REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

FORECASTING NETWORK TRAFFIC BASED ON SERVER LOGS

14011064 — Yağız Akyüz
15011087 — Görkem Şahin

SENIOR PROJECT

Advisor
Assist. Prof. Dr. Ziya Cihan TAYŞİ

January, 2021

ACKNOWLEDGEMENTS

We would like to express our deepest thanks to our dear advisor, Assist. Prof. Dr. Ziya Cihan TAYŞİ, who never stopped supporting and guiding us with his valuable knowledge, experience and cooperation.

We are grateful to Yıldız Techinal University Departmant of Computer Engineering. They provided us with the necessary facilities to carry out this study.

Last but not least, we would like to thank to our families and friends who supported and encouraged us during this project especially Harun Uz.

Yağız Akyüz
Görkem Şahin

TABLE OF CONTENTS

LIST OF SYMBOLS

LIST OF ABBREVIATIONS

ACF	Autocorrelation Function
AR	Autoregressive
CPU	Central Processing Unit
CSV	Comma Separated Values
JS	Javascript
JSON	Javascript Object Notation
MA	Moving Average
MAPE	Mean Absolute Percentage Error
PACF	Partial Autocorrelation Function
RAM	Random Acces Memory
SARIMA	Seasonal Autoregressive Integrated Moving Average

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Forecasting Network Traffic Based On Server Logs

Yağız Akyüz
Görkem Şahin

Department of Computer Engineering
Senior Project

Advisor: Assist. Prof. Dr. Ziya Cihan TAYŞİ

The aim of this project is to predict the upcoming web traffic of servers in order to be able to better manage resource allocation in advance. This is expected to reduce the amount of wasted resources while still providing a decent user experience to the clients. To do so, the dataset received from direnc.net was parsed, preprocessed, converted into different shapes of time series through various time interval and size calculation methods. Then, by analyzing the data, the data points with the highest amounts of correlation were deduced and the SARIMA model was trained accordingly. This model generates a JSON result file as a result of the training and forecasting processes. A web interface based on ReactJS and ChartJS was developed to display the results obtained. Using this interface, different prediction results can be viewed interactively.

It has been observed that the correct selection of model parameters and the data set are critical factors in the success of the system.

Keywords: Web Traffic Prediction, SARIMA, AR, MA, Regresyon, JSON, ReactJS, ChartJS

ÖZET

Sunucu Günlüğüne Dayalı Ağ Trafiği Tahmini

Yağız Akyüz
Görkem Şahin

Bilgisayar Mühendisliği Bölümü
Bitirme Projesi

Danışman: Dr. Ögr. Üyesi Ziya Cihan TAYŞİ

Bu projede web sitesi sunucu sağlayıcıların kaynak kullanımını daha efektif ve öngörülebilir hale getirmek için gelecekteki sunucu trafiğinin tahmin edilmesi hedeflenmiştir. Bunu gerçekleştirmek için çeşitli web sitelerinden alınan veri setleri işlenmiş; saniyelik, dakikalık, saatlik zaman aralıklarına bölünmüş, bu anlardaki değerler toplama, ortalama ve maksimumu bulma yöntemleri ile zaman serileri haline getirilmiştir. Daha sonrasında veri analiz edilerek korelasyonun en yüksek olduğu noktalar bulunmuştur ve buna uygun parametreler kullanılarak model eğitimleri yapılmıştır. Bu model eğitimlerinin sonucu bir adet JSON dosyaları halinde verilmektedir. Tahmin sonuçlarının görüntülenmesi için ise ReactJS ve ChartJS kullanan bir web arayüzü geliştirilmiştir. Bu arayüz kullanılarak farklı tahmin sonuçları interaktif şekilde görülebilmektedir.

Model parametrelerinin doğru seçilmesinin ve veri setinin sistemin başarısında kritik etmenler olduğu gözlemlenmiştir.

Anahtar Kelimeler: Web Trafik Tahmini, SARIMA, AR, MA, Regresyon, JSON, ReactJS, ChartJS

1
Introduction

In today's world, as a result of more and more people getting on the internet and more and more businesses digitalizing their work, there are many service providers that have to invest heavily in web servers. The amount of data transferred over the internet and the number of clients to be served are only increasing. Depending on the size of the user base, scale of the services, amount of the data utilized and the rising standards such as higher up-time requirements and lower latency times, domain providers need to ensure that they have allocated enough resources to satisfy their clients with a delightful experience. However, allocating more of these resources such as processing power and memory come with a financial cost. Under ideal circumstances a provider would want to know the exact amount of resources needed in advance to provide a good experience without wasting a dime.

The idea to develop this project derived from the aforementioned needs. The project aims to increase the efficiency of the server providers' resources by forecasting the network traffic in advance using regression, machine learning or neural networks.

Predicting the network traffic could help companies plan accordingly in advance and in a more precise manner. By having a good approximation, companies can more accurately allocate their resources such as CPU, RAM or network bandwidth so that clients are satisfied with the performance they experience while the providers do not waste unnecessary resources.

Conventional coarse-grain approaches had been used for this purpose which, instead of looking at the amount of data served or the number of requests received, predicted future CPU and RAM usage of a given server based on the previous amounts of resource demands. More accurate and fine-grain methods started being adopted with increased virtualization. These fine-grain methods allowed us to forecast multiple domains in a single system, which would not be possible with the rather old methods.

The goal of the project is to come up with the most accurate and precise predictions.

In order to achieve that, several kinds of models and parameters were tried and tested in this project. These methods and the reasoning behind them will be discussed in detail on the following chapters.

The results are produced to be consumed by users to make reasonable choices. In order to make this possible, a user-friendly solution is ought to be developed. Users should be able to give the necessary inputs -including the data set and various parameters- to the system and observe the outputs.

2
Preliminary Examination

There are a few scientific studies on web traffic forecasting publicly available today but this is a relatively new area and it's growing with the increased usage of virtualization. Essentially, web traffic forecasting problem is very similar to any other time series forecasting problem. The approaches to be utilized for forecasting the powerload of an electricity market [1] or predicting the number of cases for a disease [2], for example, are not much different. traffic-lstm

There are some studies [3] that used Weighted Random Forest Model and Deep Learning LSTM Model[4] for web server load prediction to reduce resource usage. There are also some studies that used neural network models [5] .

Since the nature of the data will be some type of time series, usage of autoregression and moving average methods were found to be beneficial. However, since the series that represent web traffic tend to show trends, integration could be also necessary, resulting in an ARIMA model. After further examination, since web traffic usage showed seasonal features as particular days of the week, specific hours of the day and some periods of the year are more prone to increased or decreased amounts of network traffic, a seasonal component was added to the ARIMA model and the regression model to be used was concluded as a SARIMA model.

Besides regression and machine learning methods, multi layer perceptrons and deep learning methods are also relevant. Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. As this model is better able to take previous data points under consideration, it is known as a very adequate model for time series predictions.

3
Feasibility

Here in this section the technical and legal details that concern this project are discussed in detail.

3.1 Technical Feasibility

Utilised software and hardware tools are discussed below.

3.1.1 Software Feasibility

3.1.1.1 Programming Language

Python, R, MATLAB and LISP are among the most popular programming languages for data engineering, data science and machine learning purposes.

Having a big community was an important criteria when it came to programming language selection since it would enable the developers to have a broad base to look at when they faced challenges. Python and R strived here with their large user base. Python's extensive arsenal of related libraries and modules, which the engineers were already familiar with, was another fact that had to be kept in mind.

Another consideration was how familiar the developers were with the mentioned programming languages, which pointed Python out once again.

Performance was another important factor: LISP wore the crown here with Python as a close second.

The cost of usage was our last consideration. Except for MATLAB, all of the aforementioned languages were free to use.

Neural Network and Deep learning ready libraries were also a quite big plus for the python.

Another advantage python had over the competing languages was ability to write server-side logic for our web application.

In conclusion Python suited our needs and knowledge the best.

3.1.1.2 User Interface for the End User

There were several candidate libraries and frameworks such as React.js, Angular.js and Vue.js that could be used for the web application part of the project. There are no big differences between them in case of performance, scalability and cost of usage so ReactJS was chosen due to developer familiarity. The charts can consist of many data points and the conversion of raw data to arrays that can be interpreted by Chart.js can be costly. React's memoized hooks were handy here.

3.1.1.3 Matplotlib

Matplotlib is a comprehensive library for creating visualization solutions in Python.

3.1.1.4 Pandas

Pandas is a data analysis and manipulation tool, built on top of the Python programming language.

3.1.1.5 Numpy

Numpy is a library for making scientific computing easier by providing many functions and data structures that vanilla Python lacks.

3.1.1.6 Statsmodels

Statsmodels is a Python module that provides many different statistical models, functions and tests.

3.1.1.7 ReactJS

ReactJS is a Javascript library for developing user interfaces for web applications that consist of reusable components and modules.

3.1.1.8 ChartJS

ChartJS is a Javascript library for creating and displaying interactive graphics.

3.1.1.9 Ant Design

Ant Design is a JavaScript library that provides styled and smart components.

3.1.1.10 Axios

Axios is a JavaScript library that provides development convenience for executing HTTP requests.

3.1.1.11 Fast API

Fast Api is an open source Python module that provides a server side logic.

3.1.1.12 PyTorch

An open source machine learning framework for Python, developed by Facebook.

3.1.1.13 Keras

An open source machine learning framework for Python that works with TensorFlow.

3.1.2 Hardware Feasibility

Deep learning applications require very high amounts of processing power, especially in training and testing stages. In machine learning, such levels of processing power is not needed. The hardware features of the computers used for machine learning and deep learning are given in the tables below.

Table 3.1 Minimum System Properties for the Project's Machine Learning Part

Hardware Name	Requirement
RAM	8GB
CPU	Intel i5
CPU Hızı	2.7 Ghz
Disk Size	60 GB

Table 3.2 Recommended System Properties for the Project's Machine Learning Part

Hardware Name	Requirement
RAM	8GB
CPU	Intel i5
CPU Hızı	2.7 Ghz
Disk Size	60 GB

Table 3.3 Minimum System Properties for the Project's Deep Learning Part

Hardware Name	Requirement
RAM	8GB
CPU/GPU	Ryzen 2400
CPU Speed	3.4 GHz
Disk Size	60 GB

Table 3.4 Recommended System Properties for the Project's Deep Learning Part

Hardware Name	Requirement
RAM	32GB
CPU	Ryzen 3700x
CPU Speed	4.4 GHz
Disk Size	120 GB
GPU	GTX 2080

3.2 Time and Workload Planning

3.2.1 Labor Force

The tasks of the people forming the labor force within the scope of the project have been determined.

Table 3.5 Labor Force

Name	Abbreviation	Type
Görkem Şahin	GŞ	Developer
Yağız Akyüz	YA	Developer

3.2.2 Tasks

Table 3.6 Gant Tasks

ID	Task Name	Start	End	Duration
1	Feasibility	October 5th	October 22th	16d
1.1	Time Feasibility	October 5th	October 7th	2d
1.2	Legal Feasibility	October 7th	October 9th	2d
1.3	Technical Feasibility	October 9th	October 19th	10d
1.3.1	Software Feasibility	October 9th	October 14th	5d
1.3.2	Hardware Feasibility	October 14th	October 19th	5d
1.4	Economical Feasibility	October 19th	October 21st	2d
2	Literature Review	October 21st	October 31st	10d
3	System Analysis	October 31st	November 10th	10d
4	System Design	November 10th	November 22nd	10d
4.1	Software Design	November 10th	November 18th	8d
4.2	Input - Output Design	November 18th	November 22nd	3d
5	Implementation of the System	October 21st	January 10th	80d
6	Performance Analysis	January 5th	January 10th	5d

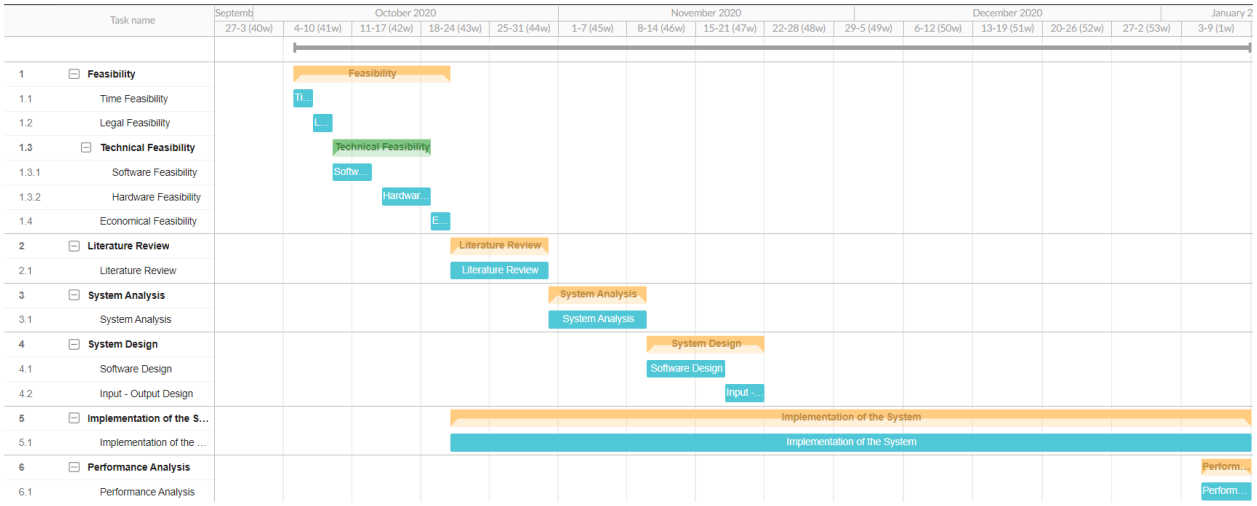


Figure 3.1 Gantt Diagram

3.3 Economic Feasibility

Total economic cost of this project is expected to fall between 6000 USD and 8000 USD. The detailed cost analysis of the project is discussed in the following sections.

3.3.1 Software Cost Analysis

The programs used in the development process of the project are either open source or free. For this reason, there is no software cost to account for.

3.3.2 Hardware Cost Analysis

There are 4 types of computer specification we examined. Two under machine learning and two under deep learning.

Table 3.7 System Costs

Hardware Name	Cost
Minimum Machine Learning System	500 USD
Recommended Machine Learning System	1000 USD
Minimum Deep Learning System	800 USD
Recommended Deep Learning System	2000 USD

3.3.3 Labor Force Cost Analysis

The project approximately takes 150 days of work to finish. Which approximately costs 5000 USD.

3.4 Legal Feasibility

Developers, product owners, supervisors and other individuals or institutions that played a role on the development and publishment of this project do not accept any responsibility regarding the consequences of its usage. All rights of the developed project are reserved, can not be copied or used without permission.

3.4.0.1 Data Usage

The data used to train and test the models developed in this project is kept confidential. Neither any information related to the owner of the data nor any other information regarding the requests and web traffic aside from the amount of the data served is saved.

3.4.1 Licenses of Software Used

All software and licenses used and planned to be used in the project development process will be examined under this title.

3.4.1.1 Python

Python, as the programming language used in this project, is licensed under a free software license called the GNU General Public License. This license gives developers various rights and freedom such as copying, distributing, working on, modifying and developing software.

3.4.1.2 Javascript

Javascript, is an open standard used widely for creating interactive web sites around the globe.

3.4.1.3 Windows

All development will be made on top of the retail type of license of the Windows operating system.

3.4.1.4 Git

Git as a versioning tool licensed under a the GNU General Public License. This license gives developers various rights and freedom such as copying, distributing, working on, modifying and developing software.

3.4.1.5 Visual Studio Code

Visual Studio Code is a free IDE used to code in Python during the development phase of this project.

4
System Analysis

Here in this section the goals and the necessary modules of the project are discussed in detail based on the preliminary examination.

4.1 Goal of the Project

The end goal of the project is to allow the end users, who are web site owners or service providers, to be ready for the upcoming traffic their web site or clients' website will receive so that they can ready their infrastructure and resources to endure that. In order to achieve this, being able to come up with accurate and precise predictions is essential. In order to start predicting, server logs describing the previous traffic is ought to be parsed and preprocessed. The last step prior to forecasting is analysing the data at hand and coming up with appropriate parameters. Once the predictions are acquired, they need to be communicated well enough to the end user so that the necessary actions can be comprehended and taken by the user.

4.2 Scope of the Project

4.2.1 Sides

This subsection is dedicated to describe the sides that participate, their goals and responsibilities.

4.2.1.1 Server Owner or Service Provider

Server owners or service providers want to be able to predict the traffic they will face in the upcoming weeks or months so that they can allocate resources in advance. This will allow them to provide a seamless user experience while wasting the minimum amount of resources possible. They will have to share the server logs with the developers so that it can be processed, trained upon and further predicted.

4.2.1.2 Engineers

Engineers' goal is to help the server owner or service provider with resource allocation by letting them know about the necessary amounts in advance. Once the server logs are handed over to the engineers, they figure out how to parse the data and turn it into a form that is consumable by the model. This step is not automated since server logs can come in many shapes based on the software or hardware used and the amount and type of data owner or provider is willing to share. Then the data is preprocessed and visualized by the engineers to eliminate invalid entries and deduce the characteristics of the data at hand. Based on these deductions, the model designed by the engineers is fed with the appropriate parameters. They share the forecasted traffic and the tools to view it with the server owners or service providers.

4.2.2 Work Flow

Here the steps to produce useful outcomes are listed in order and described.

4.2.2.1 Data Acquirement

Server owners or service providers export the logs and share them with the engineers.

4.2.2.2 Data Transformation

Server logs are transformed into a CSV file consisting of date-time and size columns where each row represent a response to a request.

4.2.2.3 Data Preprocessing

Data is preprocessed to the extend that what is left is reliable and useful, thus can be used to make deductions and utilised to train a model. This step also produces a time series based on two parameters: time interval and size calculation method.

4.2.2.4 Data Analysis

Data is analysed and its characteristics are extracted through various methods such as visualization or functional calculations.

4.2.2.5 Model and Parameter Selection

Based on the characteristics of the data, an appropriate model among regression, machine and/or deep learning methods should be chosen. The model should be fed

with appropriate parameters which are deduced based on the analysis executed at the previous step.

4.2.2.6 Testing

A variety of models and parameters should be tried and the results should be compared with one another to see which combinations of models and parameters are better forecasters.

4.2.2.7 Demonstration of the Results

A simple to use and interpret but also detailed and comprehensive way of demonstrating the results should be implemented. End user should be able to tell what the load on his website will look like in the upcoming days, weeks or months and be able to take accurate precautions beforehand.

4.2.3 Modules and Components

Most of the steps named in the work flow section are covered by the following modules and components.

4.2.3.1 Parser

This module gets rid of invalid or irrelevant information stored within the logs in case the service provider did not filter the logs beforehand, and then outputs a CSV file that only consists of the requests, their timing and the size of the data served in response.

4.2.3.2 Preprocessor

Preprocessor produces a time series with the given interval and supplies values for time points by summing the values, picking the most significant ones or through another method.

4.2.3.3 Visualizer

This module visualizes the given time series, the relationship between datapoints and the series' characteristics in order to assist the engineers to find where to begin in terms of parameters.

4.2.3.4 Forecaster

Here the model to be trained with the dataset is defined and fed with parameters. This module receives a dataset, bunch of parameters to feed the model with and outputs a JSON file that will later be consumed by the front-end application.

4.2.3.5 Server

Models, the process of training them and forecasting the future traffic need a suitable infrastructure and platform. This requires a proper back-end solution. Server will be responsible of receiving a data set and a set of parameters to train the designed models. Predictions will be served to the front-end client solution in a format that is interpretable.

Server needs to be composed of multiple modules for maintainability. The controller layer will expose end points for communication through the HTTP protocol. The parameters received should be passed onto the service layer where the actual business logic is executed, models are trained and predictions are made. Finally, the response to the initial request should be formed and served through the controller layer.

4.2.3.6 Web Application

The front-end application is where all the interaction between the user and the system takes place, therefore it should have an elegant look and a delightful user experience while providing the desired amount of functionality. Utilized models need seperate forms respectively due to them requiring different amounts and types of parameters, but the way to display results can be common as all of them boil down to a chart with two lines: actual data and predictions. User should be notified about the ongoing processes and possible occasions such as training and occurance of errors.

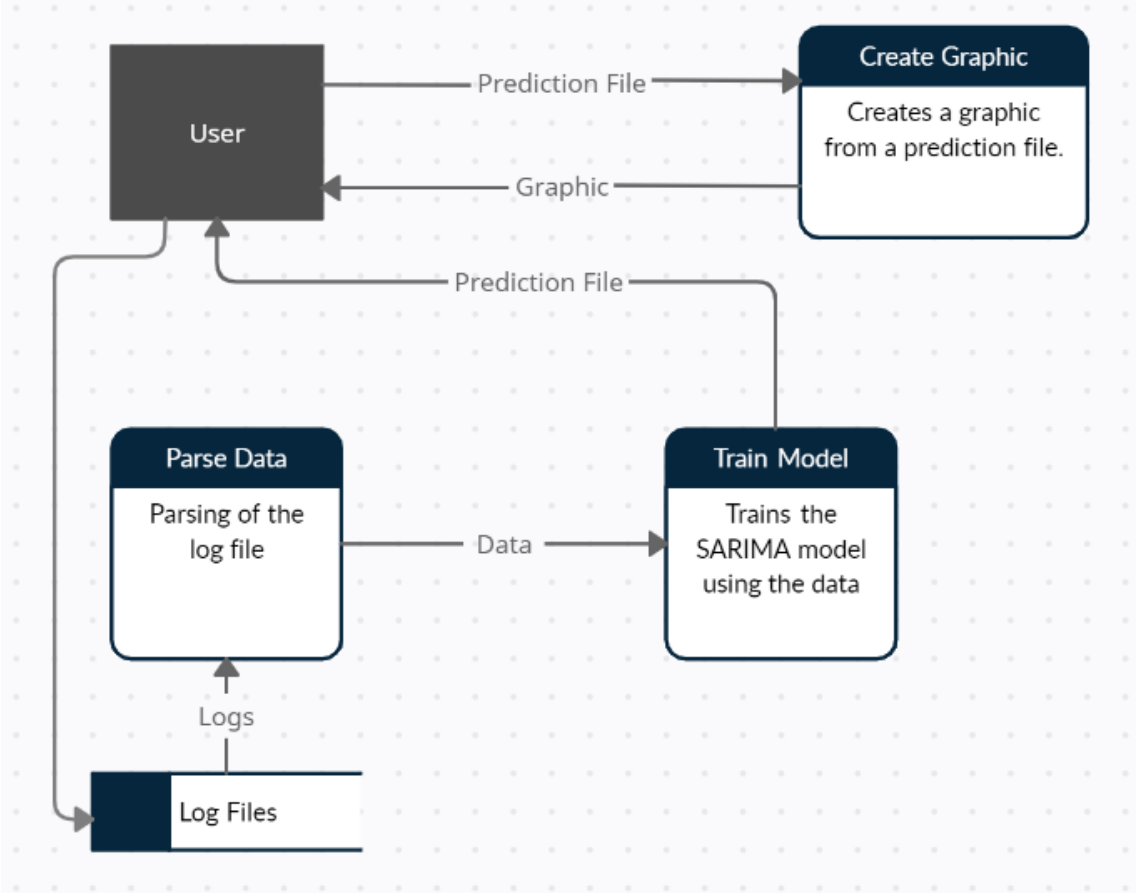


Figure 4.1 Data Flow Diagram

5
System Design

In order to estimate the traffic a web server will be exposed to in the future, having an accurate dataset with enough sample size and an appropriately selected method with well chosen parameters are essential. What has been done at this stage was to analyze the data, preprocess it to have a more manageable and meaningful dataset, visualize it to make further deductions, use various functions to determine the appropriate parameters for the time series forecasting methods to be used, and implement a solution to make these predictions accessible by the end users.

5.1 Dataset

Server logs of several websites including modasevim.com and direnc.net and different time frames withing these logs were used.

Initially the biggest continous dataset present was the one representing the traffic modasevim.com had received in June 2020, so it was chosen since the sample size of a dataset and having a broader base to train the model with is quite important for the accuracy and precision of the predictions.

Later on, the second and bigger dataset was acquired which was the traffic direnc.com had received between June and December of 2020. The logs were not continuous between months due to missing days, so multiple months could not be used at once to make predictions. September was complete as a month, so the bulk of the research was made on that month.

These datasets were acquired from the databases that stored the logs for each request these websites received during the mentioned time period.

5.2 Parser

Log files were partitioned into months and each file was around 20 GBs. Removing the irrelevant or invalid data and reducing the file size was necessary. Parser reads lines from the log files in chunks of one GB at a time, prints them onto a CSV file and reads more until all logs are consumed. Logs with invalid values are detected and eliminated. There is a lot of data per request that is irrelevant to the scope of this project, so these are not carried over to the CSV file. CSV file consists of two columns, date-time and size, where each row represents a single request.

5.3 Preprocessor

Training the model based on individual requests would take too much time since the amount of data points would be enormous. Preprocessor is a module that turns individual requests into time series based on a given interval and behaviour. Data can be grouped and timed in terms of various intervals such as days, hours, minutes or seconds. The value per time point can be chosen by a variety of methods such as by summing all the data that was served during this time period, the maximum amount of data served during a request or the average amount of data served per request within this time period.

Many combinations of different time intervals and behaviours were tried and tested. At the end of the day, the total amount of data served per hour proved to be more effective in terms of prediction quality and convenient to handle in the following steps. Seconds and minutes were too unpredictable as sudden spikes and drops were too dominant, causing high amounts of correlation between irrelevant time points. Days reduced the time series to the point where there were literally too little points to train the model with. Hours proved to be the most reliable time periods as they provided the middle ground between minutes and days. Correlation between same hours on different days were quite clear and due to the same reason, picking parameters accurately was possible, thus resulting in predictions that were way more precise and accurate.

Average, sum and max behaviours were considered when it came to assigning values to time points. Picking the maximum amount of size served in a request during a given time interval was a safe bet to begin with since allocating more resources than the actual traffic needed would mean wasting resources, but allocating less would have a direct impact on the user experience. This approach failed due to the same exact reason minutes and seconds failed as time intervals. Maximum data served per request was unreliably random. Average data served per request provided no additional

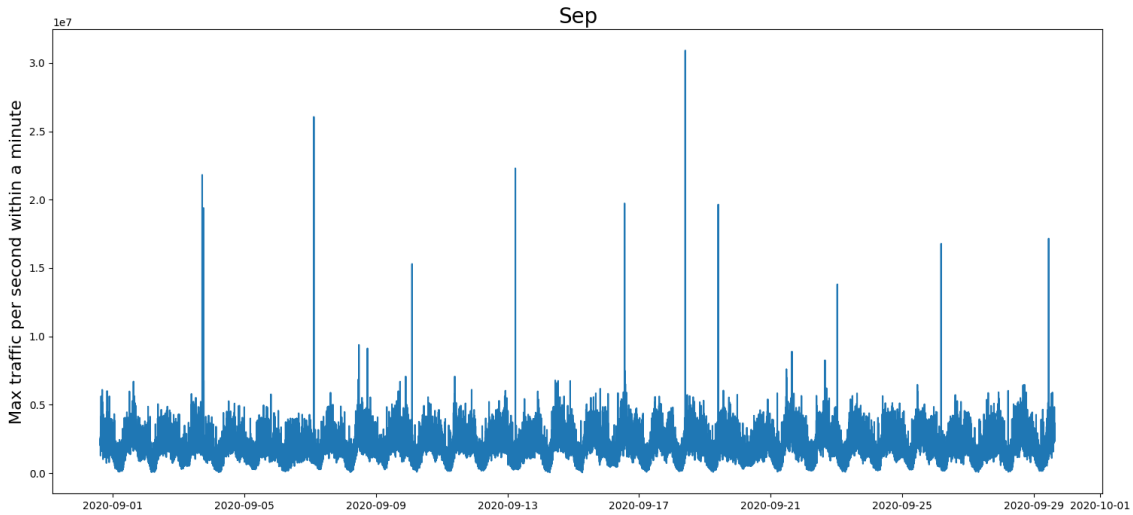


Figure 5.1 Series based on minutes and seconds had too many uninterpretable spikes which were eased out in hourly periods

benefit over summing all the requests, and came at the cost of more computation. Summing all the data served during an hour was clearly superior to others, and had less time and space complexity cost.

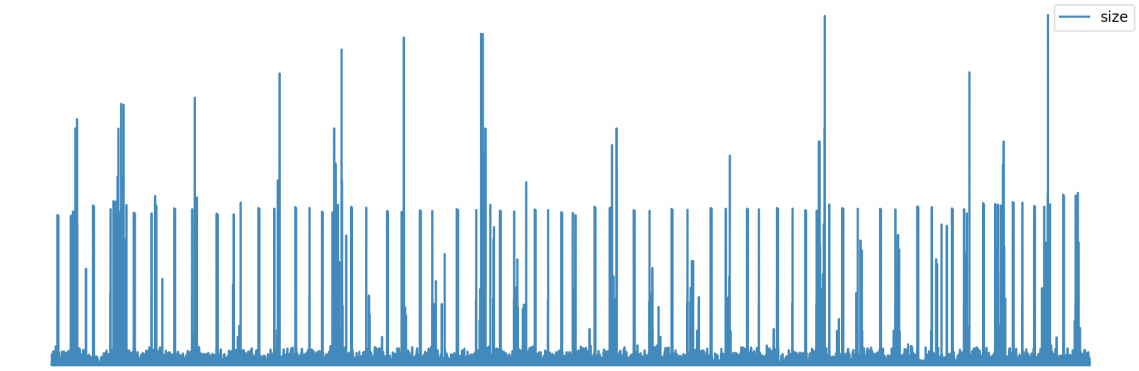


Figure 5.2 Dataset Format

The graphic above is a visualition of the raw dataset and it has a datapoint for each request received. The value on the y axis represents the amount of the data that was served for a request.

5.4 Visualizer

In order to see the characteristics of the series at hand, visualizing it is very useful. Seasonal features, trends and patterns could be observed this way. It also allows decision makers to visually see that there is nothing particularly weird about the series, thus establishing a level of confidence on the predictions based on the particular series.

The data clearly had a seasonal component as days and weeks resembled similar

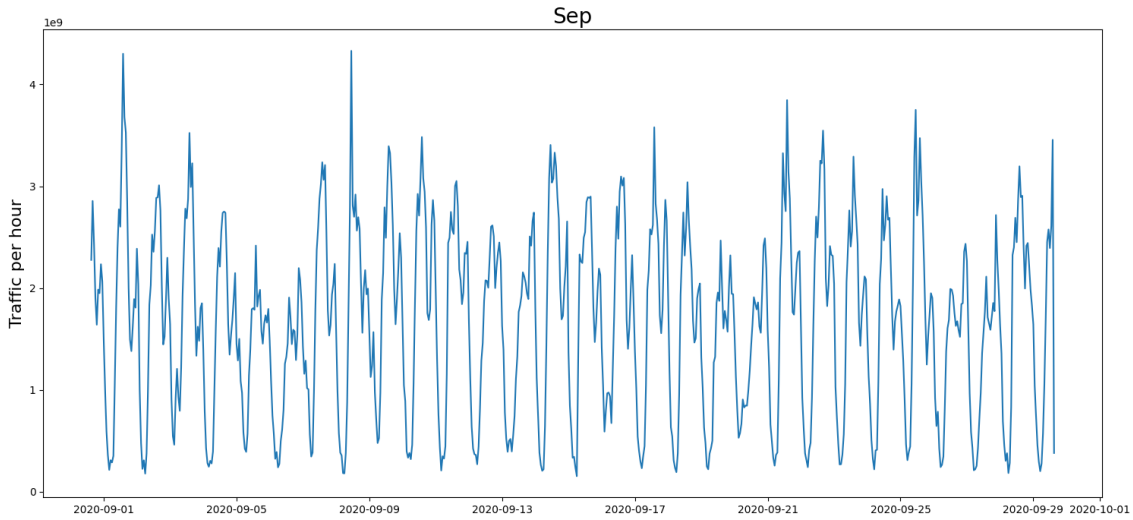


Figure 5.3 Series

patterns. There were also trends where the value keeps going up or down for various periods of time.

Series itself was not the only visualization that was made. In order to get a hold of the type and amount of correlation between different points in time, ACF and PACF were utilized. Sudden and reoccuring spikes at specific intervals meant seasonal features and increases or decreases over time pointed at trends.

5.5 Forecaster

The methods used in time series forecasting are discussed in detailed in this section.

5.5.1 Regression

Regression is one of the most important and broadly used machine learning and statistics tools. It makes predictions from data by learning the relationship between features of the data and some observed, continuous-valued response.

Machine learning is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering, computer vision and forecasting (as we will be doing in this case), where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

5.5.1.1 ARIMA

Autoregressive Integrated Moving Average, or ARIMA, is one of the most widely used forecasting methods for univariate time series data forecasting. As its name suggests, it supports both autoregressive and moving average elements. The integrated element refers to differencing allowing the method to support time series data with a trend. These components will be discussed in detail later on in the SARIMA section. The problem with ARIMA in our case is that it does not support seasonal data. ARIMA expects data that is either not seasonal or has the seasonal component removed, e.g. seasonally adjusted via methods such as seasonal differencing.

5.5.1.2 SARIMA

Seasonal Autoregressive Integrated Moving Average is an extension of ARIMA that explicitly supports univariate time series data with a seasonal component. A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA model. The seasonal part of the model consists of terms that are very similar to the non-seasonal components of the model, but they involve backshifts of the seasonal period. Since our dataset has seasonal features, SARIMA is a better fit as the model to embrace.

5.5.1.3 Components of SARIMA

5.5.1.3.1 Autoregression: A model that uses the dependent relationship between an observation and some number of lagged observations. The autoregressive model specifies that the output variable depends linearly on its own previous values and on an imperfectly predictable term; thus the model is in the form of a stochastic difference equation (or recurrence relation which should not be confused with differential equation).

5.5.1.3.2 Integration: The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary. To be able to make predictions using machine learning models, acquiring a stationary series is necessary. To achieve this, differencing the series at hand by one is usually enough. That being said, the data set used for this project was stationary to begin with. In order to scientifically confirm that the series were stationary, the Augmented Dickey Fuller test was used. A p-value smaller 0.05 indicated that the series were stationary.

5.5.1.3.3 Moving Average: A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations. A moving average is commonly used with time series data to smooth out short-term fluctuations and highlight longer-term trends or cycles. The threshold between short-term and long-term depends on the application, and the parameters of the moving average will be set accordingly.

5.5.1.3.4 Seasonality: Adds three new hyperparameters to specify the autoregression (AR), differencing (I) and moving average (MA) for the seasonal component of the series, as well as an additional parameter for the period of the seasonality. A seasonal ARIMA model uses differencing at a lag equal to the number of seasons (s) to remove additive seasonal effects. As with lag 1 differencing to remove a trend, the lag s differencing introduces a moving average term. The seasonal ARIMA model includes autoregressive and moving average terms at lag s.

5.5.1.4 Training

In order to train the SARIMA model with a dataset, appropriate values for the following parameters need to be deduced based on the characteristics of the dataset at hand.

- p: Trend autoregression order.
- d: Trend difference order.
- q: Trend moving average order.
- P: Seasonal autoregressive order.
- D: Seasonal difference order.
- Q: Seasonal moving average order.
- s: The number of time steps for a single seasonal period.

Using these two functions, the most appropriate values for these parameters need to be determined.

5.5.1.4.1 ACF: The autocorrelation function is a measure of the correlation between observations of a time series that are separated by k time units (yt and yt–k).

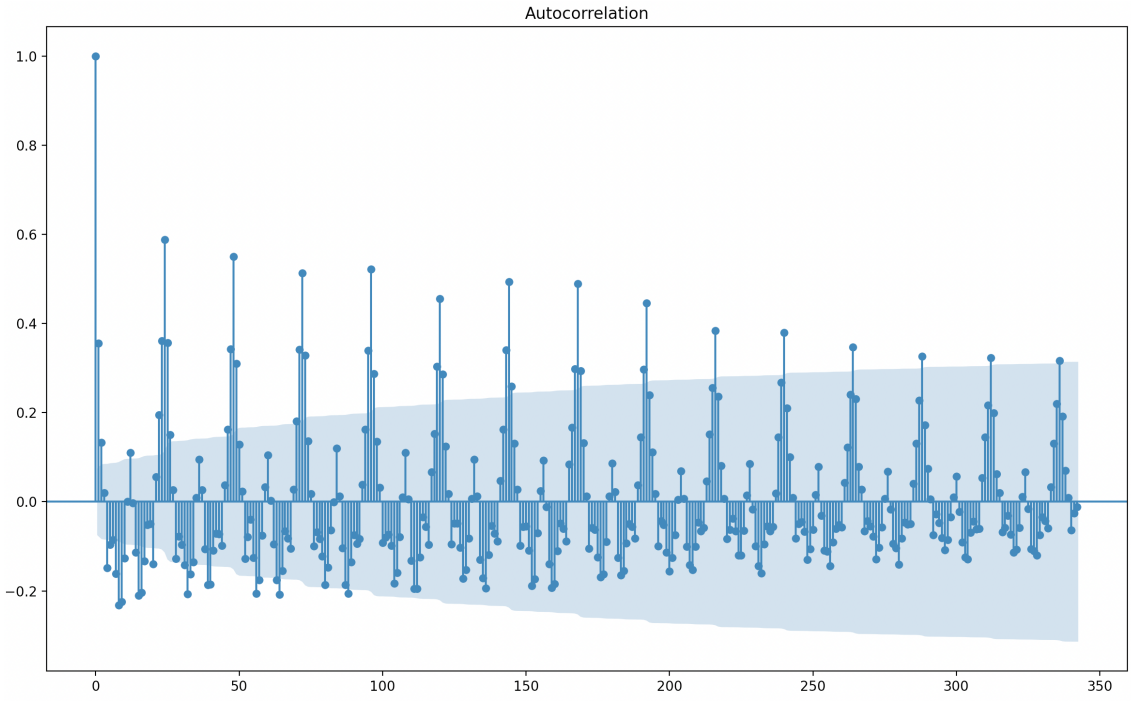


Figure 5.4 Autocorrelation Function

This will help us decide what values should be passed as p, q, P and Q by letting us know which points in time are relevant.

Based on this graphic, it can bee observed that the autocorrelation between the first few and also every 24th datapoint is quite high. It means that an autoregression with seasonal features should be implemented.

5.5.1.4.2 PACF: In time series analysis, the partial autocorrelation function states the partial correlation of a stationary time series with its own lagged values. A partial autocorrelation is the amount of correlation between a variable and a lag of itself that is not explained by correlations at all lower-order-lags. This function eliminates the correlation caused by the values in between two data points.

Here in this graphic a very high partial autocorrelation between the first, 24th and 48th datapoint can be seen. It can be explained by the 24th and 48th hours being the same hour during the previous days.

5.5.1.4.3 Parameters for the AR, I, MA and Their Seasonal Counterparts As can be seen in the previous paragraphs, seasonal auto and partial auto correlations were much higher compared to their non-seasonal counterparts, and including non-seasonal inputs only lowered the accuracy of the predictions on almost all

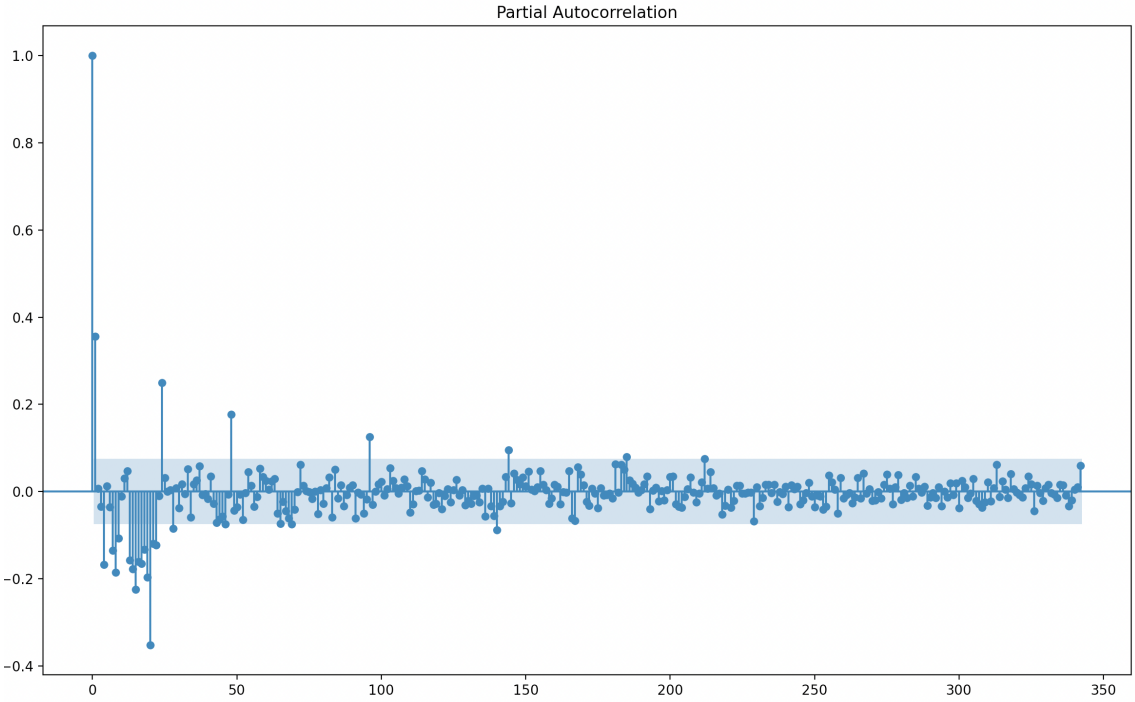


Figure 5.5 Partial Autocorrelation Function

datasets.

On the previous sections it was discussed that differencing the values was not necessary and doing so caused minor data losses, so the d value will be 0 as well.

Moving onto the seasonal components, on the ACF and PACF subsections it was discussed that there was a clear daily seasonality, which means that the value 24 should be used for s. On the PACF graphic we saw that the datapoint at a week prior to the point at hand had a very high correlation, so the value (7) should be used as Q, which skips all points until and after 7th seasonal data point, only taking the exact hour and day from the previous week under consideration.

As a result of these considerations, values 0, 0, 0, 8, 0, (7) and 24 were determined for the p, d, q, P, D, Q and s parameters respectively.

5.5.2 Multilayer Perceptron

Multilayer Perceptron is a neural network model. It has 3 sub-layers. These are input layer, hidden layer and output layer.

These 3 layers have neurons in them and all of the the neurons from one layer are connected to all of the neurons from the next layer.

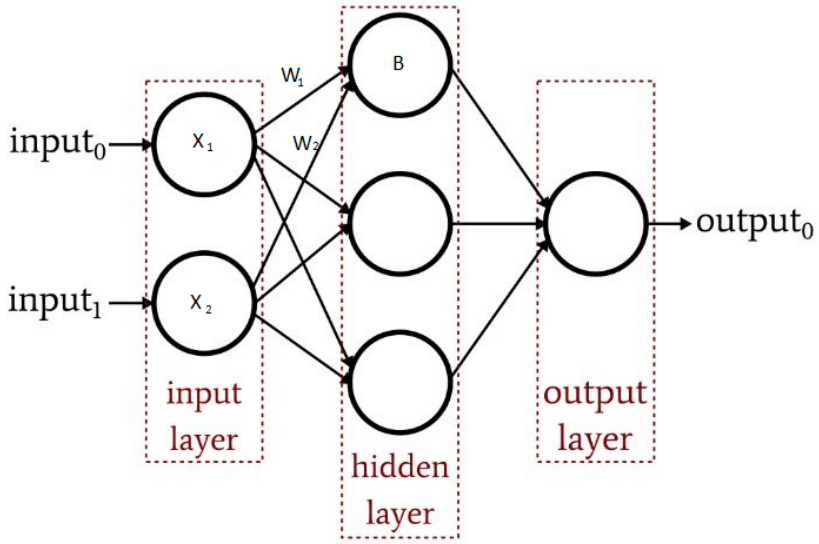


Figure 5.6 Neural Network

5.5.2.1 Hyper Parameters

To train the neural network you have to set some hyper parameters. These hyper parameters define the characteristics of the model. There hyper parameters are neuron count, dropout rate, learning rate, number of epochs.

- Neuron Count: Number of neurons in the hidden layer.
- Dropout Rate: How much of the knowledge will be dropped in each layer.
- Learning Rate: Convergence rate of the model to local minimum.
- Number of Epochs: Number of epochs the model has.

5.5.2.2 Training

To train the neural network, different hyper parameter combinations have been used and these hyper parameters' effects have been observed. MEPA of 26 has been achieved. Learning rate of 0.0008, neuron count of 256, dropout rate of 0.2 and number of epochs of 75 was the best combination in terms of MEPA.

- Neuron Count: Succes rate is increasing but also costly and may cause overfitting.

- Dropout Rate: There wasn’n a significant change.
- Learning Rate: Succes rate is increasing with lower learning rate but costly and may cause overfitting.
- Number of Epochs: Succes rate is increasing but also costly and may cause overfitting.

5.5.3 Long Short Term Memory Network

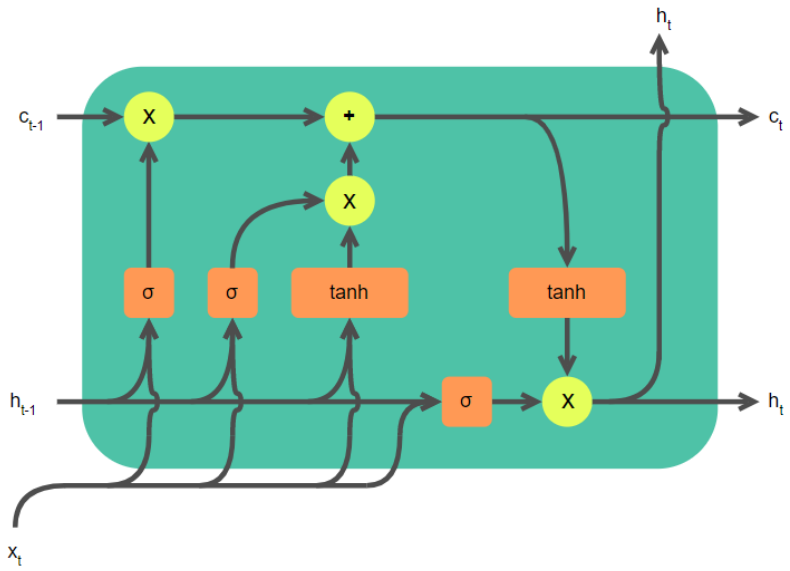


Figure 5.7 LSTM Neuron

LSTM’s (Long Short Term Memory Network) neuron structures are similar to multilayer perceptron’s but with added cell state and forget gate. With its cell state the neurons become context aware. Because of this ability LSTMs are really good at forecasting time series.

5.5.3.1 Hyper Parameters

To train the LSTM network you have to set some hyper parameters. These hyper parameters define the characteristics of the network. There hyper parameters are neuron count, layer count, learning rate, number of epochs.

- Neuron Count: Number of neurons in the hidden layer.
- Layer Count: Number of hidden layers inside the network.
- Sequance Length: Number of values that model search from.

- Learning Rate: Convergence rate of the model to local minimum.
- Number of Epochs: Number of epochs the model has.

5.5.3.2 Training

To train the LSTM network, different hyper parameter combinations have been used and these hyper parameters' effects have been observed. MEPA of 13 has been achieved. Learning rate of 0.0005, neuron count of 64, layer count of 4, sequance length of 8 and number of epochs of 1000 was the best combination in terms of MEPA.

- Neuron Count: Succes rate is increasing but also costly and may cause overfitting.
- Layer Count: Succes rate is increasing with more layers but also costly and may cause overfitting.
- Sequance Length: Succes rate was higher when the sequance length is between 8 and 16.
- Learning Rate: Succes rate is increasing with lower learning rate but costly and may cause overfitting.
- Number of Epochs: Succes rate is increasing but also costly and may cause overfitting.

5.5.4 Testing the Models

In order to be able to test our model and arguments, we seperated the 3 weeks long data set into two parts where the first two weeks were used as the training data and last one as the test data.

Predictions had a MAPE around 0.20.

The residuals, as the name suggests, tell us the difference between the predictions and the actual values. Under ideal circumstances residuals should be nothing more than white noise: Unpredictable noise with a constant standart deviation and a mean of 0.

As seen on the residuals chart, the errors are scattered around the X axis, suggesting that there is not a dominant positive or negative bias with the predictions.

Finally, forecaster also exports the predicted traffic as a JSON file where each key represents an hour and each value for the corresponding key represents the total amount of data served during that hour.

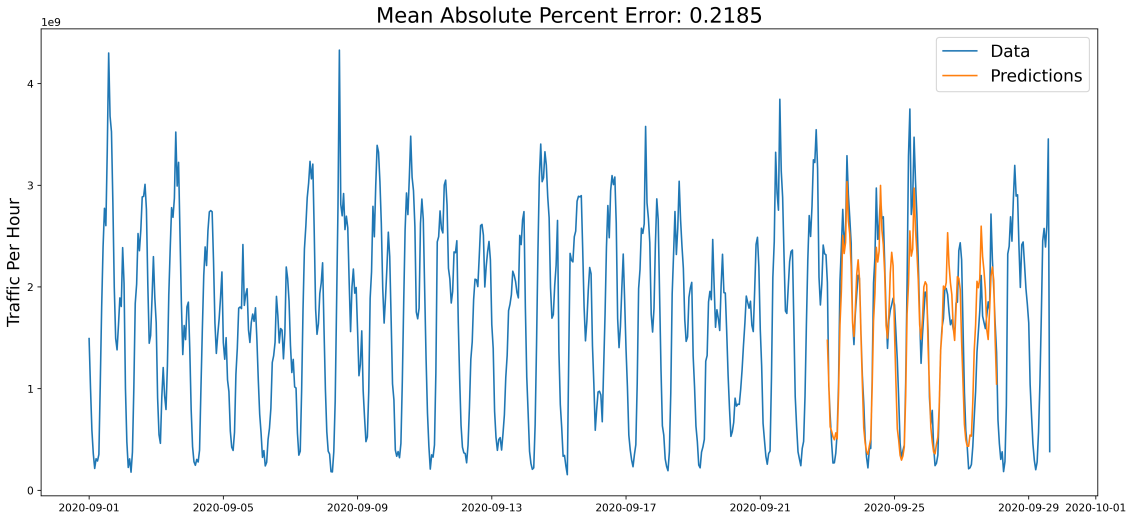


Figure 5.8 Predictions and Actual Values

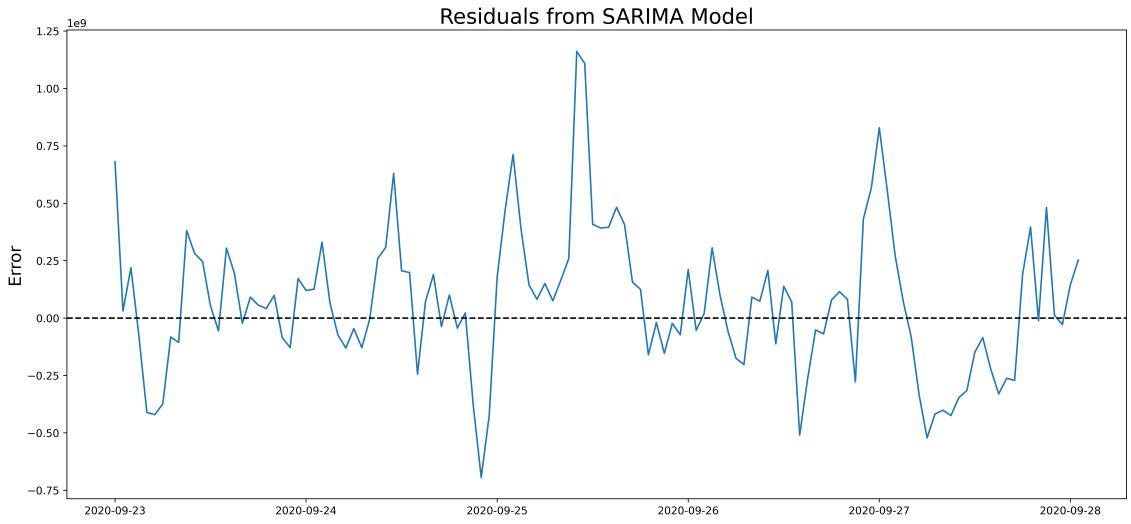


Figure 5.9 Residuals

5.6 Server

In order to run LSTM, MLP and SARIMA models with selected parameters, a server was developed using the Python programming language's Fast API library.

This server exposes three endpoints: '/lstm', '/sarima' and '/mlp'. These endpoints require a set of parameters and a dataset for the models to train with. LSTM, SARIMA and MLP models stored in the service layer are modular and reusable.

After the evaluation process, server returns a response containing the actual traffic, predicted traffic and the MAPE of the predictions.

5.7 Front End Application

In order to train the models with selected parameters and observe the results, a web application was developed using the JavaScript programming language, React.js, Chart.js, Axios and Ant Design libraries.

Earlier on the system analysis chapter the scope of the application was decided. Receiving the user input, validating it, communicating with the server to initiate the training and forecasting processes, receiving a response including the source data itself, predictions, mean absolute percentage error rate and displaying all of that are the responsibilities of this application.

First choice to be made is the kind of model to be used. A radio group with three radio buttons -SARIMA, MLP and LSTM- handles this step. Depending on the selection, the corresponding form containing input fields for the required parameters are displayed. Users are prompted to fill the form with the training parameters, data set and the dates from and until which the data set will be split into training and testing time deltas. Ant design provides good looking and useful components, matching our needs such as a form, form item, text input, date picker and file uploader. State was managed via the state-related hooks provided by React.js.

Submitting the form transfers all the information to the back-end in the shape of a JSON file and triggers the back-end services as they were laid out on the server section. Axios instance, already configured with a base URL leading to the server, receives the JavaScript object containing all the information coming from the form and dispatches the POST request. User is prompted to wait until training ends and informed about any errors. The response to the dispatched request receives a JSON object containing three fields: data, predictions and MAPE; all of which parsed and placed onto the application state.

Once the response is received, forms are replaced with charts. Mean absolute percentage error is displayed alongside actual and predicted amounts of traffic using Chart.js and React-Chartjs. Dates and hours are transformed from UNIX timestapms to JavaScript date objects. The amount of traffic is translated into megabytes.

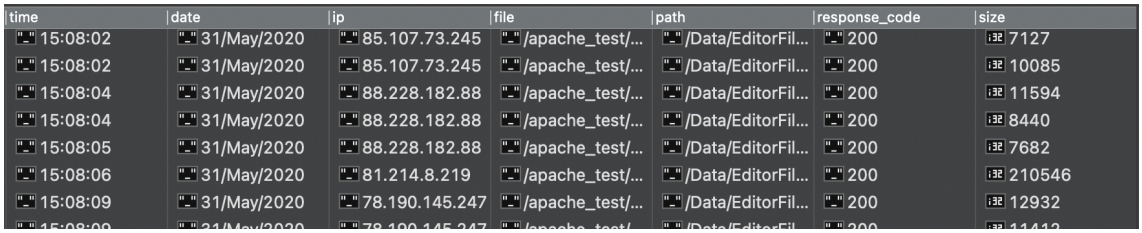
The visual implementation of the front end application can be seen on the application chapter. Theming was implemented through global theming variables of Ant Design, supplemented with Styled Components, HTML and CSS.

6
Application

6.1 Inputs and Outputs

6.1.1 Input

The input required for this application to function as expected is a server log belonging to a period of time where each row represents a request made with columns date, time and size where date and time columns store the date and time of the request and size stores the amount of data served in response to this request. This input is then parsed, preprocessed, used to train the created model and make predictions based on this model.



time	date	ip	file	path	response_code	size
15:08:02	31/May/2020	85.107.73.245	/apache_test/...	/Data/EditorFil...	200	7127
15:08:02	31/May/2020	85.107.73.245	/apache_test/...	/Data/EditorFil...	200	10085
15:08:04	31/May/2020	88.228.182.88	/apache_test/...	/Data/EditorFil...	200	11594
15:08:04	31/May/2020	88.228.182.88	/apache_test/...	/Data/EditorFil...	200	8440
15:08:05	31/May/2020	88.228.182.88	/apache_test/...	/Data/EditorFil...	200	7682
15:08:06	31/May/2020	81.214.8.219	/apache_test/...	/Data/EditorFil...	200	210546
15:08:09	31/May/2020	78.190.145.247	/apache_test/...	/Data/EditorFil...	200	12932

Figure 6.1 A Section of the Database Where Server Logs Are Stored

6.1.2 Output

The output of this application is the forecasted data traffic served within a graphical user interface that allows users to see and interpret the predictions so that they can act upon them in order to avoid facing issues later on in case the traffic exceeds their infrastructure or resources' capabilities or the traffic is much lower than what the allocated amount of resources could handle, which would mean that some of these resources had been wasted.

6.2 User Interface

User interface consists of a single page web application with nested components.

Figure 6.2 Web application

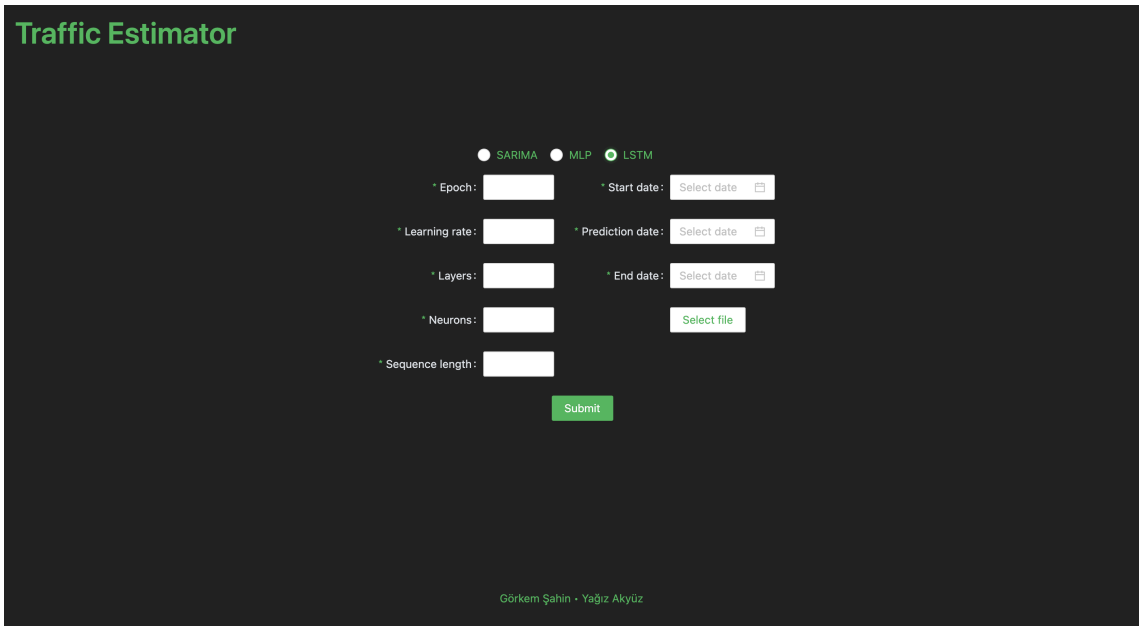


Figure 6.3 Models

User picks a model and enters the parameters.

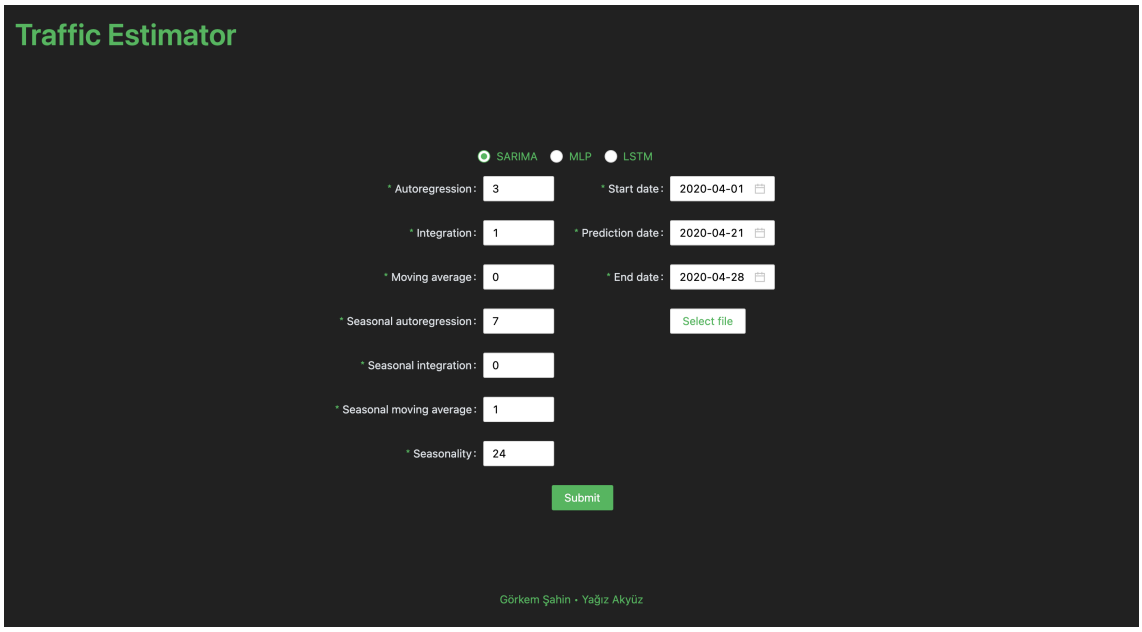


Figure 6.4 Filled parameters

After filling the form and submitting, user is prompted to wait until the model is trained and requested time frame is predicted.

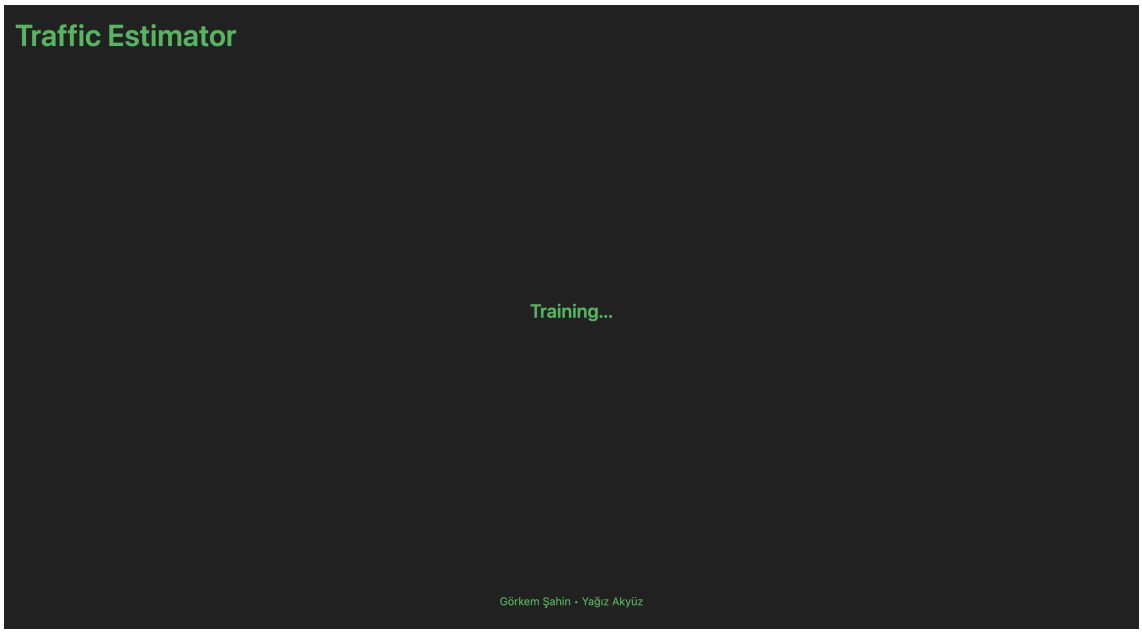


Figure 6.5 Training

User can then view the predictions, mean absolute percentage error and the actual traffic. Detailed information regarding a data point can be inspected by hoverin the mouse over it.

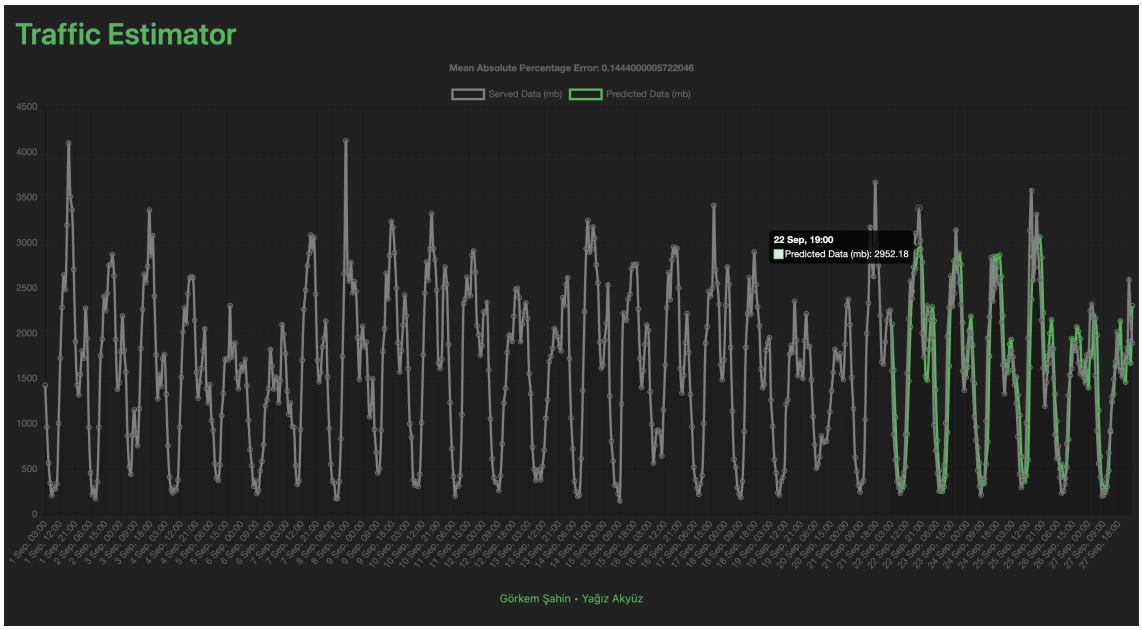


Figure 6.6 Results

7 Conclusion

7.1 Data Set

By looking at the web traffic during different months and on different servers, it had been our experience that it is crucial for the data set to have overall consistency. Months where there were no events and holidays that could have an impact on the given web site's traffic yielded more accuracy.

Working with the total amount of data served per hour resulted in best results since time periods measured in seconds or minutes contained random spikes, reducing overall consistency of trends and causing unexplainable high levels of correlation between unrelated data points with spikes.

7.2 Parameter Selection

We concluded that even though there was a correlation and partial correlation between data points to be predicted and the non-seasonal time points following up to these data points, it was better to exclude those by supplying SARIMA with 0 for p and q values and rely on their seasonal counterparts only. The seasonal correlation and partial correlations were much higher, and including other data points beside these only worsened our predictions.

Despite showing significant amounts of correlation, using non-seasonal components of SARIMA reduced the success rate because seasonal components had much higher levels of correlation. Relying on seasonality alone brought better results.

7.3 Model Selection

LSTM was the most successful model among the models we tried thanks to being able to learn the context required to make predictions in time series forecasting problems, rather than having this context pre-specified and fixed.

References

[1] Ö. Ö. B. E. Al., "Artificial neural network and sarima based models for power load forecasting in turkish electricity market," *PLOS ONE*, 2017.

[2] E. Z. M. E. A. S. da Silva; Amaury Lelis Dal Fabbro, "A sarima forecasting model to predict the number of cases of dengue in campinas, state of são paulo, brazil," pp. 436–440, 2011.

[3] Y. J. L. Chen M., "An adaption scheduling based on dynamic weighted random forests for load demand forecasting," pp. 1735–1753, 2020.

[4] X. W. Zheng Zhao Weihai Chen, "Lstm network: A deep learning approach for short-term traffic forecast," pp. 68–75, 2017.

[5] T. Koskela, "Time series prediction with multilayer perceptron, fir and elman neural networks," pp. 25–33, 1996.

Curriculum Vitae

FIRST MEMBER

Name-Surname: Yağız Akyüz
Birthdate and Place of Birth: 24.01.1996, İzmir
E-mail: yagizakyuz@gmail.com
Phone: +90 506 922 84 19
Practical Training: IBM

SECOND MEMBER

Name-Surname: Görkem Şahin
Birthdate and Place of Birth: 10.01.1997, Muğla
E-mail: gorkemsahin.1997@gmail.com
Phone: +90 554 503 14 10
Practical Training: IBM

Project System Informations

System and Software: Windows İşletim Sistemi, Python, Javascript
Required RAM: 2GB
Required Disk: 60GB