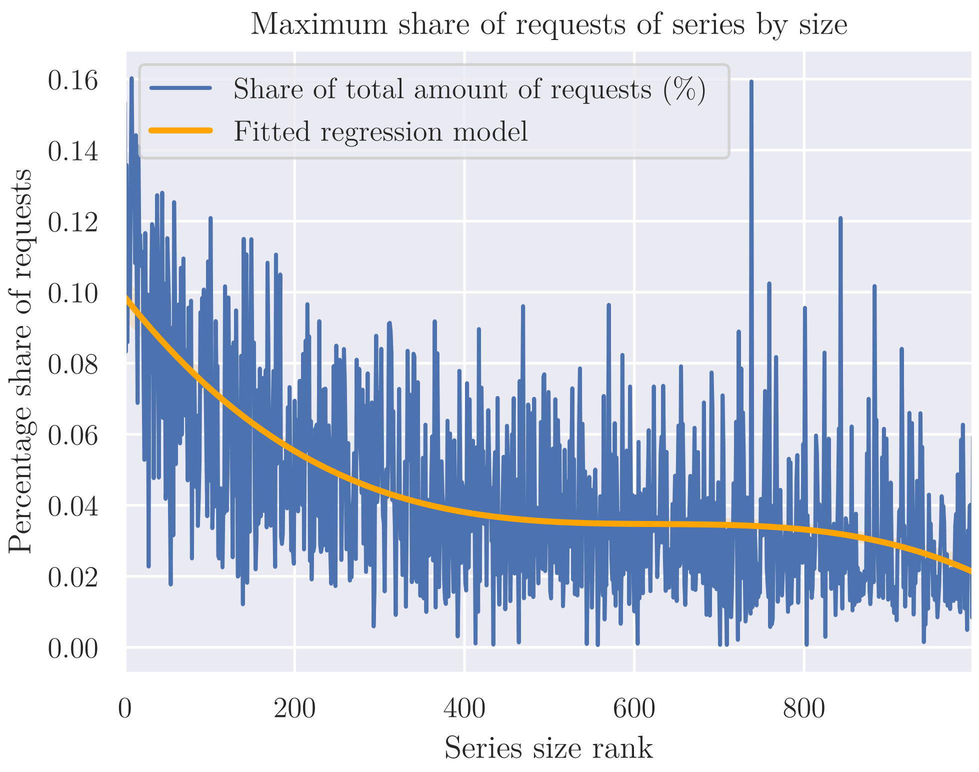

Figure 3.6: Figure shows the share of requests for the 1000 largest peak-series, in descending order.

It has been observed that the majority of series begin during the day, with the first quartile being at 9:00 AM, the median at noon and the third quantile being at 3 PM, likely corresponding to articles being published through the day. Figure 3.8 shows publish time counts. Analysis revealed that while the largest series by the total number of requests are mostly published during the day, the relationship between publishing time and the size of the total sum of series requests is not strong. In this case, the visualisation is deceiving as the majority of the weight is in the lower counts (sums) regardless of publishing hour. Even when reducing the number of observed series to the largest hundred or thousand series, no stronger relationship between publishing time and series size could be observed.

3.8.3 Growth in long-living series during short-living series peaks

Further analysis has been performed to answer how the long-living series group behaves during peaks in the short-living series. Moments where individual series exceeded 10% of the request share, have been evaluated. It has been observed that the larger a short-living peak was, the larger growth was in the long-living series. The largest measured value was around 53%. This large value is likely an outlier as the requests were highly volatile during such peaks, and the real growth was smaller. The mean increase in long-living series is 15%, the distribution and relationship is better explained in the 3.2 table and the 3.10 figure. This relationship is, however, not only positive. Smaller peaks can mean even a 24% decrease over the average requests in the long-living group than usual.

Table 3.2: Measured growth in long-living series requests during peaks.

	1. quartile	median	3. quartile
Series growth	9.91%	15.89%	22.15%

3.8.4 Short-living series - Describing peaks

With the short-living series playing a wildcard role, being able to increase the total number of incoming requests in a short period of time, it became crucial to gain an understanding of when does this occur, how much time is after that the series start until it starts to have an effect on the environment and for how long does this effect last.

3.8.5 Peak-detection method

The beginnings of request peaks, usually the only steep growth of a series, have been identified with moving averages. Fast and slow-moving averages have been used. A timestamp being marked as a beginning of a peak when the slow-moving average exceeded the fast-moving one. Before the moving averages could be used, the series sample rate and slow and fast-moving average windows had to be specified.

3.8.6 Parameter selection

The delay and response time is crucial in detecting peaks and their future impact. Too long of an interval, and the detection will be slow and miss the first peaks. As, by the time a peak is detected, assuming the method would be used in real-time, the peak is too old, or worse, the peak occurred before the method had a chance to detect it. Too small and the series becomes too volatile, and accurate detection becomes very difficult.

A large number of combinations with series sampling rates, fast and slow-moving average window sizes and the possibility of using exponential moving averages led to an examination of the moving average behaviour with different parameter choices. The outcome was that with increasing moving average window sizes, similar results to those with lower window sizes but longer sample rates were observed.

With the possibilities more clear, the next step was settling on parameters providing good detection. A small initial sample of series with different behaviour was selected on which the parameter selection and resulting performance was judged. Both exponential and normal moving averages have been observed. The series sample rate has been selected as ten seconds, and multiple promising parameter options have been found. On the small sample, the best performing option was exponential moving average with the window sizes of ten and twelve for fast and slow-moving averages. In addition, a minimum condition was specified for successful peak detection. A minimal amount of requests has to be observed within an interval to be considered. The slow-moving average has to exceed more than eight requests. This minimum has been set to prevent early marking of a peak start when the series is volatile and does not yet experience steep growth of any significance.

With the parameters set, the method has been applied to all short-lived series with over five thousand total requests. This was done to filter out less interesting series out of the thousands of series available, with most being judged unimportant for future forecasting. With the minimal number of requests set, a relatively large number of series remained (864). Even with the number of series reduced, the method failed to find any peaks for 568 of the 864 considered series. It became clear that the method had to be improved. By observing the series with undetected peaks, it was apparent that the method failed to mark series where the peak was much less steep or where the number of requests did not meet the specified criteria for the minimum amount.

Initially, the minimum number of requests was lower at 1000 requests. However, it did not present a good filter for meaningful series. Hence the requirements were raised. Nonetheless, the method has been evaluated on this number as well, with 3,714 of the 4,238 series meeting criteria have not had peaks found using the described method, this reinforced the decision to improve the method.

Once again, a small sample was taken, this time out of the undetected series, and multiple parameter options have been tried. The result was that slower exponential moving averages on slower sample rates were better than the relatively fast-paced option for the series with yet undetected peaks. This split brings problems where one set of parameters outperforms the other on a particular type of series, with the roles being reversed for another type. The series with steep growth take priority, but the remaining series can not be ignored. Both moving averages would be used to alleviate this problem, one faster, as specified before, and one slower. The slower exponential moving average is based on the 30-second series sample rate and with window sizes of ten and fourteen. This makes the minimum amount of time needed for this method mark a peak as seven minutes. For the faster-moving average, this is two minutes.

For the majority of the series, this works well. Exceptions, however, do exists, as can be observed in the figure 3.13. In the third example, the series growth is much less steep, and the condition for a minimum of requests per interval plays a more significant role than moving average crossover.

3.8.7 Method shortcomings

The result with the combined moving averages was better than using a single moving average. The number of series where a peak was detected increased from 296 to 833. A much better result. However, even with improvements, 31 (1,663 for 1000 minimal size) series before labelled as short-living series had no peaks detected. To gain a better understanding of the nature of such series, an additional analysis was conducted.

The improved method was also applied to the series meeting less strict criteria of one thousand minimum requests. The results ended up with an increase from 524 detected peaks to 2,575. However, this still left 1,663 series without a marked peak.

Comparing the two groups, series with and without a detected peak, revealed that the undetected group's series were smaller, with a shorter lifespan, smaller mean and smaller maximum values. The exact values can be found in the 3.3 table. In summary, the total number of requests per group is 8,100,901 for detected-peak and 192,788 for undetected (12,750,070 and 2,978,523 for 1000 minimal size). The visualised sum of request comparison can be seen in the 3.11 figure. The undetected group is significantly less important than the detected peak group. The undetected-peak series were not considered for further analysis.

3.8.8 First peaks in short-living series

It has been observed that 50% of the considered series have the first peak within the first 21 minutes. For 75% of the series, the peak is within 67 minutes from the first known request. Figure 3.14 shows the time until the

Table 3.3: Table shows comparison of the distributions of the measured values across both considered series groups and remainder after the minimum requests criteria.

Detected-peak group

	Series requests	Series lifespan (min)	Series maximum
1. quartile	6,273	16,938.25	9
median	7,910	34,543.66	11
3. quartile	11,318	53,900.83	15
Mean	9,725.47	35,692.94	12.66

Undetected-peak group

	Series requests	Series lifespan (min)	Series maximum
1. quartile	561	37,275.62	3
median	722	75,815.29	5
3. quartile	761	76,232.75	6
Mean	6,220.38	52,079.73	4.80

Remaining group, below 5000 total requests

	Series requests	Series lifespan (min)	Series maximum
1. quartile	198	9,462.25	2
median	461	29,554.50	3
3. quartile	1,142	54,254.75	5
Mean	887.09	33,262.55	3.90

peak in greater detail, figure 3.15 shows empirical distribution function with the probability of a first peak occurring by minutes passed from the series start. Clearly, most series show the beginning of a peak within a brief period of time since the start of their lifespan.

To better understand how steeply the peaks grow, the relationship between the size of the series and the steepness of the first peak has been explored. To determine the steepness of a series peak, a Linear Regression model has been fitted on the series from its beginning until the moment of peak detection, and the slope from the best-fit line has been measured. The relationship can be observed in the 3.12 figure. The correlation between the two variables has been measured as 0.45. No relationship can be observed.

In addition, the same has been explored for the size of the series when the peak was detected and the fitted line slope, there the relationship is very noticeable with a correlation of 0.78. The conclusion from these results was that how fast the series grows during its initial peak does not seem to affect the final series size and vice versa. The other less surprising observation is that there is a linear relationship between peak size and steepness of growth. More prominent peaks tend to grow faster.

3.8.9 First peak request drop-offs

With the beginning of the peaks know, additional questions about the series nature can be answered. How long do peaks last, how quickly do they lose requests and what proportion of the total requests associated with the series is within the marked peak?

A peak's lifespan was measured by moving average with a window of ten minutes starting from half of the window size before the marked beginning of a peak on the sample rate of five seconds. For each series, three occurrences are searched for, the 25%, 50% and 75% drop-off from the mean measured around the peak beginning. The drop-off in mean values allows measuring the longevity and pace at which requests are lost. For many series, the post-peak drop-off in mean requests is quick, happening in the first ten minutes after a peak was marked. The measured values for all measured 25%, 50% and 75% decrease in the mean are in the 3.16 figure.

To judge the series peaks' size, a percentage of the number of requests from the marked start until drop-off out of the total sum of requests has been measured. The observation results can be seen in the 3.17 figure.

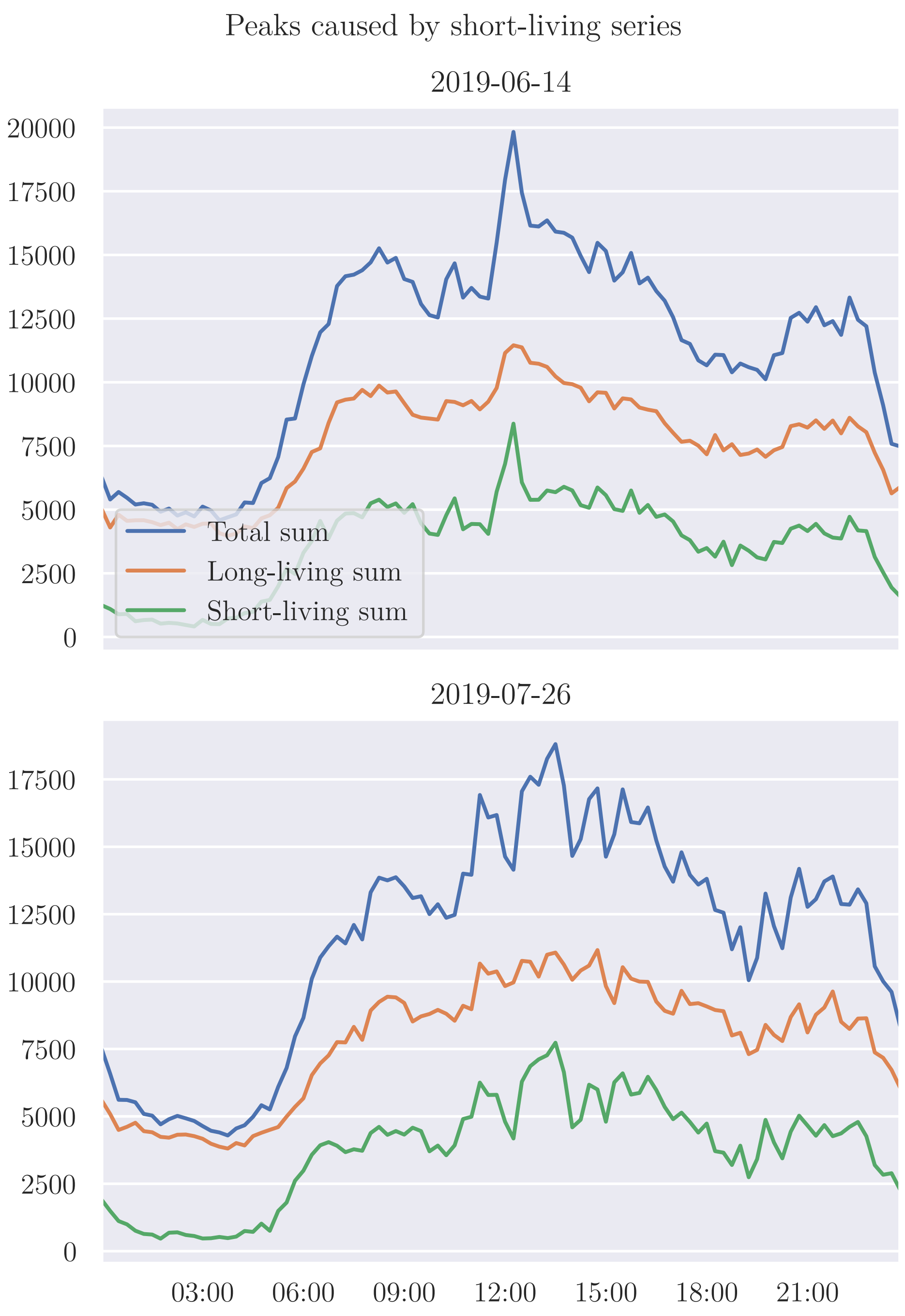

Figure 3.7: Figure shows two days, where a single peak-series exceeded its share of requests for more than 10% of the total number of requests. This leads to peak-series spikes being visible in the total sum of all requests, all while the long-living series did not experience growth at the same pace.

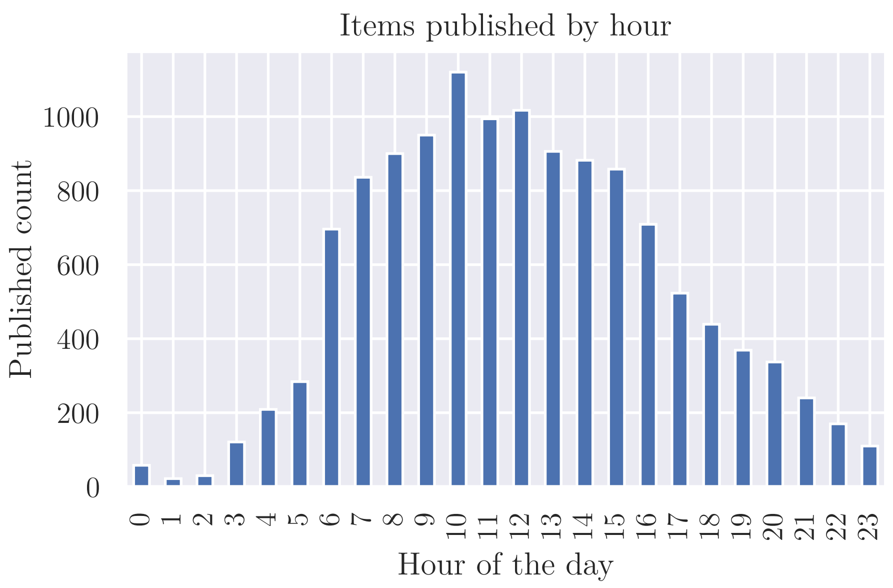

Figure 3.8: Figure shows the number of short-living series published by hour.

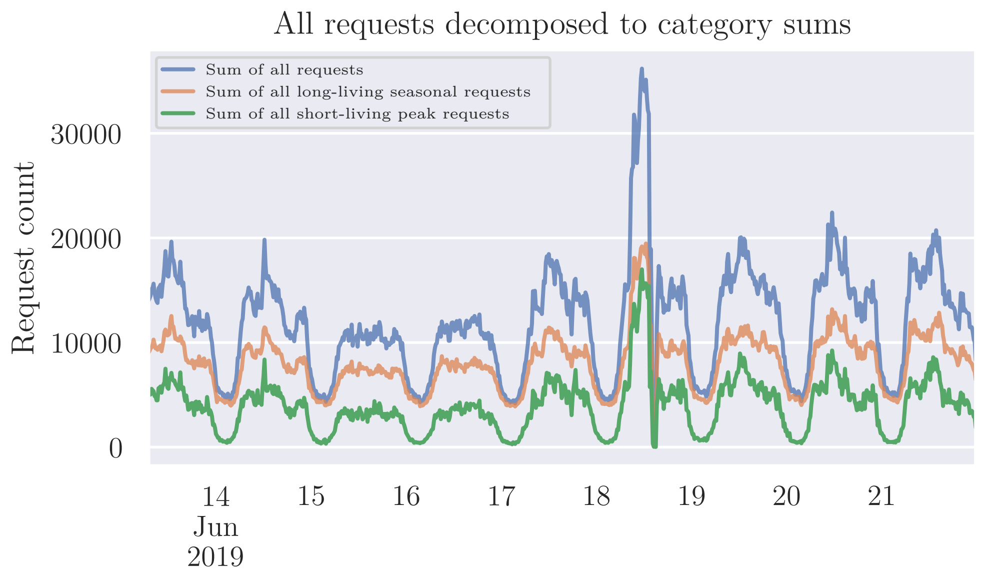

Figure 3.9: Figure shows the sum of all requests and sums of all short and long living series.

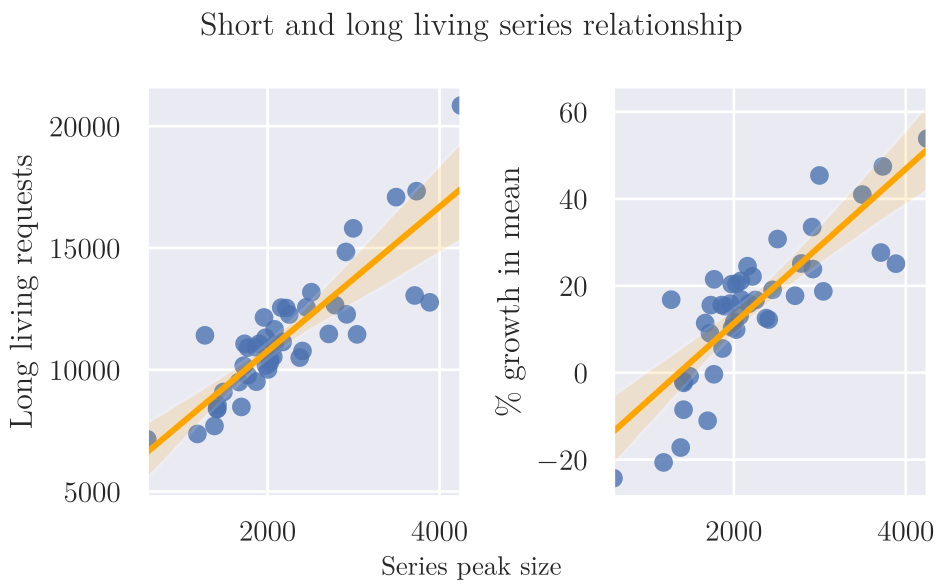

Figure 3.10: Figure explores the relationship of short and long-living series during peaks. The figure on the left compares the short-living series peak size with the requests in the long-living category measured simultaneously. The right figure compares the peak size with the percentual growth of long-living series during said peak over the mean measured at that time over the dataset.

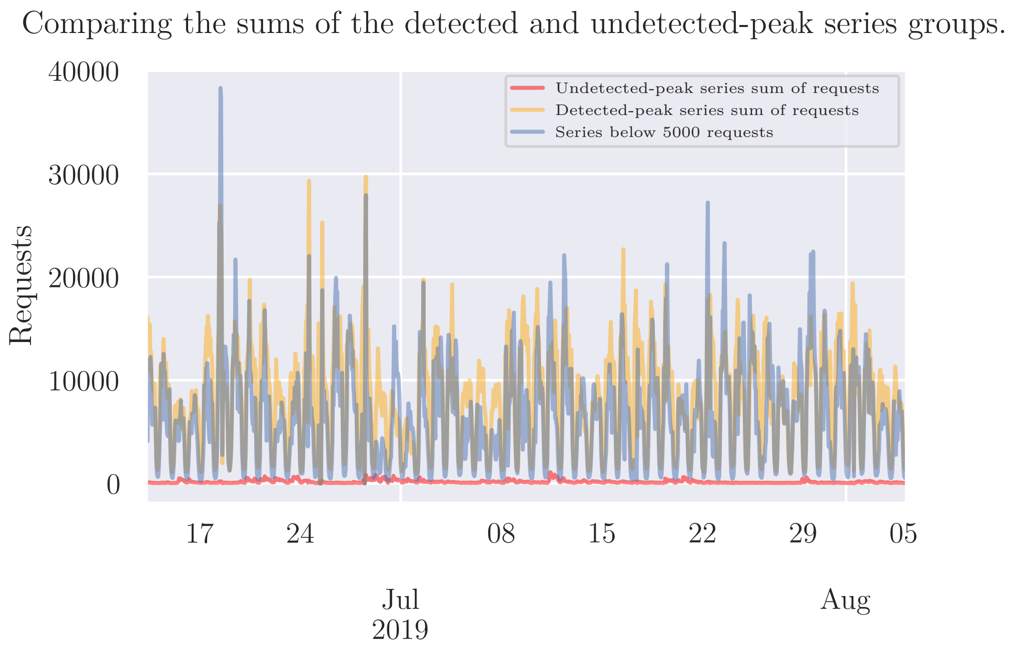

Figure 3.11: Figure compares total sums of the two series groups (with detected peaks and without).

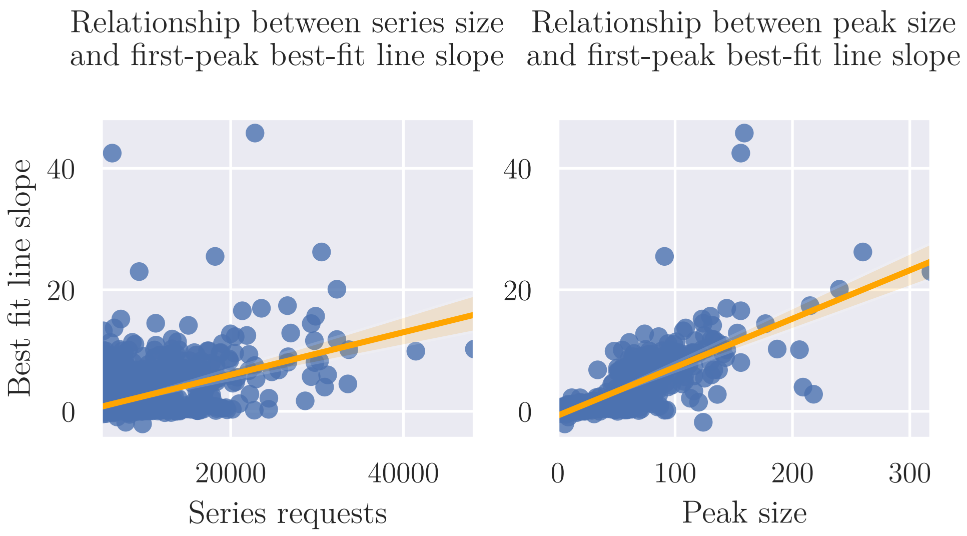

Figure 3.12: Figure shows a relationship between the total request size and the slope of the best-fit line until the first peak. The orange line is another best-fit line on the output data. The same is shown for the first peak size and best-fit slope.

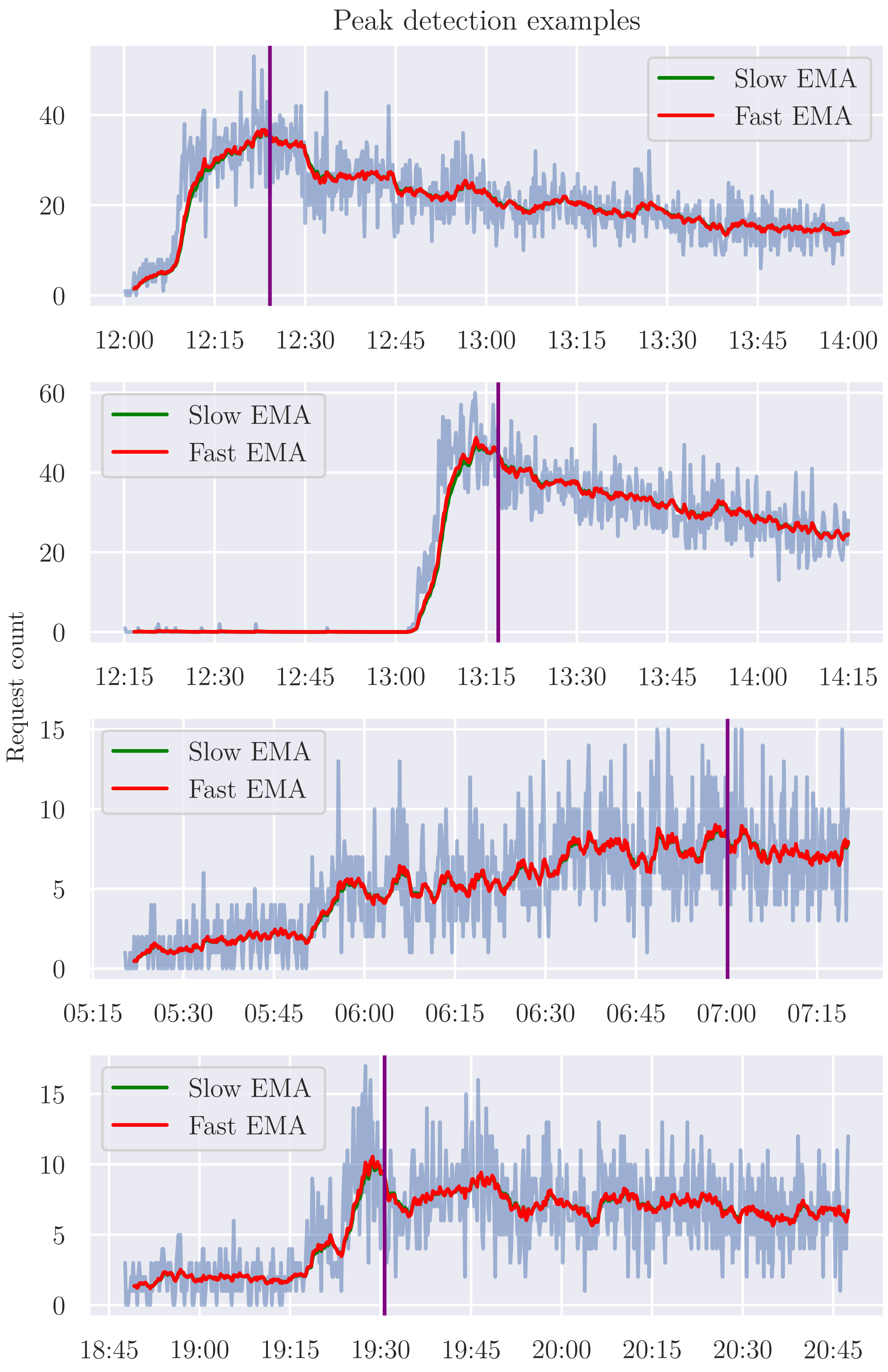

Figure 3.13: Figure shows four examples of the peak-series startup detection method by moving averages. The green series represents the slow-moving average, the red series fast-moving, and the purple vertical line indicates where the start of a peak was detected.

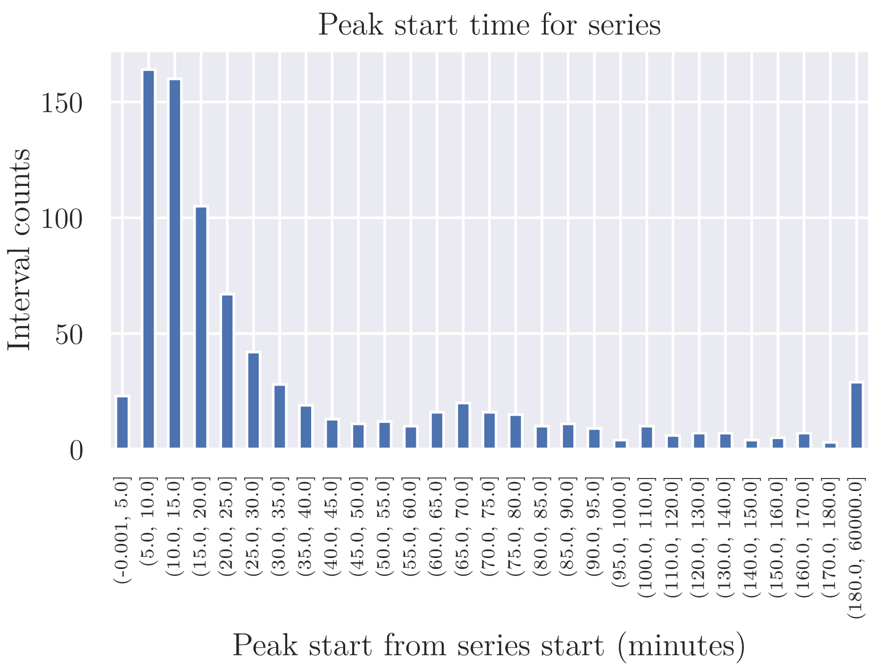

Figure 3.14: Figure shows histogram with 5 minute interval bins for series startup in minutes.

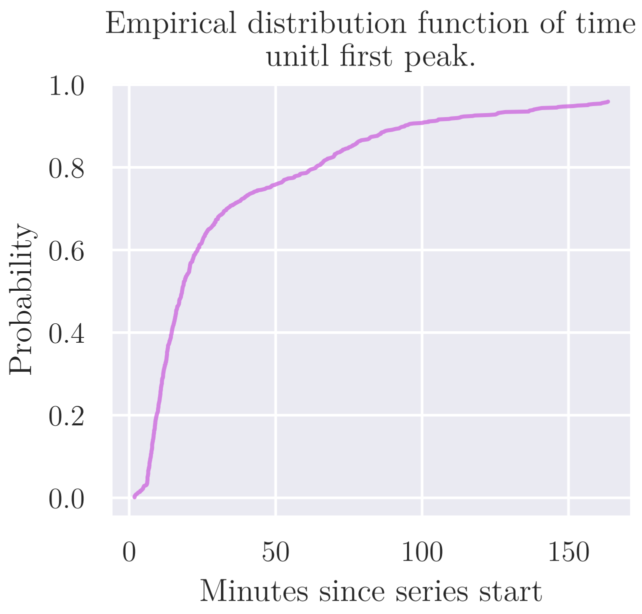

Figure 3.15: Figure shows the empirical distribution function for peak occurrence in minutes from beginning of the series.

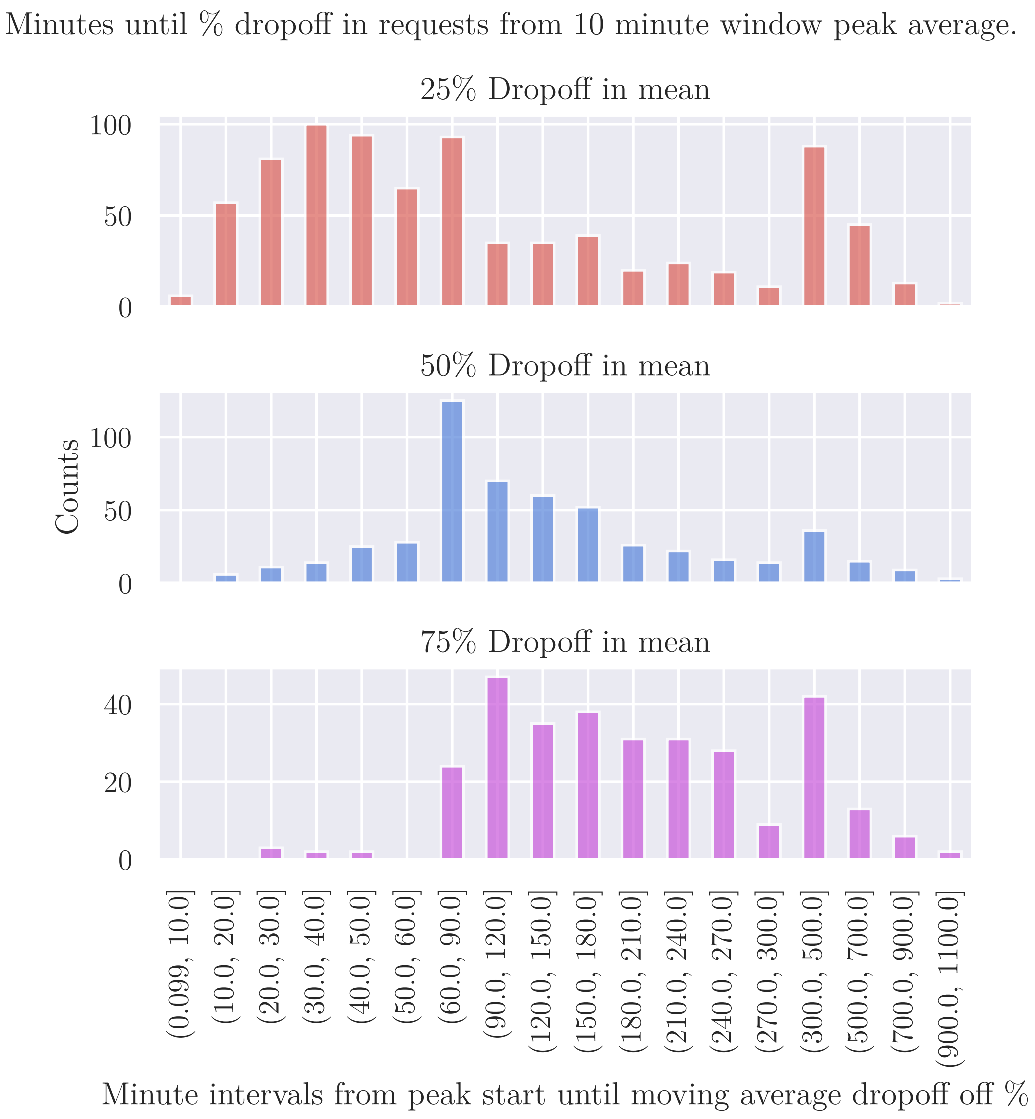

Figure 3.16: Figure shows minute interval counts of when a mean window value achieves a perceptual drop from the mean window value measured around the beginning of the peak on individual short-living series.

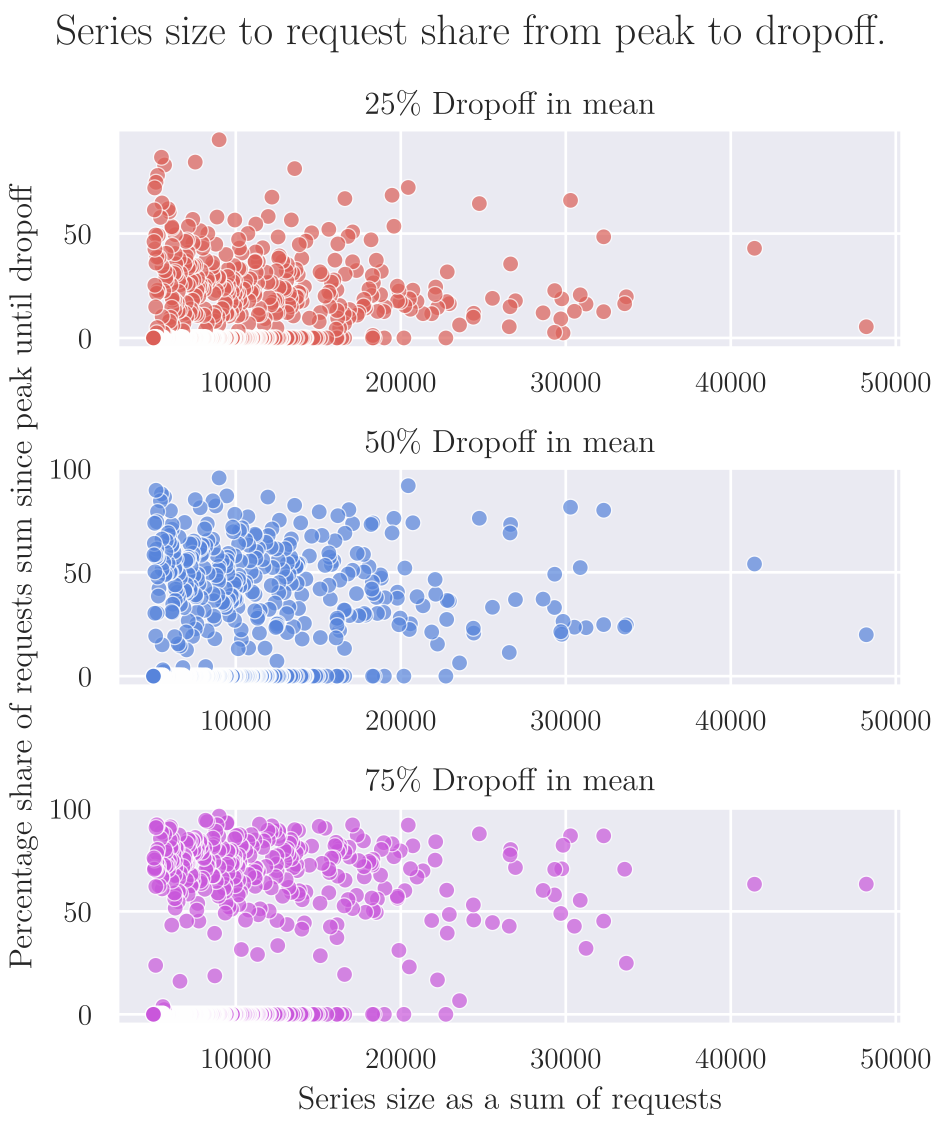

Figure 3.17: Figure shows total request size with percentage of requests out of the total series requests from peak until measured drop-off in mean.

3.9 Forecasting sudden increases

With the data analysed and better understood, a method for short-term prediction could be formed. Multiple models have been tried ARIMAX, AR(1), TBATS, benchmark models and the most complex out of the considered models, models based on HTS (Hierarchical time series). For the HTS models, the data has been split into multiple components where each was forecasted individually, the component forecasts representing bottom-levels of a simple hierarchical time series. Using a bottom-up HTS approach, the lower-level forecasts were unified to form one top-level forecast representing the incoming request's total number. Two HTS models have been used, a strict real-time model for a more conservative approach and a speculative model to allow for a situation closer to potential real-time use.

The forecasting has been attempted on one and fifteen-minute sample rates. The very fast-paced forecast on the one-minute sample rate has the potential to notice disturbances caused by sudden peaks, as observed in the analysis where peaks can happen within minutes but may fail to provide reasonable forecasts for more extended periods as the quality of the forecast deteriorates with more steps-ahead away from the last observations. The fifteen-minute sample rate forecast should provide forecasts further ahead in time. The intention is to provide a reasonable forecast for the immediate thirty minutes ahead and a medium-term forecast for the few following hours.

3.9.1 Model evaluation

The dataset has been split into test and training parts with a 70% to 30% ratio. They range from 13.6.2019 to 20.7.2019 for the training set and from the end of the training set to 5.8.2019 for the testing set. The training set has been used to fit the initial model, which was then evaluated using cross-validation with steps-ahead chosen by sample rate over the remaining testing data.

The root mean square errors and mean square errors have been observed on the short-term forecasts prepared by the cross-validation approach. This approach allowed for the evaluation of forecasts as if they were made in real-time. The cross-validation method of evaluation has been chosen to reward models performing well in the short-term instead of long-term forecasts, which would potentially benefit from a more static approach with the split of training and testing sets. The motivation being as to focus on the sudden peaks that are the topic of this thesis.

The errors were calculated by iterating over each observation in the testing set, each previous observation has been made known to the fitted model, and forecasts of three hours for the fifteen-minute sample rate model and thirty minutes for the one-minute sample rates have been made. To provide an example, if the testing set contained ten observations, ten forecasts would be made by the model, and errors would be calculated by subtracting the forecasts from the observed values over the forecasted range. The errors of the final forecasts where no comparison with observed values can be made would be zero. The final total error is given as a mean of RMSE and MSE errors on individual forecasts made.

The components used in HTS models have been evaluated differently. For components modelled by traditional time series models, the evaluation described above was used, the components where different models were employed were evaluated over multiple individual series, a more detailed description being in the 3.9.4.1 section. The individual components of the HTS models are covered in the next section.

3.9.2 HTS Forecast components

The first component consists of requests attributed to the previously identified long-living series, which together form a strong and stable seasonal component. The short-living series form the remaining components and a tracked series component where selected series, varying through time, are forecasted individually.

Two approaches were attempted with the short-living components. The first one was a strict approach where the data is used as observed in its current form in real-time. This real-time use is a very restrictive approach as the anonymised data does not allow for immediate identification of series. When a series first appears, it can not be said as to how it will behave. In a real-world scenario, the information about the series nature, such as belonging to a newly published article, would not be difficult to acquire, and assumptions could be made on the series's future behaviour.

The second, more speculative approach attempts to compensate for this loss of information in the data. Part of the peak components matching the expected behaviour of impactful articles is selected beforehand. This speculation simulates a method where the information about newly published articles is known as it appears rather than with delay as in the first method. From the previous chapters' analysis, the amount of time until a newly observed short-living series starts to have an impact can be small. Thus, the ability to forecast their impact could be crucial for successfully detecting peaks, something the first method may fail at.

In both approaches, the short-living series were split into eleven components; ten of the short-living series were picked at one point in time and forecasted individually using regression models and a remainder component containing the sum of all remaining short-living series posing as the more stable seasonal component with its volatility reduced by removing most impactful peaks.

The equation 3.1 shows how the final forecast is acquired. The $\hat{y}_{l,t}$ stands for the forecast of the long-living model at time t, the $\hat{y}_{sr,t}$ is the forecast of the remainder component of the short-living series at time t and the $\hat{y}_{tr_n,t}$ is the n-th tracked short-living series at time t.

$$\hat{y}_t = \hat{y}_{l,t} + \hat{y}_{sr,t} + \hat{y}_{tr_1,t} + \ldots + \hat{y}_{tr_{10},t} \tag{3.1}$$

In HTS matrix notation this becomes 3.2 where $y_{s,t}$ are all short-living series and the $y_{tr,t}$ represents all tracked series.

$$\begin{bmatrix} \hat{y}_t \\ \hat{y}_{l,t} \\ \hat{y}_{s,t} \\ \hat{y}_{l,t} \\ \hat{y}_{sr,t} \\ \hat{y}_{tr,t} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \hat{y}_{l,t} \\ \hat{y}_{sr,t} \\ \hat{y}_{tr,t} \end{bmatrix} \tag{3.2}$$

The first approach uses the last thirty minutes (15-minute sample rate) and ten minutes (1-minute sample rate) moving window over which sums of all short-living series are considered. The series with the most considerable sums are selected, and each is forecasted individually. This approach catches the most impactful series but may fail to do so fast enough.

The second approach attempts to re-mediate the shortcomings of the first method. Five series are selected as in the first method by their current size. The second half is tracking series previously identified and tracked by the first method. Instead of waiting for the series to be large enough to be caught by the moving window size, the previously identified series is tracked as soon as it becomes available. The newly tracked series replaces the least requested currently tracked series. To prevent this approach from replacing series before they have a chance to grow, each newly tracked series has a short, safe period when it can not be replaced. It may, however, grow large enough where it is considered one of the five largest. In this case, the series is forecasted as a part of this group freeing up space for another series.

3.9.3 Long-living series forecasting

For forecasting the long-living series, multiple models have been tried. Naive, Average and Seasonal naive models have been used as a benchmark for comparison with the Auto-regressive model with an order of one, TBATS model and an ARIMAX model with Fourier terms modelling the daily seasonality. Fourier terms for weekly seasonality have been considered as well. However, this increased the number of combinations significantly, making the process computationally tricky. In addition, the initial results showed that using weekly seasonality led to little difference in terms of AIC. For these reasons, models with multiple seasonalities have not been further evaluated. By minimising AIC, the ARIMAX order was selected as (3, 0, 9) with Fourier order of six for the fifteen-minute sample rate and (9, 1, 2) with Fourier order of eight for one minute.

Table 3.4: Long-living forecasts measured model mean RMSE values per sample rate.

Model	15-Minute RMSE	1-Minute RMSE
ARIMAX with Fourier terms	895.39	59.47
Naive	1411.14	91.06
AR(1)	1385.91	135.69
Average	1803.89	77.22
Seasonal naive	1359.26	124.64
TBATS	2450.75	111.75

On both sample rates, the best performing model ended up being the ARIMAX model with Fourier terms. The TBATS model struggled to capture the seasonality during late night and early morning, consistently forecasting fewer requests than observed. The measured errors can be observed in the 3.4 table for RMSE and in 3.5 for MSE.

Table 3.5: Long-living forecasts measured model mean MSE values per sample rate.

Model	15-Minute MSE	1-Minute MSE
ARIMAX with Fourier terms	1,704,294.04	7,162.09
Naive	3,872,358.86	12,890.16
AR(1)	3,204,636.81	23,762.77
Average	5,111,863.00	11,084.42
Seasonal naive	3,911,814.81	25,337.92
TBATS	7,967,196.16	16,633.06

3.9.4 Short-living series forecasting

The forecasting of short-living series has been split into two parts. In the first part, the individual series selected by the real-time and speculative models are forecasted. In the second, the remainder component, a strong seasonal series itself, is forecasted.

3.9.4.1 Regression for tracked series forecasting

Forecasting individual series by the same model forced certain constraints. The model should learn from all the different short-living series available in the training set and be generic enough to forecast similar but different series. This prevented the usage of traditional time series models with fitted parameters such as auto-regressive or moving average models, which attempt to best fit a particular series.

In order to use all available training data, the problem was restructured. Instead of solving a time series forecasting problem, the series data was transformed so that the problem could be solved as a more generic regression problem. This approach can take advantage of all available training data.

To do so, each available training series was iterated over, and for each of the observations at time t, a vector was taken from previous and future values. Multiple regression models have been explored together with a varying number of predictor variables as defined by 3.3.

$$predictor = \begin{vmatrix} y_{t-9} & y_{t-8} & \dots & y_{t-1} & y_t \end{vmatrix} \tag{3.3}$$

While multiple predictor variable lengths have been explored per sample rate, only one length was tried for the dependant variables (3.4) per sample rate.

$$dependent = \begin{vmatrix} y_{t+1} & y_{t+2} & \dots & y_{t+29} & y_{t+30} \end{vmatrix} \tag{3.4}$$

The summary of which feature lengths were explored per sample rate and the time-span they cover is shown in the 3.6 table.

Table 3.6: Table shows the considered predictor and dependent variable lengths.

Sample rate	Predictors	Dependent	Predictors time	Dependent time
15-Minute	12	12	3 Hours	3 Hours
1-Minute	10	30	10 Minutes	30 Minutes
1-Minute	30	30	30 Minutes	30 Minutes

Entries, where the future or historical observations were not available, were filled with zeros. The motivation for exploring such a short range of 10 minutes in the 1-minute sample rate is that some of the considered regression models are severely affected by the initial empty observations. This lack of values skewed the forecast and was magnified because a vast portion of the considered series contains a peak within 10 minutes from the start.

With the data prepared, a variety of regression models have been fit and compared. The selected models were Random Forest, Gradient Boosting (GB), AdaBoost, Linear Regression and K-Nearest Neighbours regression (KNN). The MSE and RMSE errors have been measured on a set of short-living series from within the testing data interval to evaluate the models.

While the fifteen-minute sample rate measurement went without problems, on the one-minute sample rate, the amount of data in the testing set increased

Table 3.7: The table shows MSE and RMSE errors for the considered regression models on fifteen minute sample rate.

15-Minute sample rate

Model	Predictors	MSE	RMSE	P. MSE	P. RMSE
Random Forest	12	10304	101.50	66316	257.52
Gradient Boosting	12	10779	103.82	67969	260.71
AdaBoost	12	34748	186.40	71372	267.15
Linear Regression	12	10739	103.62	67938	260.65
KNN Regression	12	15474	124.39	65205	255.35

dramatically, especially with multiple considered predictor lengths. The evaluation of the complete testing set for all models was unrealistic. While some models required only a few hours to produce testing set predictions, others, like Gradient Boosting, would take days, if not weeks. To alleviate the issue without fully losing a general overview on individual model performance, the testing set was randomly shuffled, and a subset with the size of one-tenth of the complete set was used instead. While far from ideal as valuable data was dropped from evaluation, this reduced the number of samples from 3,784,144 samples to 378,414 and should allow for a general evaluation of model performance.

Considering the method for selecting tracked series, the models' priority is to perform well around peaks and larger series. For this reason, additional error measures have been made on the series previously identified in the peak-analysis section of the thesis. The errors have been measured from the start of the series until the detected peak time extended by one hour to capture behaviour shortly after a peak.

The fifteen-minute sample rate results were close, with Random Forest achieving the best performance on total MSE and RMSE. Most models showed similar errors except for AdaBoost and KNN regression which under-performed. Outcomes on the errors around peaks were even closer. Here the KNN model showed the best score followed by Random Forest. However, the difference was not significant (MSE 1.6%) and is offset by the large difference between RF and KNN models on general errors (RF MSE being 33.4% smaller than KNN). The taken measurements can be found in the 3.7 table. On this sample rate, the models were unlikely to predict an upcoming peak. This was mainly because peaks are very likely to occur soon after the series is first observed. In terms of observations, this can mean peaks occur in the first one or two observations. Finding a model providing accurate peak prediction on so few observations remained difficult. The sample rate is likely too large to allow for the prediction of peaks.

On the one-minute sample rate, in terms of total errors, the best-performing model ended up being Linear Regression with 30 predictors followed by LR

Table 3.8: RMSE and MSE errors measured with each regression model on all testing data and testing data limited to peaks (P. MSE and P. RMSE). Models fitted on reduced peak-only data are marked with (P).

1-Minute sample rate - 10 Predictors

Model	MSE	RMSE	P. MSE	P. RMSE
Random Forest	19.68	4.43	249.09	15.78
Gradient Boosting	53.05	7.28	249.76	15.80
AdaBoost	3181.07	56.40	1763.66	41.99
Linear Regression	15.61	3.95	254.35	15.94
KNN Regression	94.73	9.73	260.62	16.14

1-Minute sample rate - 30 Predictors

Random Forest	15.63	3.95	251.84	15.86
Gradient Boosting	16.70	4.08	239.36	15.47
AdaBoost	3266.18	57.15	2012.96	44.86
Linear Regression	15.03	3.87	256.26	16.00
KNN Regression	21.41	4.62	275.91	16.61

1-Minute sample rate - 10 Predictors - Peak data

Random Forest (P)	48.16	6.93	279.94	16.73
Gradient Boosting (P)	32.84	5.73	229.87	15.16
AdaBoost (P)	2263.84	47.57	1487.57	38.56
Linear Regression (P)	58.57	7.65	246.18	15.69
KNN Regression (P)	29.02	5.38	273.79	16.54

1-Minute sample rate - 30 Predictors - Peak data

Random Forest (P)	201.42	14.19	372.55	19.30
Gradient Boosting (P)	26.77	5.17	240.67	15.51
AdaBoost (P)	1964.27	44.32	1197.99	34.61
Linear Regression (P)	58.96	7.67	268.88	16.39
KNN Regression (P)	24.86	4.98	686.69	26.20

10, RF 30 and GB 30 models. The errors are close up until the last four observed models, where errors grew dramatically. All of these poorly performing models are variants of the AdaBoost model. An interesting observation can be made about the number of predictors with KNN and GB. While most models did not show a significant difference in error with varying predictor lengths, the number of predictors was of major impact on the KNN and GB models.

By observing the models' forecasts, it became apparent that the models do not attempt to predict peaks but tend to try to predict the series outside of peaks, such as the gradual decline after peaks that form the majority of the series data. This part was handled well. However, the models were failing in the crucial ability to forecast peaks. Without this ability, the usefulness of the selected approach was threatened. The majority of forecasts started around the last observed number of requests and predicted a downward slope. This hesitance around peaks was especially apparent with Linear regression. To improve the capabilities of the models for peak prediction, a subset of the training data was created focused on peaks. Only data until the peak with a half-hour post-peak extension was used to fit additional models with 10 and 30 predictors.

The results can be observed in the 3.8 table. Example forecasts made by the GB model with 30 predictors and its behaviour can be seen in the 3.18 figure. As the peak grows, the model attempts to predict where the peak will end. This seems to manifest itself as an increase from the current number of requests followed by a steady, almost level decrease in requests. Once the peak is reached, the model forecasts a steady decrease. The decreasing trend seems to be captured well. Unfortunately, the model is unable to forecast the peak from the beginning of the series perfectly. The sought-after peak-predicting behaviour was rare in the observed models and was strongest in the selected example.

Errors measured around peaks were lowest for GB (10 predictors, trained on peaks), with three of the best scoring models being GB variants. The first fourteen model variants were close in terms of error, with the errors gradually increasing, the last six variants showing a dramatic increase.

The selected models were RF for the fifteen-minute sample rate where the choice was relatively easy, with RF achieving good results both around peaks and for whole series and KNN struggling in the total errors. The selection on the one-minute sample rate was much more difficult.

The model had to meet three criteria. It had to show attempts to forecast peaks. It had not only to try safe forecasts with downwards slope from current values but forecast actual upward surges in requests and achieve low errors around peaks. Whenever the model forecasts peaks or just a downwards slope was apparent from observing sample predictions, besides, the model had to have decent overall performance in terms of errors and had to capture the peak series trend well.

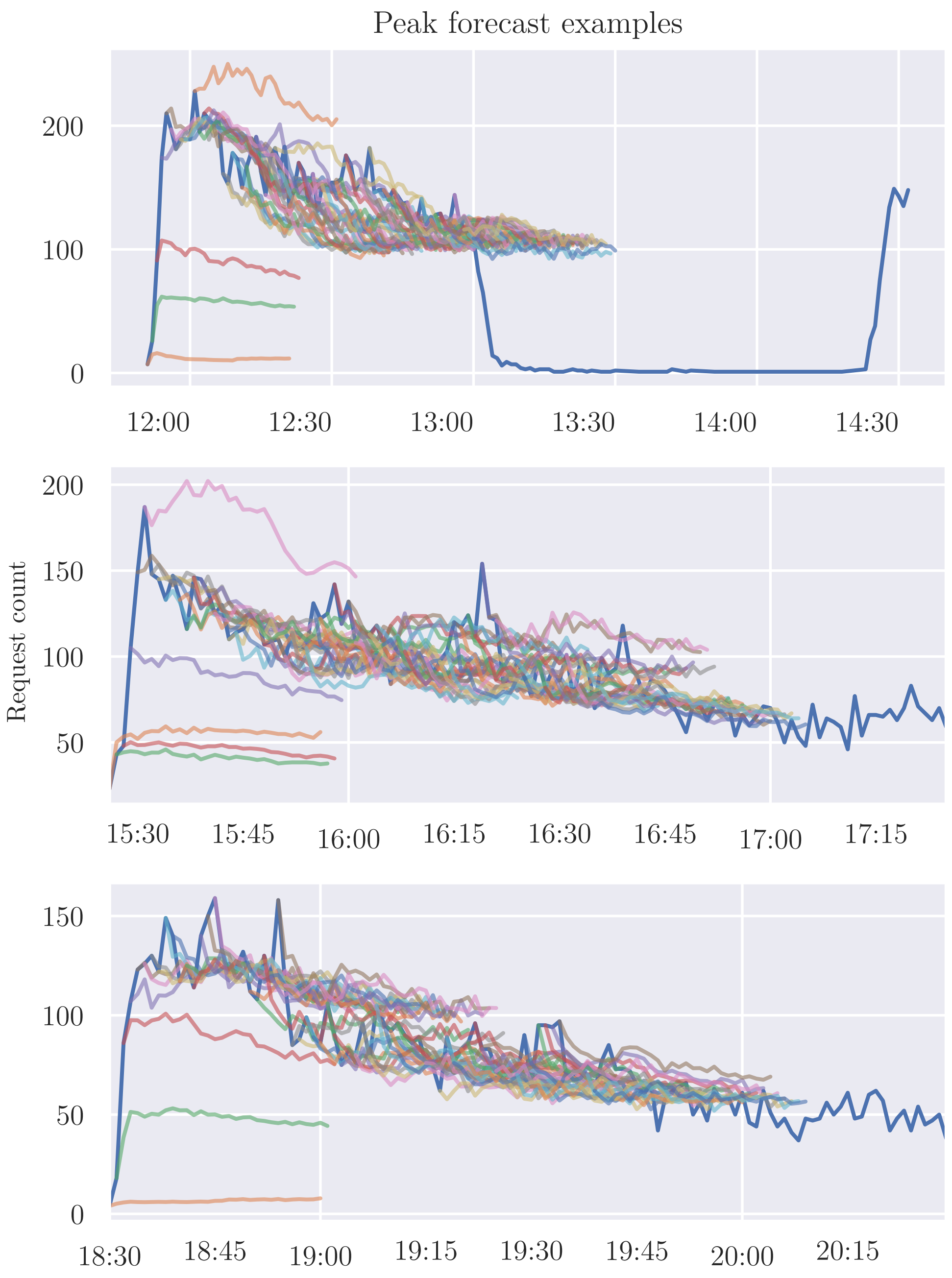

Figure 3.18: Figure shows three sample peaks from the short-living series category with forecasts made by the GB regression model on a 1-minute sample rate. At each minute, a forecast is drawn. As the peak grows, the model attempts to adjust but fails to predict the peak end. The trend of decreasing requests after the peak is captured well.

With these criteria in mind, the GB models seemed very promising. The models seemed to attempt to forecast peaks and achieve good performance around peaks in terms of error. The GB models did not do as well as other models on the whole series but did not fall too far behind.

Linear regression models performed well on the general errors and could capture the trend but failed to predict peaks. Random Forests seemed to be a decent choice, doing well in terms of error, but the models did not stand out. The KNN regression models seemed to under-perform and were heavily dependant on the number of predictors and training data. Finally, the AdaBoost models seemed to be most likely to predict more significant peaks but struggled greatly in error, being consistently the model with the most significant errors.

In the end, the GB model with 30 predictors trained on reduced peak data was selected. Other models remain viable options as the results are close but were not further explored.

3.9.4.2 Remainder series forecasting

The remainder components of both real-time and speculative methods for the short-living series has been forecasted using the TBATS and ARIMAX with Fourier terms models. The Fourier order was selected by repeated calls of `auto_arima` for each considered Fourier order (1-11) and selecting minimal AIC models. The Fourier order for a one-minute sample rate was six. For fifteen-minute, this was five and six for speculative and real-time, respectively. The selected ARIMAX orders are specified in the 3.9 table.

Table 3.9: Selected orders for ARIMAX models used in remainder series forecasting.

ARIMAX Model	ARIMA Orders	Fourier order
15-minute Real-time	(3, 0, 3)	6
15-minute Speculative	(2, 0, 4)	5
1-minute Real-time	(3, 1, 4)	6
1-minute Speculative	(3, 1, 4)	6

With similar results, the ARIMAX model ended up outperforming the TBATS model, scoring lower RMSE and MSE errors in all cases. On all considered sample rates and HTS models, the ARIMAX models were chosen and preferred over TBATS to model the remainder component. The models provided very similar day-time forecasts, but the TBATS model had a significantly worse performance during the night and early morning, where it consistently under-estimated the number of requests. This was the same outcome as in long-living series forecasts.

The measured RMSE errors can be observed in the table 3.10. The MSE errors in table 3.11 gave the same results, with ARIMAX being the superior

Table 3.10: Total sum of observed mean RMSE values on short-term forecasts.

Model	15-minute RMSE	1-minute RMSE
ARIMAX with Fourier terms (Real-time)	458.30	41.15
TBATS (Real-time)	750.97	63.21
ARIMAX with Fourier terms (Speculative)	495.89	42.62
TBATS (Speculative)	807.01	65.70

model for this series. The residuals from the automatic TBATS models did not form a normal distribution, suggesting that the model failed to capture the series.

Table 3.11: Total sum of observed mean MSE values on short-term forecasts.

Model	15-minute MSE	1-minute MSE
ARIMAX with Fourier terms (Real-time)	336,484.03	2517.73
TBATS (Real-time)	723,240.08	4676.24
ARIMAX with Fourier terms (Speculative)	399,132.52	2732.24
TBATS (Speculative)	843,236.12	5113.54

3.9.5 Forecasting methods comparison

The resulting methods leveraging request decomposition were compared to models applied to the total sum of requests on fifteen and one-minute sample rates.

To determine the Gradient Boosting and Random Forest effect for the tracked series component and the impact on the final forecast, an additional HTS model was created. This model, referred to as HTS Oracle, is purely for evaluation purposes as it removes the uncertainty in the tracked series by using the actual observed values. This means the error for this model will be caused by the long-living and remainder ARIMAX models, and it will simulate a situation where the tracked series and their peaks are forecasted perfectly.

Using mean MSE and RMSE as a metric, the HTS Oracle model showed the best results. This outcome is understandable since the model knows the series beforehand. It is not capable of actual forecasts and is meant only as a benchmark for other models. Out of the remaining models, the ARIMAX model with Fourier term ended up having the best performance on the fifteen-minute interval followed by the HTS models, speculative achieving slightly better results than the strict model. The hierarchical models offer good forecasts, but considering the complexity, themselves being formed by multiple ARIMAX and RF models, the usage of a singular ARIMAX model would be preferred. The TBATS model underperformed and failed to beat the trivial benchmark methods except for the average method on measured MSE. A more detailed comparison of RMSE and MSE model errors can be observed in the 3.20 figure. An example of model forecasts can be observed in the 3.19 figure. The measured errors can be found in the 3.12 and 3.13 tables.

Table 3.12: Total requests forecasts measured model RMSE values per sample rate.

Model	15-Minute RMSE	1-Minute RMSE
ARIMAX with Fourier terms	1,746.46	101.81
Naive	2,699.19	133.11
AR(1)	2,620.14	227.46
Average	3,466.21	125.15
Seasonal naive	2,568.07	205.31
TBATS	3,561.89	198.48
HTS Real-time	1,843.07	103.51
HTS Speculative	1,817.82	103.29
HTS Oracle	818.58	88.31

As for the results on the one-minute sample rate (figure 3.21), the HTS models ended up faring better and outperformed the ARIMAX model on mean MSE and achieving only 1.6% worse result on mean RMSE to the ARIMAX model but 6.4% better result in mean MSE (RT). This very close outcome on both sample rates between ARIMAX and hierarchical models suggested that the variable short-living series and its tracked peaks components did not have as large of an impact as was assumed prior, or the selected component models did not perform well enough to give a significant boost to the forecast accuracy.

The Oracle model showed an improvement over the one-minute sample rate over the real-time model of 14.6% in RMSE and 33.5% in MSE. On a fifteen-minute sample rate, the results are more significant and account for 55.5% for RMSE and 65.3% for MSE.

The seasonal naive method's poor performance, where even the naive method without seasonality outperformed the method on a series with such a strong seasonal pattern, may be surprising. This was due to the method of evaluation of the forecasts. On long-term forecasts, the seasonal method would perform significantly better. The short nature of the forecasts means that the errors are local and that the last observation does not diverge far from the selected interval's future values. On more extended forecasts, the repeated seasonal pattern would be superior to a naive approach.

With the models' overall performance in terms of error established and

Table 3.13: Total requests forecasts measured model mean MSE values per sample rate.

Model	15-Minute MSE	1-Minute MSE
ARIMAX with Fourier terms	6,078,994.88	24,234.51
Naive	13,442,580.23	33,192.76
AR(1)	11,099,274.31	69,731.61
Average	18,395,951.51	33,218.23
Seasonal naive	13,698,202.49	80,127.55
TBATS	1,7723,170.43	54,606.14
HTS Real-time	7,008,621.28	22,677.99
HTS Speculative	6,846,454.05	22,648.98
HTS Oracle	2,426,625.16	15,068.93

evaluated, an additional analysis into model behaviour was performed to gain a better understanding of the forecasts. Evaluation based purely on errors is insufficient as it may be misleading. The scale of the training set hid the information on how the models behave around sudden increases in requests. Furthermore, the sheer number of forecasts performed made it challenging to evaluate. The additional analysis was more visual, local and focused on request peaks.

3.9.6 Local forecast analysis

The local analysis was conducted on peaks occurring in the testing set of the data. The models selected for deeper analysis were the ARIMAX models, and all considered HTS models. In total, seventeen peaks have been observed during the testing set period. In terms of individual series impact, the shares of a short-living series ranged from 10.1% to 16.0%.

The ARIMAX, HTS real-time and HTS speculative showed similar behaviour around peaks on the fifteen-minute sample rate. That is, peaks outside of the modelled daily seasonality were not predicted but purely reacted upon. The models failed to predict peaks (such behaviour could not be expected of the ARIMAX model), but the models provide reasonable forecasts on future development during their occurrence. In the benchmark Oracle method, the more visual analysis can reflect on the previous assumptions about overall total series behaviour and how the tracked series influence it. The Oracle model forecasts the shape of the peaks, but examples exist where the model underestimates the total number of requests. The example of this occurring can be seen in one of the plots in the 3.22 figure. This suggested that the share the tracked series has, and therefore the most extensive short-living series with peaks, is lower than was initially assumed, as even a perfect forecast in tracked series can diverge from the actual observations.

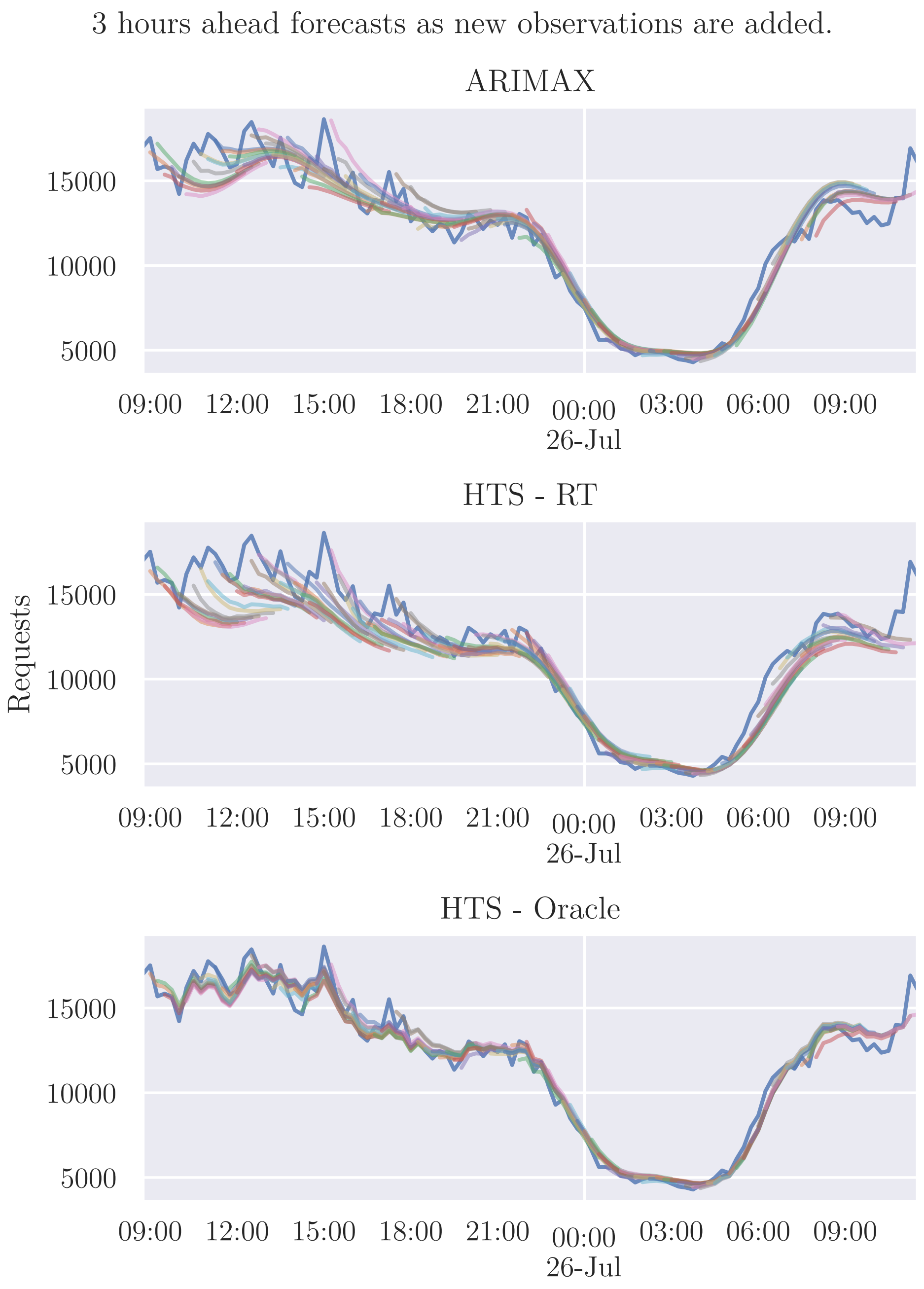

Figure 3.19: Figure shows live forecast made by the ARIMAX model.

Previous analysis showed that the peak, primarily formed by a new disruptive series, does not exist in a vacuum and that other series show similar behaviour during the occurrence of the peak, the sum of which then forms

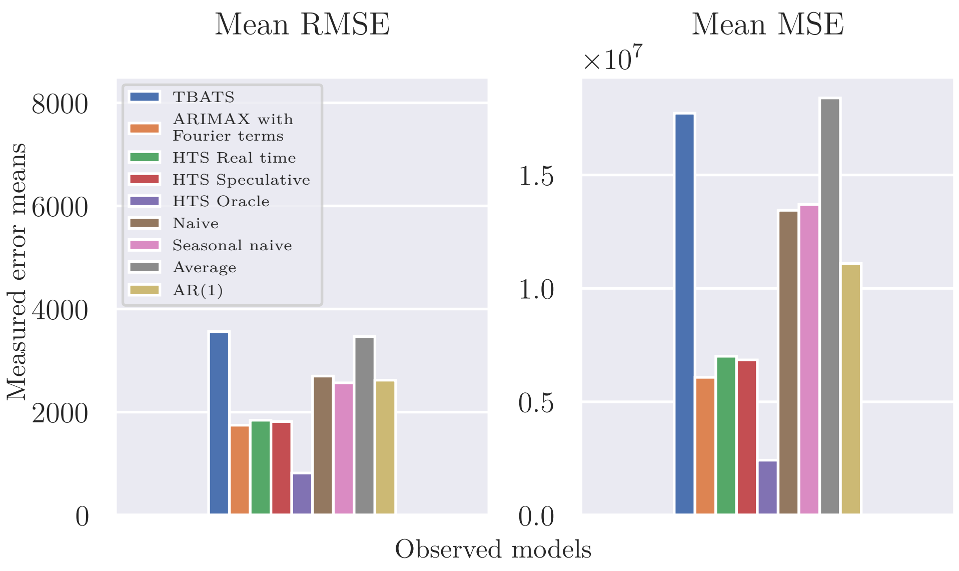

Figure 3.20: Figure shows 15-minute sample rate measured RMSE errors model comparison (Lower is better).

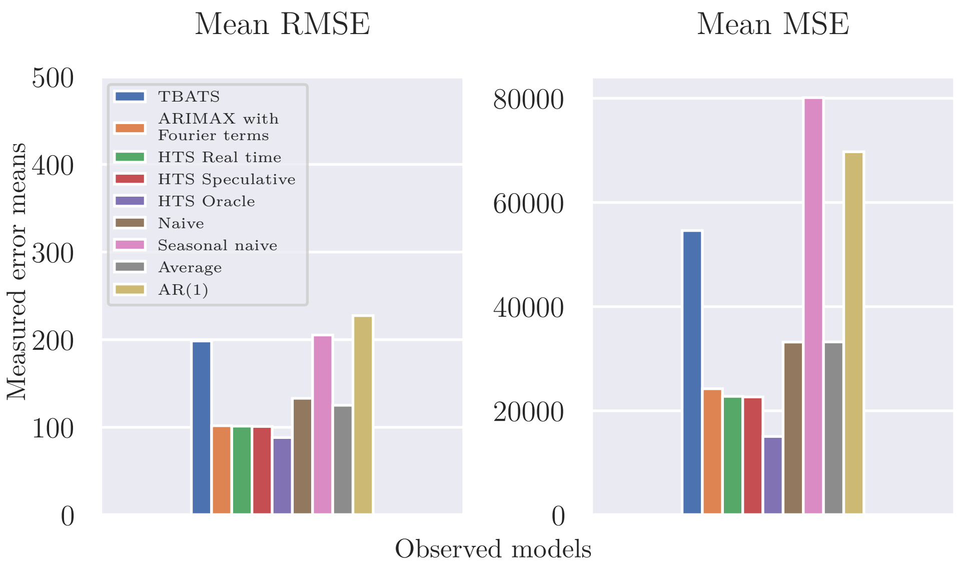

Figure 3.21: Figure shows 1-minute sample rate measured RMSE errors model comparison (Lower is better).

the total peak. A mild case of this occurrence can be observed in the first example in the 3.7 figure, where a sudden growth in the short-living series is accompanied by a lesser growth in the long-living series. The approach in this

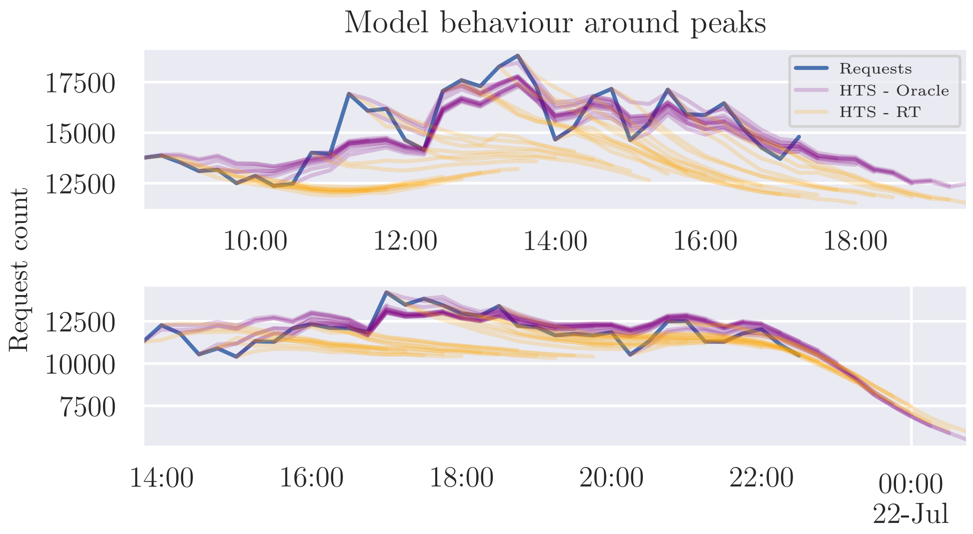

Figure 3.22: Figure shows two examples of 15-minute HTS Oracle and Real-time model behaviour around peaks. In the upper plot, the Oracle model underestimates the total number of requests, despite knowing the tracked series future values.

thesis underestimated the impact of this correlated growth behaviour between series groups. With this in mind, even if this shortcoming was addressed, the fifteen-minute sample rate would likely remain too slow to efficiently forecast sudden peaks that can occur in short periods of time. However, the models remain very capable of forecasting the near future development of the series and adapt to unforeseen changes.

The series at the one-minute sample rate is much more noisy and challenging to interpret. The visualisations become cluttered and unclear. Additional issues arise when considering simulation, with the available simulator being suited for a fifteen-minute sample rate. The steps in the simulator are chosen as 15 minutes. Besides, the logic used for scaling decisions based on the individual forecasts would likely have to be very complex and robust to accommodate volatile forecasts. Different approaches for transforming the one-minute sample rate forecasts into less volatile and robust forecasts have been explored to remediate the issues and allow for a closer analysis of model behaviour and simulation, such as averaging and merging multiple forecasts.

One of the simplest methods is to convert the forecasts to the same format as those seen on the fifteen-minute sample rate. The 30 steps ahead have been aggregated to two steps, and the same timestamp index has been used. This means forecasts made in between fifteen minutes were dropped. This leads to a significant loss of information. If this were the selected option, the advantages of using a small sample rate would disappear. It can, however, serve as a proof of concept, and indeed the observed summed forecasts seemed to fit the series relatively well.

Another approach was with using the means of the forecasts. When applied to all forecasts made during the half an hour that is the forecast window size, the mean fits the series very well, including peaks. This is not surprising, as this turns the forecasts into one step ahead since all forecasts need to be made before their mean can be computed.

A balance between those approaches had to be found. The benefits of the one-minute sample rate had to be preserved, and the outcome should have a minimal range of forecasts and should be on the fifteen-minute sample rate so that it could be used in the simulations.

While not ideal, the solution was to move to the fifteen-minute sample rate using a mean of the first five forecasts, that is, forecasts made over the first five minutes. This way, volatility is reduced, a significant portion of the information is preserved in the mean, and for the first step, it leaves a 10-minute in-advance window of when forecasts are available. This is especially important in the simulation, where the number of machines changes between steps and the time needed for machine instance start needs to be accounted for. This approach provides reasonable forecasts but can not accurately predict peaks. It can, however, forecast a trend around those. An example of such forecasts can be seen in the 3.23 figure. The usefulness of such a method should be determined by simulation, as a failure to accurately predict a peak in its full extent, but only a portion of it may still lead to a decision to scale up.

In general, a forecast that is at least hinting at a potential upcoming peak is unlikely to occur with any significant head-start. The time bought by this approach is potentially in a matter of few minutes rather than any significant block of time. This is reinforced by the peak occurrence analysis, which suggests little available time for reaction. Combined with the need for the regression model to react on more observations, leveraging the forecast is likely to prove difficult.

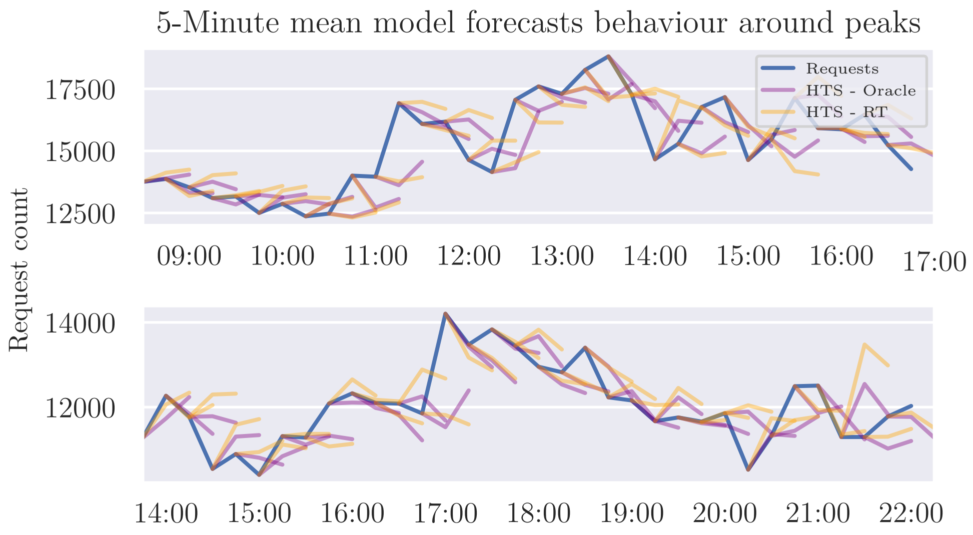

Figure 3.23: Figure shows mean of the first five forecasts made by the 1-minute models with 15-minute intervals.

3.10 Environment Scaling simulation

With the models compared, the forecasts from the HTS and ARIMAX models were taken and used in environment scaling simulations. Their behaviour has been compared not only between each other but also with other purely reactive forms of autoscaling and a static approach where the number of instances does not change. The simulator used was developed by Vondra [4] and was briefly introduced in the theoretical part of the thesis.

3.10.1 Simulation process

The simulation runs over the entire testing data set, for which forecasts are available.

Additional two functions combining reactive approaches with the available forecasts for auto-scaling were created. The more straightforward function, which only considers forecasts for scaling up, is defined by the 1 algorithm, the second one, which uses forecasts for both up and downscaling, is defined in the 2 algorithm.

The functions are basing the future number of machines on the forecasts made by a model of choice. This forecast is combined with observations of utilisation to anchor the forecasted series to the environment. The reactive elements step in with extreme cases where the forecast fails to predict the actual development in requests. The anchoring with utilisation is critical. Without

Algorithm 1: Logic used for forecast based scaling.

```
input  : $up$ - Change in requests per second for request growth
         $down$ - Utilisation threshold for down-scaling
         $utilisation_t$ - Percentage of environment utilisation at time $t$
         $h$ - Forecast steps ahead
         $on\_cooldown$ - Marks whenever a scaling decision was
         recently made, used to reduce volatility, when used marked
         with (cd).
output: $machines_{t+1}$ - Returns the number of machines needed at
         time $t + 1$.
if $utilisation_t \geq 75\%$ then
    // Model failed to forecast a surge, reactive scale up.
    return $machines_t + 1$;
else if $utilisation_t \leq 40\%$ & $\neg on\_cooldown$ then
    // Reactive scale down.
    return $machines_t - 1$;
if $requests_t + up < \hat{y}_{t+h}$ then
    return $machines_t + 1$;
return $machines_t$;
```

it, the decision would be made blindly regardless of the current state. This could then turn disastrous in cases where the forecast fails or gets too detached and would likely force a much more complicated implementation, which would likely end up suffering from the same detached symptoms. If the state of the environment is not aligned with the forecasting logic, the approach does not work. An example of such a scenario can be made even when all forecasts are successful. Regularly, at the end of the day, a decrease in requests can be observed. The models are usually able to predict this accurately. When the environment is undergoing this decline, each step where the forecasts are considered will notice this and notify the environment. This seems like a desirable outcome. However, without information about the current state of the environment, the next step will arrive to the same conclusion as the number of steps during the decline is larger than one. For each of the steps, the forecasts will be accurate, the decision correct, but in sequence, this will cause the environment to quickly reach the minimum allowed number of running machines which will lead to the environment being overwhelmed. Both functions contain a cooldown condition for down-scaling, which prevents down-scaling right after up-scaling. This reduces volatility and improves service quality at the cost of more used machine hours.

For each of the considered approaches (static, reactive and forecast based), a parameter sweep was run. This search shows hints to the behaviour of each method, but, most important in this section, it allows finding the optimal choice of parameters for each approach for the observed time series.

In addition, for each of the considered models in the forecast based approaches, the steps ahead are evaluated independently. That is, the parameter sweep process was applied multiple times per model, with different choice of h steps ahead.

As the large number of combinations from the parameter sweep was overwhelming, A set of conditions was placed on the results. From observations of the considered series, approaches with a large number of starts or stops were judged too volatile and were discarded. Results, where the reactive approach achieved good results in terms of Apdex but did not differ much from the static approach, were dropped as well. Additional results dismissed were based on the Apdex C. metric, results where the metric exceeded 2% were discarded, when no results could be found with a lower score, the smallest percentage achieved was used instead.

Out of these reduced results, the ones with the lowest Apdex C. and S. metrics, machine hours and lowest number of starts and stops and were chosen. When a reasonable trade-off between worse Apdex measures and lower starts and stops was found, it was added for consideration.

The simulation process was repeated twice with different simulator parameter for service demand. In the first option, this was chosen as 25 milliseconds. The results with this value were good for the majority of the considered methods, including reactive, which could deliver good efficiency and no service violations. The range of needed machines with this value is from four to roughly twenty around the peaks. This is a reasonable number of machines. This, however, did not allow for a good comparison as achieving good results was easy for the majority of the considered approaches. A second value chosen as 80 milliseconds comes from measurements of the web application, where 80 milliseconds was the measured average of one thousand HTTP requests collected by using the application. This value increases the strain on the environment, and the number of machines needed to provide service grew. This number of machines is less realistic, but the additional stress and volatility allow for a better comparison of the evaluated approaches.

Algorithm 2: Logic used for forecast based scaling.

```
input  : $up$ - Change in requests per second for request growth
         $down$ - Change in requests per second for request decline
         $utilisation_t$ - Percentage of environment utilisation at time $t$
         $h$ - Forecast steps ahead
         $on\_cooldown$ - Marks whenever a scaling decision was
         recently made, used to reduce volatility, when used marked
         with (cd).
output: $machines_{t+1}$ - Returns the number of machines needed at
         time $t + 1$.
if $utilisation_t \geq 75\%$ then
    // Model failed to forecast a surge, reactive scale up.
    return $machines_t + 1$;
else if $utilisation_t \leq 40\%$ & $\neg on\_cooldown$ then
    // Model failed to forecast request decline, reactive
       scale down.
    return $machines_t - 1$;
// No need for immediate reaction. Consider forecasts.
if $utilisation_t \geq 70\%$ then
    // Utilisation is high, consider forecast only for
       scaling up, else maintain current number.
    if $requests_t + up > \hat{y}_{t+h}$ then
        return $machines_t + 1$;
    return $machines_t$;
else if $utilisation_t \leq 50\%$ then
    // Utilisation is low consider forecast only for
       scaling down, else maintain current number.
    if $requests_t - down < \hat{y}_{t+h}$ & $\neg on\_cooldown$ then
        return $machines_t - 1$;
    return $machines_t$;
else
    // Utilisation is within reasonable bounds, consider
       forecast for scaling up or down.
    if $requests_t + up > \hat{y}_{t+h}$ then
        return $machines_t + 1$;
    else if $requests_t - down < \hat{y}_{t+h}$ & $\neg on\_cooldown$ then
        return $machines_t - 1$;
// Maintain current number, no growth or decline expected.
return $machines_t$;
```

3.10.2 Simulation results

The selected results from the simulation are in the 3.14 and 3.15 tables. The static approach serves as a baseline for comparison to other approaches, while the real observation approach uses the actual observed values instead of forecasts and can demonstrate the forecast-based method's capabilities.

In the case of 25-millisecond service demand, the measurements offer excellent results but are not very useful for direct comparison. The reactive utilisation approach achieves results that avoid SLO violations and are good in machine hours while avoiding volatility. The forecast based approaches offer similar results with reduced machine hours over the static approach while offering no or little to no SLO violations. The results vary in terms of starts and stops and machine hours, but the differences are not significant, and with parameter, tweaking can be replicated by other predictive approaches. Further forecast steps ahead were measured and considered but did not offer any benefits. For clarity, they are mostly not shown in the result tables. With these conditions using the utilisation based approach makes the most sense.

With service demand being chosen as 80 milliseconds, the results are more interesting as the difference between reactive and proactive approaches grows larger. In the reactive group, the utilisation approach achieves the best performance with results that significantly reduce machine hours over static approach or lesser reduction for slightly fewer SLO violations. For predictive approaches, the real observations offer the best possible result with eleven thousand machine hours and Apdex C. of 0.73%, a nearly identical result to the static approach with a significant saving of hours, that is when using 42 instances for comparison, which are better suited to deal with peaks, on 32 instances the advantage in hours is smaller. However, clear benefits in Apdex metrics can be observed. The SLO violations observed with 32 static instances are purely caused by sudden peaks. Other predictive approaches offer good results as well. The Mean HTS RT method using the algorithm 2 achieves the same results for Apdex metrics as with the real observations with a slight increase in machine hours. With a cost of the significant increase to hours, the mean speculative approach can replicate this using the 1 algorithm. A surprising result is that the Mean HTS Oracle method could reproduce the same result as Mean HTS RT only with a significant increase in machine hours. Overall, while differences have been measured, they are insignificant and can not serve as strong evidence for the superiority of the considered predictive methods. Even when using observed values as forecasts or the Oracle methods, the observed SLO violations are around extreme peaks, and other predictive approaches achieve close results. However, the predictive approaches appear to outperform the reactive approaches offering better or similar results in terms of SLO violations with lesser hours.

The scaling decisions made by selected methods can be observed in the 3.24 figure.

Table 3.14: Table shows selected results for different autoscaling approaches grouped by type using 25 ms service demand.

Forecast based - Algorithm 2

Model	*up*	*down*	*h*	Hours	Starts	Stops	Apdex S.	Apdex C.
Real Observations	23	16	1	3879.75	194	199	0	0.00%
ARIMAX	29.0	12.5	1	4048	165	170	0	0.13%
HTS RT	12.5	26.5	1	4265.5	165	170	0	0.00%
HTS Speculative	13.5	26.5	1	4110.25	185	190	0	0.06%
HTS Oracle	5.0	24.0	1	4268.5	169	174	0	0.00%
Mean ARIMAX	15.0	25.5	1	4130.00	193	198	0	0.06%
Mean HTS RT	30.0	21.0	1	4094.25	188	193	0	0.13%
Mean HTS RT	7.0	17.5	1	4133.00	203	208	0	0.06%
Mean HTS Speculative	19.0	22.5	1	4127.75	191	196	0	0.06%
Mean HTS Oracle	26.5	24.5	1	4089.50	177	182	0	0.06%

Forecast based - Algorithm 1

Model	*up*	*down*	*h*	Hours	Starts	Stops	Apdex S.	Apdex C.
Real Observations	30.0	45	1	3881.50	202	209	0	0.00%
ARIMAX	30.0	40	1	4134.25	195	201	0	0.00%
HTS RT	30.0	40	1	4054.25	176	182	0	0.00%
HTS Speculative	30.0	40	1	4071.0	180	186	0	0.00%
HTS Oracle	30.0	40	1	4043.00	173	179	0	0.00%
Mean ARIMAX	30.0	40	1	4332.00	214	220	0	0.00%
Mean HTS RT	30.0	40	1	4371.75	220	226	0	0.00%
Mean HTS Speculative	30.0	40	1	4203.25	198	204	0	0.00%
Mean HTS Oracle	30.0	40	1	4187.0	193	194	0	0.00%

Reactive

Method	*up*	*down*	*h*	Hours	Starts	Stops	Apdex S.	Apdex C.
Utilisation	55	35	—	4489.25	174	179	0	0.00%
Utilisation	60	40	—	4076.50	170	176	0	0.13%
Utilisation	65	45	—	3686.75	163	170	3	0.39%
Queue	10	5	—	3389.0	194	200	67	17.57%
Latency	35	30	—	2790.5	277	282	21	6.12%
Hybrid latency-utilisation	30	15	—	4939.75	16	14	8	0.59%
Hybrid latency-utilisation	30	20	—	4193.50	39	40	11	0.86%
Hybrid latency-queue	30	5	—	3409.50	190	196	11	1.99%

Static

Instances	*up*	*down*	*h*	Hours	Starts	Stops	Apdex S.	Apdex C.
16	—	—	—	6005.00	6	0	0	0.06%

Table 3.15: Table shows selected results for different autoscaling approaches grouped by type using 80 ms service demand.

Forecast based - Algorithm 2

Model	*up*	*down*	*h*	Hours	Starts	Stops	Apdex S.	Apdex C.
Real Observations	16	8	1	10985.50	413	415	9	0.73%
Real Observations	17	18	2	11307.00	397	397	9	0.73%
ARIMAX	5	21	1	12108.00	413	407	10	0.86%
HTS RT	7	23	1	11759.50	389	386	13	0.99%
HTS Speculative	8	22	1	11751.50	394	391	13	0.99%
HTS Oracle	8	29	1	12154.75	378	372	10	0.86%
Mean ARIMAX	6	29	1	12157.25	406	400	9	0.79%
Mean HTS RT	5	27	1	12335.25	405	399	9	0.73%
Mean HTS Speculative	6	12	1	11846.50	429	423	10	0.93%
Mean HTS Oracle	8	23	1	12029.00	397	392	10	0.93%

Forecast based - Algorithm 1

Model	*up*	*down*	*h*	Hours	Starts	Stops	Apdex S.	Apdex C.
Real Observations	16.0	50	1	11311.5	455	455	9	0.73%
Real Observations	13.0	40	2	12988.50	432	428	8	0.66%
ARIMAX	18.0	45	1	12412.50	432	428	11	0.93%
ARIMAX	23.0	55	1	10710.00	456	459	12	1.06%
HTS RT	12.5	45	1	12611.00	403	399	11	0.93%
HTS Speculative	8.0	60	1	11656.75	515	507	11	0.93%
HTS Oracle	8.5	55	1	12877.75	484	485	11	0.86%
Mean ARIMAX	19.5	40	1	12937.75	389	374	11	0.86%
Mean HTS RT	26.5	30	1	13869.00	294	285	10	0.86%
Mean HTS Speculative	10.0	45	1	14965.75	473	441	9	0.73%
Mean HTS Speculative	14.0	50	1	13283.00	440	436	10	0.86%
Mean HTS Oracle	15.5	50	1	12413.50	427	428	11	0.86%

Reactive

Method	*up*	*down*	*h*	Hours	Starts	Stops	Apdex S.	Apdex C.
Utilisation	70	25	—	12505.75	180	175	15	1.19%
Utilisation	60	25	—	14027.00	246	238	10	0.79%
Queue	20	15	—	15139.50	428	434	81	7.78%
Latency	90	85	—	8139.75	471	476	118	10.58%
Hybrid latency-utilisation	85	25	—	11178.75	113	108	30	2.39%
Hybrid latency-utilisation	85	30	—	10642.25	169	168	36	3.12%
Hybrid latency-queue	85	10	—	10224.50	293	295	80	6.79%

Static

Instances	*up*	*down*	*h*	Hours	Starts	Stops	Apdex S.	Apdex C.
42	—	—	—	15771	0	0	10	0.73%
32	—	—	—	12016	0	0	49	3.52%

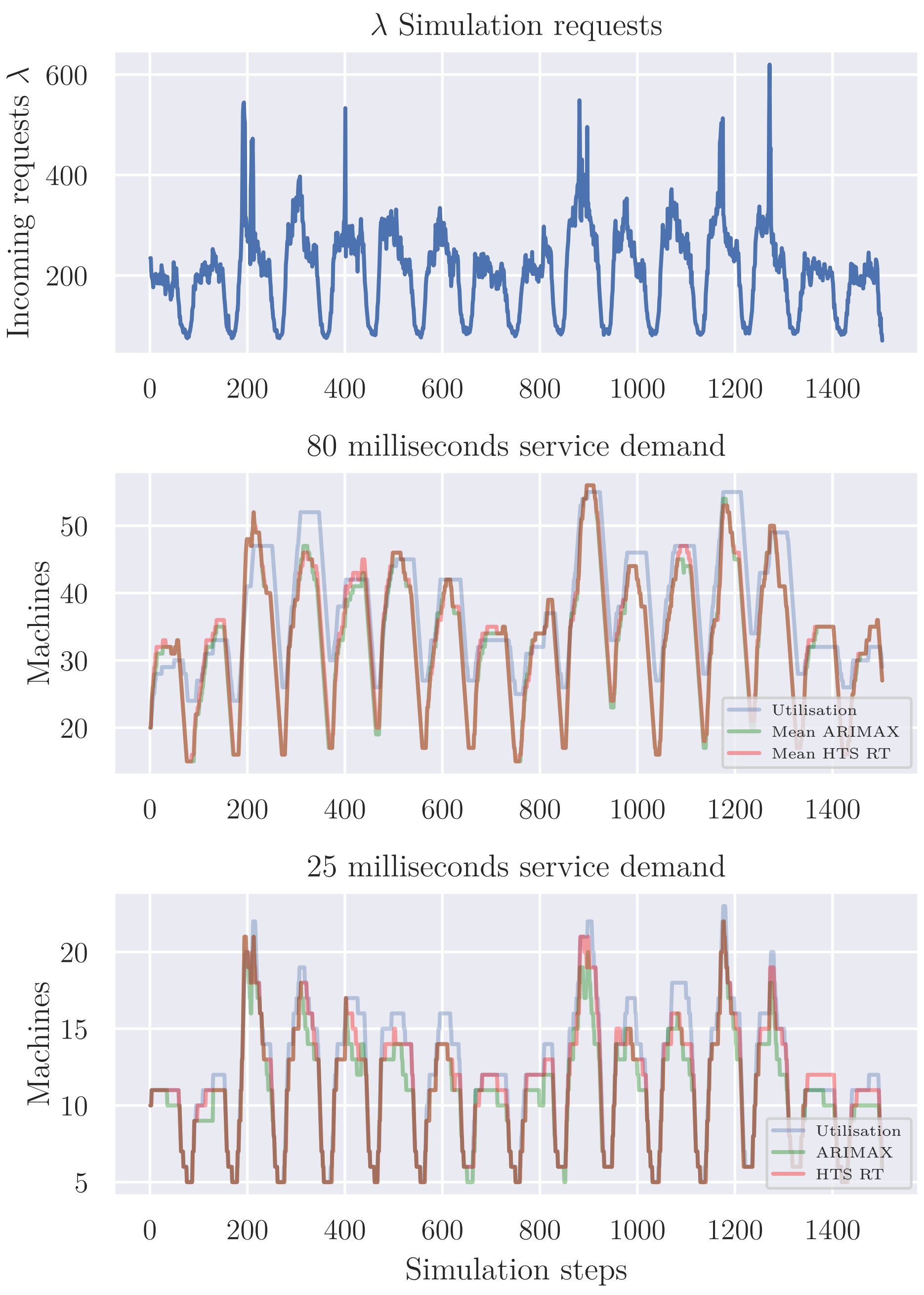

Figure 3.24: The top figure shows incoming requests rescaled to match the total volume of data noted by λ. The figures below show selected methods for autoscaling for two simulation scenarios with 80 ms and 25 ms service demand, respectively. In the 80 ms scenario, the Mean HTS RT forecasts mostly lead to the same decisions as with the ARIMAX forecasts but seem more aggressive around peaks. This is a desirable characteristic.

3.11 Discussion

The quality of the data played a significant role in this thesis. Even with efforts to split and label the data, the outcome was destined to remain speculative. A method with access to the original non-anonymised data should see improvements as the method for selecting tracked peaks could determine the importance of a series in real-time without delay. Even the speculative approach likely does not take full advantage of the data.

Additionally, the HTS method should see better results if the correlation of the series groups and their shared growth is leveraged. This varies through various encountered peaks. In some, this would be of minor significance or even negative, while it would be of significant impact in others.

The observed article peaks are shorter and with a smaller impact than initially assumed. No explored model was able to predict such sudden peaks accurately. The one-minute sample rate methods offer greater capability in forecasting around the peaks as the model is quick to react to the sudden surge even with a delay, the more long-term method on a higher sample rate is too slow to notice the peak, and frequently the sudden peak ceases to have a significant effect on the environment within the steps contained in one forecast.

The one minute models are the only models with a chance to predict peaks, but even if handled perfectly, leveraging them becomes difficult, mainly with the predictive autoscaling methods used. In this thesis, the forecasts had to be handled with a significant loss of information as the evaluation was otherwise unworkable. With a more granular approach to simulation with smaller steps in time (this significantly increases the complexity of the simulation), the results could be better evaluated. However, the best possible outcome is likely models reacting to a peak a few minutes ahead rather than any significant period of time.

Section 2.6 cites multiple publications with a focus on article prediction using additional features. Such features could compensate for the difficulties encountered during time-series peak prediction, where predicting the total peak size was unsuccessful. In addition, additional information about the articles could be used to potentially achieve a larger head-start and more accurate prediction.

The simulation outcome is mostly indecisive, with good results being achieved and methods making good autoscaling decision close to results achieved using real data. While the HTS models did seem to fare slightly better, the results are not significant enough to support the usage of such a method. This outcome suggests that further improvements to the forecast are unlikely to make a significant difference, and a more straightforward method may be preferable unless a predictive autoscaling approach that can better utilise the future peaks is found.

The main focus of this thesis was on time series forecasting using a combination of article peak forecasts with total request series for autoscaling. This proved a difficult task. As even an unlikely perfect forecast is of little value when it can not be leveraged by the autoscaling method, further focus should be placed on the autoscaling method and its adaptation to peak prediction.

The idea of keeping the time series forecasting simple and shifting the peak prediction attempts to the autoscaler component seems like a possible way forward. Implementing such a method could be more straightforward as using a single ARIMAX model is not difficult and offers good general forecasts during the entire day. This forecast could then be used as a component in more complex autoscaler logic making on both the forecast and additional article tracking component.

Chapter 4

Conclusion

During this thesis, the web traffic data has been cleaned, transformed and an analysis was conducted. The patterns appearing in the data have been identified and the total requests decomposed into individual series. Both the individual series and total requests have been corrected to remediate outages and anomalies where necessary.

Further efforts have been made to revert the loss of information formed by the anonymisation the data suffered. The data was split into preconceived behavioural groups based on the known structure of the application. With the use of hierarchical time series clustering, the process was successful, and the resulting split matched expectations.

With the data split finished, the resulting groups were analysed deeper. Special attention has been given to a large number of series with temporal peaks and their impact. Such series forming a large share of requests observed in the news application. The nature of the impactful initial peaks of such temporal series was analysed, and an understanding of the peaks resulted from the analysis. The individual impact of these temporal peaks was smaller than initially expected, and the time window until the full peak impact was found to be short.

Multiple forecasting approaches have been attempted and compared, spanning from forecast benchmark methods to established time series forecasting models and a combined approach using HTS models attempting to leverage the specific temporal series component of news application traffic. In addition, potential best-case variants of the HTS models have been observed to properly evaluate the theoretical gains from such approaches.

The resulting models have been compared using error measurements and visual inspection. As a result, the best performing models have been identified, and a deeper analysis focused on behaviour around request peaks has been performed.

A cloud environment simulation has been conducted to evaluate the usefulness of the resulting models. The simulation showed good performance but did not distinguish the peak-focused approaches from other predictive approaches in a major way. The explored methods are unlikely to provide more than a few minutes ahead head start to sudden peaks. Even with perfect forecasts, the bottleneck is the autoscaling method. A more sophisticated autoscaling method would be needed to fully take advantage of forecasted peaks regardless of the model.

Bibliography

1. CASTILLO, Carlos; EL-HADDAD, Mohammed; PFEFFER, Jürgen; STEMPECK, Matt. Characterizing the life cycle of online news stories using social media reactions. *Proceedings of the 17th ACM conference on Computer supported cooperative work social computing.* 2014. ISBN 9781450325400. Available from DOI: `10.1145/2531602.2531623`.
2. JAVŮREK, Karel. Na čem běží Seznam.cz: Běžný standard už nestačí, přechází na vlastní cloud i servery. *Connect.cz* [online]. 2018 [visited on 2021-04-28]. Available from: `https://connect.zive.cz/clanky/na-cem-bezi-seznamcz-bezny-standard-uz-nestaci-prechazi-na-vlastni-cloud-i-servery/sc-320-a-196137/default.aspx`.
3. SINGH, Parminder; GUPTA, Pooja; JYOTI, Kiran. TASM: Technocrat ARIMA and SVR Model for Workload Prediction of Web Applications in Cloud. *Cluster Computing.* 2019, vol. 22, no. 2, pp. 619–633. ISSN 1386-7857. Available from DOI: `10.1007/s10586-018-2868-6`.
4. VONDRA, Tomáš. *Automatic Scaling in Cloud Computing.* Prague, 2017. Available also from: `http://hdl.handle.net/10467/72822`. PhD thesis. Czech Technical University in Prague. Faculty of Electrical Engineering. Department of Cybernetics. Supervised by Jan ŠEDIVÝ.
5. PEREIRA, P.; ARAUJO, J.; MACIEL, P. A Hybrid Mechanism of Horizontal Auto-scaling Based on Thresholds and Time Series. In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC).* 2019, pp. 2065–2070. Available from DOI: `10.1109/SMC.2019.8914522`.
6. *Common Log Format* [online]. 2021 [visited on 2021-03-23]. Available from: `https://httpd.apache.org/docs/1.3/logs.html#common`.
7. HYNDMAN, R.J.; ATHANASOPOULOS, G. *Forecasting: Principles and Practice (3rd ed)* [online]. Melbourne, Australia: OTexts, 2021 [visited on 2021-02-07]. Available from: `https://Otexts.com/fpp3/`.

8. WANG, Xiaozhe; SMITH, Kate; HYNDMAN, Rob. Characteristic-based clustering for time series data. *Data mining and knowledge Discovery*. 2006, vol. 13, no. 3, pp. 335–364. Available from DOI: `10.1007/s10618-005-0039-x`.

9. HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. *The elements of statistical learning: data mining, inference and prediction* [online]. 2nd ed. Springer, 2009 [visited on 2021-03-18]. Available from: `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`.

10. PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, vol. 12, pp. 2825–2830.

11. *Hierarchical clustering* [online]. 2021 [visited on 2021-02-16]. Available from: `https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering`.

12. VAN CANNEYT, Steven; LEROUX, Philip; DHOEDT, Bart; DEMEESTER, Thomas. Modeling and predicting the popularity of online news based on temporal and content-related features. *Multimedia Tools and Applications*. 2018, vol. 77. Available from DOI: `10.1007/s11042-017-4348-z`.

13. KENESHLOO, Yaser; WANG, Shuguang; HAN, Eui-Hong; RAMAKRISHNAN, Naren. Predicting the Popularity of News Articles. In: 2016, pp. 441–449. Available from DOI: `10.1137/1.9781611974348.50`.

14. HYNDMAN, R.J. *Forecasting with long seasonal periods* [online]. 2010 [visited on 2021-03-13]. Available from: `https://robjhyndman.com/hyndsight/longseasonality/`.

15. SMITH, Taylor G. et al. *pmdarima: ARIMA estimators for Python* [online]. 2017 [visited on 2021-03-02]. Available from: `http://www.alkaline-ml.com/pmdarima`.

16. HYNDMAN, R.J.; ATHANASOPOULOS, G. *Forecasting: Principles and Practice (2nd ed)* [online]. Melbourne, Australia: OTexts, 2018 [visited on 2021-02-07]. Available from: `https://Otexts.com/fpp2/`.

17. LIVERA, Alysha; HYNDMAN, Rob; SNYDER, Ralph. Forecasting Time Series With Complex Seasonal Patterns Using Exponential Smoothing. *Journal of the American Statistical Association*. 2010, vol. 106, pp. 1513–1527. Available from DOI: `10.1198/jasa.2011.tm09771`.

18. *Forests of randomized trees* [online]. 2021 [visited on 2021-03-04]. Available from: `https://scikit-learn.org/stable/modules/ensemble.html#forest`.

19. SHALEV-SHWARTZ, Shai; BEN-DAVID, Shai. *Understanding Machine Learning: From Theory to Algorithms*. USA: Cambridge University Press, 2014. ISBN 1107057132.

20. DUDEK, Grzegorz. Short-Term Load Forecasting Using Random Forests. In: 2015, vol. 323, pp. 821–828. ISBN 978-3-319-11309-8. Available from DOI: 10.1007/978-3-319-11310-4_71.

21. *Nearest Neighbors* [online]. 2021 [visited on 2021-03-18]. Available from: https://scikit-learn.org/stable/modules/neighbors.html#regression.

22. LIU, Zitao; YAN, Yan; HAUSKRECHT, Milos. A Flexible Forecasting Framework for Hierarchical Time Series with Seasonal Patterns: A Case Study of Web Traffic. In: *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval*. Ann Arbor, MI, USA: Association for Computing Machinery, 2018, pp. 889–892. SIGIR '18. ISBN 9781450356572. Available from DOI: 10.1145/3209978.3210069.

23. HARRIS, Charles R. et al. Array programming with NumPy. *Nature*. 2020, vol. 585, no. 7825, pp. 357–362. Available from DOI: 10.1038/s41586-020-2649-2.

24. VIRTANEN, Pauli et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 2020, vol. 17, pp. 261–272. Available from DOI: 10.1038/s41592-019-0686-2.

25. TEAM, The pandas development. *pandas-dev/pandas: Pandas*. Zenodo, 2020. Version 1.2.2. Available from DOI: 10.5281/zenodo.3509134.

26. HUNTER, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*. 2020, vol. 9, no. 3, pp. 90–95. Available from DOI: 10.5281/zenodo.4268928.

27. WASKOM, Michael; TEAM, the seaborn development. *mwaskom/seaborn*. Zenodo, 2020. Version 0.11.1. Available from DOI: 10.5281/zenodo.592845.

28. HYNDMAN, Rob J; KHANDAKAR, Yeasmin. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*. 2008, vol. 26, no. 3, pp. 1–22. Available from DOI: 10.18637/jss.v027.i03.

29. SEABOLD, Skipper; PERKTOLD, Josef. Statsmodels: Econometric and statistical modeling with python. In: *9th Python in Science Conference*. 2010. Available from DOI: 10.25080/Majora-92bf1922-011.

30. POLÁK, Adam. Server Load Clustering. *eClub Prague, Unpublished*. 2015.

Appendix A
Project structure

- 0 Initial data transformation - Set of notebooks for initial data preparation and transformation.
  - 0.1 Data preparation - Notebook dedicated for preparing the data into easily accessible CSV format.
  - 0.2 Data resampling - Short notebook useful for transforming data between sample rates.
- 1 Data analysis - Initial analysis of the data.
- 2 Data preprocessing - Data outage correction.
- 3 Clustering features - Clustering feature calculation from series.
- 4 Clustering - Labelling series.
  - 4.0.1 Clustering template - Template notebook for clustering with different features.
  - 4.0.2 Clustering methods - A collection of reusable methods for clustering.
  - 4.1.1 All features - Clustering using all features.
  - 4.1.2 All features multiple clusters - Clustering using all features and multiple clusters.
  - 4.2 Model features - Clustering using model based features.
  - 4.3 Lifespan features - Clustering using features related to lifespan.
  - 4.4 Behaviour features - Clustering using behavioural features.
  - 4.5 Value features - Clustering using measured value features.
  - 4.6 Clustering review - The review of all considered clustering approaches.
  - 4.7 Review plots - Plotting the series by label for final evaluation.

5 Downtime data fix - Improvements in the initial data correction with improvements for forecasting.

6 Cluster analysis - Comparison analysis of the resulting clusters.

7 Peak analysis - A collection of notebooks for analysis of the short-living series and their peaks.

   7.1 Moving average options - Exploring the parameter options for moving averages.

   7.2 Method selection - Selecting the method for detecting peaks in the series.

   7.3 Undetected series group analysis - Analysing the short-living series without detected peak.

   7.4 Peak-series lifespan - Analysing the lifespan of short-living series with a peak.

8 Forecasting - Collection of notebooks covering forecasts of components and their combination.

   8.1 Forecasting methods - Reusable methods for forecasting.

   8.2.1 1-minute seasonal model - 1-minute sample rate forecasting model for long-living series.

   8.2.2 15-minute seasonal model - 15-minute sample rate forecasting model for long-living series.

   8.3.1 1-minute peaks model - 1-minute sample rate forecasting model for short-living series.

   8.3.2 15-minute peaks model - 15-minute sample rate forecasting model for short-living series.

   8.4.1 1-minute combined model - Combined model on 1-minute sample rate for total requests.

   8.4.2 15-minute combined model - Combined model on 15-minute sample rate for total requests.

   8.5.1 1-minute combined model - Combined model on 1-minute sample rate for total requests.

   8.5.2 15-minute combined model - Combined model on 15-minute sample rate for total requests.

9 Simulation plots - Simulation plots and simulation parameter search analysis.

Appendix **B**

Acronyms

AIC Akaike information criterion

ANN Artificial neural network

ARIMA Autoregressive integrated moving average

ARIMAX Autoregressive integrated moving average model with an exogenous variable

BIC Bayesian information criterion

CLF Common Log Format

CSV Comma-separated values

GB Gradient Boosting

HTS Hierarchical Time Series

HTTP Hypertext Transfer Protocol

KNN K-Nearest Neighbours

MSE Mean Square Error

RF Random forest

RMSE Root Mean Square Error

SLO Service Level Objective

SVR Support Vector Regression

Appendix C

Contents of enclosed CD

```
  readme.txt ............ the file with CD contents description
├── src ................. source codes directory
│   ├── notebooks ....... project notebooks
│   └── cloud-sim ....... modified cloud simulator files
├── plots ............... generated plots directory
│   └── plots.tar.gz .... archived plots
├── data ................ data directory
│   └── data.tar.gz ..... archived data, features, results
└── text ................ the thesis text directory
    ├── thesis.pdf ...... the thesis text in PDF format
    └── thesis.tar.gz ... archived LaTeX thesis source
```