Aristotle University of Thessaloniki

Master Thesis

Real-time Traffic Prediction using Multiple Data Sources

a Neural Network Approach

Author:
Kiriakos Adam

Supervisor:
Athena Vakali

A thesis submitted in fulfillment of the requirements
for the degree of Masters of Science

in the

OSWinds Lab
Department of Informatics
Faculty of Science
Aristotle University of Thessaloniki

February 28, 2018

ARISTOTLE UNIVERSITY OF THESSALONIKI

Abstract

Faculty of Science
Department of Informatics

Master thesis

Real-time Traffic Prediction using Multiple Data Sources
a Neural Network Approach

by Kiriakos Adam

Intelligent Transport Systems (ITS) is a field, developed rapidly over the last two decades, driven by the growing need for better transport network management strategies and by the continuing improvements in computing power. This thesis provides a literature review of traffic prediction methodologies, together with a novel method for time series aggregation, and a neural network model for traffic prediction. In particular, traffic prediction is studied as a supervised learning task, using regression and multi class classification methods. Working with multiple data sources, Data Visualization techniques, as well as, Descriptive Statistical methods have been introduced for a deeper understanding of the spatio-temporal properties of the data and better extraction of important and meaningful features. Experiments are conducted upon real Floating Car time series Data (FCD) and weather condition information in the city of Thesssaloniki. Our approach has been implemented as an online prediction framework using cutting edge -industry ready- technologies.

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Περίληψη

Faculty of Science
Τμήμα πληροφορικής

Μεταπτυχιακή Διατριβή

Τεχνικές πρόβλεψης κυκλοφοριακής κίνησης σε πραγματικό χρόνο χρησιμοποιώντας πολλές πηγές δεδομένων

Αδάμ Κυριάκος

Πλήθος έργων και εφαρμογών έχουν υλοποιηθεί στα πλαίσια των έξυπνων πόλεων με ποικίλους αισθητήρες που παρέχουν πληροφορίες για τη δημιουργία εφαρμογών χρήσιμων σε πολίτες. Στα πλαίσια της παρούσας εργασίας αρχικά πραγματοποιείται μία βιβλιογραφική επισκόπηση πάνω στο υπάρχων ερευνητικό έργο με σκοπό την επίλυση του κυκλοφοριακού προβλήματος. Στη συνέχεια γίνεται η παρουσίαση της συνεισφοράς μας μέσω ενός πλαισίου λογισμικού το οποίο οπτικοποιεί, αναλύει και δημιουργεί προβλέψεις για την κίνηση στους δρόμους της Θεσσαλονίκης. Αυτό, χρησιμοποιώντας ως πηγή πληροφοριών χωροχρονικές παρατηρήσεις αυτοκινήτων στην πόλη, συνδυασμένες με καιρικές συνθήκες. Για την επίτευξη αυτού τα σύνολα δεδομένων προ-επεξεργάζονται και μετασχηματίζονται με απώτερο σκοπό την ανάλυση τους από νευρωνικά δίκτυα καθώς επίσης και από αμοιγώς στατιστικά μοντέλα. Η παραπάνω διαδικασία αντιμετωπίζει το πρόβλημα με μεθόδους επιβλεπομένης μάθησης τόσο ως πρόβλημα ταξινόμησης όσο και ως πρόβλημα παρεμβολής.

Acknowledgements

First and foremost I wish to thank my supervisor, professor Athena Vakali, director of the OS Winds Laboratory. She has been supportive since the days I was an undergraduate. She helped me come up with the thesis topic and guided me over almost half a year of development.

I would also like to thank Josep Maria Salanova Grau and Panagiotis Tzenos for contributing the dataset used for the purposes of this thesis, and the PhD candidate Ilias Dimitriadis for his contribution. Without their passionate participation and input, the validation survey could not have been successfully conducted.

I would also like to express my gratitude towards the committee, assistant professor Apostolos Papadopoulos and assistant professor Grigorios Tsoumakas for their valuable input and constructive discussion in the examination of the thesis.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Author
Kiriakos Adam

Contents

List of Figures

List of Tables

List of listings and algorithms

List of Abbreviations

FCD	**F**loating **C**ar **D**ataset
IOT	**I**nternet **O**f **T**hings
VTL	**V**irtual **T**rip **L**ines
GPS	**G**lobal **P**ositioning **S**ystem
GIS	**G**eographic **I**nformation **S**ystems
OSRM	**O**pen **S**ource **R**outing **M**achine
OSM	**O**pen **S**treet **M**ap
ARIMA	**A**utoregressive **I**ntegrated **M**oving **A**verage
RNN	**R**ecurrent **N**eural **N**etworks
LSTM	**L**ong **S**hort **T**erm **M**emory **N**etworks
FNN	**F**eedforward **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
DBN	**D**eep **B**elief **N**etworks
MLANN	**M**ultilayer **A**rtificial **N**eural **N**etwork
HMM	**H**idden **M**arkov **C**hain
TDNN	**T**ime **D**elay **N**eural **N**etwork
SVR	**S**upport **V**ector **R**egression
GARCH	**G**eneralized **A**uto **R**egressive **C**onditional **H**eteroskedasticity
RESTfull	**R**epresentational **S**tate **T**ransfer
JSON	**J**avaScript **O**bject **N**otation
API	**A**pplication **P**rogramming **I**terface
ITS	**I**ntelligent **T**ransportation **S**ystem
OWM	**O**pen **W**eather **M**ap
WGS	**W**orld **G**eodetic **S**ystem
SMOTE	**S**ynthetic **M**inority **O**ver-sampling **T**echnique
ACC	**A**ccuracy
RMSE	**R**oot **M**ean **S**quare **E**rror
MAE	**M**ean **A**bsolute **E**rror
MSE	**M**ean **S**quare **E**rror
GCRS	**G**oedetic **C**oordinate **R**eference **S**ystem
PCRS	**P**rojected **C**oordinate **R**eference **S**ystem
WGS	**W**orld **G**eodetic **S**ystem

Chapter 1

Introduction

In this introduction chapter we approach the main topic of this thesis, that is the issue of traffic congestion and the way in which it can be predicted by using multiple data sources. First, we will discuss the main problem and subsequently we will refer to the way other cities have organized their traffic flow trying to eliminate this modern problem of urban areas. Finally, we will discuss our contribution in order to make accurate predictions trying to encapsulate the congestion and avoid congested roads.

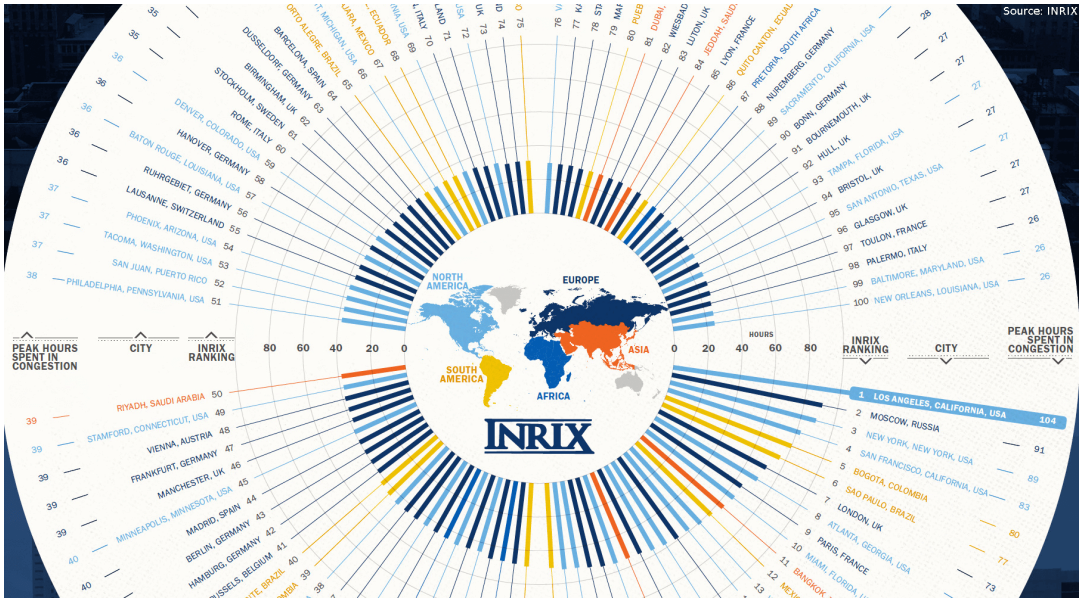

Figure 1.1: Traffic in big cities

1.1 Problem definition

Traffic congestion is a condition of transport networks that occurs as use increases, and is characterized by slower speeds, longer trip times, and increased vehicular queueing. When traffic demand is great enough that the interaction between vehicles slows the speed of the traffic stream, this results in some congestion. As congestion increases, the hours spent by citizens stuck into their cars increase to. Taking the top spot was Los Angeles, where drivers spent an average 104 hours stuck in the city's legendary traffic jams. The cost of this congestion, measured in wasted time and fuel, was $9.7 billion – a number that sums up to $2,408 per driver. (INRIX, Global Traffic Scorecard). On the other hand, USA

lost $ 133 billion in 2015 in hidden costs and the cost is predicted to increase $175 billion in 2020 (Carrealtime, Digital age transportation) by moving foods to suppliers, manufacturers and markets, a cost passed along to consumers in the form of higher prices. Furthermore, 56 billion pounds of additional carbon dioxide were released in the air due to congestion in 2011.

A major problem in big cities is the traffic congestion caused by commuters traveling around the cities. One of the main reasons of **traffic congestion** is that the majority of people prefer using their own cars instead of public transportation. Another factor is that most people live in the suburbs outside the city center and they tend to travel at the same time of the day. Moreover, cars and road space are not used optimally. As a result, this causes traffic jams during the rush hour. As it is presented in figure1.1 the traffic in urban areas exists in large cities and creates a **sequence of problems**, such as air and noise pollution (Bengio, Simard, and Frasconi, 1994) (Abbaspour et al., 2015). Another problem that can appear is stress which is a consequence of these two. Stress in turn impairs the mental health of urban residents. Research shows that urban residents have worse mental health than rural residents. In particular, they have much higher levels of mood and anxiety disorders (Hennessy and Wiesenthal, 1999).

Transportation problems have been studied by many researchers and many solutions have been proposed to solve them. Improving transportation includes many ways to improve traffic performance in major cities and motorways. To help reduce traffic congestion, cities long ago developed various means of public transportation such as buses, subways, and light rail. Some cities have better public transportation than others.

Smart cities, adopt a spatially distributed network of autonomous intelligent sensors to measure a variety of physical or environmental parameters, such as pressure, noise, temperature, sound, ambient light levels, gas concentration, and traffic status for efficient urban management. This monitoring capability can be effectively used in the emerging initiatives of smart city. Every second, massive amounts of heterogeneous data are being produced by these sensors that pose the challenge to develop platforms to share unambiguous meaning of collected data. This can be achieved by the concept of Semantic Sensor Network (SSN). SSN enables sharing and reusability of sensor data from various sensors in a meaningful way and endures system to process sensor data in heterogeneous environment. In Stockholm, for example, they use electronic pricing which means that the citizens have to pay a fee if they want to enter the central city. In Barcelona traffic control cameras are connected by fibre optics to the transport authority to monitor traffic in real time, providing the control centre means to increase or reduce the frequency of green lights according to the traffic conditions. On the other hand, transport for London's online journey planner provides instant advice on routes in the UK capital, with users able to opt for multiple modes of transport, including walking, tube train, bus, overground train, river transport and bicycle (WEF, Cities are tackling traffic). The key to the success of the integrated journey planner is the willingness of operators to share information and to provide it to the general public.

1.2 Our contribution

Historically, traffic monitoring systems have been mostly limited to highways and have relied on data feeds from a dedicated sensing infrastructure, which often includes loop detectors, radars, video cameras. The reason why fixed-location sensors were preferred when transportation engineers first sought to measure traffic conditions was evident, as there was no easy way of tracking any significant subset of vehicles. By placing fixed-location sensors, traffic engineers ensured that they could record data from nearly every vehicle passing by a given location. Apparently, the obvious disadvantage of placing sensors in this manner is that data is only recorded within a short distance of the sensor. Thus, to cover a significant portion of the road network, it is necessary to use other sources of information as well.

An essential step in mitigating the effects of traffic congestion is the creation of real-time traffic monitoring systems. This dissertation proposes a global approach to designing traffic information systems with a specific focus on estimating arterial traffic conditions from sparse GPS probe data. Arterials (also known as the secondary network) are major city streets (not highways) that aid that aid travelling both within and between the cities. Probe data refers to location and speed measurements provided by a subset of vehicles traveling through the road network. This work leverages the increasing availability of mobile devices (i.e. cell phones) with sensors, such as GPS, that are capable of providing detailed information about the traffic conditions experienced by the driver carrying the device. This thesis suggests a solution for urban areas with traffic problems, by proposing the use of GPS services which are provided by mobile devices, a fact that can be easily achieved, since mobile devices are commonly used by the majority of people nowdays. The proposition, in order to solve the posed problem of traffic congestion, is to make predictions of traffic flows. By using this information the car drivers will know a priori which road is congested or not.

This thesis focuses on multiple data sources and the way they can be used so as to make accurate predictions on traffic congestion in Thessaloniki. Our contribution is to create a framework that can predict the congestion on the roads of Thessaloniki using historical data and external sources. The goal is to ultimately provide the most accurate, timely estimations and forecasts of traffic conditions, allowing thus, the citizens, to consult this framework about their daily transportations so as to avoid congested roads and areas. The above framework could be used by citizens and companies in order to:

- Travel in less time, since citizens can consult the framework and choose no congested routes
- Reduce stress in daily base, since as mentioned human health can be affected by traffic
- Reduce noise into urban areas
- Reduce the pollution from stucked cars, since the reduction of stopped cars affects noise and air pollution
- Reduce gas consumption, since as mentioned traffic congestion affects the gas consumption greatly

1.3 Thesis structure

This master thesis is organized as follows:

Chapter 1 presents an introduction about the problem that exists in large cities about traffic congestions and our contribution as solution.

Chapter 2 provides the theoretical background and a review of the literature on short term traffic prediction and a discussion of the state-of-the-art techniques in short term prediction in transport using statistical methods and neural networks.

Chapter 3 presents the methodology used for data cleaning ,map matching, our approach for the time series aggregation, as well as, some stationarity and seasonality tests, and the proposed traffic prediction frameworks.

Chapter 4 deals with the architecture and the implementation of the framework, as well as, the data model that was used in order to create predictions.

Chapter 5 deals with the data sources and how we have pre-processed them in order to feed the prediction methods using the most appropriate aggregations to increase the discretion power of the classifiers, as well a section of descriptive statistics for better understanding the whole procedure.

Chapter 6 summarises the findings of this research and suggests avenues for further research.

Chapter 2

Background

This chapter outlines the basic background about the technologies used to analyse traffic and examines the existing research in the subject of traffic prediction in urban areas. It presents a variety of techniques and datasets with case studies in cities and countries using several sensors and IOT architectures with the main purpose to examine traffic congestion.

2.1 Time Series Fundamentals

A time series is a set of observations or data values in periods, where periods have the same length. The study of time series evolves around methods for analysing of datasets, extracting valuable statistics and other characteristics using sequences of past observations. In more detail, there are two targets, on the one hand the observation of events per time and on the other the prediction of the values to come based in previous values.

As a sample, given the time serie $X_1, X_2, ..., X_T$, T indicates the time or data points with same length of time observation. $x_1, x_2, ...x_T$ are specific values for $X_1, X_2, ..., X_T$ under a uniform sampling that are random. We call the observations of $X_1, X_2, ..., X_T$ belonging to a infinite sequence of random variables a *stochastic process* and can be represented as $\{X_T\}$. According to statistics the stochastic procedure refers to the population. Each stochastic procedure can be described using a probability function $f(x_1, x_2, ..., x_T)$ which is indefinite. If the function was known, the procedure of predicting the probability of an instance in time from the current time would be easy. Due to indefinite nature, the analysis of time series is done by formulating models for the description of the stochastic procedure resulting from the series.

The current value of a random variable X, is expressed as a function of previous values, ie those with a time lag on the sum sample of the series. Conversely, in a regression model the random variable consists of a function given i general interpretive variables. A stochastic procedure is called stationary, if the variance and the median do not change in time. The following applies:

- $E(X_1) = E(X_2) = E(X_T) = \mu$, where $E(X_i)$ indicates the expected value of the random variable X_i and $i = 1, 2, ..., T$
- $\sigma^2 = var(X_T) = E(X_T - \mu)^2, \sigma^2$ or $var(X_T)$ is the variance that is the expected value of the square deviation from the mean value of the random variable, μ is the average numerical value of random variables.

- $X_i, i = 1, 2, ..., T$ and $E(X_T - \mu)$ is the expected deviation of the variable from the median.

- $\gamma_\kappa = E[(X_T - \mu)(X_{T+\kappa} - \mu)]$, where $\gamma_\kappa$is the covariance with κ time lags, $X_{T+\kappa}$ the random variable $X_i, i = 1, 2, ..., T$ with time lag κ and $E[(X_T - \mu)(X_{T+\kappa} - \mu)]$ the expected deviation of X_i from the median and the lagged X_i from the median.

2.2 Forecasting Time Series methods

There are several methods to predict time series. In this section two types of prediction methods are discussed and the way these methods can be used in order to make predictions is explored. The first is based on artificial intelligence and more specifically on neural networks, and the second on pure statistics with an Autoregressive Integrated Moving Average model.

2.2.1 Neural networks and Deep learning

In the field of neural networks and time series analysis RNN and LSTM are the most commonly used techniques for prediction of travel time and congestion (G. Polson and Sokolov, 2017) (Epelbaum et al., 2017) (Yi, Jung, and Bae, 2017). In this section we will discuss the basic functionality of these networks and subsequently we will delve into an analysis of specific forecasting methods that have already been used. One of the appeals of RNNs is the idea that they are able to connect previous information to the present task (Schmidhuber, 1990), such as using previous traffic congestion condition to inform the understanding of the present congestion. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on previous computations. Another way to think about RNNs is that they have a “memory” which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back for only a few steps.

$X_1, X_2, ..., X_T$ being the time series, x_t is the input at time step t, x_1 could be a one-hot vector corresponding to the second congestion of a sentence. The hidden state of time step t is s_t and It’s the "memory" of the network. s_t is calculated based on the previous hidden state and the input at the current step:

$$s_t = f(Ux_t + Ws_{t-1}) \tag{2.1}$$

With $f$ussualy been a nonlinearity such as $tanh$ or $ReLU$.U being the matrix connecting the input layer and W being the one connecting the hidden. s_{-1}, which is required to calculate the first hidden state, is typically initialized to all zeros. o_t is the output at step t. For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary. $o_t = \text{softmax}(Vs_t)$. V being connecting the output layer with the previous one.

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by (Hochreiter and Schmidhuber, 1997a). They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem (Bengio, Simard, and Frasconi, 1994) (Hochreiter and Schmidhuber, 1997a). Remembering information for long periods of time is practically their default behavior. All recurrent neural networks have the form of a chain of repeating modules of neural networks. In standard RNNs, this repeating module will have a very simple structure, such as a single *tanh* layer. The first step in our LSTM is to decide what information we are going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer". The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates, where cell state is kind of like a conveyor belt. It connects the entire chain, with only some minor linear interactions. It looks at h_{t-1} and , x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1} . A 1 represents "completely keep this" while a 0 represents "completely get rid of this". Taking the example of traffic model trying to predict the congestion based on all the previous ones. In such a problem, the cell state might include the time congestion of the present congestion, so that the correct time segment can be used. When we see for example a new day, we want to forget the previous congestion situation and create a prediction based on the current day. The next step is to decide what new information we are going to store in the cell state. This has two parts. First, a *sigmoid* layer called the “input gate layer” decides which values will be updated. Next, a *tanh* layer creates a vector of new candidate values, C_t, that could be added to the state. In the next step, we will combine these two to create an update to the state.

In the example of our traffic model, we want to add the time segments for prediction per weak on specific days.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.2}$$

$$C_t = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{2.3}$$

It is now time to update the old cell state, C_{t-1}, into the new cell state C_t. The previous steps already decided what to do, we just need to actually do it.

We multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add $i_t * C_t$. This is the new candidate values, scaled by how much we would update each state value. In the case of the traffic model, this is where we would actually drop the information about the old time segments and add the new information, as we decided in the previous steps.

$$C_t = f_t * C_{t-1} + i_t * C_t \tag{2.4}$$

Finally, we need to decide what we are going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we are going to output. Then, we put the cell state through *tanh* (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

2.2.2 Statistical inference

In an Autoregressive integrated moving average (ARIMA) model, we express x_t as a function of past value of x and/or past errors (as well as a present time

error). When we forecast a value past the end of the series, on the right side of the equation we might need values from the observed series or we might, in theory, need values that have not been observed yet.(Box and Jenkins, 1970).

Consider the AR(2) model $x_t = \delta + \phi_1 x_{t-1} + \phi_2 x_{t-2} + W_t$. In this model,$x_t$ is a linear function of the values of x at the previous two times. Suppose that we have observed n data values and wish to use the observed data and estimated AR(2) model to forecast the value of $x_n + 1$ and $x_n + 2$, the values of the series at the next two times past the end of the series. The equations for these two values are

$$x_{n+1} = \delta + \phi_1 x_n + \phi_2 x_{n-1} + W_{n+1} \tag{2.5}$$

$$x_{n+2} = \delta + \phi_1 x_{n+1} + \phi_2 x_n + W_{n+2} \tag{2.6}$$

To use the first of these equations, we simply use the observed values of x_n and x_{n-1} and replace W_{n+1} by its expected value of 0 (the assumed mean for the errors). The second equation for forecasting the value at time $n + 2$ presents a problem. It requires the unobserved value of x_{n+1} (one time past the end of the series). The solution is to use the forecasted value of (the result of the first equation). In general, the forecasting procedure, assuming a sample size of n, is as follows:

- For any W_j with $1 \leq j \leq n$, use the sample residual for time point j
- For any $W_j$with $j > n$, use 0 as the value of W_j
- For any x_j with $1 \leq j \leq n$, use the observed value of x_j
- For any x_j with $j > n$ use the forecasted value of x_j

For convenience the ARIMA model can represented as ARIMA(p, d, q) where the parameter p is the number of Auto-Regressive (AR) terms, for example, if p is 5, the predictors for $x(t)$ will be $x(t-1), ..., x(t-5)$.The parameter q is the number of Moving Average (MA) terms. MA terms are lagged prediction errors in the prediction equation, for example, if q is 5, the predictors for $x(t)$ will have such lagged prediction errors $e(t-1), ..., e(t-5)$, where $e(i)$ is the difference between the moving average and the actual value at the i th instant. The parameter d is the number of differences when the time series becomes stable.

Variable	Description
x_1	One-hot vector (2.2)
h_{t-1}	Previous vector (2.2)
x_t	Time series at time step t (2.1)
c_t	Cell state (2.3)
s_t	Hidden step of time step t (2.1)
f	Activation function (2.1)
U	Matrix, connecting the input layer (2.1)
W	Matrix,connecting the hidden layer (2.1)
O_t	Output at step t (2.4)
V	Connecting the output layer with the previous values (2.4)
x_t	Function of past value (2.5)
x	Past errors (2.5)
W	Expected value (2.5)
ϕ	Slope coefficient (2.5)
δ	Constant (2.5)

Table 2.1: Chapter 2 notations

Table 2.1 is a summary table of variables of section for better understanding of equations. For each variable there is a description.

2.3 Traffic and data sources

in the past 50 years a number of sensors designed to collect various types of traffic data have been designed. In general, traffic data includes flows, density, occupancy, velocity and travel time. One additional data type possible are vehicle trajectories, which are always represented by a sequence of discrete time/location pars for each vehicle. There are several technologies used in order to capture information about the traffic status some of which are presented in figure 2.1.

Figure 2.1: Traffic and data sources

Loop Detectors
Inductive loop detectors are built into the roadway so that they can detect each vehicle that passes over them. They work by detecting the metal of a vehicle as it passes over the detector. Properly calibrated, a loop detector is capable of providing high-accuracy flow and occupancy data, the latter of which can be used to infer density. When two loop detectors are placed close together, velocity can be measured by looking at consecutive crossing times. Loop detectors are not capable of directly measuring travel times.

Radar
Radar detectors are devices placed on poles along the road enabling them to collect flow, occupancy and velocity data. In general loop detectors provide higher accuracy data than radar (Jo and Jung, 2014). Radars are generally not well suited to mass data collection on arterials and due to the fact that accuracy decreases in arterial environments (Jo and Jung, 2014). For this reason and the fact that almost no radar data exists on arterials, they are not considered viable inputs into an arterial traffic information system.

Video
Video recording can be used in two ways. The first one is to use high resolution cameras to track all vehicles within the view of camera and the second one is to place the camera and record vehicle plate numbers in order to analyse the traffic by car. Using high ratio cameras can not track real time traffic data due to the large amount of post-processing work that needs to be done, besides, this technology is expensive to deploy and can only cover little segments of road.

Bluetooth
Bluetooth technology has been developed in order to transmit information within a small range. Bluetooth readers are capable of scanning the surrounding airways for Bluetooth enabled devices. If readers are place along side a roadway, travel time can then be measured for all vehicles carrying a Bluetooth device. This devices suffer from the problem of having a relatively high detection range. This is good in sense rarely miss a vehicle, but it is difficult to determine the precise time that a vehicle passed the reader as it might be detected for more than one minute.

Wireless Sensors
Wireless sensors are small devices embedded into the roadway (similar to loop detectors), but the detection mechanism in these sensors allows for re-identifying vehicles at subsequent sensor locations. Thus, these sensors provide travel times for a large percentage of the flow of traffic. The primary advantage of these sensors over other travel time measurement sensors is that they are much cheaper to produce, potentially allowing for large-scale deployment on arterial roads.

Virtual Trip Lines (VTL)
Virtual trip lines (VTL) comprise the basis of a “participatory sensing” system that allows individuals to download an application in their GPS-enabled smartphone that both sends traffic data as well as receives traffic information and alerts. A VTL is a virtual line drawn on the road. The basic idea is that the phone monitors its own GPS position every few seconds and has downloaded a list of VTLs in the general region that the phone resides in. When the phone crosses one of the VTLs, it sends an update to the central VTL server indicating its velocity and time of crossing as well as the travel time from the previous VTL it crossed.

The accuracy of the velocity data generated by frequent GPS sampling varies greatly with the type of GPS chip in the phone and can be very good in some cases and very bad in others. It is generally accurate enough for highway traffic estimation when properly filtered (Jo and Jung, 2014). For arterials, the velocity measurements are not reliable, so the travel time measurements are the only data that is suitable for arterial traffic estimation.

GPS Signal

The basic idea behind the GPS system is to determine precise location using triangulation. Based on the intersection of a group of satellites' signals, triangulation or satellite ranging is used to calculate a location on earth by measuring the distance from each of several satellites in space. The satellites act as reference points in space. Knowing the distances from the satellites to a point on the Earth's surface allows a position to be accurately determined.

Four satellite measurements are needed to determine exact position in three-dimensional space. In order for triangulation to work, a receiver measures the amount of time a radio wave takes to travel from a satellite to the receiver. Both the satellite and receiver generate a set of digital codes called pseudo-random codes at exactly the same time. The pseudo-random code repeats itself every millisecond and is carried on radio waves. Each satellite transmits two carrier signals. Thus, the difference between the satellite's code and receiver's pseudo random code will give the distance between the two.

According to (Prof. Dr. Bernhard Hofmann-Wellenhof, 1993) there are sensitive on errors from various components. Among errors have developed several ways to correct the accuracy of the transmitted signals.

Causes of GPS Signal errors	
Satellite Orbit (Position)	Multipath Errors (bounce off buildings, etc.)
Earth's ionosphere (charged particles)	Local Weather (moisture in air, lightning)
Earth's troposphere (moisture)	Poor Satellite Geometry (GDOP)
Receiver Noise (local conditions, radio interference)	Receiver Clock Errors
Ways to improve the data of GPS collected signals	
Establish protocols for your applications	Employ averaging techniques
Perform mission planning	Utilize DGPS
Understand how the selection of datums and coordinate systems affect accuracy	

Table 2.2: Causes of GPS Signal errors and ways to improve the data of collected signals

2.4 Related work

As discussed in section 2.2 there are two basic procedures in **traffic congestion prediction** using **time series** datasets. On the one hand there is pure statistics and on the other hand is **artificial intelligence** and neural networks. Some of the most widely used traffic prediction models are those based on spectral analysis, **Autoregressive Integrated Moving Average** (ARIMA) time series models and Kalman filtering (Lu, Wu, and Vemuri, 1993) This method was initially proposed by (Kalman, 1960).

One of the very first studies on the topic of traffic prediction was conducted by (Iokibe, Mochizuki, and Kimura, 1993). The authors have implemented a fuzzy logic traffic prediction model using counter sensors on tunnels by traffic variation

patterns. The traffic based on the traffic variation pattern is retained as a reference traffic, the actual traffic at preceding control period is compared with the traffic of pattern at every prediction and the traffic of pattern at the current control is corrected to obtain a predicted traffic congestion calculated from pattern traffic for one preceding hour and measured traffic.

Combining technologies by using sensors and mathematical models a number of traffic prediction algorithms have been developed or proposed over the last two decades. Their structure varies in the degree of sophistication, complexity and data requirements.

Research on traffic prediction is a common use case around a time series problem. Autoregressive Integrated Moving Average (ARIMA) and Neural Networks are algorithms the appear to perform best in this area. For example (Zeng et al., 2008) explores the variations of the linear model ARIMA and non-linear Neural Network using a hybrid approach of ARIMA and Multilayer Artificial Neural Network (MLANN) and in 2010 claims support vector regression model (SVR) has been widely used to solve non-linear time series problems (Hong et al., 2010). The models are modified to cater for the randomness of the so called unknown factors that affect traffic. This is also known as ARIMA-GARCH. GARCH is algorithms and models that account for the errors.

As we can see there are several approaches in the prediction models and algorithms that have been used but there are also differences on the type of the datasets. There are several studies trying to predict the congestion of a city using loop detectors (G. Polson and Sokolov, 2017). On this approach G Polson et al, created a deep learning model using data from loop detectors with Feed forward Neural Network, Convolutional Neural Network, Recurrent Neural-Network and Long Short Term Memory layers. (Epelbaum et al., 2017) have used the same deep learning structure to predict the traffic speed of the external Paris Ring road using FCD datasets with graph based database. On the other hand (Yi, Jung, and Bae, 2017) using 3 layers of neural network and FCD dataset have transform the prediction problem to a binary classification problem using congested and non congested classes.

(G. Polson and Sokolov, 2017) approaches the prediction problem using several data sources, such as special event by the Chicago Bears football game days, traffic incidents on snowstorm forecasts and loop detectors using off-on switches. The specific work used deep learning models in order to predict the traffic speed for 40 minutes time intervals.(Song, Kanasugi, and Shibasaki, 2016) used several neural networks, such as RNN, LSTM, DLSTM, TDNN, and HMM using the GPS signals of users without the speed of their vehicles.(Cheng et al., 2017) used a binary vector from hundreds of locations in Beijing, the problem transformed into a multi class problem using as labels (fluency, slow, congestion, extreme congestion). Their approach used 15,30,45,60 minutes time window in order to predict the correct class based on previous observations.

(Jia, Wu, and Xu, 2017) introduces a deep belief network (DBN) and long short-term memory (LSTM) to predict urban traffic flow considering the impact of rainfall, this work was based on 10 and 30 minutes prediction.(Keay and Simmonds, 2005) used statistical methods trying to associate the rainfall impact on traffic prediction and they have found a strong association between wet days and

high values of traffic. (Dunne and Ghosh, 2013) used stationary wavelet transform instead of ANN trying to correlate the weather conditions and the traffic into urban arias. (Pan et al., 2013) address the problem of detecting and describing traffic anomalies using crowd sensing with two forms of data, human mobility and social media, on this paper researchers used heterogeneous data sources such as social media, and GPS trajectories.

(Song, Kanasugi, and Shibasaki, 2016) used heterogeneous data source such as human mobility data and transportation network data trying to understand how humans move and select the transportation mode throughout a large-scale transportation network is vital for urban congestion prediction and transportation scheduling flow prediction using recurrent neural network (RNN) and LSTM.

		Data										Results			
Reference	**Year**	**Data Source**	**Amount of data**	**Type of Data**	**Methods**	**Smoothing**	**Rsults**	**evaluation**	**External sources**	**Topological neighbors**	**Topological estimation**	**Prediction**	**Prediction values**	**Technology**	**Visualizing**
Deep Neural Networks for Traffic Flow Prediction	2017	Location of users inside inside specific links	0.5 million of points, 1 month data	GPS Signals without speed	AdaGrand, RNN, DNN and CNN		4 days memory (99% accuracy)		No	No	Road	4 days memory (99% accuracy)	Binary, congestion/ No congetion	Tensrflow, Keras	No
Deep Learning applied to Road Traffic Speed forecasting	2017	User information per 3 minutes on external ring road of Paris	10 month data with average on missing values using detectors	Gps signals with speed	Deep Learning, Feedforward Neural Network, Convolutional Neural Network, Recurrent Neural-Network, Long Short Term Memory Adam optimizer	Yes, on speed		RMSE and Q-score	No	No	Road	10-60 minutes	Speed prediction	-	No
Deep learning for short-term traffic flow prediction	2017	21 sensor data estimating the speed by length highway on highway I-55 in Chicago	-	Data from sensors, off-on switches	Deep Learning		40 minutes	Monte Carlo simulations	Social Media by team,Traffic incidents,Specific event detection, Chicago Bears football team, weather status by day	Yes (LARS) method	1/21 road	40 minutes	Speed prediction	H2O	No
Traffic Flow Prediction with Rainfall Impact Using a Deep Learning Method	2017	2 minute intervals from segment from the 2nd and 3rd Ring Road in Beijing	2 months observations, August, 2013, and June to August, 2014	Data from sensors	LSTM,DBN		10 minutes and 30 minutes	MAE MAPE RMSE	Weather Hourly rainfall by (NMC)			10 minutes and 30 minutes	Speed prediction	-	No
DeepTransport: Learning Spatial-Temporal Dependency for Traffic Condition Forecasting	2017	Congestion measures (fluency, slow, congestion, extreme congestion)	binary vector from 349 locations on Beijing	Linkert annotated	CNN, RNN, ARIMA				No	No	Nodes	15, 30, 45, 60 minutes	Compared linkert with arima	-	No
DeepTransport: Prediction and Simulation of Human Mobility and Transportation Mode at a Citywide Level	2016	Location of users	1.6 million users over three years	GPS Signals without speed	RNN, LSTM, DLSTM, TDNN, HMM			MAPE, MSE	No	No	Map	-	Speed prediction	-	Yes
Our contribution	2018	FCD dataset, one per 100m for 800 cars	98 million of points over 3 months, 14.5 GB	Gps signals with speed	LSTM	No	5, 10, 15, 20, 25, 30 minutes		Weather Hourly status by OWM	No	Subset of map	5, 10, 15, 20, 25, 30, 45, 60 minutes	Speed prediction, Congestion using 3 classes	Tensrflow, Keras	No

Table 2.3: Comparison with related work

Table 2.3 summarises the existing work that has been discussed on this section. The related work differs on data sources as well as the extrernal sources used in order to increase the accuracy of the models. These references are using the most popular methods. Our contribution differs on the data source, using an FCD dataset, at the external sources and on prediction models.

Data sources
Sensors
Radars
Cameras
GPS
RFID
Bluetooth
Wireless sensors
Loop detectors
Virtual Trip Lines
Social networks
Floating Car Dataset

Prediction models
Kalman filters
k-nearest neighbor
Artificial Neural Networks
Back Propagation Neural Network (BPNN)
Box–Jenkins
ARIMA (KARIMA)
Subset ARIMA
Vector autoregressive
ARIMA explanatory variables (ARIMAX)
Moving Average (ARMA)
Space Time ARIMA
Seasonal ARIMA (SARIMA)
Weighted Support Vector Regression (SVR)
Linear Genetic Programming
Multilayer Perceptron
Fuzzy Logic (FL)
Bayesian Network
Markov Chain
Markov Random Fields (MRFs)
Support Vector Regression (SVR)
local Weighted Learning (LWL)

Prediction values
Binary (Congested & Non congested)
Congested - Moderated - Non congested
Fluency & Slow & Congestion & Extreme congestion
Neavily congested - Congested - Lightly congested - Non congested - Free flow
Speed prediction

Table 2.4: More often used data sources, prediction models and prediction values.

As we have seen in this section there are several approaches proposed on the field of transportation and mobility with different implementations, data sources, architectures and prediction values.Table 2.4 summarizes the most popular.

Chapter 3

Traffic prediction methodology

In this section the pipeline and the phases of our methodology in order to extract the correct congestion predictions are described. As it is explained in figure 3.1 the GPS signals come via the communication between cars and satellites. The FCD dataset that has been used in this thesis was collected from CERTH HIT. This dataset is presented and analysed in section 5.1. In order to create accurate predictions there have been implemented several modules with the capability of transferability. The whole process has been divided into phases. The base stages of phase one contain the cleaning of dataset, the map matching procedure (section 3.1) as well as road segmentation using the OSM data model (section 3.2). After the end of phase one the dataset is ready for refactoring using the aggregation phase as it is described in section 3.3 using several different window sizes. After the aggregation of the time segments the datasets driven in the procedure of prediction and validation using machine learning techniques as it is described in section 3.4. Each phase implements different functionality and will be discussed thoroughly.

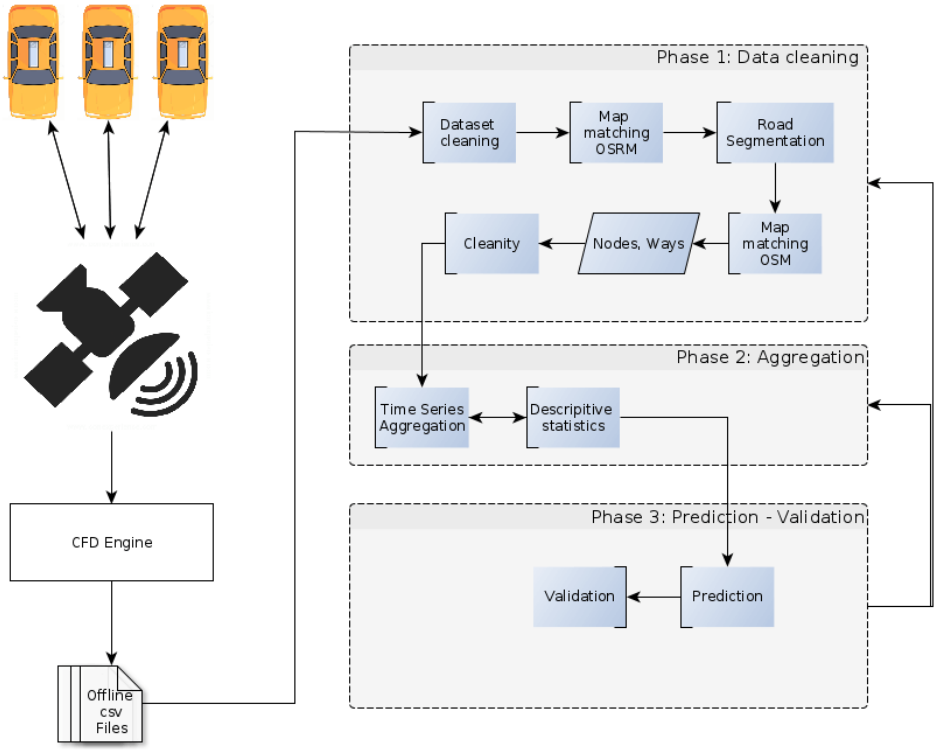

Figure 3.1: Traffic prediction methodology

3.1 Map matching

Mapping a part of the earth's surface involves a multi-step process to get from a point on the earth to a point on a map. During that process, the point on the earth is assigned a geodetic coordinate, which is then transformed into the map's coordinate system, resulting in the location on the map. Several concepts are needed for creating a map, which should be discussed here:

Geodetic Datum is a reference used for geodetic surveying. It connects an ellipsoid and a prime meridian (defining the coordinate system root) with one or more reference points on the earth's surface.

Geodetic Coordinate Reference System (GCRS) consists of a geodetic datum and an ellipsoidal coordinate system, the GCRS allows for assigning a pair of coordinates (usually longitude/latitude) to every point of the datum's area.

Projected Coordinate Reference System (PCRS) projects a GCRS into a different coordinate system, e.g. a Cartesian system for 2D maps.

The mapping process effectively uses three steps to get to a map location for any given point on the earth:

1. Map the real point to the datum's ellipsoid surface, using the geoid and the datum reference(s)

2. Map the ellipsoid point to an ellipsoid coordinate, using the GCRS

3. Map the ellipsoid coordinate to a map location, using the PCRS

The mapping applications discussed in this thesis use the World Geodetic System 1984 (WGS84) as their reference system (Imagery and Agency, 1984). WGS84 uses the aforementioned EGM96 geoid and defines its own geodetic datum and Geodetic Coordinate Reference System. Based on WGS84, different projections can be used to create a map.

As we have discussed in the previous section the whole dataset is coordinate tuples (longitude, latitude). To make the coordinates valuable information we have to match them with something stable. In GIS this is called geodesy (Kleusberg and (eds.), 1998), and can be done using the trajectories and is defined here as the path of a vehicle on a road network (Zheng, 2015).The objective of trajectory map matching is to estimate this path from noisy position data (in our case from GPS devices) .

There are several techniques to make this possible by using graphs (Willard, 2014). In our approach to accomplish this we matched the location we extract from the original dataset to a real location on earth as (Luxen and Vetter, 2011) suggest with an open source library (OSRM).

OSRM is a C++ routing engine that works using topological information. More specifically we were using a pbf data file with a map of Thessaloniki. After the initialization OSRM builds a RESTful API witch can be triggered using http requests. The main services OSRM supports:

- Nearest - Snaps coordinates to the street network and returns the nearest matches.

- Route - Finds the fastest route between coordinates.

- Table - Computes the duration of the fastest route between all pairs of supplied coordinates.
- Match - Snaps noisy GPS traces to the road network in the most plausible way.
- Trip - Solves the Traveling Salesman Problem using a greedy heuristic.
- Tile - Generates Mapbox Vector Tiles with internal routing metadata.

OSRM also supports "profiles". Profiles representing routing behavior for different transport modes like car, bike and foot. You can also create profiles for variations like a fastest/shortest car profile or fastest/safest/greenest bicycles profile.

For the purposes of this thesis we have only used the match service using the car profile.

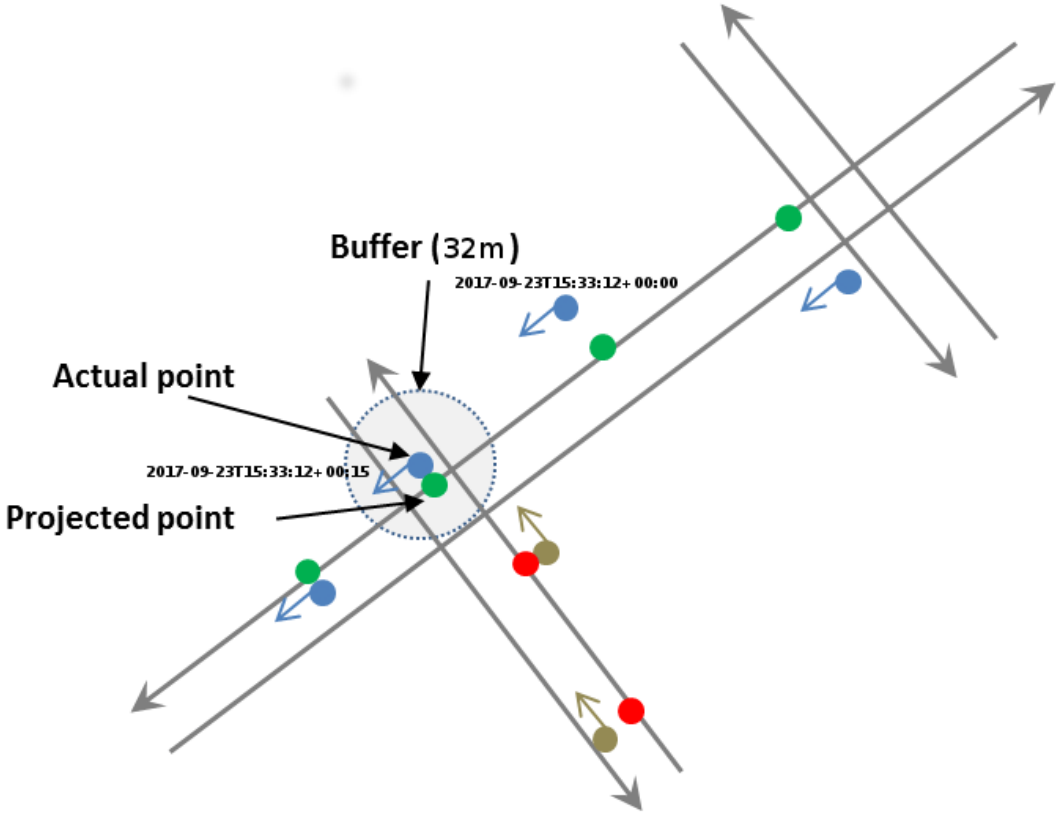

Figure 3.2: Map matching methodology

As we can see in figure 3.2 a 32m buffer has been used for each point which means that for each instance we are trying to find the best projected point on this perimeter. For each point we are setting the current timestamp and the orientation of the point. OSRM calculates the travel path using the trajectory for each car using the sequence of points and if it is impossible to travel the distance between the points in the specific time, or the orientation of points are vertical or on the opposite direction the projected point is recalculating. If all conditions are positive we find for each blue the green map matched point.

3.2 Road segments clustering

In this subsection it is described how the matched points clustered into road segments. To achieve this each point is first matched into point on nearest road based on the trajectory of each car as described in section 3.1 in the following subsection the procedure of clustering these points into roads segments will be discussed. A road segment represents a part of the road that has been divided.

The main goal is to describe the points into groups based on the road that are appeared on. As an example we can see in figure 3.3 a road separated into seven segments.

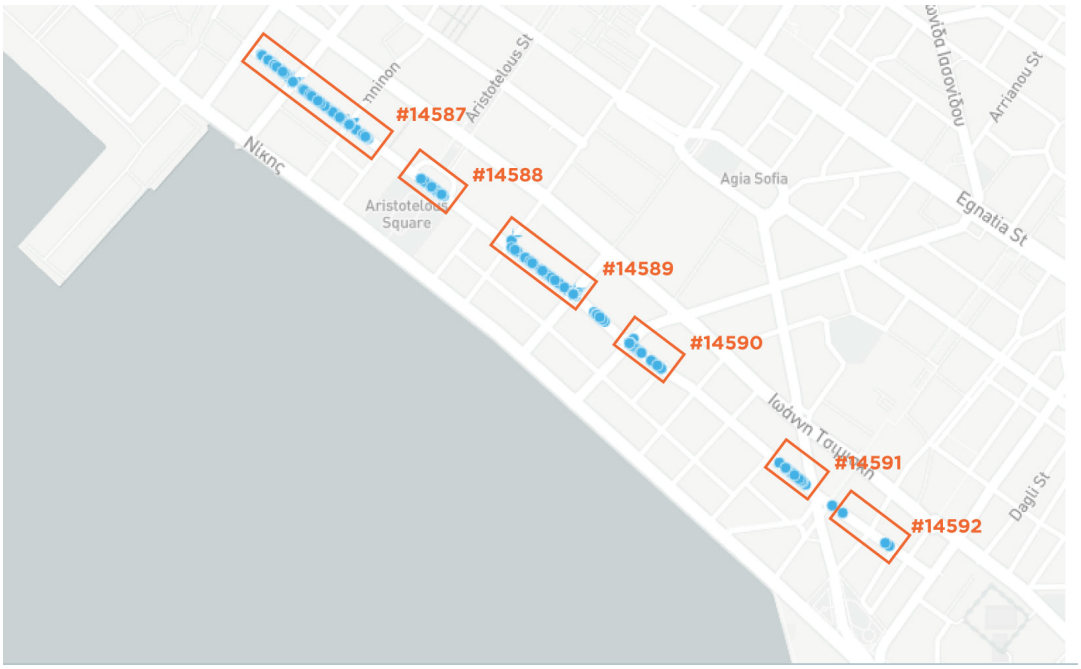

Figure 3.3: Matched points into road segments

There are segments for each road based on OpenSteetMap way elements. Each OSM segment is defined by linear features and area boundaries. Each way is defined as a json file that contains a list of nodes, where nodes defines a point in space.

In order to transform the location of each point (longitude,latitude) to id's the Nominatim geocoding API has been used which is based on OpenSteetMap. For each point a query is structured based on the coordinates of the point and a response is gathering as node or way. If the response is a a way as A.2 shows the whole process is terminated and the point clustered on a specific osm way segment with an id as the identifier. If the response is a node as figure A.3 shows the information is stored and these points are going to be cleaned as it is described in section 4.2.3.

For each response as explained in lstlisting A.2 and A.3 the returned information is strored into a local database as it is described in section 4.4.

A way is represented as a line of nodes and a node represents a point on map. Its way is divided into segments based in special nodes and tags, such as:

- Traffic lights

- Cross roads
- Crosswalk
- Bridges

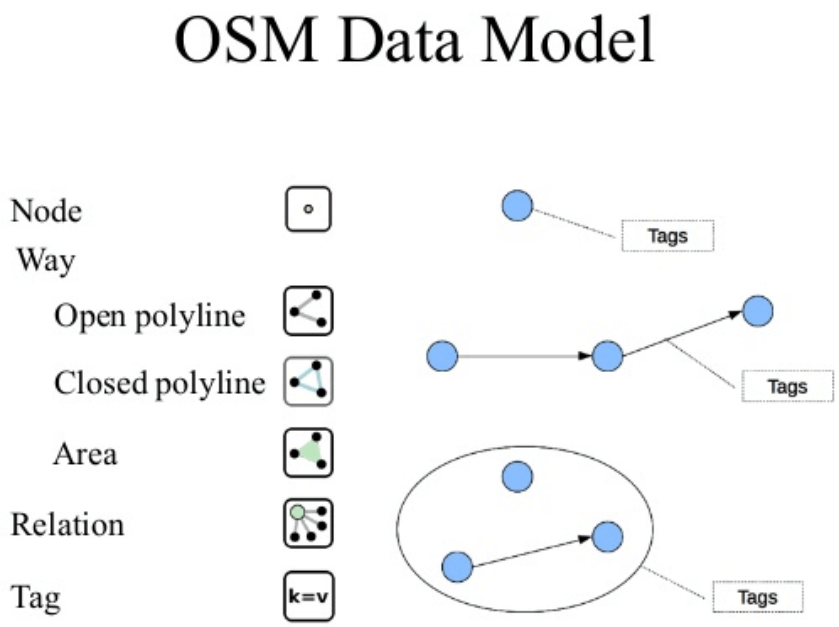

Figure 3.4: OSM Data model

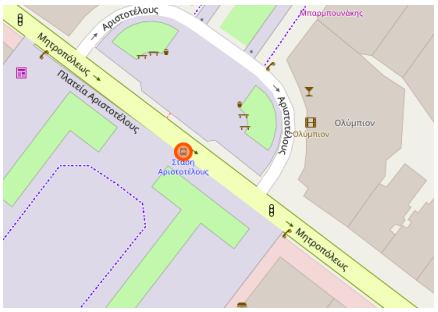

Figure 3.5: OSM node representation

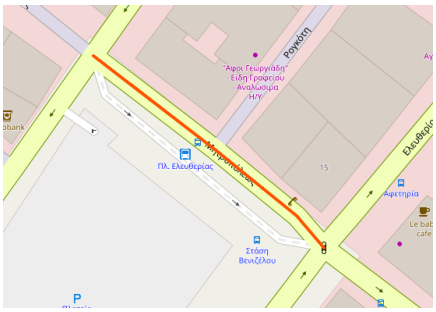

Figure 3.6: OSM way representation

Figure 3.4 explains the osm data model and shows the relation between nodes and ways, as figure 3.5 represents a node and figure 3.6 how a list of nodes can construct a way using the OSM data model.

3.3 Time series aggregation

In this subsection the time series aggregation and the methodology used in order to aggregate the dataset using time is discussed. Our schema of dataset represents car location over the time, with latency beside the point (4-6 seconds). The amount of data that can produced is about 46926 points per hour with 800 active cars . For trending and visualization reasons an aggregation phase has been created using the dataset with time windows sizes of 5,10,15,20,25,30 minutes. The aggregation phase is described in algorithm 3.1. This algorithm runs through the dataset as a sliding window for each time segment size and:

1. Selects documents on the time segment
2. Keeps the needed fields
3. Loops over the valid fields
4. Computes the metrics

Algorithm 3.1: Create time segments

Data: Sorted ways by time
Result: Ways separated by time
1 **while** *not at end of ways* **do**
    get firstTimestamp(way)
    getPointsOrdered(way)
    initialize variables
    while not at end of points **do**
      getPoint()
      maxSpeed,minSpeed=compute()
      computeTimeDifference()
      if numberOfPoints $\geq$ minNumberOfPointsPerWay and minDeference $\geq$ timeWindow **then**
        median, mean, standard deviation= compute()
      end if
    end while
2 **end**

Figure 3.7 depicts an example of road segment trying to visualize the whole process of aggregation using the sliding window technic. As we can see, in the figure the speed of the cars, the number of them and the statistics that can be examined from the data points using different window sizes. In this figure we can understand that the time series segmentation has been done on the geographical and time dimension for each segment of each road using all the available data points.

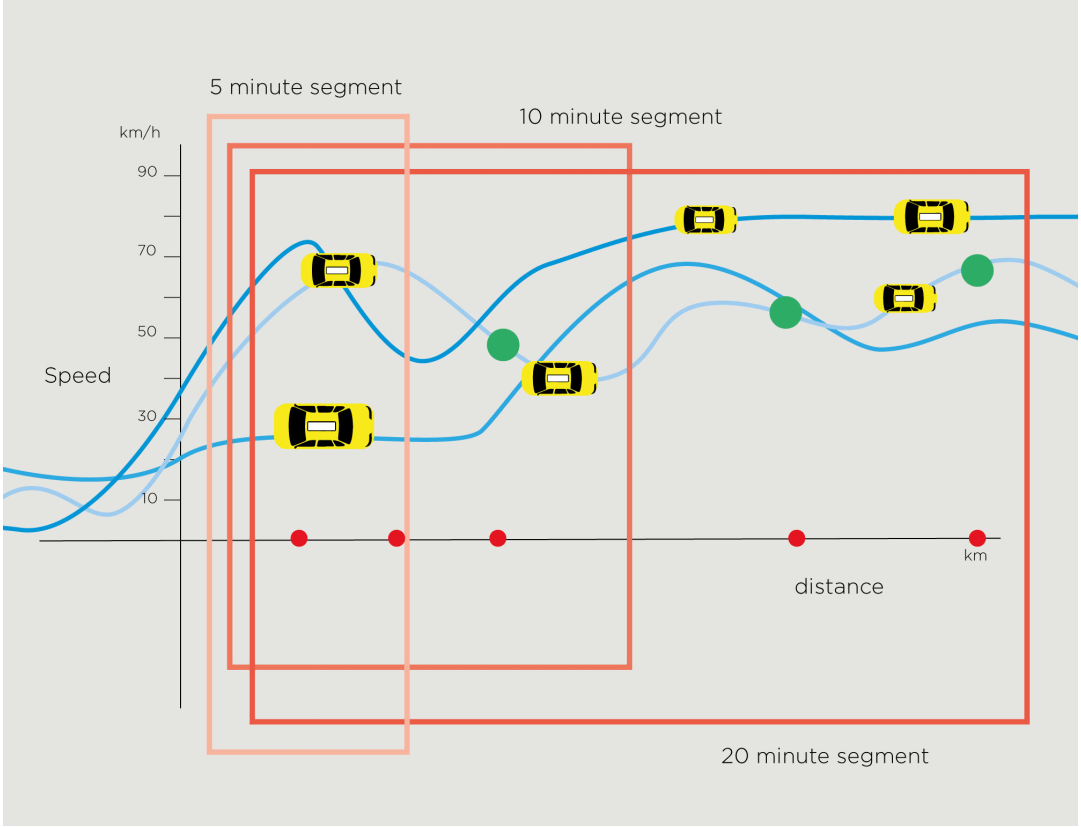

Figure 3.7: Time series aggregation

After the whole process ended standard statistics and metrics were computed in order to use them for prediction such as mean speed, median speed, standard deviation, mode the metrics are changing as the window size changes. For example, on the 5 minute windows size there exist two cars with mean speed 50 km/h, from the other hand on 20 minute windows size there are six cars with min speed 15 km/h and max speed 80 km/h and the mean speed is 60 km/h. The algorithm 3.1 iterates over the road segments using different size of window trying to find the best fit for each timestep of prediction and produces statistics for each one of them. Our initial hypothesis is confirmed and is more easy to predict the congestion of a segment three days later using 60 minute time window rather that as with 5 minute window size.

3.4 Supervised Prediction

In this section the prediction and validation phase is discussed. The main pipeline of the process follows 6 basic steps:

1. Transform time series for supervised learning
2. Create the labels based on the speed using three main classes
3. Feature selection
4. Transform values
5. Model evaluation

6. Plot results

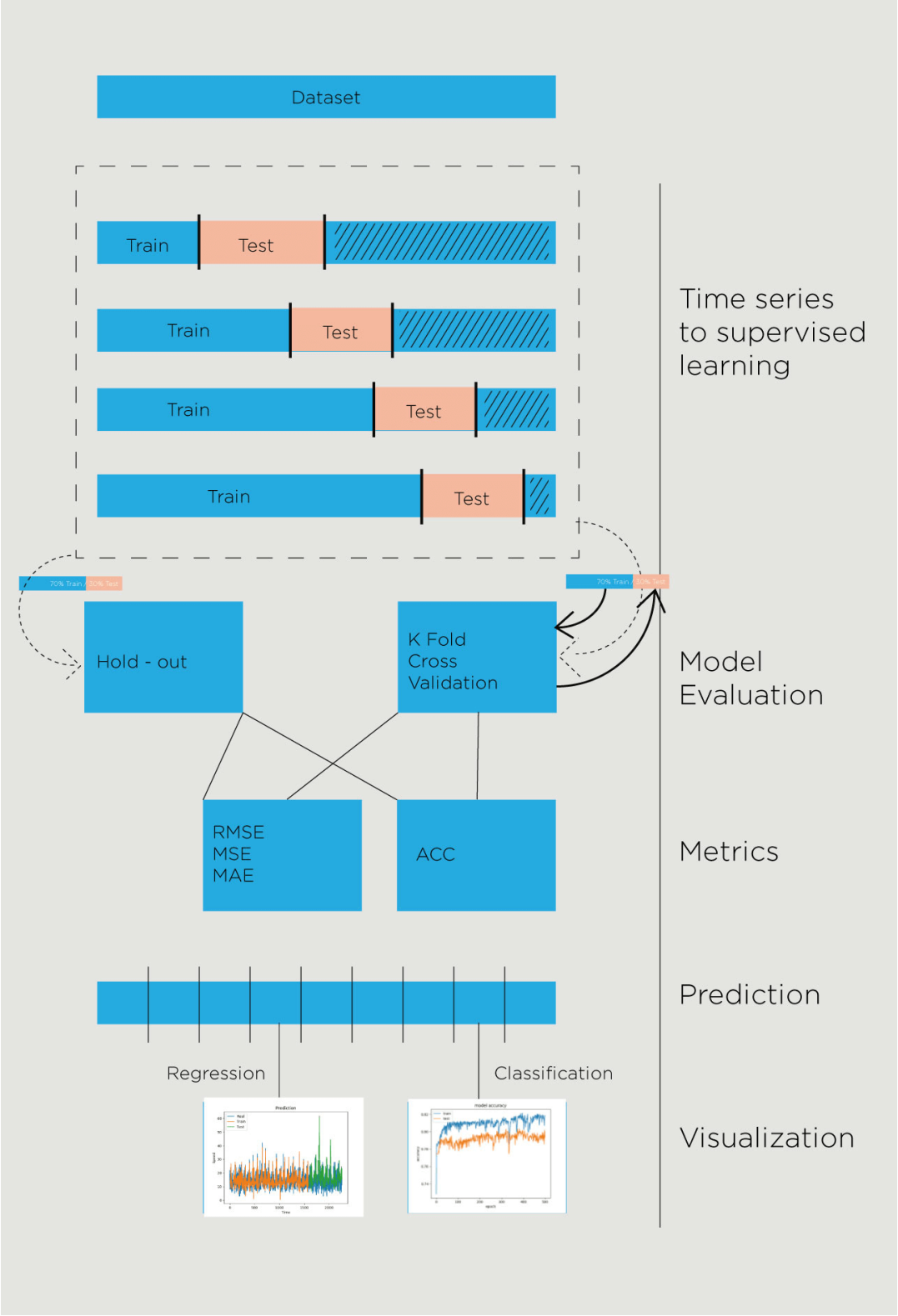

Figure 3.8: Prediction model and methodology

In figure 3.8 it is described the whole process of prediction from the phase of dataset along to visualization process per method and methodology. The whole process is splitted into the phase of time series to supervised learning, as we can see in figure 4.5 the whole process transform the dataset in order to use supervised technics. In this phase the look back variable describes the time (in minutes) used per step. For example if we only want one hour of observations and we have a 10 minute time window, in order to use this observations the time series to supervised routine will shift the dataset six steps. So for the timestemp t the process will use the observations $(t-1), ..., (t-10)$. So the look back parameter describes the points that have been observed into this period of time. An example of the procedure is described on table 4.5. After this step the dataset is splitted using the train/test percentage. If the hold-out method used for the model evaluation phase the train set used for training and the test set for prediction, in case of k-fold cross validation the dataset splits and the train part is used for training and test the model among fold and the test set is used for the evaluation of the model in order to avoid overfitting . On the next phase the appropriate metrics used for regression and for classification. On the last phase prediction is produced for each model and then the appropriate visualization used in order to create charts for learning curve and for the real values of predictions.

3.4.1 Time series to supervised learning

As it is described in algorithm 3.2 for each time instance of the dataset is calculated the n-th previous steps and the k-th next steps. On each run a separate values has been used for input and output prediction values for experimental reasons, each input and output value corresponds to the time windows that have been created on the time aggregation phase.

Algorithm 3.2: Time series to supervised

Data: Time series data model, input shape, output shape
Result: Data model for supervised learning

```
while not end of data do
    initialize set
    getFirstTimestamp(way)
    getPoints(way)
    while points≤ input shape do
      inputSequencePoints.append()
    end while
    while points≤ output shape do
      outputSequencePoints.append()
    end while
end
```

In figure 3.9 we can see an example with m =150 and p =2. This means that the last 150 time steps have been used in order to predict the 2 next moments. If a 5 minute time segment has been used , in order to predict the congestion for the next 10 minutes will need to get the last 12.5 hours congestion status.

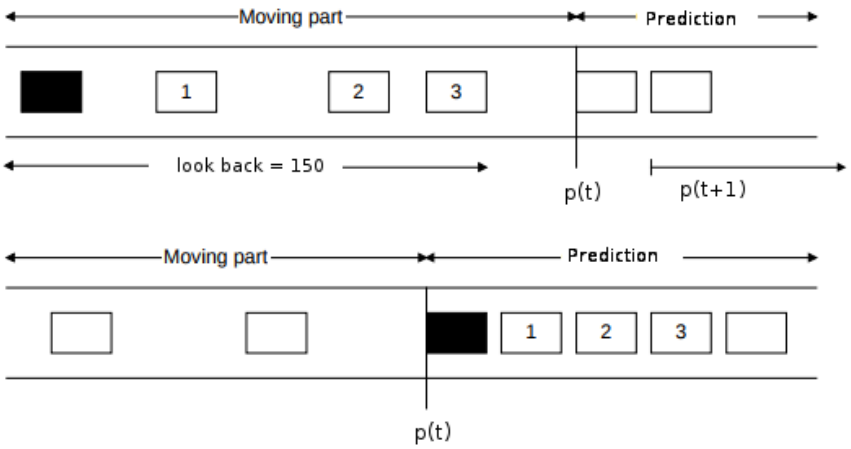

Figure 3.9: Input - output prediction

The phase of prediction is divided into two separate threads. The one that uses the average speed from the time series aggregation that is explained in section 3.3 as classes, on this thread the whole prediction process can be described as a supervised classification problem. The second thread uses the data as it is stored on database without the classes which means that the problem is described as a regression problem using the real speed values. These values are used as a sequence of mean speeds per segment, using the same series aggregation process.

3.4.2 Real values to classes

In order to create the classes for the congestion status, an average flow speed has been calculated for each segment of the road based on previous observations on the time series aggregation phase. The main process is described in algorithm 3.3

Algorithm 3.3: Real values to classes

```
Data: Speed observations
Result: Congestion classes
    initialize counters
    getFirstTimestamp(way)
    for each way segment do
      getPointsOrdered(way)
      sum points speed
      sum observed cars
      calculate average speed
      if averageSpeed≥ 0.4 then
        speedClass="green"
      else if averageSpeed <0.4 and averageSpeed ≥ 0.2 then
        speedClass="orange"
      else if averageSpeed <0.2 then
        speedClass="red"
      end if
    end for
```

The main purpose of this operation is to transform the prediction into multi class classification problem.

3.5 Statistical prediction

In this section the statistical prediction phase that has been used in order to achieve accurate predictions of real speed values is presented. To accomplish this an evaluation has been done trying to avoid seasonality and trends in our dataset. After the evaluation of the dataset the ARIMA method has been used with several different approaches has be done on autoregression, and movig average models.

3.5.1 Time series evaluation

Before the processing of the series is essential to evaluate the time series dataset (FCD) in order to detect if it is stationary or not. To make this possible it is assumed that all cars have pretty much the same behavior as they flow in the same town and on the same roads.

To confirm this hypothesis the dataset has been evaluated using a stationarity test where **stationary** process is one whose statistical properties do not change over time. More formally, a strictly stationary stochastic process is one where given $t_1, ..., t_\lambda$ the joint statistical distribution of $X_{t1}, ..., X_{t\lambda}$ is the same as the joint statistical distribution of $X_{t1+\tau}, ...X_{t\ell+\tau}$ for all ℓ and τ . This is an extremely strong definition: it means that all moments of all degrees (expectations, variances, third order and higher) of the process, anywhere are the same. It also means that the joint distribution of (X_t, X_s) is the same as (X_{t+r}, X_{s+r}) and hence cannot depend on s or t but only on the distance between s and t, i.e $.s - t$.

Stationarity means that mean and the variance of a stochastic process do not depend on t (that is they are constant) and the autocovariance between X_t and $X_{t+\tau}$ only can depend on the lag τ (τ is an integer, the quantities also need to be finite). Hence, for stationary processes, $\{X_t\}$, the definition of autocovariance is:

$$\gamma(\tau) = cov(X_t, X_t + \tau)$$

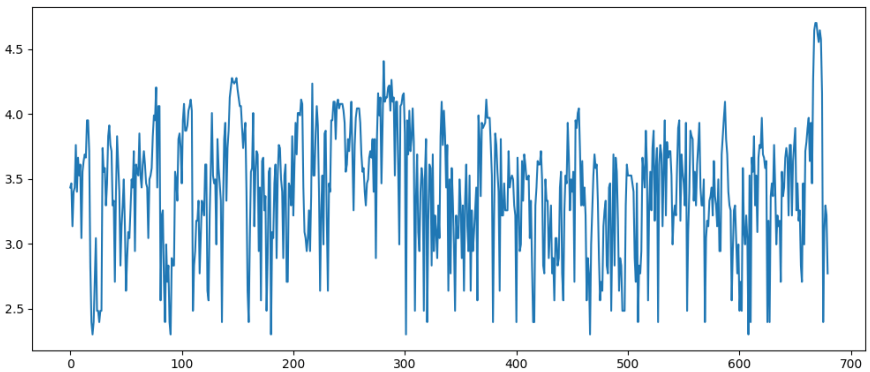

Figure 3.10: Time series stationarity

As shown in figure 3.10 there is no trend on the time series and the speed of the car is between 15 and 45 km/h this means that for all the time window that is shown in the figure there are no trends or seasons with different behavior. To be more accurate a split of dataset has been used trying to figure out the mean speeds per split in order to understand the big picture of the time series and as assumed, mean speed on first split is 36.802 km/h and 32.500 km/h on the second half. On contrary the variance of this mean is 25.011 and 28.071 respectively.

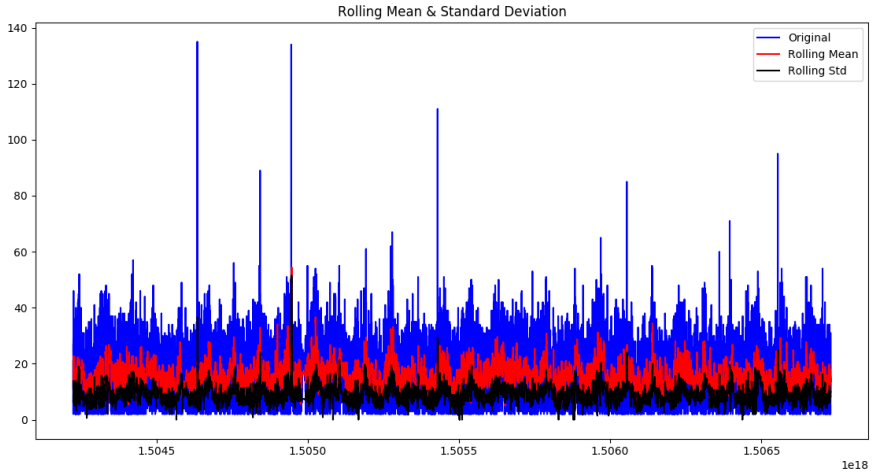

Figure 3.11: Stationarity test using real values,st. deviation, mean values

As the results from the mean values were not enough a Dickey-Fuller Test run on FCD dataset in order to test stationarity and **seasonality** thoroughly. Figure 3.11 presents the real value of speed, the mean value and the standard deviation of observed speed values.

Test Statistic	-41.823732
p-value	0.05065929
#Lags Used	76.000000
Number of Observations Used	192360.000000
Critical Value (1%)	-3.430384
Critical Value (5%)	-2.861555
Critical Value (10%)	-2.566778

Table 3.1: Stationarity test using real values,st. deviation, mean values

Table 3.1 presents values from the Dickey-Fuller test. As assumed the mean and standard variations have small variations with time. Also, the Dickey-Fuller test statistic is less than the 10% critical value, thus the time series is stationary with 90% confidence.

3.5.2 ARIMA method

In order to create statistical prediction ARIMA model has been used. ARIMA is an acronym that stands for AutoRegressive Integrated Moving Average. It is a generalization of the simpler AutoRegressive Moving Average and adds the notion of integration. This acronym is descriptive, capturing the key aspects of the model itself. Briefly, they are:

AR: Autoregression. A model that uses the dependent relationship between an observation and some number of lagged observations. Autocorrelations, are numerical values that indicate how a data series is related to itself over time. More precisely, it measures how strongly data values at a specified number of periods apart are correlated to each other over time. The number of periods apart is usually called the "lag". For example, an autocorrelation at lag 1 measures how values 1 period apart are correlated to one another throughout the series. An autocorrelation at lag 2 measures how the data two periods apart are correlated throughout the series. Autocorrelations may range from +1 to -1. A value close to +1 indicates a high positive correlation while a value close to -1 implies a high negative correlation. These measures are most often evaluated through graphical plots called "correlagrams". A correlagram plots the auto- correlation values for a given series at different lags. This is referred to as the "autocorrelation function" and is very important in the ARIMA method.

I: Integrated. 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.

MA: Moving Average. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations. Moving average models look very similar to the AR model, the concept behind them is quite different. Moving average parameters relate what happens in period t only to the random errors that occurred in past time periods, i.e. $E(t-1), E(t-2)$, etc. rather than to $X(t-1), X(t-2), (X_t - 3)$ as in the autoregressive approaches. A moving average model with one MA term may be written as follows: $X(t) = -B(1) * E(t-1) + E(t)$. The term B(1) is called an MA of order 1. The negative sign in front of the parameter is used for convention only and is usually printed out auto- matically by most computer programs. The above model simply says that any given value of X(t) is directly related only to the random error in the previous period, E(t-1), and to the current error term, E(t). As in the case of autoregressive models, the moving average models can be extended to higher order structures covering different combinations and moving average lengths.

3.6 Evaluation

Accuracy metric is used to evaluate the prediction of multi class classification and conversely Root mean squared error and Mean Absolute Error, Mean square error, used to evaluate the prediction of the regression problem both on neural networks and ARIMA model.

Accuracy (ACC) indicates how close a measurement is to the actual value. Accuracy has been used after the cleaning of the dataset from unbalanced observation using the Synthetic Minority Over-sampling Technique (SMOTE) technique as it is described in section 4.5.1.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	TP	FN
	Negative	FP	TN

Table 3.2: Confusion matrix

Where:

- **True Positives (TP):** number of positive examples, labeled as such.
- **False Positives (FP):** number of negative examples, labeled as positive.
- **True Negatives (TN):** number of negative examples, labeled as such.
- **False Negatives (FN):** number of positive examples, labeled as negative.

As our predictions is a multi class classification problem the confusion matrix is transformed as table 3.3 shows corresponds on true positive and Errors between the samples.

		Predicted Class		
		Green	**Orange**	**Red**
Actual Class	**Green**	TP_G	E_{GO}	E_{GR}
	Orange	E_{OG}	TP_O	E_{OR}
	Red	E_{RG}	E_{RO}	TP_R

Table 3.3: Confusion matrix multi class

The overall accuracy in our multi class classification problem is transformed into this:

$$ACC = \frac{TP_G + TP_O + TP_R}{TP_G + E_{CO} + E_{GR} + E_{OG} + TP_O + E_{OR} + E_{RG} + E_{RO} + TP_R}$$

Mean squared error (MSE) is a quadratic scoring rule that also measures the average magnitude of the error. It is the average of squared differences between prediction and actual observation.

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_i - \hat{y}_j)^2$$

Where:

- n = the number of errors

- $\sum$ = summation of observations
- $(y_i - \hat{y}_j)$ = error between observations

Root mean squared error (RMSE) is a quadratic scoring rule that also measures the average magnitude of the error. It's the square root of the average of squared differences between prediction and actual observation.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_i - \hat{y}_j)^2}$$

Where:

- n = the number of errors
- $\sum$ = summation of observations
- $(y_i - \hat{y}_j)$ = error between observations

Mean Absolute Error (MAE) measures the average magnitude of the errors in a set of predictions, without considering their direction. It is the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_i - \hat{y}_j|$$

Where:

- n = the number of errors
- $\sum$ = summation of observations
- $|y_i - \hat{y}_j|$ = the absolute error

The metrics that have been discussed in this subsection were used accordingly to the prediction model. On regression procedure the MSE, RMSE, MAE has been used as metrics, and on multi class classifiers the accuracy of each model. The fit of models has been evaluated using the loss per prediction.

As this chapter explains in order to create the framework, a pipeline was designed based on the methodology that has been described with respect on nature of the problem. Since the dataset comes from the real world the datasets has been pre-processed and evaluated using several studies on time series forecasting with evaluation that corresponds on each phase of the framework.

Chapter 4

Implementation

This chapter explains the procedures that have been implemented in order to archive the main goal of this thesis. To accomplice this the main pipeline has been divided into sections and can be described as modules for each step. First of all the basic architecture is discussed trying to understand the whole pipeline. After this it is explained the implementation and the tools that have been used trying to map match original data points into road segments, using several techniques and tools. In second phase the datasets goes throw a pipeline of cleaning. Finally, the prediction phase is presented. This phase is divided into regression and multi class classification using neural networks and pure statistics trying to find the best prediction models.

4.1 Architecture

In this section we will discuss the architecture and the pipeline that we have created in order to implement the traffic predictions from the initial datasets. In figure 4.1 is the main diagram of our system.

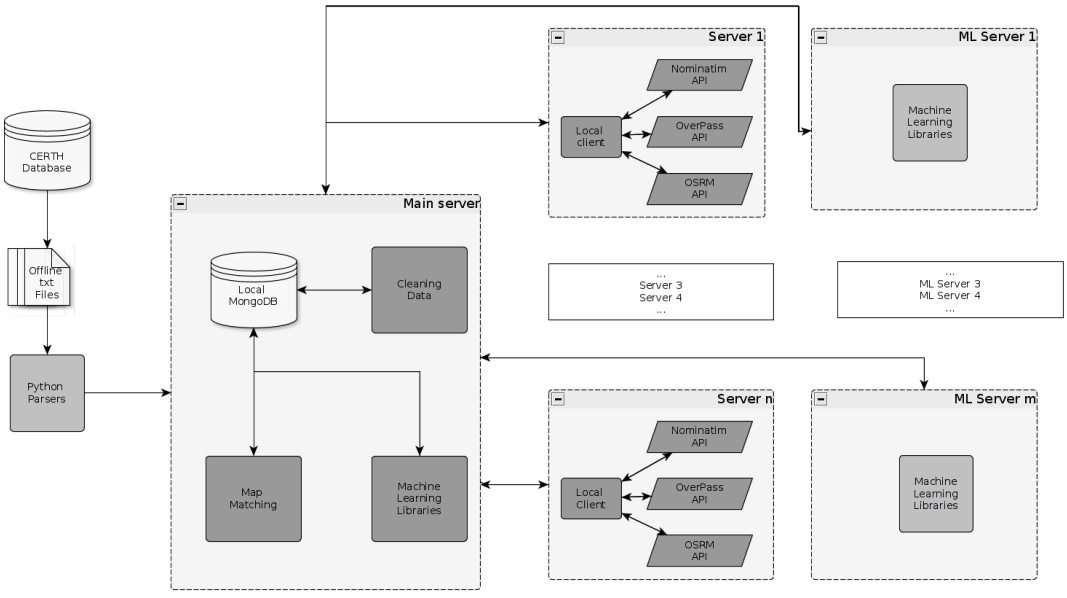

Figure 4.1: System Architecture

The whole procedure is divided into steps. Each step is an individual process and can run modular beside steps. The basic steps can described as follows:

1. Obtain data
2. Read data
3. Clean data
4. Mach points to map
5. Match coordinates to ways
6. Match mismatched nodes to ways
7. Compute time segments
8. Predict congestion

The whole process is designed taking into account the nature of the problem for online processing all the algorithms and routines have implemented in multi-thread parallel architecture. As it was not easy to run our experiments using multiple computers the framework has tested experiments as it is described in section 4.6 using the testing environment. The basic diagram of modules of our system can be found in figure 4.2.

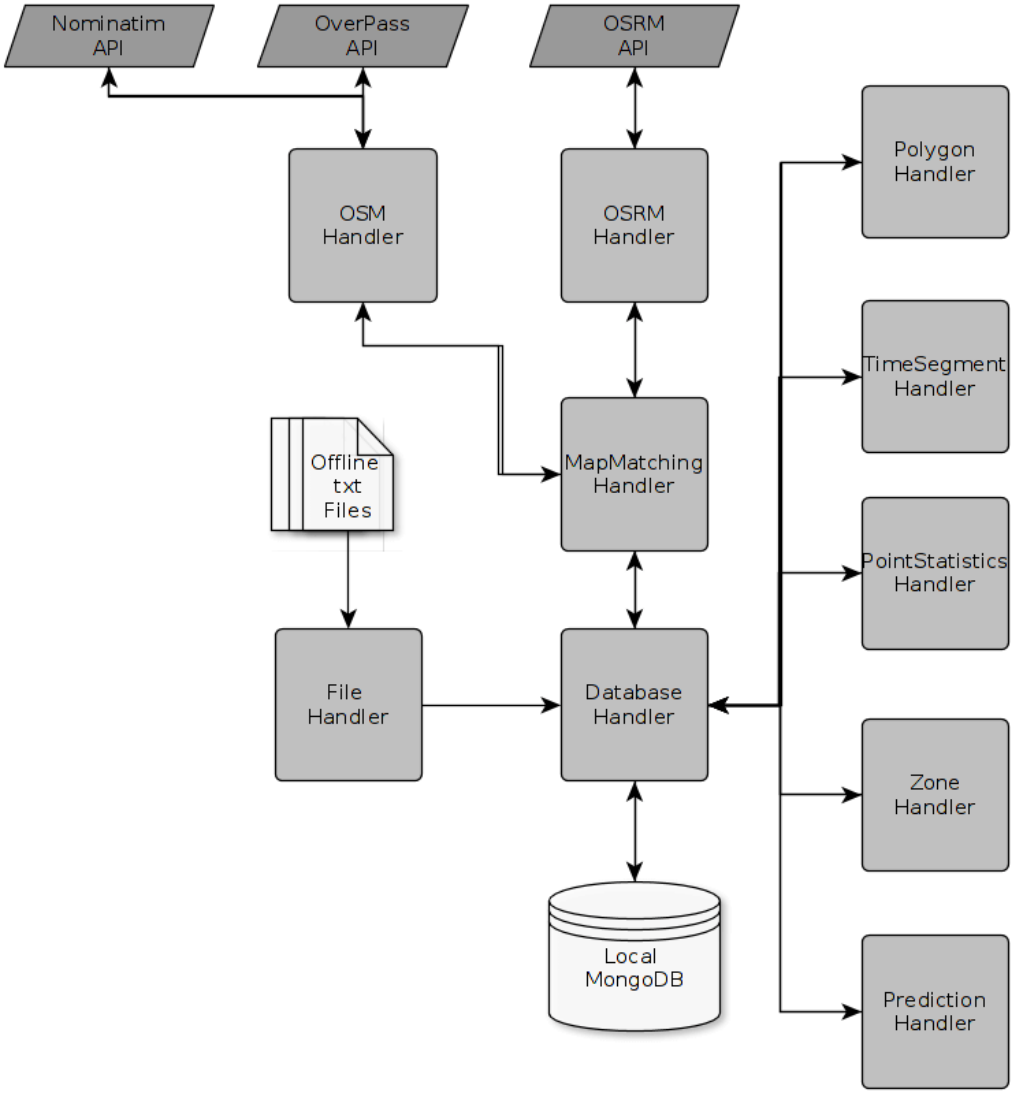

Figure 4.2: System modularity

Each rectangle in figure 4.2 represents a module that can communicate with the database handler that transects information to our local Mongo database. As we can see there are two modules that cannot communicate directly with the database hander. This design was used in order to make our map matching approach more scalable. To make this happen we have used OSM, Nominatim and OSRM APIs as they are appeared with rhombus. The process of observations has be done using the RESTfull architecture with multiple queries. The whole process is been divided into threads and each thread works asynchronously into each separate phase. The main idea behind the distributed architecture is described more extensively in section 6.3.

4.2 Map matching implementation

As we have discussed in section 2.3 the signal of GPS is relative to the height of the buildings near the device and the weather conditions. This can be confirmed in figure 4.3 as the original data points are not linked on roads. To make this

happen we used a map matching approach to match the points on roads and segments using a map based approach with an open source tool as explained in 3.1. This process has be done in order to predict the traffic per road and not per region.

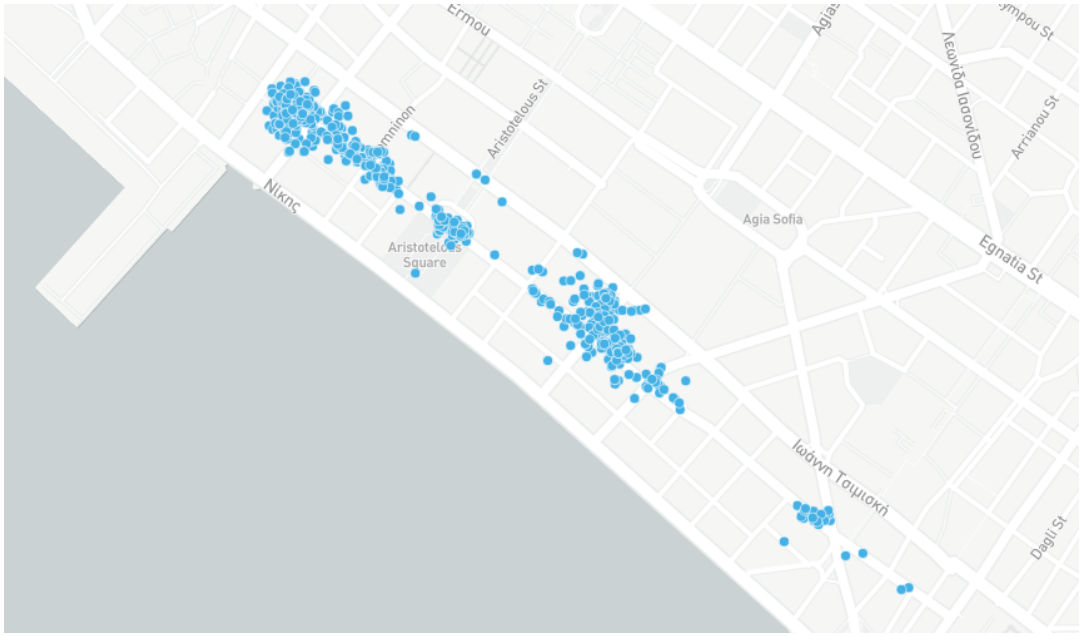

Figure 4.3: Original points

In figure 4.3 we can see a subsample of the original dataset which represents data points near to a segment id. This figure corresponds to 2547 points on segments road name "Mitropoleos" with five different segments.

The vehicle's path is sampled in a sequence of position measurements $(z0, ..., zT)$ during a time interval $[0, 1, ..., T]$. Each position measurement zt with $t \in [0, T]$ corresponds to a position on the map. In our approach we sorted the dataset on timestamp because as we can see in table 5.2 the coordinates are not stored with order and we have to order them to get the sequence and we have group the timestamps by the car id to get a sequence of continues points and get the trajectories. As a result we can see in figure 4.4 which represents the original trace of a travel for a time of period.

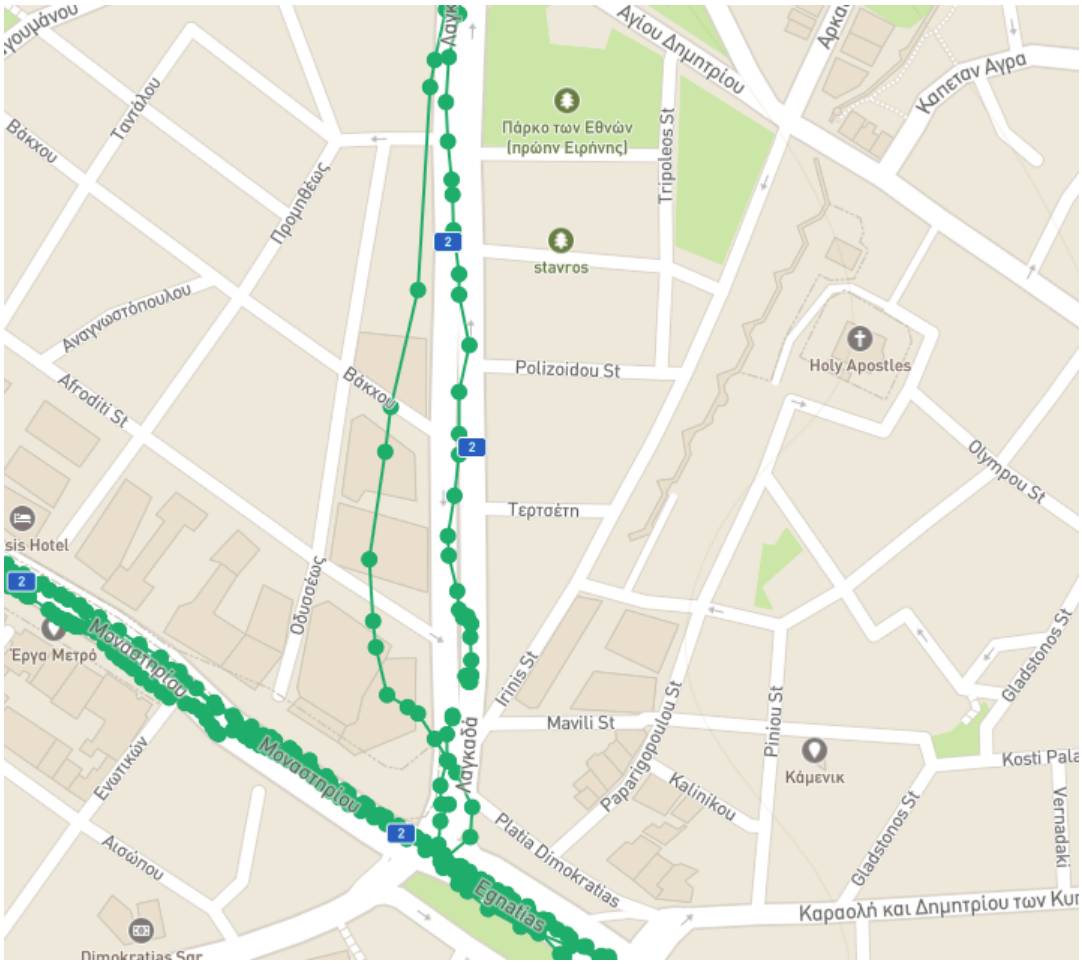

Figure 4.4: Original points in line string

Our target is to match original traces, as we can see in figure 4.5 to matched traces as in figure 4.6 on segments roads

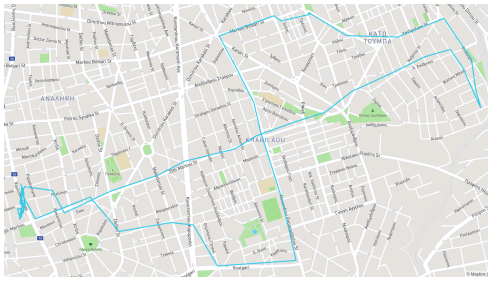

Figure 4.5: Original trajectory

Figure 4.6: Matched trajectory

As we can see in 4.6 the original trace has been moved from its original position and is matched on the road.

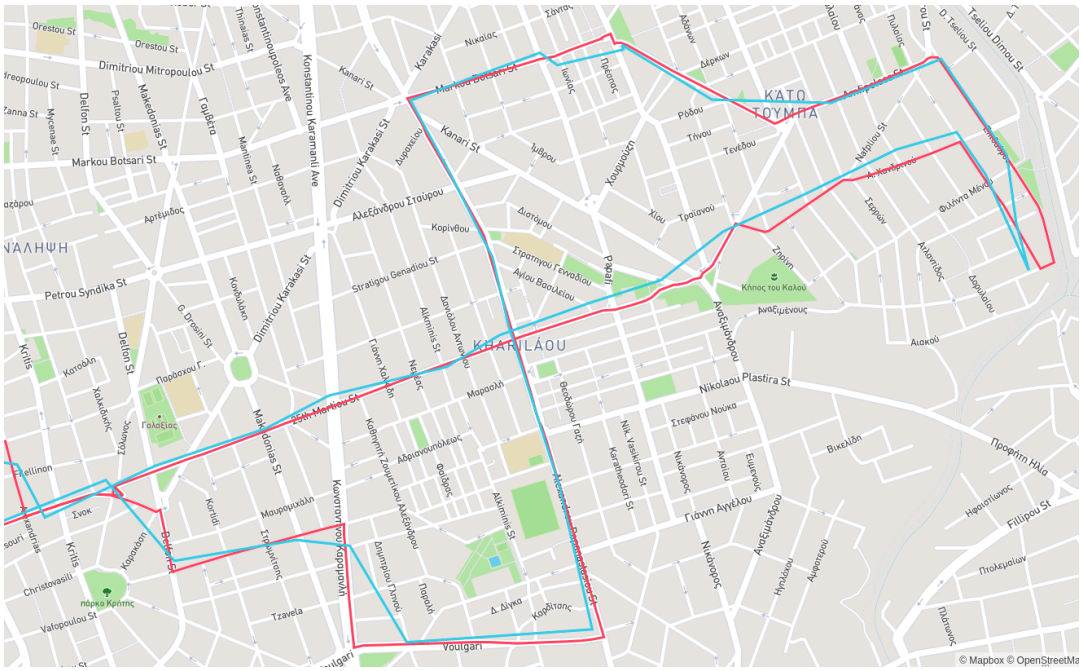

Figure 4.7: Matched line string

In figure 4.7 we can see the differences of the original line string of points compared with the matched line string of points on a random trace of the dataset. As the response for each query there were a returned trajectory with more points than the query, the used points for each trajectory were the most closed points from the original trace.

4.2.1 Map matching using OSRM

As we will discuss in 4.6 the overall process was designed to run parallel on multiple computers using all their computational power. To make this possible we have to create the line strings of the routes using direct asynchronous urls with a batch of points and their features. So we have created a routine that gets the car ids and extracts urls that can be handled from OSRM as we have explained in section 3.1 . The algorithm that creates the batches of urls is described in

algorithm 4.1.

Algorithm 4.1: Create urls for OSRM

```
Data: FCD dataset
Result: urls for map matching using batchs of points
initialization;
getDistinctCarIds();
while not at end of car ids do
    readPoints();
    if points.size>2 then
        point.get(longitude,latitude,orientation,timestamp);
        if numberOfPoints> batch size then
            OSRMStack.append(url) ;
        else
            url.append(point);
        end
    else
        getNext();
    end
end
return the list of valid urls;
```

After the whole process ends the result is a list of URLs for each thread where each URL has a trajectory and looks as the url bellow:

/match/v1/car/22.959107,40.64122;22.959047,40.641295;22.959047,40.641295;
22.959047,40.641295;22.959047,40.641295?timestamps=1504261051;
1504261141;1504261143;1504261145;1504261148&radiuses=32;32;32;32;32
&overview=full&steps=true&geometries=geojson&annotations=true
&tidy=false

To make this happen with the most accurate way we have used four features for its point :

- Timestamp
- Orientation
- Speed
- Radius

As it is argued in section 3.1 these features have been used in order to split the line strings on multi line strings in order to succeed more accurate segments. These features are handled by OSRM to cut the separate the line string by the time and the speed based on locations. This process is based on a defined radius in meters near each point. For example, if a point is away from the current location or is irregular to travel from on point to the other based on the time the routing engine separates the strings to different trajectories.

The whole process is based on car profile using OSRM. After the execution of each url query we are getting a JSON response as can be found in lstlisting A.4 with the required information about the matching. Along side the map matching

with OSRM and the correct location of points we use the response information from the geojson to build a data model, as it is described in 4.4 with structure oriented on ways for further use.

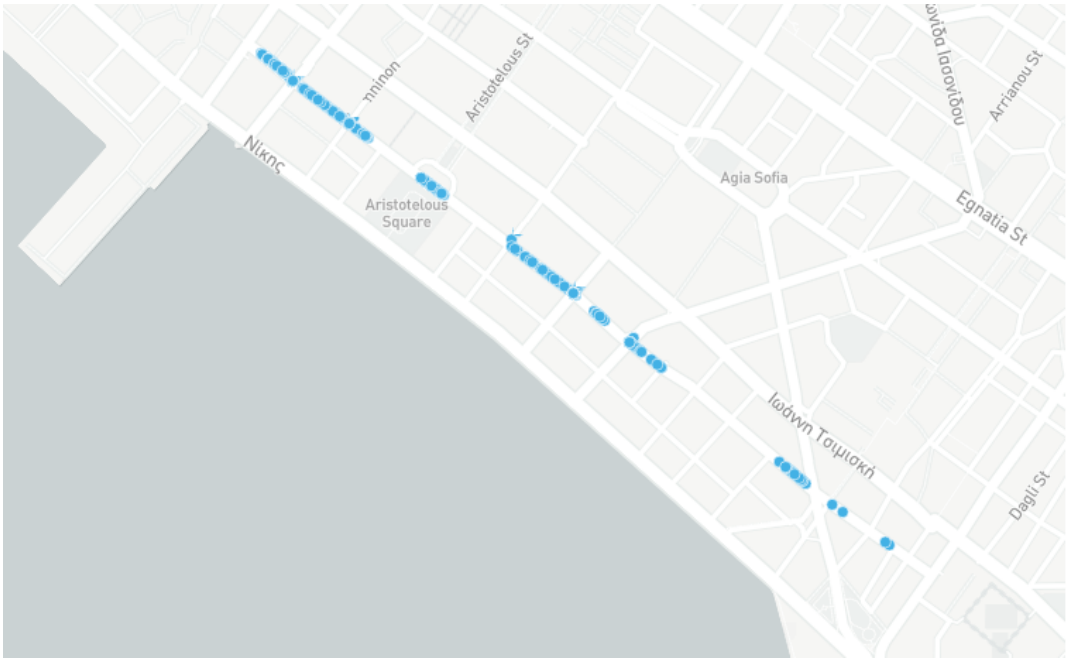

Figure 4.8: Matched points

In figure 4.8 we can see the matched points after the whole process on the same road as points matched on a road on OpenStreetMap using OSRM. We can easily understand the deferences between the figure 4.3 with the original points. As we can see in figure 5.3 each instance flaged by full matched, or not if we have matched the point to the correct position and if we have the road name returned by the OSRM geojson response. We can now cluster the matched coordinations(longitude, latitude) to OpenSteetMap segments using the full matched points.

4.2.2 Road clustering using OSM

As it is suggested in section 3.1 one of the basic phases of this implementation is to cluster each road to a specific id. To accomplish this we have clustered each road to a OpenStreetMap id. In this subsection, the way in which this process was accomplished is further explained.

As we examined in section 4.6 the whole process was implemented on a parallel environment using threads. For the purpose of clustering roads we have created a batch of URL for each thread using the matched points. As we have explained in section 4.1 the reverse geolocating system we used is Nominatim OSM API and the queries structure is represented as follows: /reverse?format=xml&lat=40.63267&lon=22.94089&zoom=18& addressdetails=1, an example of a response can be found in lstlisting A.5.

For each URL Nominatim response a json file with information about the specific coordinates. This information has been stored to our data model for further processing 4.4. As we can see for each point we get a osm-type feature and a

osm-id feature. These two attributes can cluster all our data points into road segments. Using the bounding box information returned from the reverse transform we can now represent out points into segments as in figure 4.9.

Figure 4.9: Way segments and road clustering

If a point can be found into a bounding box then it is clustered into the specific way segment. A cleaning process is used in order to fix mis clustered points as described in section4.2.3.

Figure 4.10 presents full matched points to segments.

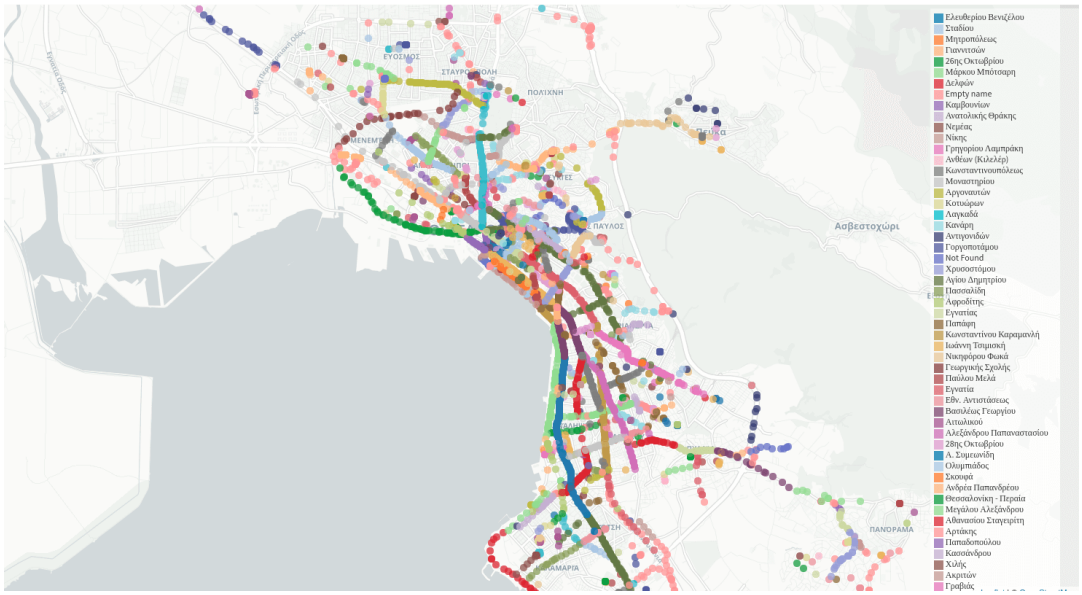

Figure 4.10: Matched points to roads

4.2.3 Cleaning map matched points

In this subsection we discuss the routine which was created in order to clean the matched points and increase the correct matched points into ways. To achieve this we have used as data input the matched locations of nodes and ways and we have rematched the near nodes to their ways. In algorithm 4.2 this is presented in detail. The whole procedure was used because as we can understand from figure 4.11 only the 53% of the dataset have clustered as ways.

Figure 4.11: Way and nodes ratio

Algorithm 4.2: Matched points on ways

Data: Matched points on nodes and ways
Result: Matched points on ways

```
1 while not at end of trajectories do
      if point.type=node and previousPoint.type=way then
        if point.id in previousPoint.nodes then
          update(previousPoint,osmWayId)
        else if point.name=previousPoint.name then
          update(previousPoint,osmWayId)
        else
          getNextPairs()
        end if
      else if point.type=way and previousPoint.type=node then
        if previousPoint.id in point.nodes then
          update(previousPoint,osmWayId)
        else if previousPoint.name=point.name then
          update(previousPoint,osmWayId)
        else
          getNextPairs()
        end if
      end if
2 end
```

For better understanding figure 4.12 shows an example of a trajectory on road "Egnatia". As the figure shows there are points matched into way segments 1458749, 1458746, but there are also three points 1258320,1458300,1258721 that have been clustered as nodes. Using the algorithm 4.2 and the trajectory the green point with id #1258320 will reclustered into way #1458749. The same procedure will cluster the red node into way #1458749. If a point cant be clustered on a previous way the next way used as cluster. In this example if the yellow node is in 10m radius from the #1458746 will clustered there as well. After this routine the reclustered dataset contains 73% of ways and 27% of nodes.

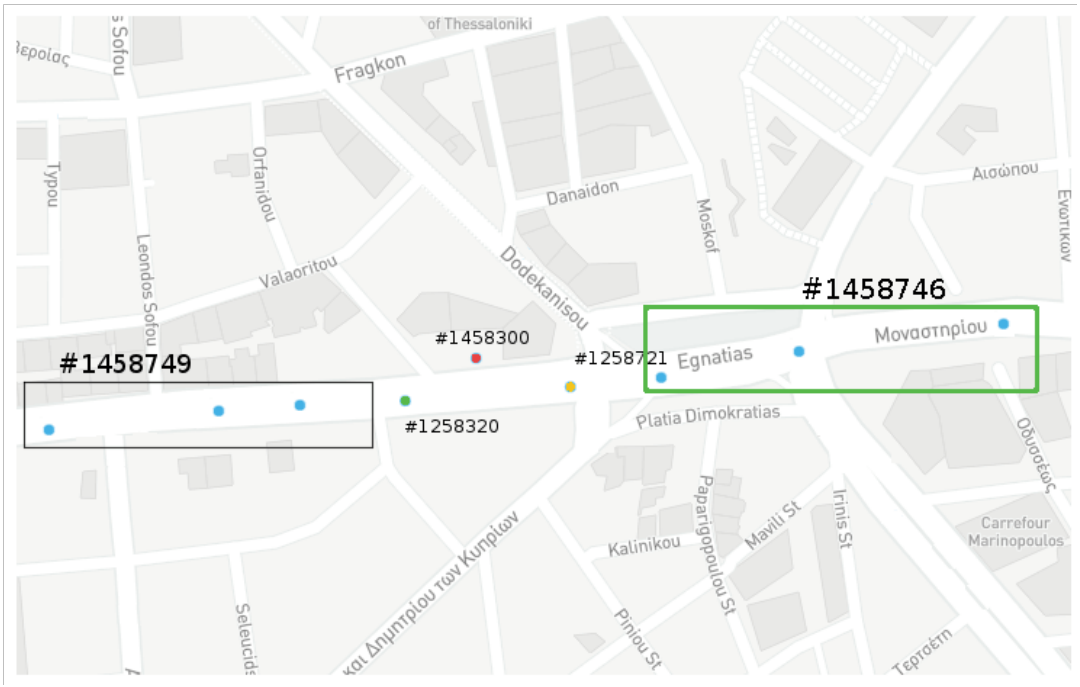

Figure 4.12: Node and ways representation on road segment

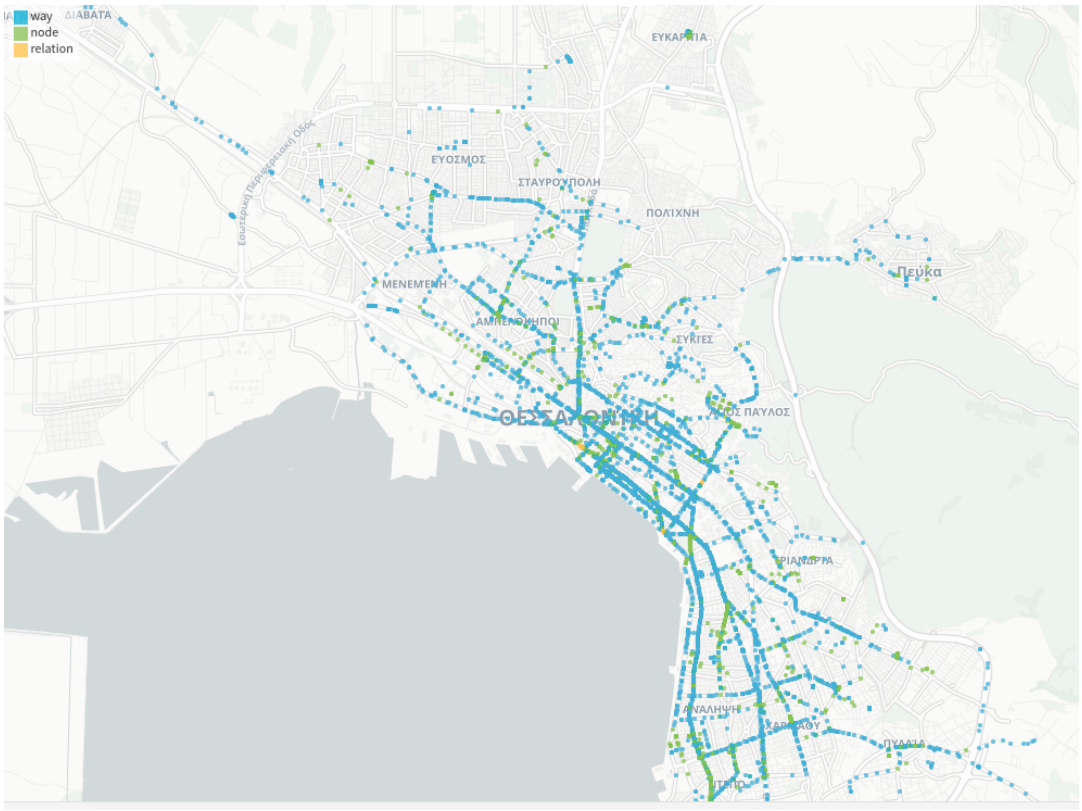

Figure 4.13: Matched points on nodes and ways

In figure 4.13 we can see a subsample of the whole dataset representing the matched points. The full matched points are noted in blue color while the points matched as nodes are noted in green. Analogically the 79% of the dataset is full matched to ways and the 21% is matched to nodes.

4.3 Time series aggregation

As it is described in section 3.3 datasets have been aggregated on time in order to find the best size of time window and downscale the amount of information that have to be processed on the prediction phase. To achieve this we used an aggregation window of 5,10,15,20,25,30,45,60 minutes respectively. At this point it is deemed to crucial to clarify that there is no overlap between the sliding windows.

The whole process computes the time windows using the time and the space dimension. As it is described the time aggregation has been done for each distinct way segment using only valid ways with minimum amount of cars per window 10 cars. That means that if in a 5 minute window does not present at least 10 cars the whole process reconstructs the initial phase. First of all, the initial start time is defined which is the first observation of a point. After this all segments get the numeric value index of the initialization and the time window slides among all segments. For each slide our approach computes the mean and median, mode speed of observed cars, the numbed of them, as well the minimum and maximum speed. For statistical confirmation also standard deviation has been computed.

For each time window the whole process produces two different collections. The one that includes the way segments with the observed values, and one with the distinct way's, the second one includes information about the statistics that has be observed on each specific segment.

The whole process computes the time segments in parallel mode using multi threads. A new thread starts for each way segment and computes the time segments repetitively. A time id is aligned on each time segment in order to sort the segments for the time series analysis faster.

4.4 Data Model

In this subsection the base model is described as it has been used in order to construct this framework driven by the prediction models. This base model can be divided as follows:

1. Road segments
2. Time segments
3. Weather conditions
4. Statistics

According to this base lines the data model can be found in figure 4.14. As it is described in 3.2 roads have been divided into segments using way attribute from OpenStreetMap. This struct is explained in details in section 5.3. Weather conditions have been extracted from dataset as described in section 5.2. After the struction of the base model, statistics and aggregations methods used in order to finish up the spatio temporal data model as shown in figure 4.14.

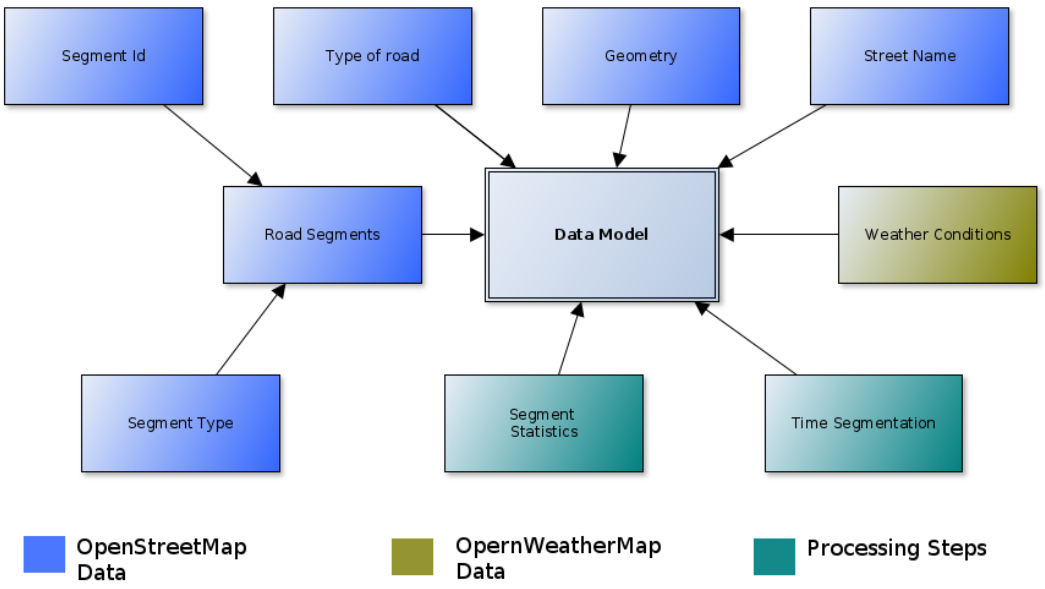

Figure 4.14: Data Model

	FCD	OSM	OSM Metadata	5 min segment	10 min segment
Documents	98.344.433	348.000	60.083	2.596.208	2.086.357
Avg Doc size	435.7 B	193.6 B	531.4 B	386.4 B	386.6 B
Total Doc size	39.9 GB	64.3 MB	30.4 MB	956.6 MB	769.2 MB
Num indexes	10	3	2	2	2
Total Index size	7 GB	13.4 MB	1.4 MB	33.4 MB	26.8 MB

	15 min segment	20 min segment	25 min segment	30 min segment	45 min segment
Documents	1.779.818	1.568.132	1.411.001	1.289.867	1.043.848
Avg Doc size	386.8 B	386.9 B	387.0 B	387.1 B	387.3 B
Total Doc size	656.5 MB	578.6 MB	520.8 MB	476.2 MB	385.6 MB
Num indexes	2	2	2	2	2
Total Index size	22.9 MB	20.1 MB	18.1 MB	16.6 MB	13.4 MB

	60 min segment	5 min segment stats	10 min segment stats	15 min segment stats	20 min segment stats
Documents	891.175	7.226	7.226	7.226	7.226
Avg Doc size	387.5 B	113.0 B	113.0 B	113.0 B	113.0 B
Total Doc size	329.3 MB	797.4 KB	797.4 KB	797.4 KB	797.4 KB
Num indexes	2	2	2	2	2
Total Index size	11.5 MB	116.0 KB	124.0 KB	116.0 KB	124.0 KB

	25 min segment stats	30 min segment stats	40 min segment stats	60 min segment stats	Weather
Documents	7.226	7.226	7.226	7.226	42113
Avg Doc size	113.0 B	113.0 B	113.0 B	113.0 B	330.5 B
Total Doc size	797.4 KB	797.4 KB	797.4 KB	797.4 KB	13.3 MB
Num indexes	2	2	2	2	2
Total Index size	116.0 KB	116.0 KB	124.0 KB	116.0 KB	848.0 KB

Table 4.1: Dataset Model

Table 4.1 presents the whole dataset as it is presented in database. The basic characteristics among others are the number of records per collection and the indexes that have been used in order to implement fast access throw queries. As we can see in the table there is a statistic collection for each time window size that holds values for each segment. Weather dataset, FCD and OSM datasets are presented as different collections, however the OSM dataset has an auxiliary collection with metadata.

4.5 Prediction

The Intelligent Transportation System (ITS) provides data availability which has enabled traffic scientists to develop a number of traffic information prediction methods, including data driven statistical and neural network models. In this section the two main methods developed in our framework are discussed. The two methods developed are namely ARIMA and neural networks, which were used so as to create the most accurate model.

For each method there were free variables that can change the behavior of the predictions driven by the discrimination power between classes and real speed values. These variables are different per supervised learning technique.

4.5.1 Supervised prediction

This section presents the main procedure of the supervised learning as is described the main process splits the problem into regression and classification. In order to create the models for this purposes, two models have been implemented. The model for the regression model is described in figure 4.16 and the classification model is described in figure 4.19. These two models use three types of hidden layers. There have been used LSTM layers (Hochreiter and Schmidhuber, 1997b), dropout (Srivastava et al., 2014) and dense layers. For example on classification there is only one LSTM layer and on regression there are two, one as a sequence of the other. On the other hand, the dropout layers that have been used by the classification model are beside a dense layer and before the last dense layer, and on the regression model the dropout layer exists before the last one. All models were trained using hold-out and cross validation as it is explained in figure 3.8.

Features and encoding

At this prediction stage, the feature of selection process based on Principal component analysis is discussed. As we can see on scree plot in figure 4.15 and at the correlation matrix on table 4.2 there are strong correlations equal to 1 between the temperature, temperature min and temperature max thereat only temperature has been used in order to predict using either classification or regression as the values of other variables where the same.

Correlation	day_of_week	time_time	temp	temp_min	temp_max	icon	clouds	humidity	rid
day_of_week	1	-0.003	-0.084	-0.084	-0.084	-0.122	-0.183	0.157	0.065
time_time	-0.003	1	0.292	0.292	0.292	0.064	0.085	-0.223	-0.07
temp	-0.084	0.292	1	**1**	**1**	-0.056	-0.012	-0.651	0.083
temp_min	-0.084	0.292	**1**	1	**1**	-0.056	-0.012	-0.651	0.083
temp_max	-0.084	0.292	**1**	**1**	1	-0.056	-0.012	-0.651	0.083
icon	-0.122	0.064	-0.056	-0.056	-0.056	1	0.861	0.05	-0.559
clouds	-0.183	0.085	-0.012	-0.012	-0.012	0.861	1	-0.053	-0.314
humidity	0.157	-0.223	-0.651	-0.651	-0.651	0.05	-0.053	1	-0.126
rid	0.065	-0.07	0.083	0.083	0.083	-0.559	-0.314	-0.126	1

Table 4.2: Correlation matrix

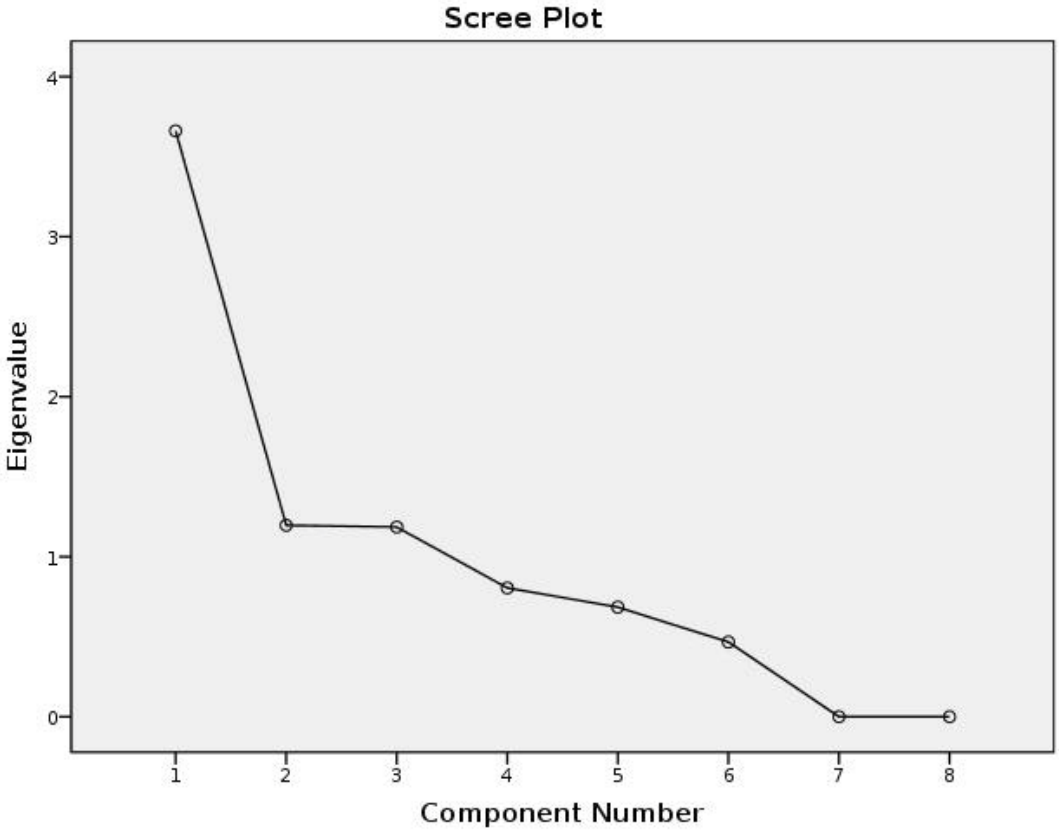

Figure 4.15: Scree plot using all variables

As we can understand from the scree plot the variables on position 7 and 8 can be omitted as they provide no additional information. These variables are the minimum and maximum temperature with correlation equal to 1, as seen on table 4.2.

The used features have been transformed in order to increase the discrimination power of the classifiers using min max scalers and label encoders. The dataset was divided into train and test sets with respect on the time series structure for the cross validation step and the plots has been used for visualization purposes.

In table 4.3 we can see an example that has been used for our prediction models for five observations using the osm data model, the observed points and the weather information about the specific time stamps .

Day of week	Time of obs	Temp	Temp Min	Temp Max	Aggregation Icon	Humidity	Clouds	Desc	Main Desc	Rid	Pressure	Prediction Value
4	0	19	19	19	1	77	0	Clear	Sky is clear	800	1017	12
4	0	19	19	19	1	82	0	Clear	Sky is clear	800	1015	10
4	0	19	19	19	1	80	0	Clear	Sky is clear	800	1010	15
4	0	19	19	19	1	75	0	Clear	Sky is clear	800	1015	10
4	0	19	19	19	1	73	0	Clear	Sky is clear	800	1014	20

Table 4.3: Time series model features

The strong correlated values were removed and the final dataset looks similar with the table 4.4. These values were used in order to train our models and create predictions based on past observations.

Day of week	Time of obs	Temp	Aggregation Icon	Humidity	Clouds	Description	Main Description	Rid	Pressure	Prediction Value
4	0	19	1	77	0	Clear	Sky is clear	800	1017	12
4	0	19	1	82	0	Clear	Sky is clear	800	1015	10
4	0	19	1	80	0	Clear	Sky is clear	800	1010	15
4	0	19	1	75	0	Clear	Sky is clear	800	1015	10
4	0	19	1	73	0	Clear	Sky is clear	800	1014	20

Table 4.4: Time series model features without very strong correlations

As it is described in section 3.4.1 the whole dataset was transformed from time series observations into supervised dataset on the phase of prediction using the look back technique. The look back variable takes values, such as time in minutes and uses the previous observations in order to build the essential dataset. In table 4.5 we can see a sample table using only the temperature as the main feature with look back equals to one.

Day of week (t)	Time of obs (t-1)	Temp (t-1)	Desc (t-1)	Prediction Value (t-1)	Day f week (t)	Time of obs (t)	Temp (t)	Desc (t)	Prediction Value (t)
4	0	18	Clear	13	4	0	17	Clear	12
4	0	17	Clear	15	4	0	18	Clear	10
4	0	17	Clear	12	4	0	16	Clear	15
4	0	18	Clear	17	4	0	17	Clear	10
4	0	19	Clear	20	4	0	19	Clear	20

Table 4.5: Supervised learning model Features

A lot of differentiations have been applied in order to find the best model. Some of them can be described as follows:

1. Different time window size.
2. Different numbers of learning rate.
3. Different data model.
4. Different train-test sets.
5. Different connections (full or non-full) between the processing nodes.
6. Different number of hidden nodes.
7. Different transfer functions.

8. Different error back-propagation schemes.

For the feature extraction has been used a lot of different combinations. The characteristics that have been used can be explored with more details from the section 5.5 as it is plotted values for descriptive statistical purposes.

Regression

The regression process is used in order to predict real speed values on segments. In this phase real values of speed using GPS signals for each segment were used. This means that for each segment of road we observe the speed for each car regardless of the id of the car. We create the mean speed using the time series aggregation and these values is used in order to create predictions. The prediction model is presented in figure 4.16. As described there are two LSTM layer in sequence, a dropout layer connected on a dense layer with linear activation and the output layer that predict the real speed on segments.

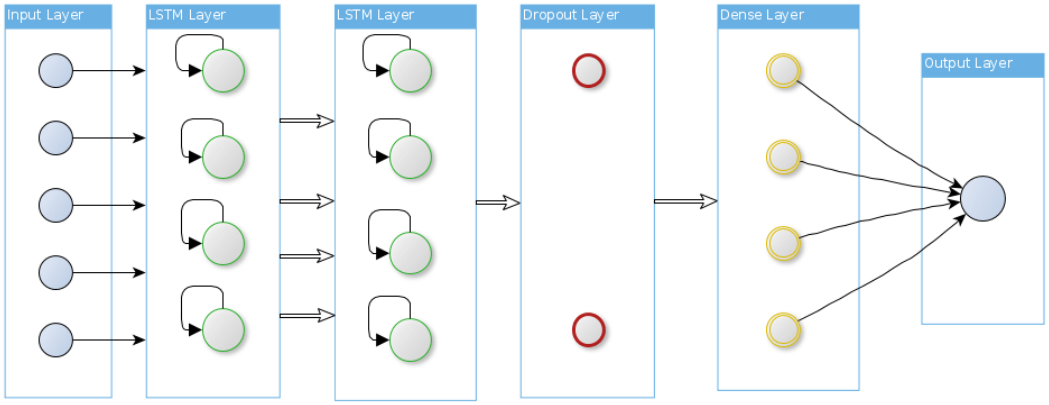

Figure 4.16: Neural network regression model

Figure 4.17 shows the prediction of mean speed using the regression model of an LSTM neural network with 32 neurons using the hold out evaluation, with blue color is plotted the real speed values, with orange color the speed on train phase and with green is plotted the prediction speed for the 30% of the dataset. In figure 4.18 the leaning curve among epochs on the same level is described

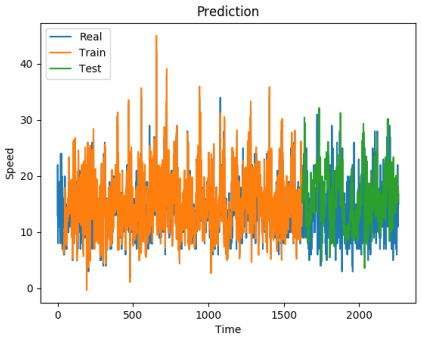

Figure 4.17: Speed prediction using hold-out

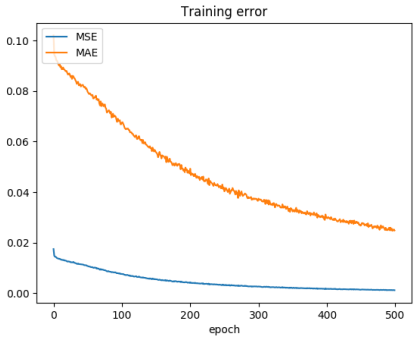

Figure 4.18: Learning curve using hold-out

Section 6.2 portrays each model with the test error rate per model and the predicted values using hold-out and cross validation. As it was computational expensive there have been only some approaches in order to find the best model.

Classification

The classification model uses five hidden layers. The first one is an LSTM layer with several different approaches on the number of neurons. On the third layer exists a dense layer with half number of neurons, and on the last layer there is an dense layer with exact number of neurons as the first layer. The red layers represent the dropout layers between the other layers. The activation function used on dense layers was rectified linear unit (ReLU). As the prediction classes is three a softmax activation used in the last layer with three neurons. The Adam optimizer used for each model (Kingma and Ba, 2014) as optimizer. The final model is described in figure 4.19

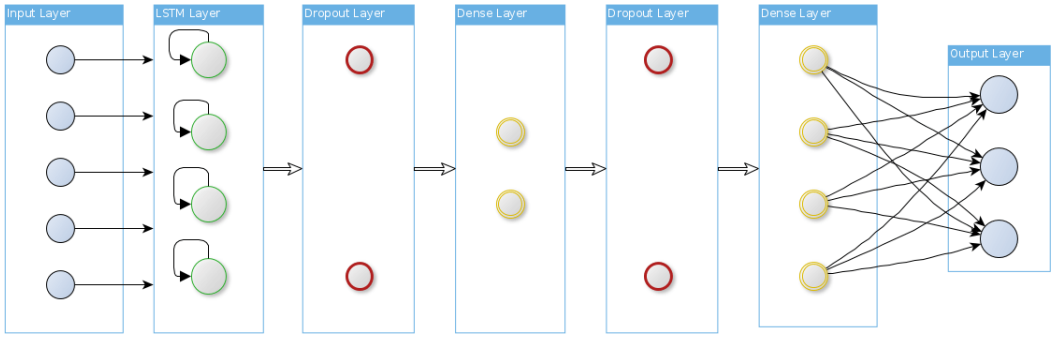

Figure 4.19: Neural network classification model

As (Yanminsun, Wong, and Kamel, 2011) explained on a review about class imbalanced datasets there are several problems using unbalanced observations. The most significant problem using such datasets is the big errors on prediction process. There has been a lot of research in order to avoid such accuracy problems to adjust the class distribution of a data set (i.e. the ratio between the different classes/categories represented). In this thesis Synthetic Minority Over-sampling Technique (SMOTE) was used as is described in (Jia, Wu, and Xu, 2017). This process uses the number of classes of minority and majority class in order to smooth the number of samples trying to balance the dataset on same number of observations per class. As we can notice in figure 4.20 the accuracy of the model is on the range of (75-83) % as the learning curve is beside the range (0.43-0.38). Respectively in figure 4.22 and in figure 4.23 we can observe the same prediction model using balanced dataset after the SMOTE procedure.

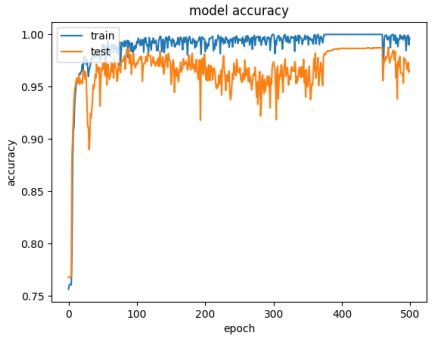

Figure 4.20: LSTM model classification accuracy with unbalance dataset

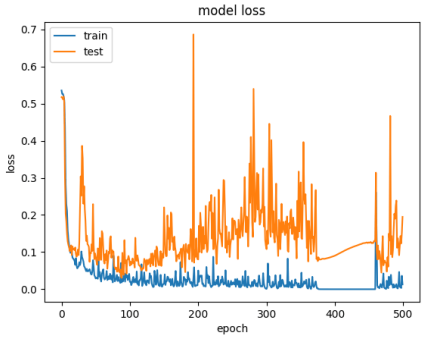

Figure 4.21: LSTM model classification learning curve with unbalance dataset

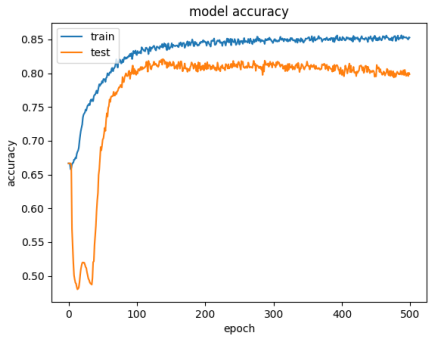

Figure 4.22: LSTM model classification accuracy with balanced dataset

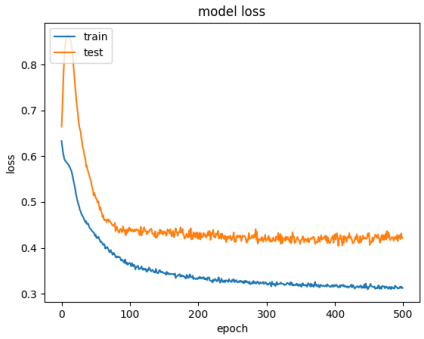

Figure 4.23: LSTM model classification learning curve with balanced dataset

Section 6.1 outlines each model with the test error rate per model and the accuracy per class on the phase of train and test using hold-out and k-fold cross validation. As it was computational expensive there have been only some approaches in order to find the best model.

4.5.2 Statistical prediction using ARIMA

Each of these components is explicitly specified in the model as a parameter. A standard notation is used of ARIMA(p,d,q) where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used.

The parameters of the ARIMA model are defined as follows:

p: The number of lag observations included in the model, also called the lag order.

d: The number of times that the raw observations are differentiated, also called the degree of differencing.

q: The size of the moving average window, also called the order of moving average. A linear regression model is constructed including the specified number and type of terms, and the data is prepared by a degree of differencing in order

to make it stationary, i.e. to remove trend and seasonal structures that negatively affect the regression model.

A value of 0 can be used for a parameter, which indicates to not use that element of the model. This way, the ARIMA model can be configured to perform the function of an ARMA model, and even a simple AR, I, or MA model.

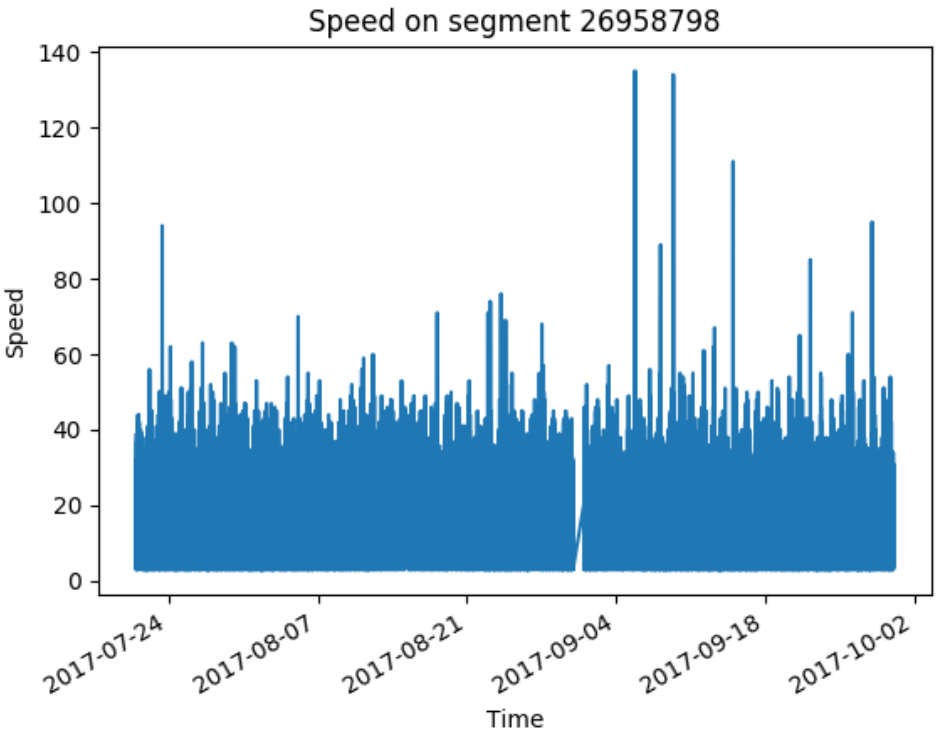

Figure 4.24: Speed observations on segment

The target is to predict the speed of figure 4.24. As described above there were different tests in order to find the best model for the prediction phase. The ARIMA model can be described with different values on variables (p,d,q) using different methods, such as Css-mle, mle and css where “css-mle”, the conditional sum of squares likelihood is maximized and its values are used as starting values for the computation of the exact likelihood via the Kalman filter. If “mle”, the exact likelihood is maximized via the Kalman Filter. If “css” the conditional sum of squares likelihood is maximized. The models that have been tested can be found in table 4.6:

ARIMA(1,0,0)	ARIMA(4,0,0)	ARIMA(0,0,2)
ARIMA(2,0,0)	ARIMA(5,0,0)	ARIMA(0,0,3)
ARIMA(3,0,0)	ARIMA(0,0,1)	ARIMA(0,0,4)
ARIMA(15,0,0)	ARIMA(0,0,15)	

Table 4.6: ARIMA models

For each model the error rate of the predicted observations visualized as real values a line plot of the residual errors, suggesting that there may still be some

trend information not captured by the model and as a density plot of the residual error values, suggesting the errors are Gaussian.

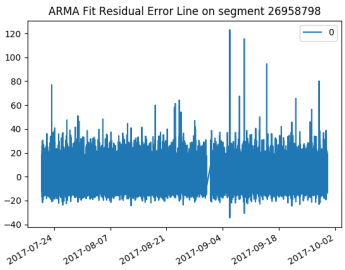

Figure 4.25: ARIMA fit residual line

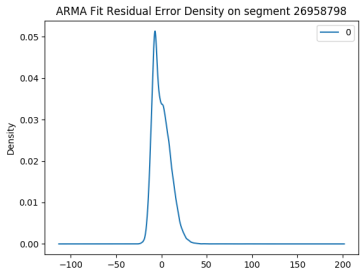

Figure 4.26: ARIMA residual density

For each model an autocorrelation figure describes the correlation as figure 4.27 explains.

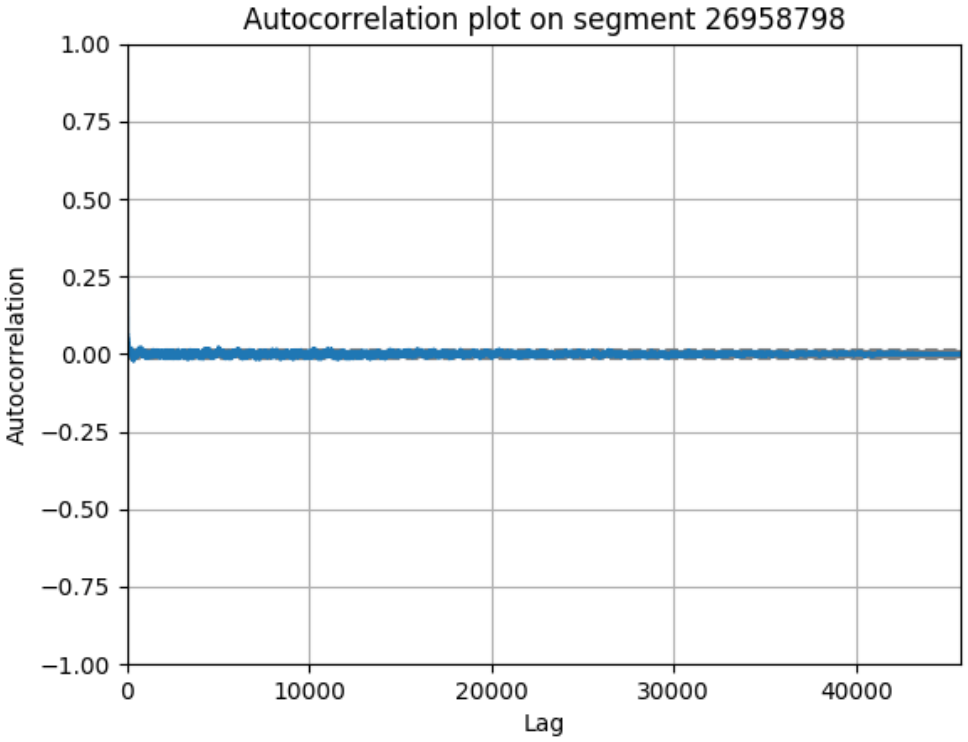

Figure 4.27: Autocorrelation plot using ARIMA (5,0,0)

After the prediction of each segment the prediction values were described as error rate and predicted values and have been visualized as prediction values as figure 4.28 shows.

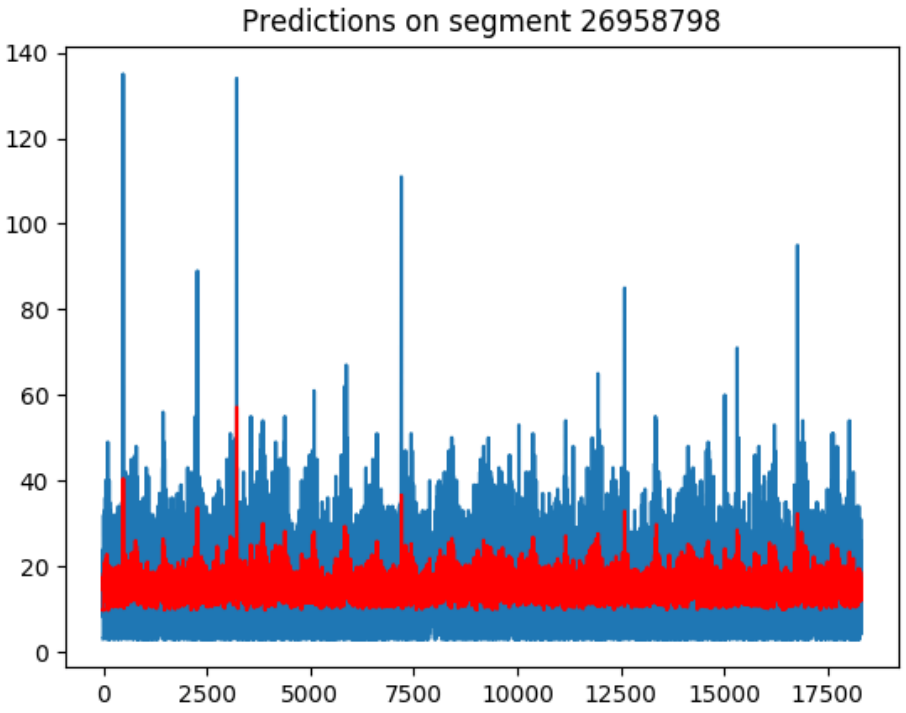

Figure 4.28: Speed predictions on segment using ARIMA (5,0,0)

As we can understand from autocorrelation in figure 4.28 and from the prediction figure 4.28 our prediction model seems that there may still be some trend information not captured by the model.

In 6.1 each model with test error rate per model is described. As it was computational expensive there have been only some approaches in order to find the best prediction model.

4.6 Testing environment

This subsection discusses the environment used by our framework to pre-process the datasets, create the data models, discriminate the time windows, train and test our prediction models and evaluate the results. As it was discussed earlier all phases were designed as a parallel architecture to make the whole process scalable. For this purpose threads and clever queries with indexes in database have been used.

Here are the specifications of our infrastructure both in hardware and in software design.

- Operating System:Linux 4.10.0-42-generic #46 16.04.1-Ubuntu x64 GNU/Linux
- CPU: Intel(R) Xeon(R) CPU E3-1240 v3 @ 3.40GHz, 4 cores, 8 threads
- RAM: 2 x DIMM DDR3 Synchronous 1600 MHz (0,6 ns) 4GB
- Hard Drive: Kingston SSDNow V300 120GB
- Databases: MongoDB 3.4.10 Community Edition, PostgreSQL 9.5.10
- APIs: Nominatim, osmAPI, OSRM
- Programming language, frameworks and libraries: Python 3 (Pandas, keras, sklearn, tensorflow, imbalanced-learn, matplotlib, numpy, pymongo, math, statsmodels, bson, urllib2, urllib3, json, geopy, shapely, threading, Queue, osrm), MongoDB, SPSS, Dataiku, Mongodb Compass, LaTeX

The trend to use NoSQL databases in place of relational databases has increased in certain use cases. This, as requirements of data models can change frequently. In 2013, Lu´ıs A. Basti~ao Silva et al explain that document-based databases do not have the limitation of RDBMS databases (Bastião et al., 2014). They demonstrated that Lucene, MongoDB and CoucheDB, show high performance levels compared to typical SQL databases. In our system we used MongoDB for storing our dataset and postgreSQL to store the dataset of OpenStreetMap using several indexes based on geographical objects.

Conclusions

As this chapter explains the implementation of this framework has be done using several technologies. The whole framework was created in a abstraction level and can be modified in order to serve multiple requests as an online prediction framework.

Chapter 5

Dataset understanding

Before the presentation of the results extracted as figure 3.1 shows based on the framework methodology it is necessary to describe the data included in our data set. This chapter presents in detail the values included in the data model and explains the way each dataset is presumed to affect the traffic prediction models.

5.1 FCD Dataset

In the following section the dataset, which has been used as the main data source to make predictions, is discussed. The dataset is a 98.3 millions comma separated values file which have been given via CERTH-HIT (Center for Research and Technology Hellas - Hellenic Institute of Transport) and contains information about TAXI floating car in the city of Thessaloniki. The whole dataset is about 12.7GB as raw data and 41.2 GB on mongoDB database as geojson format and contains information about the floating cars. In particular the dataset is separated by line and each line contains the speed, the orientation the altitude the location of the floating car with longitude and latitude and the timestamp of the observation .

The specific dataset that we process and evaluate our approach contains data from 1 July 20017 to 30 Sept 2017. The amount of records is about 102.500 million points. As it is discussed the dataset has been splitted by dates on the phase of machine learning. A simple example of the dataset can be found in table 5.1.

Car id	Timestamp	Longitude	Latitude	Altitude	Speed	Orientation	Zone id
22690	2017-09-29 23:59:59.630	22.97575	40.52456	0	0	5	36
22766	2017-09-29 23:59:59.833	22.96938	40.65935	8	48	273	10
23430	2017-09-29 23:59:59.873	22.95399	40.63538	4	0	132	66
23459	2017-09-29 23:59:59.877	22.95956	40.60470	3	10	43	65
22693	2017-09-29 23:59:59.883	22.94902	40.63336	3	47	303	63
23372	2017-09-29 23:59:59.923	22.95595	40.59709	2	24	32	19
23077	2017-09-29 23:59:59.940	23.03357	40.59144	5	11	33	6
22900	2017-09-29 23:59:59.960	22.92257	40.64074	1	10	149	56
23182	2017-09-29 23:59:59.960	22.94944	40.65536	9	50	196	64
22684	2017-09-29 23:59:59.970	22.94065	40.63594	1	0	137	62

Table 5.1: Sample of FCD dataset

Where car id identifies a unique id for each car. Ids changes on each day for anonymity reasons. longitude, latitude represents the location of the cars in World Geodetic System (WGS 84) format. Altitude is in meters from the sea. Orientation is in radius from north and the zone id represents the region the taxi is locating at.

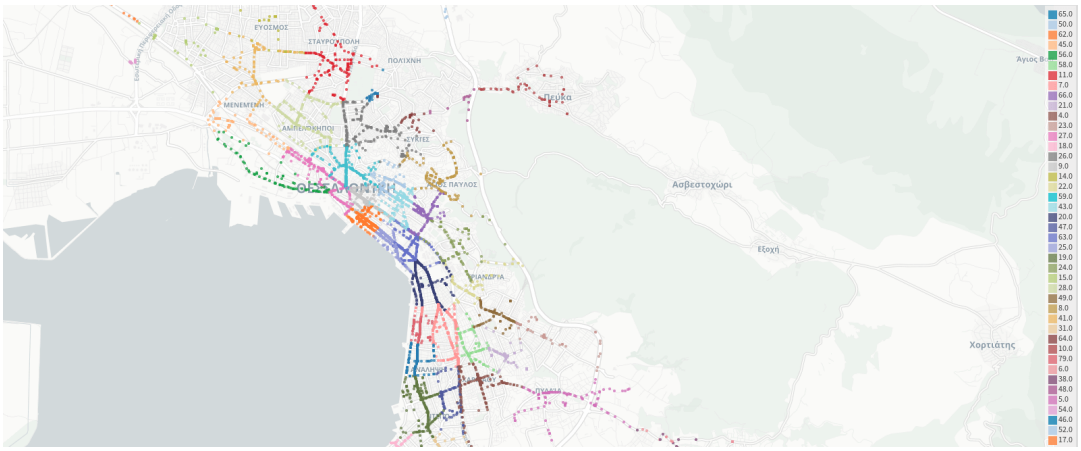

Figure 5.1: Matched points in zones

In figure 5.1 are plotted the points clustered into zones defined by the provider. We can easily figure that the zones are regions around the urban area of Thessaloniki and can be used for better map matching.

5.1.1 Data pre-processing

As the dataset represents information from the real world we faced some problems with the GPS signals and the real location of the floating cars. To clear non valid tuples of coordinates the tuples has been validated the tuples to obey the range of longitude and latitude as follows.

Longitude		**Latitude**		
Min	**Max**	**Min**	**Max**	**Non valid**
-180	180	-90	90	0

Table 5.2: Valid coordinates

In order to avoid data from stationary taxis influencing the analysis we have implement a routine in order to clean the dataset from stopped points on taxi stands. To make this possible we extracted the taxi stands from the official GIS of the municipality of Thessaloniki using the necessary polygons to clean the points based on the observations that its not on these polygons if they are stopped. After the basic cleaning of the dataset the whole amount was decreased to 98.344 millions points and the size of the set is 39.9 GB.

Algorithm 5.1 shows the procedure that has been used in order to clean the dataset from stopped cars on taxi stations.

Algorithm 5.1: Clean points of stopped taxi

```
Data: All data points
Result: Cleaned dataset from stopped taxis
    getNonValidPolygons()
    for all non valid polygons do
      while not at end of points do
        getPoint()
        if polygon.contains(point) then
          point.delete()
        else
          continue
        end if
      end while
    end for
```

5.2 Weather Dataset

In this subsection the weather dataset, that has been used in order to understand the impact of the climate on the congestion prediction and create better predictions, is discussed.

The historical dataset that is used is json file from OpenWeatherMap (OWM) that contains hourly information about the current weather conditions in the city of Thessaloniki. The whole dataset contains information from 1-Oct-2012 until 1-Oct-2017. For the purpose of this thesis we have used a subset from 1 July 20017 to 30 Sept 2017. A simple example of the dataset can be found in table 5.3

Timestamp	Min Temp	Max Temp	Temp	Clouds	Wind speed	Main Description	Desc	Pressure	rid	Humidity	Wind degree	True icon
2017-09-01T00:00	268.15	268.15	268.15	0	2	Sky is Clear	Clear	1017	800	77	110	01n
2017-09-01T01:00	264.863	264.863	264.863	0	2	Sky is Clear	Clear	1016	800	82	100	01n
2017-09-01T02:00	268.15	268.15	268.15	0	2	Sky is Clear	Clear	1016	800	82	110	01n
2017-09-01T03:00	268.15	268.15	268.15	0	2	Sky is Clear	Clear	1016	800	77	100	01n
2017-09-01T04:00	268.15	268.15	268.15	0	1	Sky is Clear	Clear	1016	800	77	120	01n
2017-09-01T05:00	268.15	268.15	268.15	0	1	Sky is Clear	Clear	1016	800	77	0	01d
2017-09-01T06:00	267.15	267.15	267.15	0	1	Sky is Clear	Clear	1016	800	64	0	01d
2017-09-01T07:00	268.15	268.15	268.15	0	1	Sky is Clear	Clear	1015	800	56	57	01d
2017-09-01T08:00	268.15	268.15	268.15	0	2	Sky is Clear	Clear	1015	800	60	300	01d
2017-09-01T09:00	269.15	269.15	269.15	0	2	Sky is Clear	Clear	1015	800	57	290	01d
2017-09-01T10:00	269.15	269.15	269.15	0	3	Sky is Clear	Clear	1015	800	53	300	01d
2017-09-01T11:00	269.15	269.15	269.15	0	3	Sky is Clear	Clear	1014	800	47	300	01d
2017-09-01T12:00	270.15	270.15	270.15	0	4	Sky is Clear	Clear	1013	800	57	310	01d
2017-09-01T13:00	269.15	269.15	269.15	0	4	Sky is Clear	Clear	1013	800	39	190	01d

Table 5.3: OWM weather dataset

As we can see on the sample table there are attributes that can clarify the weather conditions per hour using the minimum, the maximum as well as the current temperature in kelvin scale, counter of clouds, sum metrics for the wind and the orientation in degrees, a summary description and a main description. Among others we have the humidity an rid integer and the true icon. The explanation of true icon can be found in figure 5.2. On the other side the rid can be found in table 5.4 and its explain the weather condition using a well defined

code system. There are several tables about the code system for each weather condition and can be found on OWM condition explanation

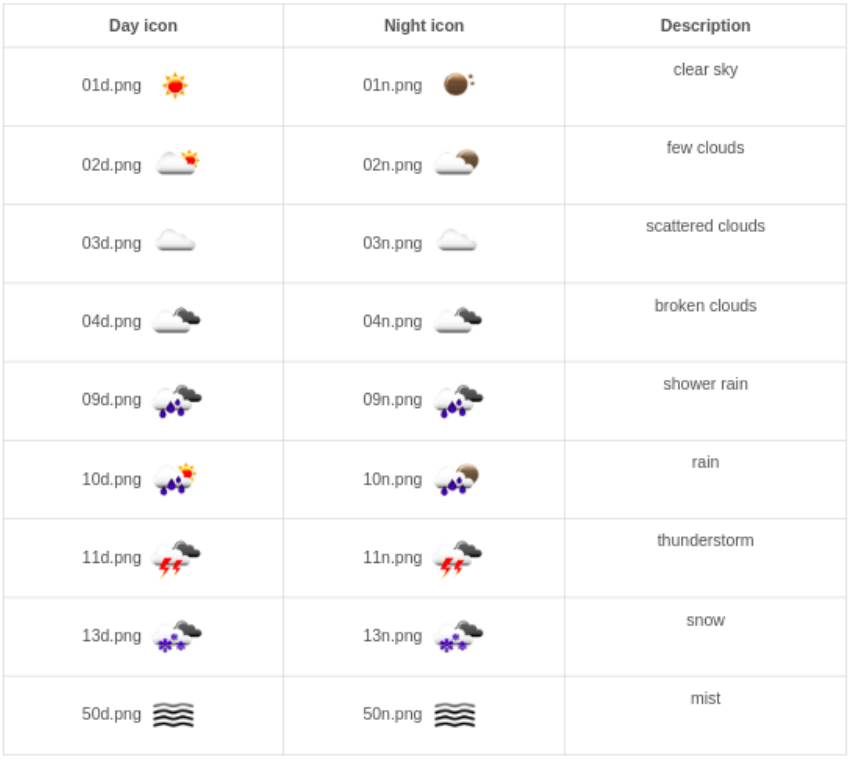

Figure 5.2: OWM weather icons

ID	Meaning	Icon
200	thunderstorm with light rain	11d
201	thunderstorm with rain	11d
202	thunderstorm with heavy rain	11d
210	light thunderstorm	11d
211	thunderstorm	11d
212	heavy thunderstorm	11d
221	ragged thunderstorm	11d
230	thunderstorm with light drizzle	11d
231	thunderstorm with drizzle	11d
232	thunderstorm with heavy drizzle	11d

Table 5.4: Grouping OWM icons by condition

As we can observe in the grouping table OWM have clustered the weather condition with more accurate definition using the rid attribute. The true icon 11d can appear in several tensions for the condition thunderstorm. The same transformation applies on each icon and can be found on the code system of OWM.

5.2.1 Weather dataset pre-processing

In this subsection the transformation that has been done on the dataset is discussed. For visualization purposes from kelvin to celsius scale and aggregation

routines has been used on the icon symbol in order to transform it into an integer as the true icon contains information about the time using the (n,d) symbols we have remove this letter as we have this information from the timestamp attribute. Table 5.5 shows the main differences from the original dataset.

Timestamp	Min Temp	Max Temp	Temp	Clouds	Wind speed	Main Description	Desc	Pressure	rid	Humidity	Wind degree	True icon	Aggr icon
2017-09-01T00:00	19	19	19	0	2	Sky is Clear	Clear	1017	800	77	110	01n	1
2017-09-01T01:00	18	18	18	0	2	Sky is Clear	Clear	1016	800	82	100	01n	1
2017-09-01T02:00	17	17	17	0	2	Sky is Clear	Clear	1016	800	82	110	01n	1
2017-09-01T03:00	18	18	18	0	2	Sky is Clear	Clear	1016	800	77	100	01n	1
2017-09-01T04:00	18	18	18	0	1	Sky is Clear	Clear	1016	800	77	120	01n	1
2017-09-01T05:00	18	18	18	0	1	Sky is Clear	Clear	1016	800	77	0	01d	1
2017-09-01T06:00	21	21	21	0	1	Sky is Clear	Clear	1016	800	64	0	01d	1
2017-09-01T07:00	24	24	24	0	1	Sky is Clear	Clear	1015	800	56	57	01d	1
2017-09-01T08:00	24	24	24	0	2	Sky is Clear	Clear	1015	800	60	300	01d	1
2017-09-01T09:00	25	25	25	20	5	few clouds	Clouds	1011	801	35	170	02d	2
2017-09-01T10:00	26	26	26	20	5	few clouds	Clouds	1010	801	37	170	02d	2
2017-09-01T11:00	27	27	27	0	3	Sky is Clear	Clear	1014	800	47	300	01d	1
2017-09-01T12:00	27	27	27	0	4	Sky is Clear	Clear	1013	800	57	310	01d	1
2017-09-01T13:00	30	30	30	0	4	Sky is Clear	Clear	1013	800	39	190	01d	1

Table 5.5: OWM weather aggregated dataset

5.3 OSM Dataset

OSM Dataset is described as the main OpenStreetMap information and the OSM meta data. As described in figure 3.4 using the base model there are two main types included into the dataset, nodes and ways. For each instance exists a dataset and a meta data collection.

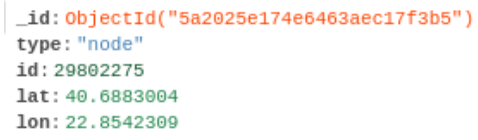

Figure 5.3: OSM Node data

Figure 5.4: OSM Node meta data

As we can see in figure 5.3 exists information about the Node id and the coordinates (longitude, latitude) on WGS84 system point. In figure 5.4 shows the meta data of an other example point. Among others contains information about the type of the specific node, which is a traffic light.

Figure 5.5: OSM Way data

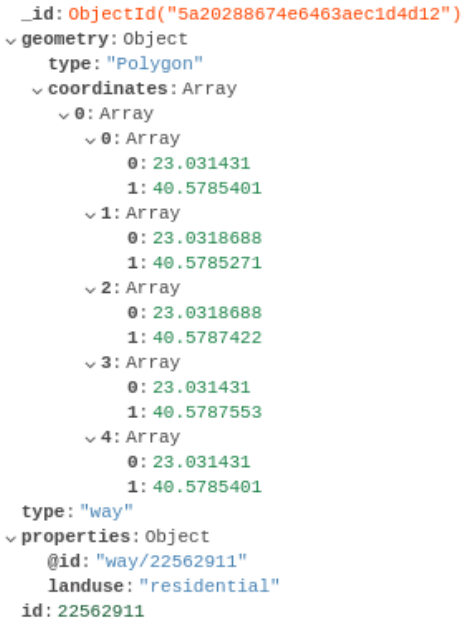

Figure 5.6: OSM Way meta data

Figure 5.5 contains information about a specific way, such as road name, way type, etc. It also contains information about the node that is composed as node ids. On tag property information is stored about the name of the way and the property of oneway which describes if the road has two or more lines per direction. In figure 5.6 we can see the meta data of a way. The meta data of the ways have been used to detect the Polygon that represents the way.

5.4 Data Visualization

For visualization purposes several tools has been tested, among other Tableau and elasticsearch using kibana. For compatibility reasons we selected MongoDB Compass and the Dataiku platform selected.

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size
fcd_cars	98,344,433	435.7 B	39.9 GB	10	7.0 GB
osm_data	348,000	193.6 B	64.3 MB	3	13.4 MB
osm_data_meta	60,083	531.4 B	30.4 MB	2	1.4 MB
paths	168	2.3 KB	391.5 KB	3	48.0 KB
paths_data	478,800	74.9 B	34.2 MB	2	6.4 MB
time_segments_10	2,086,357	386.6 B	769.2 MB	2	26.8 MB
time_segments_15	1,779,818	386.8 B	656.5 MB	2	22.9 MB
time_segments_20	1,568,132	386.9 B	578.6 MB	2	20.1 MB
time_segments_25	1,411,001	387.0 B	520.8 MB	2	18.1 MB
time_segments_30	1,289,867	387.1 B	476.2 MB	2	16.6 MB
time_segments_45	1,043,848	387.3 B	385.6 MB	2	13.4 MB
time_segments_5	2,596,208	386.4 B	956.6 MB	2	33.4 MB
time_segments_60	891,175	387.5 B	329.3 MB	2	11.5 MB
time_segments_stats_10	7,226	113.0 B	797.4 KB	1	132.0 KB
time_segments_stats_15	7,226	113.0 B	797.4 KB	1	116.0 KB
time_segments_stats_20	7,226	113.0 B	797.4 KB	1	124.0 KB
time_segments_stats_25	7,226	113.0 B	797.4 KB	1	116.0 KB
time_segments_stats_30	7,226	113.0 B	797.4 KB	1	116.0 KB

Figure 5.7: MongoDB Compass example

Mongodb Compass is used in order to analyze the documents and display rich structures within the collections, which allows to quickly visualize and explore the schema, to understand the frequency, types and ranges of fields in the data sets. Among others the results can be displayed both graphically and as sets of JSON documents using geographical queries.

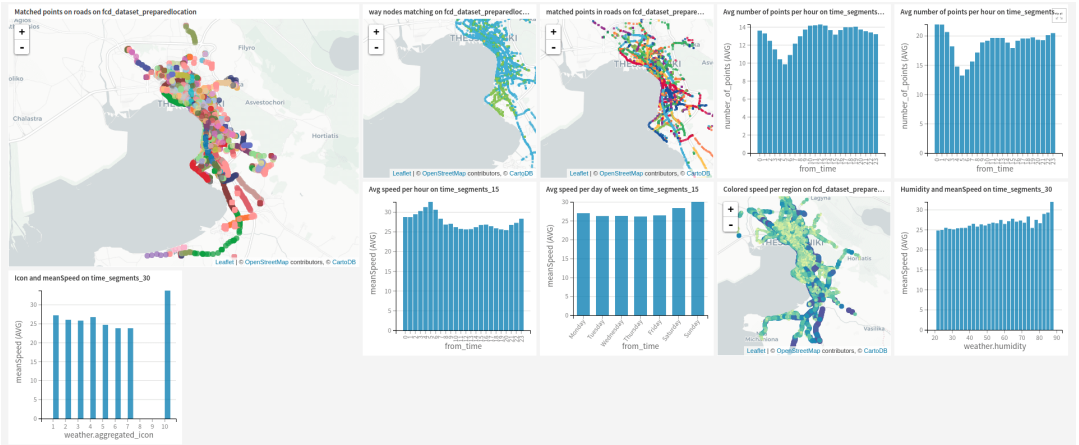

Figure 5.8: Dataiku example

Dataiku DSS is the collaborative data science software platform for data scientists, data analysts, and engineers to explore, prototype, build, and deliver their own data products. Among others dataiku is used in order to extract Histogram, maps, scatter plots and boxplots.

5.5 Descriptive statistics

In this section we will discuss the phenomena and the observations that have been studied in order to be used as features to our models for prediction. Moreover, we will thoroughly examine our dataset in order to make an evaluation by using SPSS and Dataiku.

As the figure 5.9 shows, people are using a taxi more often at the weekends and as we can conclude based on the second half of the figure there are few of taxi on road at early hours. Base on this diagram we can assume that the day of the week as a feature can cause changes on the prediction model as the number of taxis that are moving into the urban areas changes by time.

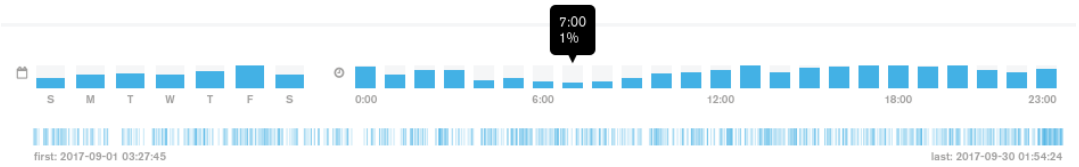

Figure 5.9: FCD recorded timestamps

As we can see in figure 5.10 the most significant region of city with more than the half of routes are near the sea with altitude between 0-5 meters. From this we can understand that most of the citizens are using taxi in order to travel within the city. As this attribute shows unbalanced behaviors, The feature of altitude was not taken into account since correlation between altitude and speed is non-existent.

Figure 5.10: FCD car altitude

Figure 5.11 shows the speed of cars inside the urban area captured as a mean value. Using the colors of the diagram is confirmed that inside the urban area the cars traveling with slow speed and on highways and big roads the speed increases. From this figure it is assumed that each road segment must have a separate threshold as free flow speed.

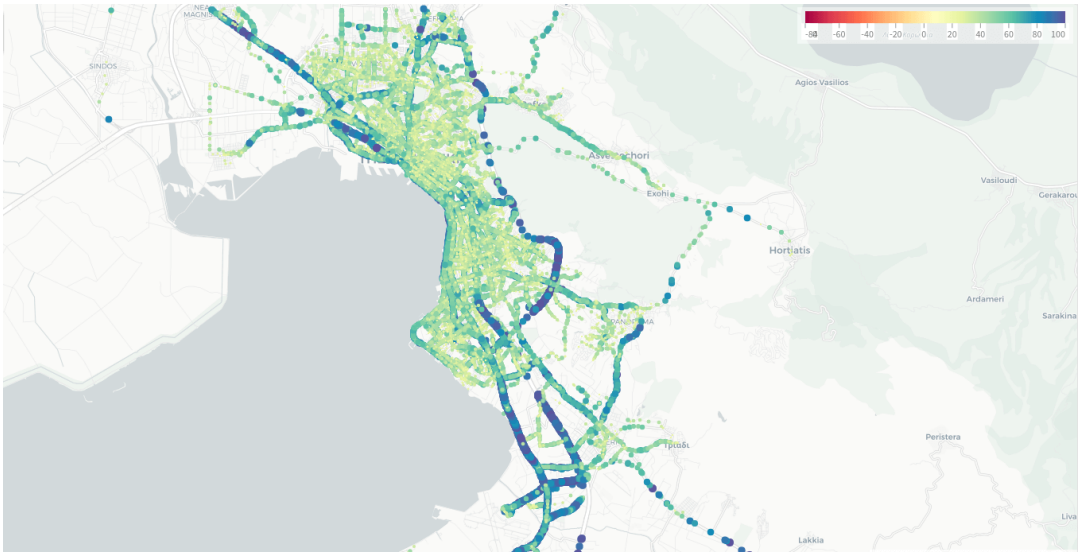

Figure 5.11: FCD car speed per region

Studying figure 5.12 we understand that the day of the week can affect the car speed and the day of the week was used as feature as it increases the discrimination power of prediction models. Taxis travel faster at the weekends and in particular on Sundays. From this figure was decided to use the day of the week as an extra feature on our prediction models.

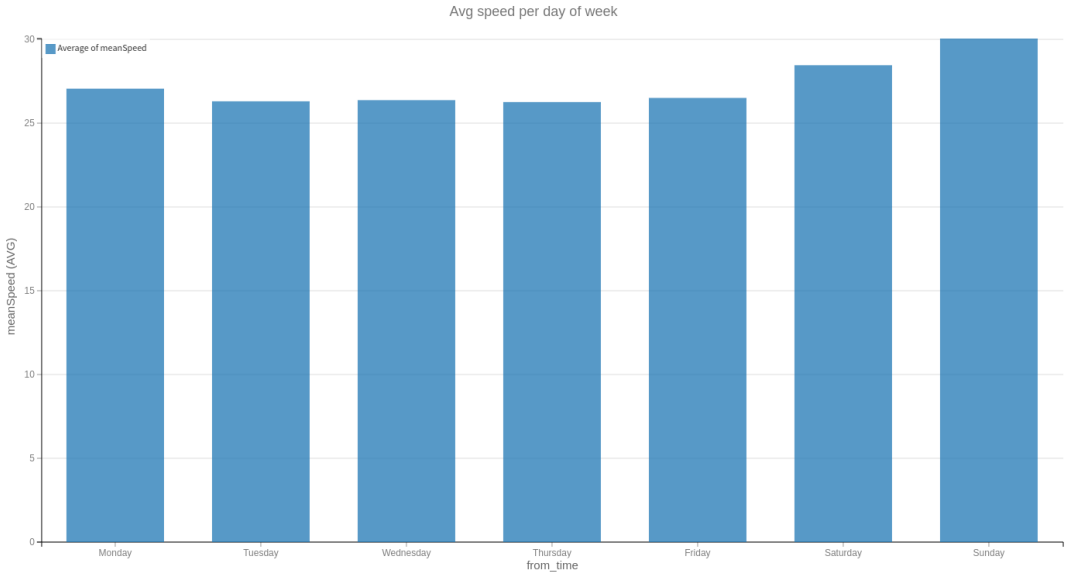

Figure 5.12: FCD average speed per day of week

Figure 5.13 describes the mean average speed per hour of day for all the days of three months. As we can see the traveling speed increases at early hours and the number of cars at this hours are significantly fewer and can confirmed in figure 5.14.

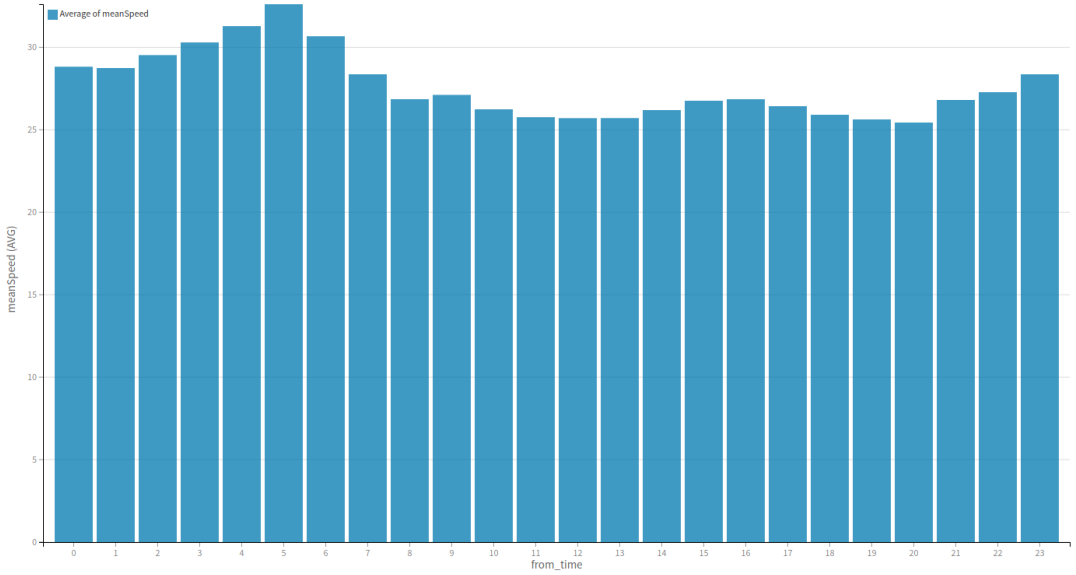

Figure 5.13: FCD average speed per hour

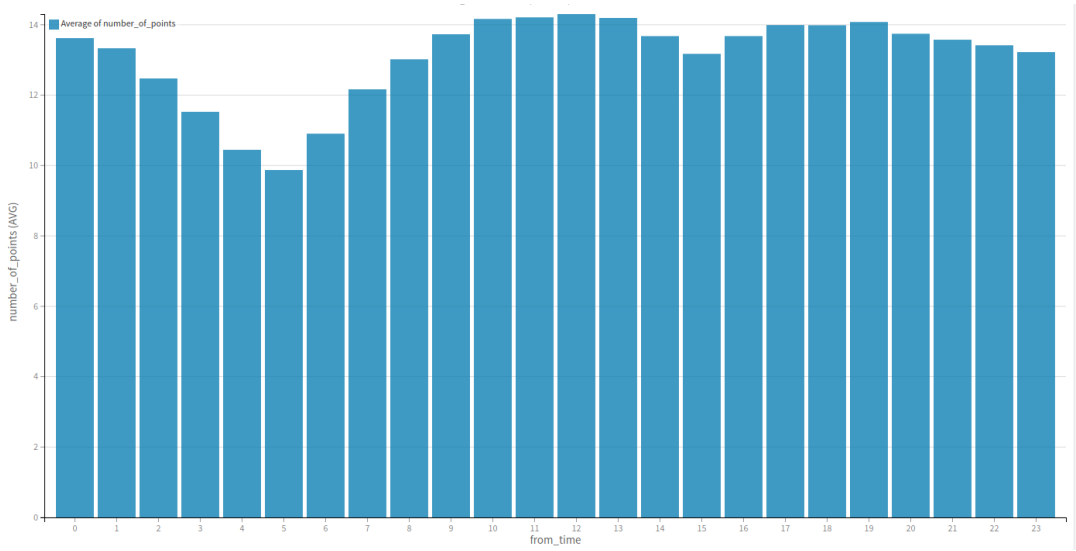

Figure 5.14: FCD average number of cars per hour

Based on the above figures, it can be assumed that the hour of an observation as a feature can lead to a better prediction model. The hypothesis can be confirmed from the correlation matrix in table 4.2.

Figure 5.15 shows the average mean speed using time segments of 30 minutes per way type. As we can see there is massive impact on the average speed per type of road, and overlap between road types.

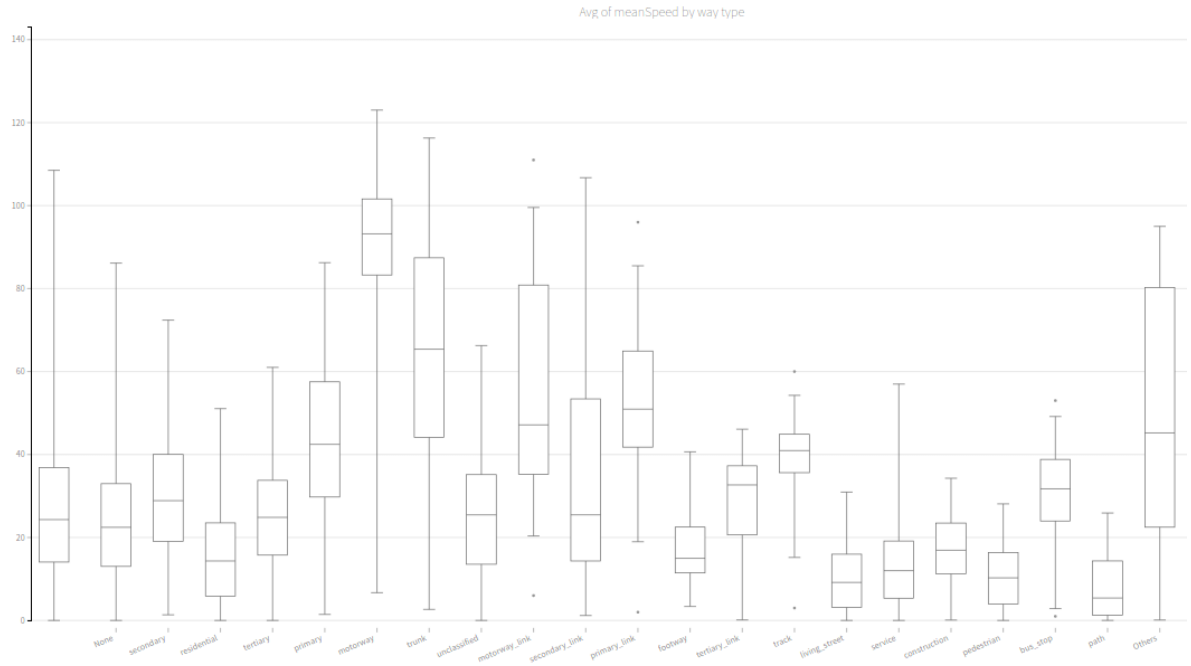

Figure 5.15: Average mean speed per way type using 30 minutes segments

Figure 5.16 shows the average median speed using time segments of 30 minutes per way type. As we can see there is massive impact on the average speed per type of road. The same procedure has been done in order to find the average mean and median speed on all time window size.

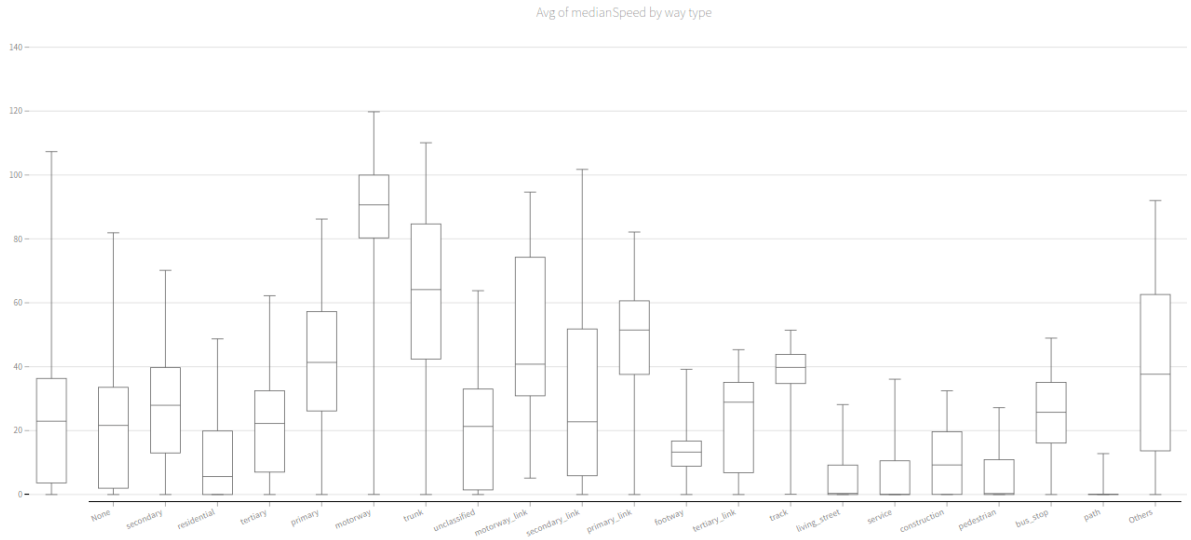

Figure 5.16: Average median speed per way type using 30 minutes segments

At this point it is studied the impact of weather in flow speed. In figure 5.17 is plotted the mean speed of all segments and the humidity.

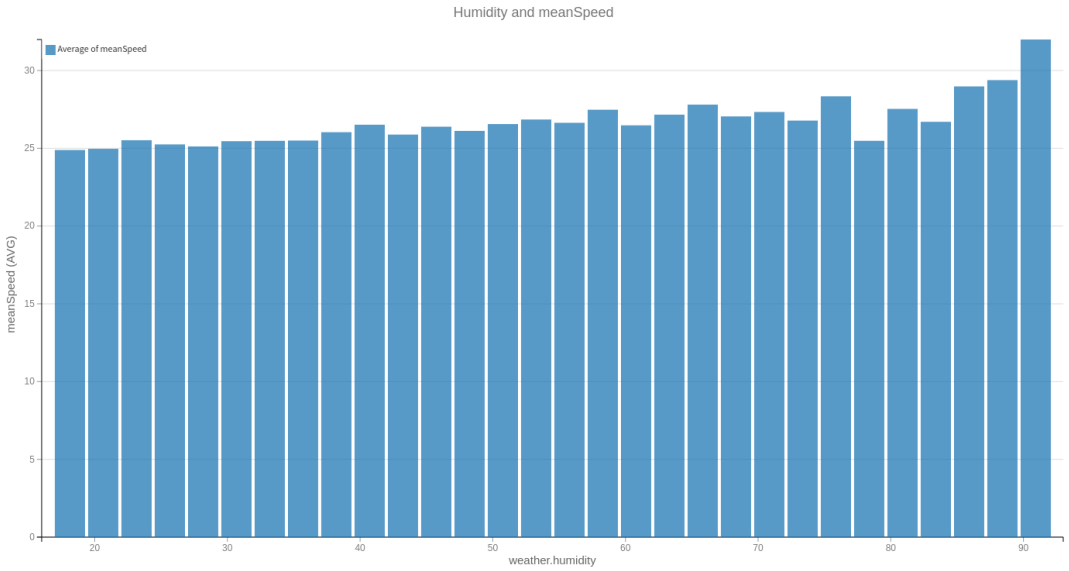

Figure 5.17: Humidity and speed on roads

As figure shows there is a positive correlation between speed and humidity on very high values of humidity.

A conclusion to be drawn in this chapter is that the datasets can be used in several ways in order to extract knowledge by them and the amount of data were remarkable for predictions and further processing after the cleaning phase, and the whole data model can presented as a unique tool for predictions in order to eliminate the problem of traffic congestion.

Chapter 6

Results - Future work

This chapter presents the future work and results using diagrams and tables. The results has been divided in statistical methods and in supervised learning. Beyond the statistical models were ARIMA models has been used, on supervised learning the results were divided into regression and classification. For each procedure the best results has been noted, At the end of each sub section there exists a summarise matrix for each technique using different parameters trying to find the best model, and some of the results can be found using figures in appendix A.2.

6.1 Statistical Models Results

As it is described in section 4.5.2 there were several approaches trying to find the best feet using the ARIMA method. In table 6.1 there is the matrix that describes the AR and MA variables as well the test error rates for each model. The train size for each model is 60% of the entire dataset and for each model exists a column that describes the run time of the models in minutes.

Model	Method	No. Obs	MSE	MAE	RMSE	Time mins
ARIMA(1,0,0)	Css-mle	18297	94.423	7.838	9.717	**63.43**
ARIMA(2,0,0)	Css-mle	18297	93.592	7.793	9.674	138.16
ARIMA(3,0,0)	Css-mle	18297	93.408	7.779	9.665	226.38
ARIMA(4,0,0)	Css-mle	18297	93.096	7.765	9.649	313.49
ARIMA(5,0,0)	Css-mle	18297	92.92	7.758	9.639	464.01
ARIMA(0,0,1)	Css-mle	18297	95.537	7.905	9.774	63.73
ARIMA(0,0,2)	Css-mle	18297	94.186	7.837	9.705	153.89
ARIMA(0,0,3)	Css-mle	18297	93.89	7.814	9.69	232.61
ARIMA(0,0,4)	Css-mle	18297	93.57	7.798	9.673	341.81
ARIMA(15,0,0)	Css	18297	**92.23**	**7.724**	**9.607**	8017.21
ARIMA(0,0,15)	Css	18297	92.28	7.726	9.609	8018.21

Table 6.1: ARIMA model results

As table 6.1 described the best model for our tra The main problem in classical Box-Jenkins is trying to decide which ARIMA specification to use -i.e. how many AR and / or MA parameters to include. It depended upon graphical and numerical evaluation of the sample autocorrelation and partial autocorrelation functions. However, the computational power needed in order to find a good model using ARIMA was prohibitive in our testing environment and the process was stopped after ten models. There have been eleven approaches and the best model for our

traffic prediction using real speed is ARIMA(15,0,0). As it is described the test test of the prediction models are the 40% of the entire dataset. As it is explained in section 6.3 these predictions can be better.

6.2 Supervised Learning Results

In this section the result set is presented by using tables and figures in order to visualize the predicted values either using regression process or classification. The results have been splitted per prediction method and model, as well as each model has been evaluated. For the evaluation of models the methods hold out and k-fold cross validation used and for each method exists a matrix with summarized results. The metrics are different per model as for the regression Mean Square Error, Root Mean Square Error, Mean Absolute Error has been used and as we can see of table as described in section 3.6. In order to find good models irrelevant to the method a lot of tries have been made, and each model has been evaluated using hold-out and k-fold cross validation. Using the **bold** annotation the smallest values for errors, time and the biggest for the accuracy on classification models per column for all methods can be found as special values.

6.2.1 Regression

Table 6.2 presents results of the regression process using the mean speeds per time windows size. The prediction on this phase is done only for the next time step. As we can see there were several variables of our models that have been tweaked in order to create a good prediction model. For example, the num neurons is the number of neurons for each layers as is described in figure 4.16. The windows size describes the size of the window that has been used on the time series aggregation phase. The look back variable is the number of time steps feeded into each phase in order to create the prediction for each step. The bach size describes the number of observation that will be used in order to create the prediction for each separate step. The model that has been used in order to predict these results has been trained using some standard characteristics. Such as:

LSTM Model

- Timestemps to predict: 1
- Learning rate: 0.001
- Train size:70%, test size: 30%
- Number of epochs: 500
- Optimizer: Adam

As explained on table 3.8 the error rates can change per method. For each model the errors were represented for each train, test sets separately for evaluation of the methods. For each window size the annotated model preset the best fit for each metric. In example the model with id 3 fits better on train set with 0.973 km/h deviation from the real speed value, but this is not happening on the

test set respectively were the model 1 fits better on the test set with 5.493 km/h deviation from the real value.

The evaluation of models has been done separately per prediction method as figure 3.8 described. As the timesteps to predict is equal to one, the predictions on the table correspond on minutes equal to the window size. For example in model with id 21 the model predicts the mean speed for the next 60 minutes with 2.5 km/h deviation from the actual speed value, where the look back variables used as multiplier by the window size, this means that on the same model using window size 60 and look back 16 means that we are using the last 16 hours as an input data for each prediction, and it is running for 26 minutes in order to create this prediction and it is the best prediction model as it uses a big time window and a lot of information of previous steps.

						Train			Test			
id	**num. neurons**	**Window size**	**Look back**	**Bach size**	**Params**	**MSE**	**RMSE**	**MAE**	**MSE**	**RMSE**	**MAE**	**Time mins**
1	64	5	200	20	255299	1.688	1.299	1.012	48.488	6.963	**5.493**	1.81
2	32	5	200	20	115363	2.991	1.729	1.346	66.877	8.178	6.527	**1.40**
3	64	5	120	20	173379	1.572	8.216	**0.973**	67.501	8.216	6.479	3.63
4	64	10	15	10	65859	11.559	3.521	2.384	11.559	6.618	**5.037**	3.42
5	64	10	60	20	111939	3.662	2.125	**1.391**	33.662	7.261	5.541	2.21
6	32	10	100	20	64163	12.935	3.597	2.71	60.444	7.775	6.173	**0.85**
7	32	15	66	20	46755	3.132	1.77	1.359	54.81	7.403	5.76	**0.60**
8	64	15	66	20	118083	3.475	1.864	1.511	41.994	6.48	**5.056**	0.72
9	120	15	66	20	302043	**1.508**	1.228	0.917	53.055	7.284	5.686	1.05
10	120	15	40	20	252123	1.932	6.656	**0.978**	44.301	6.656	5.324	2.34
11	120	20	30	20	232923	3.061	5.679	**1.282**	32.25	5.679	**4.474**	1.83
12	32	20	50	20	38563	7.317	2.705	2.013	50.918	7.136	5.457	**0.49**
13	64	25	24	20	75075	1.673	1.356	**0.962**	30.629	5.534	4.453	0.94
14	120	25	24	20	221403	2.026	1.215	1.049	26.282	5.126	**3.891**	1.37
15	64	25	24	20	75075	2.037	**1.215**	1.032	39.423	6.279	4.821	1.03
16	32	30	33	20	29859	3.675	1.917	1.508	45.479	6.744	5.386	**0.37**
17	120	30	20	20	213723	5.043	2.868	1.728	31.362	5.6	**4.296**	1.18
18	120	45	22	20	217563	3.8	**1.949**	1.557	29.986	5.476	4.227	0.43
19	32	45	22	20	24227	5.678	2.383	1.889	21.302	4.615	**3.288**	**0.29**
20	120	60	16	20	206043	2.665	**1.632**	1.261	24.325	4.932	3.925	0.38
21	32	60	16	20	21155	6.409	2.532	1.923	**9.970**	**3.158**	**2.588**	**0.26**

Table 6.2: LSTM model, 1 timesteps to predict, learning rate:0.001, train size:0.7, epochs:500 with unbalanced datasets using hold out

						Train			Test			
id	**num. neu-rons**	**Time window size**	**Look back**	**Bach size**	**Params**	**MSE**	**RMSE**	**MAE**	**MSE**	**RMSE**	**MAE**	**Time mins**
1	120	5	120	20	405723	**0.4136 (+/- 0.4907)**	**0.5562 (+/- 0.3229)**	0.4112 (+/- 0.2350)	**7.7113 (+/- 15.7785)**	**2.0060 (+/- 1.9202)**	1.4830 (+/- 1.5589)	66.6
2	32	5	120	20	74403	0.8328 (+/- 0.6043)	0.8545 (+/- 0.3205)	0.6364 (+/- 0.2565)	10.0481 (+/- 21.1187)	2.2796 (+/- 2.2026)	1.7648 (+/- 1.7622)	**33.2**
3	120	5	30	10	232923	0.9246 (+/- 1.3428)	0.8093 (+/- 0.5193)	**0.6025 (+/- 0.3655)**	12.6540 (+/- 19.4057)	2.8839 (+/- 2.0826)	**0.6025 (+/- 0.3655)**	82.0
4	32	10	15	10	20643	5.4537 (+/- 3.6289)	2.2123 (+/- 0.7480)	1.6088 (+/- 0.5472)	22.2947 (+/- 15.5398)	4.4642 (+/- 1.5379)	**1.6088 (+/- 0.5472)**	40.6
5	120	10	60	20	290523	**0.9164 (+/- 1.0765)**	**0.8310 (+/- 0.4752)**	**0.6132 (+/- 0.3684)**	11.6156 (+/- 23.2557)	2.4409 (+/- 2.3786)	1.7335 (+/- 1.6571)	38.3
6	64	10	60	20	111939	1.1286 (+/- 1.5846)	0.8930 (+/- 0.5754)	0.6659 (+/- 0.4277)	**9.9070 (+/- 19.8724)**	2.2814 (+/- 2.1684)	1.6382 (+/- 1.5046)	25.8
7	32	10	60	20	43683	1.6020 (+/- 1.5885)	1.1463 (+/- 0.5367)	0.8640 (+/- 0.4059)	10.4569 (+/- 19.9822)	**2.4192 (+/- 2.1458)**	1.8128 (+/- 1.5429)	**22.2**
8	120	15	40	20	252123	0.3398 (+/- 0.4329)	0.5028 (+/- 0.2949)	0.3652 (+/- 0.2210)	8.7526 (+/- 18.1703)	**2.0641 (+/- 2.1194)**	**1.5245 (+/- 1.6115)**	27.2
9	32	15	40	20	33443	0.5591 (+/- 0.6355)	0.6652 (+/- 0.3415)	0.5000 (+/- 0.2536)	**8.5110 (+/- 15.8939)**	2.1775 (+/- 1.9416)	1.6388 (+/- 1.4398)	**16.8**
10	64	15	40	20	91459	**0.4187 (+/- 0.5101)**	**0.5680 (+/- 0.3100)**	**0.4242 (+/- 0.2382)**	8.9247 (+/- 19.1158)	2.1020 (+/- 2.1228)	1.5836 (+/- 1.6064)	18.7
11	120	20	30	20	232923	**0.4346 (+/- 0.4949)**	**0.5721 (+/- 0.3276)**	**0.4188 (+/- 0.2345)**	**7.2885 (+/- 13.5706)**	**2.0274 (+/- 1.7827)**	**1.5102 (+/- 1.3560)**	21.5
12	32	20	30	20	28323	0.5852 (+/- 0.7402)	0.6586 (+/- 0.3891)	0.4949 (+/- 0.2864)	9.3930 (+/- 18.5285)	2.2202 (+/- 2.1127)	1.7017 (+/- 1.5802)	**13.2**
13	120	25	24	20	221403	**0.3107 (+/- 0.4012)**	**0.4767 (+/- 0.2889)**	**0.3533 (+/- 0.2109)**	8.3647 (+/- 17.8986)	2.0333 (+/- 2.0568)	1.4688 (+/- 1.4198)	17.4
14	120	25	24	20	221403	0.3536 (+/- 0.4175)	0.5159 (+/- 0.2957)	0.3828 (+/- 0.2165)	**7.9004 (+/- 16.7801)**	**1.9840 (+/- 1.9910)**	**1.4651 (+/- 1.4556)**	16.0
15	32	25	24	20	25251	0.5838 (+/- 0.7221)	0.6647 (+/- 0.3768)	0.5100 (+/- 0.2918)	8.0366 (+/- 15.6138)	2.0869 (+/- 1.9187)	1.5631 (+/- 1.3609)	**10.5**
16	32	30	20	20	23203	1.0969 (+/- 1.0684)	0.9303 (+/- 0.4811)	0.7024 (+/- 0.3739)	**7.0411 (+/- 11.0529)**	2.1194 (+/- 1.5967)	1.5765 (+/- 1.1557)	**9.1**
17	120	30	20	20	213723	**0.5869 (+/- 0.6490)**	**0.6622 (+/- 0.3853)**	**0.4909 (+/- 0.2893)**	7.0503 (+/- 12.8202)	**2.0192 (+/- 1.7242)**	**1.4907 (+/- 1.2029)**	14.3
18	32	45	13	20	19619	0.9150 (+/- 1.2092)	0.8045 (+/- 0.5175)	0.6162 (+/- 0.4032)	5.1693 (+/- 6.9177)	1.8957 (+/- 1.2552)	1.4182 (+/- 0.8503)	**6.5**
19	64	45	13	20	63811	0.6282 (+/- 0.6900)	0.6950 (+/- 0.3811)	0.5281 (+/- 0.3018)	**5.0292 (+/- 8.2234)**	**1.7792 (+/- 1.3652)**	**1.3177 (+/- 0.9663)**	6.6
20	120	45	13	20	200283	**0.5570 (+/- 0.7142)**	**0.6310 (+/- 0.3985)**	**0.4888 (+/- 0.3262)**	5.5116 (+/- 8.8602)	1.8744 (+/- 1.4136)	1.3288 (+/- 0.9503)	9.7
21	64	60	10	20	60739	0.3735 (+/- 0.4271)	0.5352 (+/- 0.2949)	0.3999 (+/- 0.2196)	3.6115 (+/- 6.0265)	1.4919 (+/- 1.1772)	**1.1162 (+/- 0.8400)**	5.2
22	32	60	10	20	18083	0.6127 (+/- 0.6241)	0.6923 (+/- 0.3654)	0.5490 (+/- 0.2905)	5.0607 (+/- 7.6114)	1.7992 (+/- 1.3503)	1.3341 (+/- 0.9027)	**5.1**
23	120	60	10	20	194523	**0.2966 (+/- 0.4143)**	**0.4484 (+/- 0.3090)**	**0.3350 (+/- 0.2358)**	**4.8466 (+/- 9.5003)**	**1.6274 (+/- 1.4827)**	1.2132 (+/- 1.0505)	7.6

Table 6.3: LSTM model, 1 timesteps to predict, learning rate:0.001, epochs:500 with unbalanced datasets using 10 fold cross validation

Table 6.3 shows the evaluation of the LSTM models using 10 fold cross validation in order to create robust prediction models for further use. This procedure has been done using the LSTM model with some standard characteristics. Such as:

- Timestemps to predict: 1
- Learning rate: 0.001
- Number of epochs: 500
- Optimizer: Adam
- 10 fold cross validation

As expected the best model using the k- fold cross validation uses 60 minutes time windows in order to predict the mean speed of the next 60 minutes, but this metric is valuable in order to extract knowledge for the mean speed per our and it can not used for recommendations about the traffic status in short time. On the other hand though using the 5 minute time window, 120 neurons for the first layer, 60 for the dense and look back equal to 30, which means 150 minutes memory for each step the model predicts on the test set with MAE 0.6025 and standard deviation 0.3655 near to this value in 82 minutes. This model can be used only on a production environment in order to predict the real mean speed of cars for the next 5 minutes as the time that require to make the prediction on our testing environment it makes it prohibitive.

As there was impossibility to show the results of all tests for visualization reasons in Appendix A.3 exists prediction diagram along with the learning curves diagram for some of the models that have been tested.

6.2.2 Classification

Table 6.4 summarizes some of the models that have been tested trying to find the best prediction model according to figure 3.8. The results shows that there are different results on fitting and on validation accuracy as the time window size changes using different timestemp prediction. As we can see on table the model used on this phase can explained as follows:

- Timestemps to predict: free variable
- Learning rate: free variable
- Look back: free variable
- Train size: free variable
- Number of neurons: free variable
- Hold out
- Number of epochs: 250
- Optimizer: Adam

*free variable: several values has been tested and can be found per model on table

As can be examined by the table has been used different time windows, train sizes and number of neurons per layer, the look back variable corresponds in timesteps accordingly the time window size, for example using a window size of 5 minutes and look back variable equal to 60, the model will use 300 minutes of observations on each step.

id	Window size	Timestamps to predict	Look back	Train size	Learning rate	Number of neurons	Total params	Accuracy	Validation accuracy	Loss	Validation Loss	Time mins
1	5	1	60	0.6	0.001	120	75.423	0.78	0.77	0.52	0.58	3.85
2	5	2	20	0.6	0.0001	150	116.778	0.88	0.85	0.29	0.38	10.32
3	5	3	60	0.6	0.001	120	75.423	0.79	0.77	0.50	0.57	3.83
4	5	4	40	0.6	0.01	150	116.778	0.76	0.77	0.53	0.52	19.13
5	5	5	60	0.6	0.001	120	75.423	0.78	0.77	0.51	0.56	3.83
6	5	6	20	0.6	0.0001	150	116.778	0.88	0.85	0.27	0.42	10.31
7	5	7	40	0.6	0.0001	150	116.778	0.87	0.83	0.30	0.41	18.94
8	5	8	20	0.6	0.0001	150	116.778	0.87	0.85	0.29	0.42	10.40
9	5	9	60	0.6	0.001	100	52.853	0.78	0.76	0.53	0.59	5.54
10	5	10	60	0.6	0.001	120	75.423	0.79	0.78	0.50	0.55	3.85
11	5	11	60	0.6	0.001	100	52.853	0.79	0.77	0.50	0.56	5.24
12	5	12	60	0.6	0.001	120	75.423	0.79	0.77	0.50	0.58	3.86
13	10	1	50	0.7	0.0001	150	116.778	0.82	0.80	0.40	0.43	13.32
14	10	2	10	0.7	0.0001	150	116.778	0.85	0.84	0.36	0.38	3.66
15	10	5	90	0.7	0.0001	150	116.778	0.81	0.79	0.43	0.46	23.03
16	10	6	10	0.7	0.0001	150	116.778	0.85	0.83	0.35	0.39	3.47
17	10	9	30	0.7	0.0001	150	116.778	0.83	0.82	0.39	0.41	7.30
18	10	10	20	0.7	0.0001	150	116.778	0.84	0.83	0.37	0.40	4.46
19	10	11	10	0.7	0.0001	150	116.778	0.85	0.84	0.35	0.37	2.12
20	15	1	6	0.7	0.0001	150	116.778	0.86	**0.86**	0.35	0.34	1.76
21	15	2	6	0.7	0.0001	150	116.778	0.85	**0.86**	0.35	0.36	2.17
22	15	3	6	0.7	0.0001	150	116.778	0.85	**0.86**	0.36	0.36	2.16
23	15	4	6	0.7	0.0001	150	116.778	0.86	**0.86**	0.35	0.35	2.29
24	15	5	66	0.7	0.0001	150	116.778	0.83	**0.82**	0.40	0.43	13.37
25	15	6	6	0.7	0.0001	150	116.778	0.86	**0.86**	0.35	0.35	1.96
26	15	7	6	0.7	0.0001	150	116.778	0.86	**0.86**	0.35	0.35	1.98
27	15	8	6	0.7	0.0001	150	116.778	0.86	**0.86**	0.35	0.34	2.21
28	15	9	6	0.7	0.0001	150	116.778	0.86	**0.86**	0.35	0.35	1.81
29	15	10	6	0.7	0.0001	150	116.778	0.85	**0.85**	0.36	0.36	1.98
30	15	11	6	0.7	0.0001	150	116.778	0.86	**0.85**	0.35	0.35	1.57
31	15	12	13	0.7	0.0001	150	116.778	0.85	**0.85**	0.37	0.38	2.04

Table 6.4: LSTM model, train, with balanced datasets using hold out

According to table the range of predictions are between (78 - 86 %) and the models with bigger window size as model 20, fits better and predicts the next timestemp with the better accuracy compared to other models.

In order to understand our models better, each model evaluated using 5 fold cross validation, for each timestemps for prediction using different window size per model. Table 6.5 summarizes the results for some of tested models.

id	Window size	Timesteps to predict	Look back	Accuracy	Validation accuracy	loss	Validation loss	Time mins
1	5	1	100	83.99% (+/- 4.91%)	74.30% (+/- 8.54%)	0.34	0.51	6.95
2	5	2	60	84.02% (+/- 4.99%)	74.49% (+/- 8.13%)	0.34	0.51	7.68
3	5	3	80	83.92% (+/- 5.04%)	74.19% (+/- 8.69%)	0.34	0.51	8.35
4	5	4	120	84.04% (+/- 4.95%)	74.71% (+/- 8.37%)	0.34	0.51	10.26
5	5	5	160	83.89% (+/- 5.09%)	74.39% (+/- 8.51%)	0.34	0.51	6.87
6	5	6	100	84.03% (+/- 4.94%)	74.38% (+/- 8.53%)	0.34	0.51	7.94
7	5	7	180	83.97% (+/- 4.93%)	74.42% (+/- 8.57%)	0.34	0.51	8.84
8	5	8	120	83.99% (+/- 4.92%)	74.34% (+/- 8.49%)	0.34	0.51	9.94
9	5	9	80	83.97% (+/- 4.95%)	**74.53% (+/- 8.42%)**	0.34	**0.51**	6.78
10	5	11	120	**84.03% (+/- 4.89%)**	74.38% (+/- 8.59%)	0.34	0.51	7.79
11	5	12	180	83.98% (+/- 4.98%)	74.43% (+/- 8.55%)	0.34	0.51	12.47
12	10	1	30	83.01% (+/- 5.48%)	70.59% (+/- 10.29%)	0.36	0.58	5.51
13	10	2	80	83.05% (+/- 5.37%)	70.63% (+/- 10.35%)	0.36	0.57	5.24
14	10	3	50	83.03% (+/- 5.42%)	70.64% (+/- 10.53%)	0.36	0.57	**4.69**
15	10	4	200	83.04% (+/- 5.43%)	70.49% (+/- 10.42%)	0.36	0.58	5.80
16	10	5	50	82.99% (+/- 5.47%)	70.53% (+/- 10.34%)	0.36	0.57	6.26
17	10	8	40	83.03% (+/- 5.58%)	70.59% (+/- 10.20%)	0.36	0.58	5.51
18	10	9	50	83.03% (+/- 5.51%)	70.41% (+/- 10.34%)	0.36	0.57	6.24
19	10	10	10	82.96% (+/- 5.50%)	70.49% (+/- 10.26%)	0.36	0.57	5.25
20	15	1	6	82.49% (+/- 6.03%)	63.72% (+/- 12.30%)	0.37	0.68	5.06
21	15	2	40	82.58% (+/- 6.03%)	63.99% (+/- 11.94%)	0.37	0.69	6.89
22	15	3	60	82.62% (+/- 5.87%)	63.85% (+/- 11.97%)	0.37	0.68	8.56
23	15	4	6	82.57% (+/- 5.95%)	63.55% (+/- 12.38%)	0.37	0.68	5.37
24	15	5	133	82.59% (+/- 5.95%)	63.90% (+/- 12.15%)	0.37	0.68	6.72
25	15	6	20	82.64% (+/- 5.87%)	63.96% (+/- 12.29%)	0.37	0.68	7.00
26	15	7	66	82.65% (+/- 5.86%)	63.86% (+/- 12.17%)	0.37	0.68	6.00
27	15	8	6	82.58% (+/- 5.97%)	63.69% (+/- 12.13%)	0.37	0.68	6.02
28	15	9	133	82.56% (+/- 5.91%)	63.90% (+/- 12.14%)	0.37	0.68	5.93
29	15	10	20	82.70% (+/- 5.86%)	63.88% (+/- 11.95%)	0.37	0.68	5.60
30	15	11	33	82.62% (+/- 5.92%)	63.89% (+/- 12.25%)	0.37	0.68	6.82
31	15	12	33	82.58% (+/- 5.89%)	63.90% (+/- 12.25%)	0.37	0.68	7.28

Table 6.5: LSTM model, learning rate:0.00001, epochs:250 with balanced datasets using 5 fold cross validation

6.3 Future work

This subsection explores the future work that needs to be done both in theoretical and practical level. The section is divided into architecture, prediction and dataset. The architecture section created in order to make the framework more scalable. As on prediction section is described our problems in prediction procedure trying to create predictions in all roads of Thessaloniki and how we are planning to resolve them.

As it is discussed in section 4.1 the whole system designed parallel using threads among servers using REST APIs. In order to use more constructive asynchronous architecture between phases, for future work an architecture using a messaging platform between servers has been designed. The specific architecture can be studied in figure 6.1.

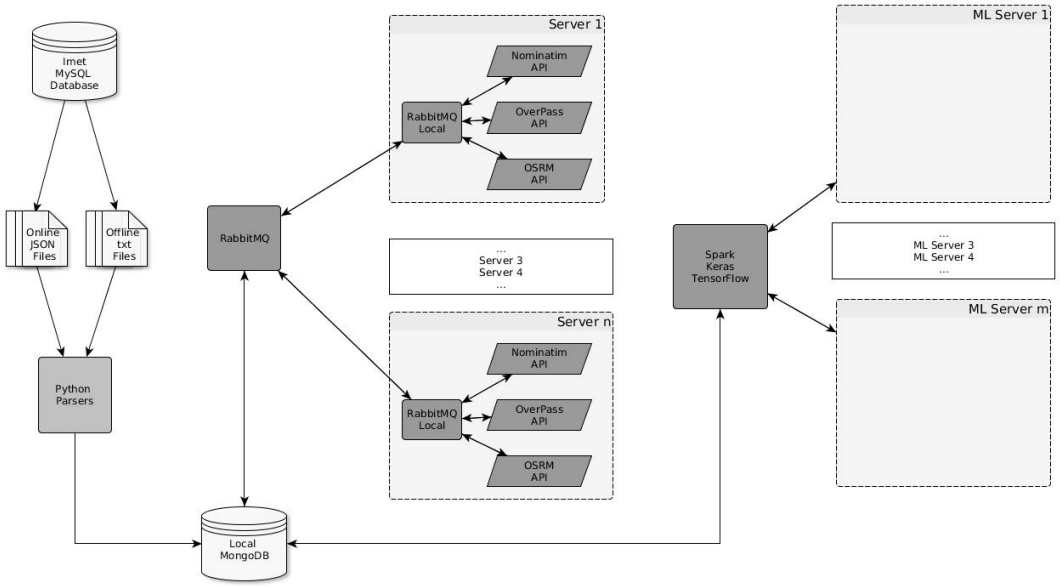

Figure 6.1: Future work architecture

After this adjustments, the framework will be ready to serve online predictions in more than one city irrelevant with the size of city using more reliable hardware as infrastructure. in section 4.5 the implementation of prediction phase is discussed. As our computational power was limited, is suggested for future work to create the predictions on all roads of town (7.226 osm ways) in order to understand the traffic with more details.

For this research weather stations have been used to build a model for traffic analyses. There is no guarantee on the time of the weather observation is recorded. At this point we know from other research that weather does effect road conditions (Dunne and Ghosh, 2013). As this study is exploring the dynamics of roads over a large geographical location. It is required to know that not all roads are under the same whether condition at one time or even receive the same level of condition. It is suggested for future work to collect the weather condition from more stations around the city with more precise topical characteristics.

Conclusions a conclusion to be drawn is that the predictions that have be presented in this chapter differ on error rates per model, but after the adjustments that have been described in previous chapters the models can predict accurately the traffic congestion as well as the real speed values, and these models can used separately accordingly with the time window. For example models that predict values more accurate using big time windows can be used in order to predict the congestion days after the observations as appropriate with small time windows.

Appendix A

Appendix

A.1 Main pipeline

```
import time
from osmHandler import osmHandler
from wayHandler import wayHandler
from zoneHanlder import zoneHandler
from osrmHandler import osrmHandler
from fileHandler import fileHandler
from polygonHandler import polygonHandler
from databaseHandler import databaseHandler
from onlineDataHandler import onlineDataHandler
from timeSegmentHandler import timeSegmentHandler
from mapMatchingHandler import mapMatchingHandler
from projecttStatistics import projecttStatistics
from predictionsAI import predictions as supervPrediction
from predictionsStats import predictions as statsPrediction


## project settings ##
server_ip="localhost"
osmAPIServer=server_ip
nominatimAPIServer=server_ip
osrmAPIServer=server_ip
osrmAPIPort=5000
number_of_osrm_threads=8
number_of_osm_threads=30
number_of_segments_threads=1000
number_of_url_per_thread=100
number_of_points_per_match=100
mongoDb="mongodb://username:password"+server_ip+"/thesis-database"


## datasets locations ##
__datasets__root="../../__datasets"
_fcd_location="/fcd"
_polygon_location="/non_valid_polygons"
fcd_dataset=__datasets__root+_fcd_location
polygons_dataset=__datasets__root+_polygon_location

## initialize ##
_databaseHandler = databaseHandler('thesis-database-production', "fcd_cars", "ways", "
    connected_ways","time_segments_5","time_segments_10","time_segments_15","
    time_segments_20","time_segments_25","time_segments_30","time_segments_45","
    time_segments_60","time_segments_stats_5","time_segments_stats_10","
    time_segments_stats_15","time_segments_stats_20","time_segments_stats_25","
    time_segments_stats_30","time_segments_stats_45","time_segments_stats_60","osm_data","
    osm_data_meta","weather", mongoDb)
_osmhandler = osmHandler(_databaseHandler,osmAPIServer,nominatimAPIServer,osrmAPIServer,
    number_of_osm_threads)
_wayHandler = wayHandler(_databaseHandler)
_polygonHandler = polygonHandler(_databaseHandler)
_fileHandler = fileHandler(_databaseHandler, _osmhandler,_polygonHandler)
```

```
44  _osrmhandler = osrmHandler(osrmAPIServer,osrmAPIPort,number_of_osrm_threads,
        number_of_url_per_thread)
45  _mapMatchingHandler = mapMatchingHandler(_databaseHandler,_osrmhandler,
        number_of_points_per_match)
46  _statisticsHandler= projecttStatistics(_databaseHandler)
47  _timeSegmentHandler= timeSegmentHandler(_databaseHandler,number_of_segments_threads,
        _statisticsHandler)
48  _zoneHandler = zoneHandler(_databaseHandler)
49  _statsPredictionHandler=statsPrediction(_databaseHandler)
50  _supervisedPredictionHandler=supervPrediction(_databaseHandler)
51
52
53  ## create indexes ##
54  print ("Creating indexes...")
55  _databaseHandler.createIndex(_databaseHandler.car_collection,"location","2dsphere",False)
56  _databaseHandler.createIndex(_databaseHandler.car_collection,"matched_location","2dsphere",
        False)
57  _databaseHandler.createIndex(_databaseHandler.car_collection,"matched_location_name",1,False)
58  _databaseHandler.createIndex(_databaseHandler.car_collection,"osm_way_id",1,False)
59  _databaseHandler.createIndex(_databaseHandler.car_collection,"recorded_timestamp",1,True)
60  _databaseHandler.createIndex(_databaseHandler.car_collection,"car_id",1,False)
61  _databaseHandler.createIndex(_databaseHandler.car_collection,"osm_type",1,False)
62  _databaseHandler.createIndex(_databaseHandler.car_collection,"speed",1,False)
63  _databaseHandler.createIndex(_databaseHandler.car_collection,"osm_type,osm_id,
        recorded_timestamp",1,False)
64
65  _databaseHandler.createIndex(_databaseHandler.ways_collection,"way_cordinates","2dsphere",
        False)
66  _databaseHandler.createIndex(_databaseHandler.ways_collection,"osm_way_id",1,True)
67
68  _databaseHandler.createIndex(_databaseHandler.connected_ways,"osm_way_id",1,True)
69  _databaseHandler.createIndex(_databaseHandler.connected_ways,"connected_with_ways",1,False)
70
71  _databaseHandler.createIndex(_databaseHandler.way_segments5,"osm_id",1,False)
72  _databaseHandler.createIndex(_databaseHandler.way_segments10,"osm_id",1,False)
73  _databaseHandler.createIndex(_databaseHandler.way_segments15,"osm_id",1,False)
74  _databaseHandler.createIndex(_databaseHandler.way_segments20,"osm_id",1,False)
75  _databaseHandler.createIndex(_databaseHandler.way_segments25,"osm_id",1,False)
76  _databaseHandler.createIndex(_databaseHandler.way_segments30,"osm_id",1,False)
77  _databaseHandler.createIndex(_databaseHandler.way_segments45,"osm_id",1,False)
78  _databaseHandler.createIndex(_databaseHandler.way_segments60,"osm_id",1,False)
79
80
81  print ("Read data from files...")
82  _fileHandler.readData(fcd_dataset)
83
84
85  print ("Clean dataset from stopped taxi...")
86  _polygonHandler.CleanDatasetFromStoppedTaxis(0,20,6,polygons_dataset)
87
88
89  print ("Match points on map...")
90  _mapMatchingHandler.matchAllPointsToCorrectWays()
91
92
93  print ("Match coordinates to ways...")
94  _osmhandler.matchLonLatToWays()
95
96
97  print ("Match mismatced nodes to ways...")
98  _osmhandler.matchNodesToWays()
99
100
101 print ("Compute time Segments..." )
102 _timeSegmentHandler.createTimeSegments(60)
103 _timeSegmentHandler.createTimeSegments(45)
104 _timeSegmentHandler.createTimeSegments(30)
105 _timeSegmentHandler.createTimeSegments(25)
106 _timeSegmentHandler.createTimeSegments(20)
```

```
_timeSegmentHandler.createTimeSegments(15)
_timeSegmentHandler.createTimeSegments(10)
_timeSegmentHandler.createTimeSegments(5)


print("Check stationarity...")
_statisticsHandler.stationarityTest()


print ("Predict congestion using multi class...")
_supervisedPredictionHandler.predictCongestion("classification")


print ("Predict congestion using real values...")
_supervisedPredictionHandler.predictCongestion("regression")


print ("Predict congestion using real values ARIMA...")
_statsPredictionHandler.predictTraffic()
```

Listings A.1: Main pipeline

```
"osm_id_34389787 : {{
  "version": "0.6",
  "copyright": "OpenStreetMap and contributors",
  "license": "http://opendatacommons.org/licenses/odbl/1-0/",
  "way": {
     "id": "34389787",
     "visible": "true",
     "version": "7",
     "changeset": "29525656",
     "timestamp": "2015-03-16T19:40:49Z",
     "user": "JayCBR",
     "uid": "2700471",
     "nd": [{"ref": "29802964"},
           {"ref": "436318586"},
           {"ref": "1711548286"},
           {"ref": "29802950"}],
     "tag": [{"k": "highway","v": "secondary"},
        {"k": "name","v": "mitropoleos"},
        {"k": "name:en","v": "Mitropoleos street"},
        {"k": "name:prefix","v": "street"},
        {
           "k": "oneway",
           "v": "yes"
        }
     ]
  }
}}
```

Listings A.2: OSM way structure

```
osm_id_878655876{
  "version": "0.6",
  "copyright": "OpenStreetMap and contributors",
```

```
   "attribution": "http://www.openstreetmap.org/copyright",
   "license": "http://opendatacommons.org/licenses/odbl/1-0/",
   "node": {
      "id": "878655876",
      "visible": "true",
      "version": "3",
      "changeset": "42467856",
      "timestamp": "2016-09-27T12:47:46Z",
      "user": "wheelmap_android",
      "uid": "524500",
      "lat": "40.6326746",
      "lon": "22.9413547",
      "tag": [{"k": "highway","v": "bus_stop"},
         {"k": "name","v": "bus stop aristotelous"},
         {"k": "route_ref","v": "5;6;12"},
         {"k": "wheelchair","v": "no"}
      ]
   }
}
```


Listings A.3: OSM node structure


```
   "matchings": {{
      "confidence": 0,
      "geometry": {
         "coordinates": [
            [22.959029, 40.641235],
            [22.95907, 40.641284]
         ],
         "type": "LineString"
      },
      "legs": [{
         "annotation": {
            "nodes": [163183488, 1737078440],
            "datasources": [0],
            "speed": [16.1],
            "weight": [0.4],
            "duration": [0.4],
            "distance": [6.455811]
         },
         "steps": [{
            "intersections": [{
               "out": 0,
               "entry": [true],
               "bearings": [32],
               "location": [22.959029, 40.641235]
            }],
            "geometry": {
               "coordinates": [
```

```
                  [22.959029, 40.641235],
                  [22.95907, 40.641284]
               ],
               "type": "LineString"
            },
            "mode": "driving",
            "duration": 0.4,
            "maneuver": {
               "bearing_after": 32,
               "type": "depart",
               "modifier": "right",
               "bearing_before": 0,
               "location": [22.959029, 40.641235]
            },
            "weight": 0.4,
            "distance": 6.5,
            "name": "Eptapirgiou"
         }, {
            "intersections": [{
               "in": 0,
               "entry": [true],
               "bearings": [212],
               "location": [22.95907, 40.641284]
            }],
            "geometry": {
               "coordinates": [
                  [22.95907, 40.641284],
                  [22.95907, 40.641284]
               ],
               "type": "LineString"
            },
            "mode": "driving",
            "duration": 0,
            "maneuver": {
               "bearing_after": 0,
               "location": [22.95907, 40.641284],
               "bearing_before": 32,
               "type": "arrive"
            },
            "weight": 0,
            "distance": 0,
            "name": ""
         }],
         "distance": 6.5,
         "duration": 0.4,
         "summary": "Eptapirgiou",
         "weight": 0.4
      }],
```

```
      "distance": 6.5,
      "duration": 0.4,
      "weight_name": "routability",
      "weight": 0.4
   }],
   "tracepoints": [{
      "alternatives_count": 22,
      "waypoint_index": 0,
      "location": [22.959029, 40.641235],
      "name": "Eptapirgiou",
      "matchings_index": 0
   }, {
      "alternatives_count": 32,
      "waypoint_index": 1,
      "location": [22.95907, 40.641284],
      "name": "Eptapirgiou",
      "matchings_index": 0
   }]
```


Listings A.4: OSRM geojson respone


```
{
   "timestamp": "Mon, 25 Dec 17 10:10:31 +0000",
   "attribution": "Data  OpenStreetMap contributors, ODbL 1.0. http:
      //www.openstreetmap.org/copyright",
   "querystring": "format=xml lat=40.63267 lon=22.94089 zoom=18
      addressdetails=1",
   "result": {
      "place_id": "80109789",
      "osm-type": "way",
      "osm-id": "37080247",
      "lat": "40.6329185",
      "lon": "22.9406380513023",
      "boundingbox": "40.6327965,40.6330246,22.9404824,22.9407763",
      "#text": "7, Aristotelous Square, Tsinari, Ano poli,
         Thessaloniki, Thessaloniki Regional Unit, Central
         Macedonia Region, Macedonia - Thrace, 54124, Greece"
   },
   "addressparts": {
      "house_number": "7",
      "pedestrian": "Aristotelous Square",
      "neighbourhood": "Tsinari",
      "suburb": "Ano poli",
      "city": "Thessaloniki",
      "county": "Thessaloniki Regional Unit",
      "state_district": "Central Macedonia Region",
      "state": "Macedonia - Thrace",
      "postcode": "54124",
      "country": "Greece",
```

```
        "country_code": "gr"
    }
}
```

Listings A.5: Nominatim reverse geocoding response

A.2 Extended results

A.2.1 Results using ARIMA model

ARIMA(1,0,0)

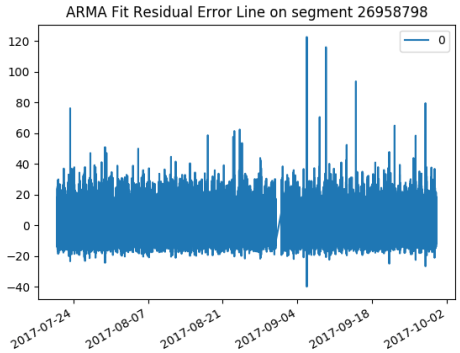

Figure A.1: Fit residual line ARIMA(1,0,0)

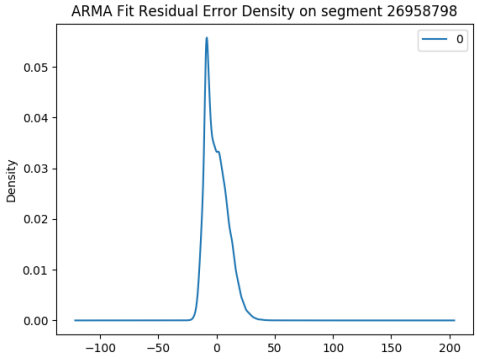

Figure A.2: Fit residual density ARIMA(1,0,0)

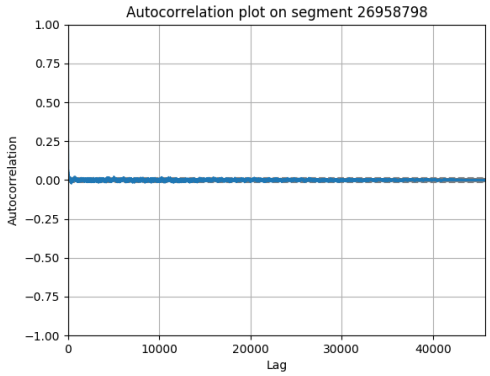

Figure A.3: Autocorrelation using ARIMA(1,0,0)

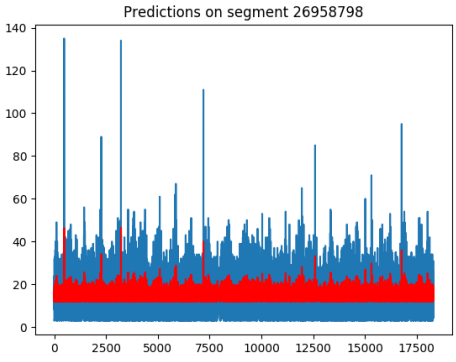

Figure A.4: Prediction using ARIMA(1,0,0)

ARIMA(2,0,0)

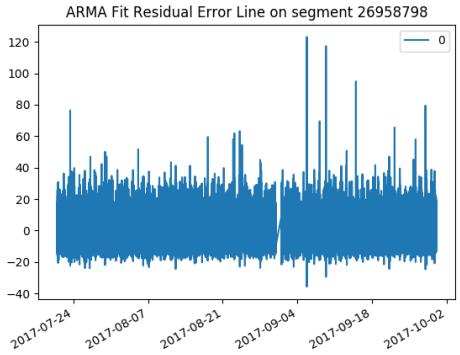

Figure A.5: Fit residual line ARIMA(2,0,0)

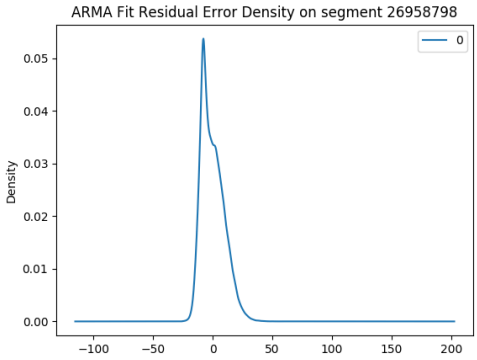

Figure A.6: Fit residual density ARIMA(2,0,0)

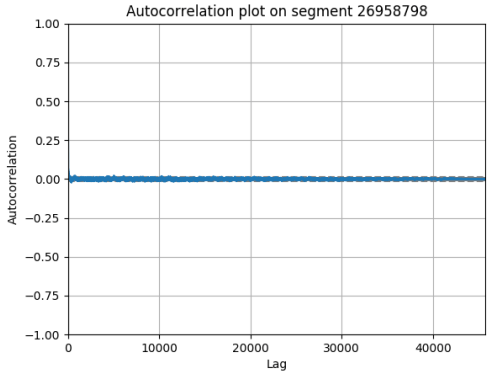

Figure A.7: Autocorrelation using ARIMA(2,0,0)

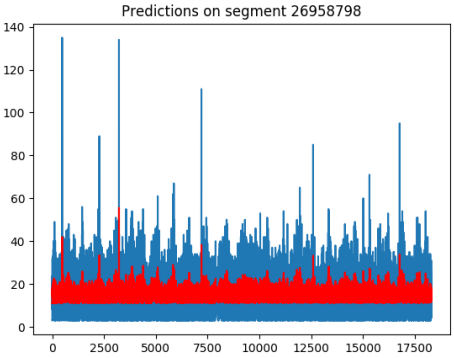

Figure A.8: Prediction using ARIMA(2,0,0)

ARIMA(3,0,0)

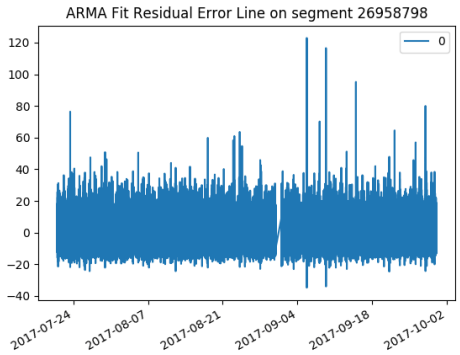

Figure A.9: Fit residual line ARIMA(3,0,0)

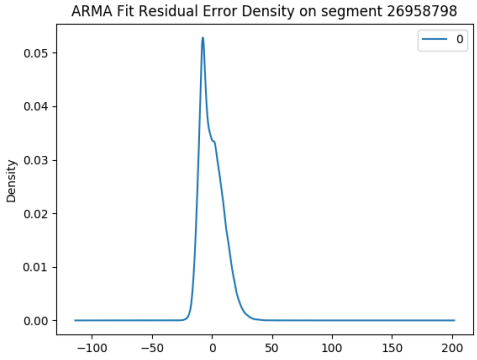

Figure A.10: Fit residual density ARIMA(3,0,0)

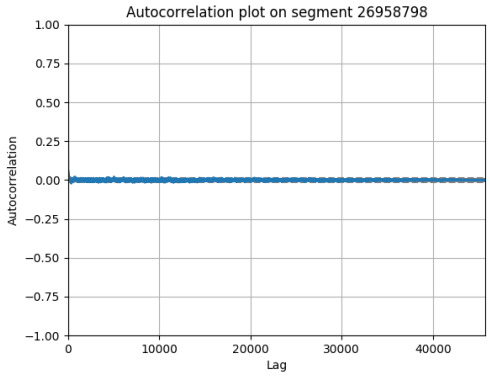

Figure A.11: Autocorrelation using ARIMA(3,0,0)

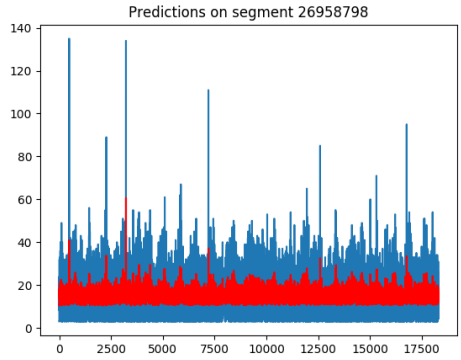

Figure A.12: Prediction using ARIMA(3,0,0)

ARIMA(4,0,0)

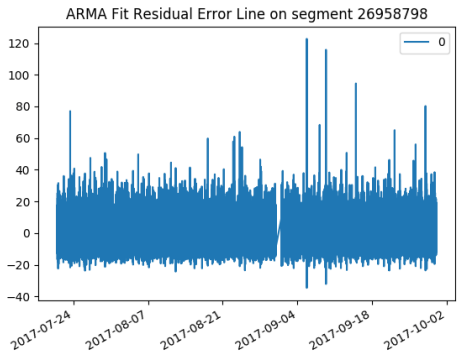

Figure A.13: Fit residual line ARIMA(4,0,0)

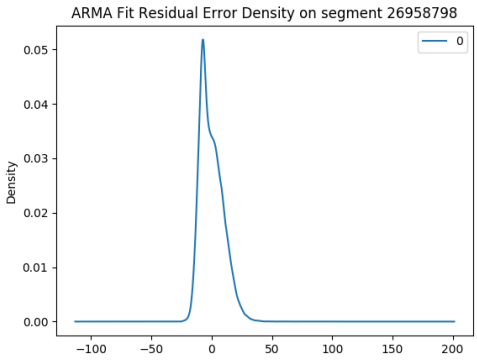

Figure A.14: Fit residual density ARIMA(4,0,0)

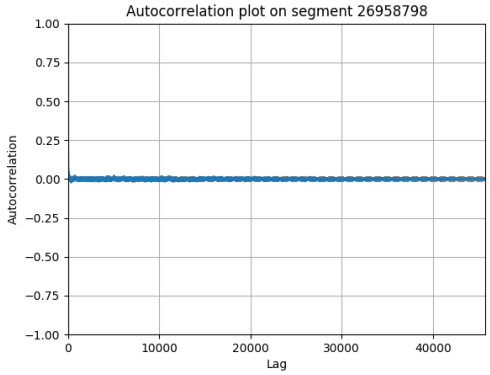

Figure A.15: Autocorrelation using ARIMA(4,0,0)

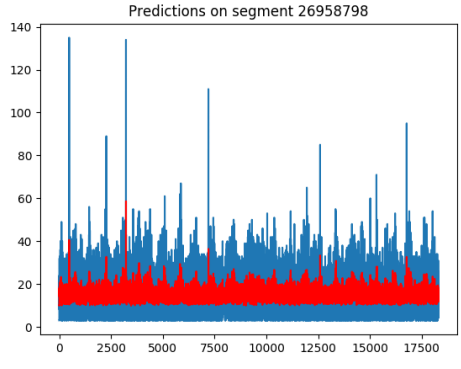

Figure A.16: Prediction using ARIMA(4,0,0)

ARIMA(5,0,0)

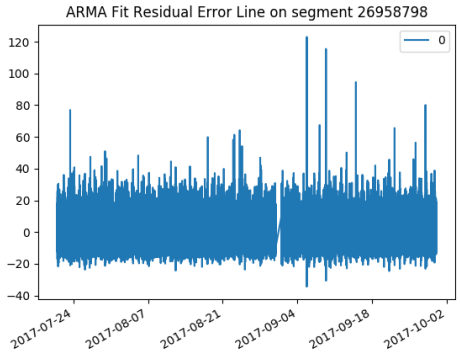

Figure A.17: Fit residual line ARIMA(5,0,0)

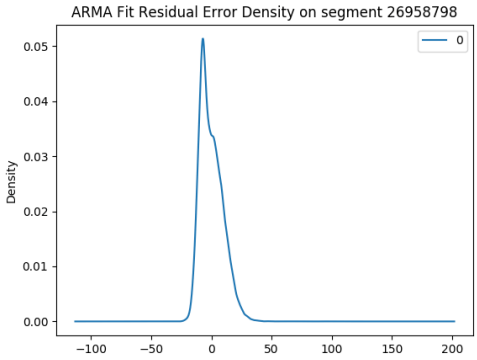

Figure A.18: Fit residual density ARIMA(5,0,0)

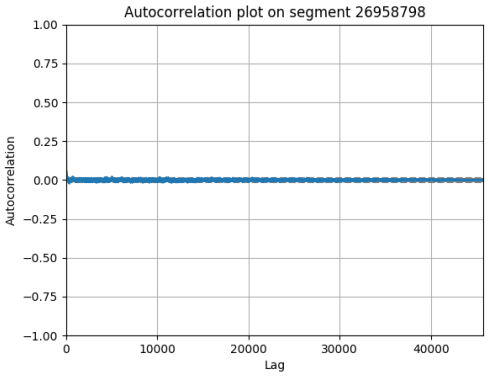

Figure A.19: Autocorrelation using ARIMA(4,0,0)

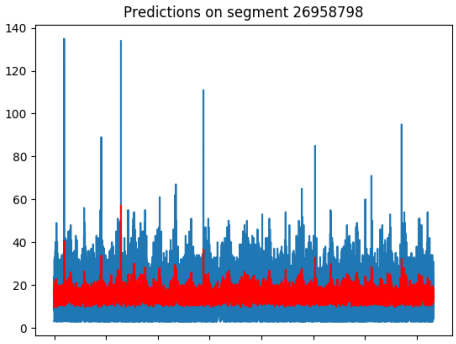

Figure A.20: Prediction using ARIMA(5,0,0)

ARIMA(0,0,1)

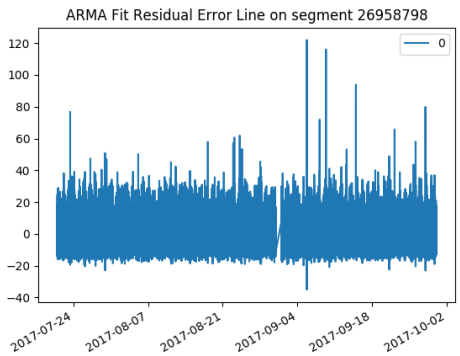

Figure A.21: Fit residual line ARIMA(0,0,1)

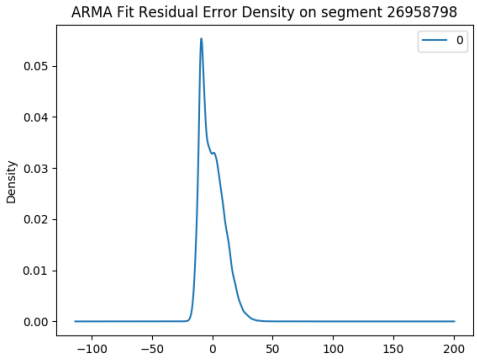

Figure A.22: Fit residual density ARIMA(0,0,1)

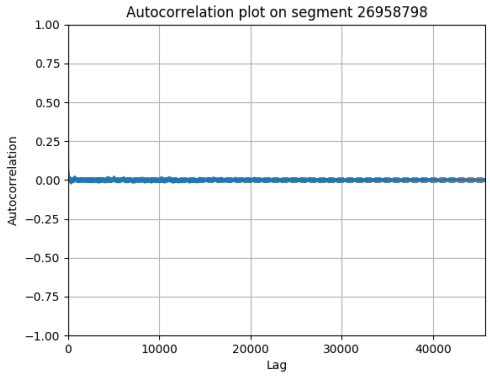

Figure A.23: Autocorrelation using ARIMA(0,0,1)

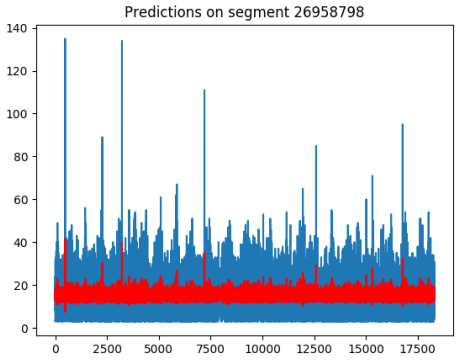

Figure A.24: Prediction using ARIMA(0,0,1)

ARIMA(0,0,2)

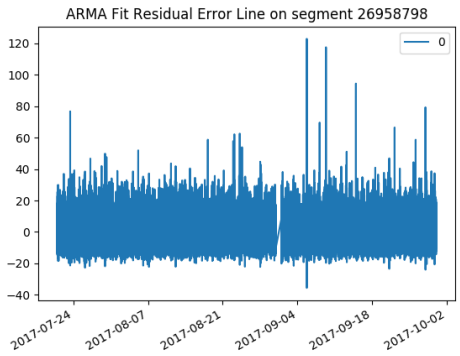

Figure A.25: Fit residual line ARIMA(0,0,2)

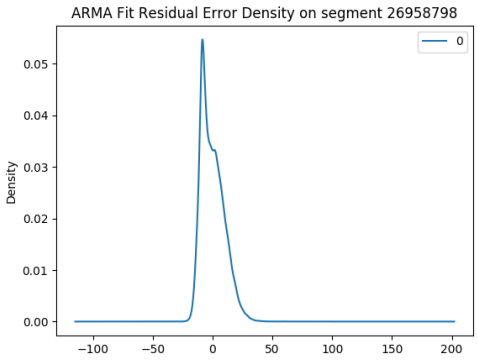

Figure A.26: Fit residual density ARIMA(0,0,2)

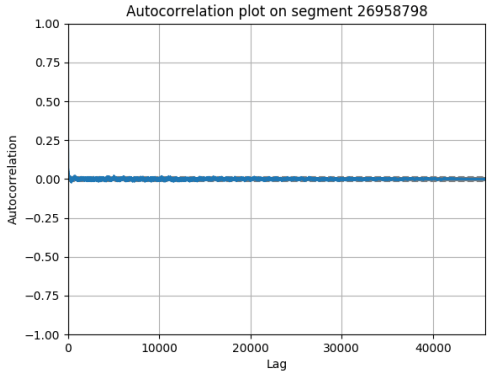

Figure A.27: Autocorrelation using ARIMA(0,0,2)

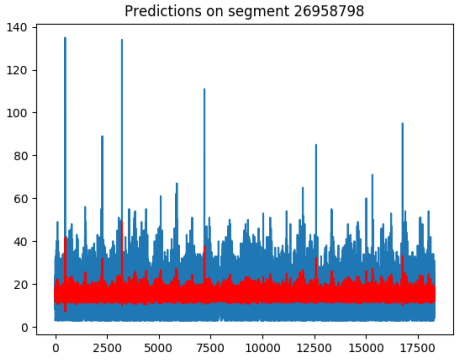

Figure A.28: Prediction using ARIMA(0,0,2)

ARIMA(0,0,3)

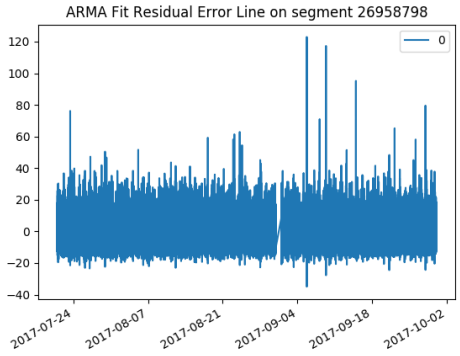

Figure A.29: Fit residual line ARIMA(0,0,3)

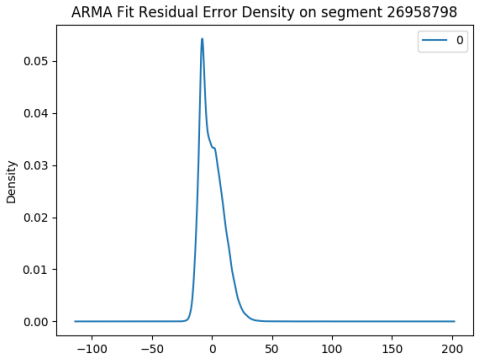

Figure A.30: Fit residual density ARIMA(0,0,3)

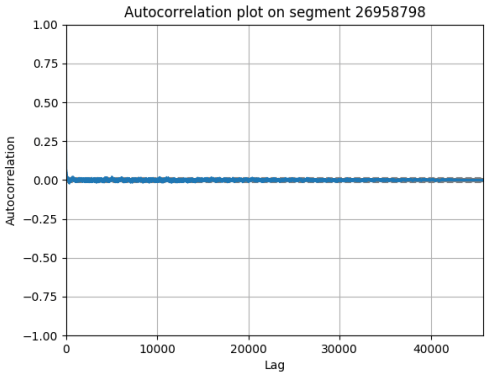

Figure A.31: Autocorrelation using ARIMA(0,0,3)

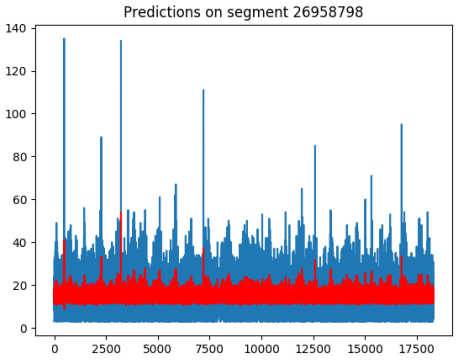

Figure A.32: Prediction using ARIMA(0,0,3)

ARIMA(0,0,4)

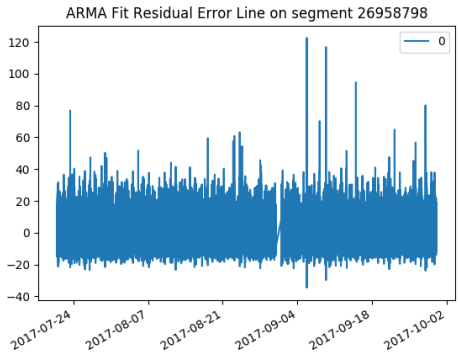

Figure A.33: Fit residual line ARIMA(0,0,4)

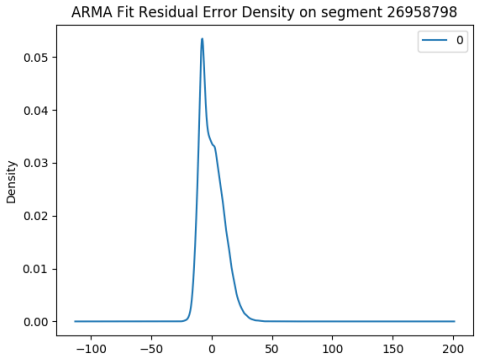

Figure A.34: Fit residual density ARIMA(0,0,4)

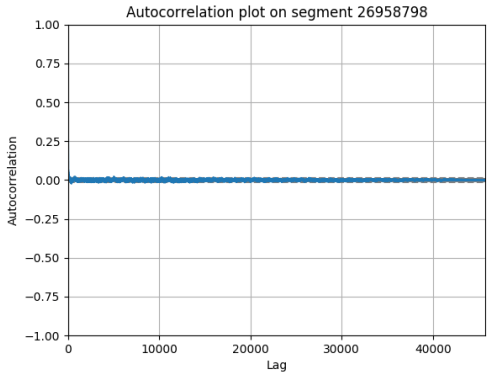

Figure A.35: Autocorrelation using ARIMA(0,0,4)

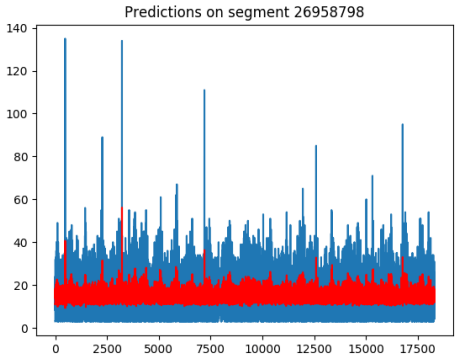

Figure A.36: Prediction using ARIMA(0,0,4)

A.3 Results using supervised learning (Regression)

Models per time window

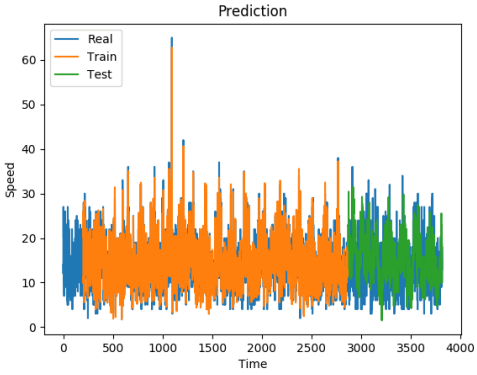

Figure A.37: Prediction using 5 minute time window, epochs: 500, test MAE: 5.493

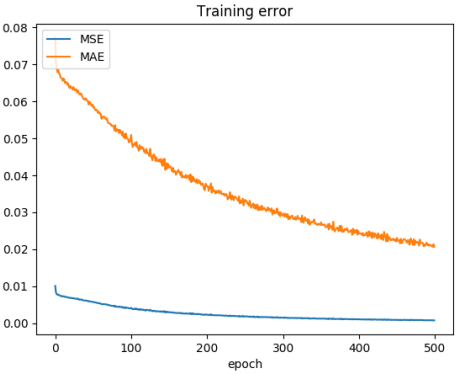

Figure A.38: Learning curve using 5 minute time window, epochs: 500, test MAE: 5.493

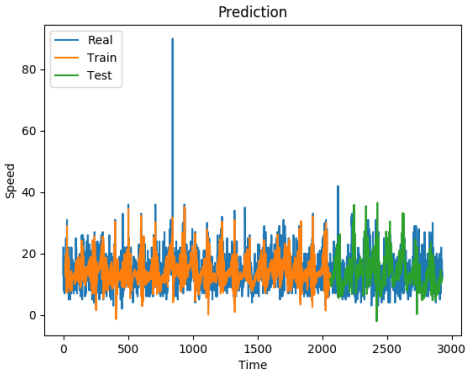

Figure A.39: Prediction using 10 minute time window, epochs: 500, test MAE: 5.037

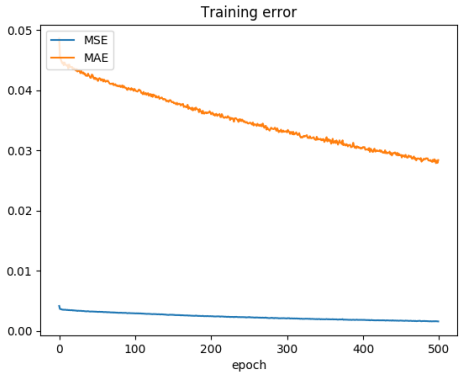

Figure A.40: Learning curve using 10 minute time window, epochs: 500, test MAE: 5.037

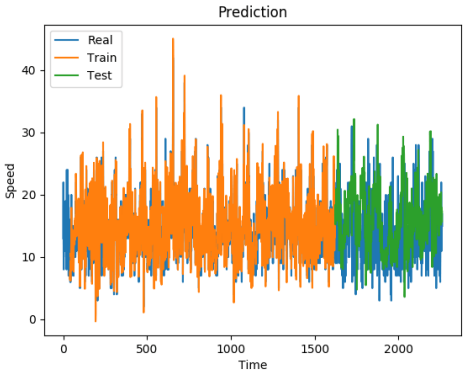

Figure A.41: Prediction using 15 minute time window, epochs: 500, test MAE: 5.324

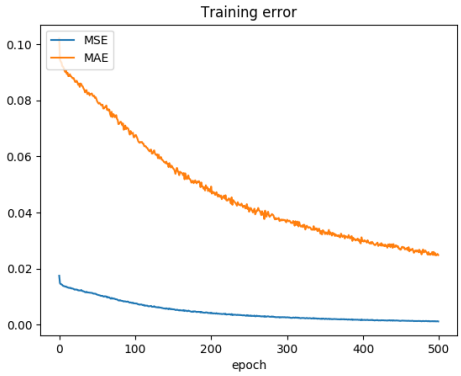

Figure A.42: Learning curve using 15 minute time window, epochs: 500, test MAE: 5.324

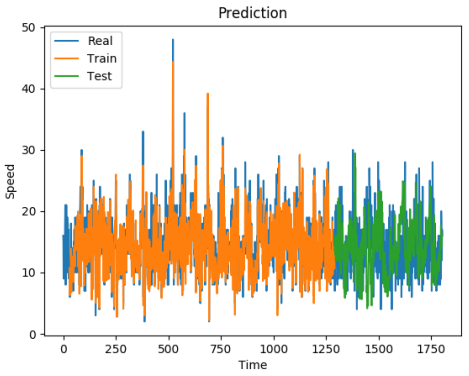

Figure A.43: Prediction using 20 minute time window, epochs: 500, test MAE: 4.474

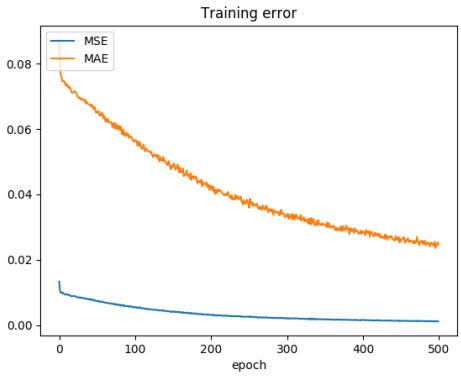

Figure A.44: Learning curve using 20 minute time window, epochs: 500, test MAE: 4.474

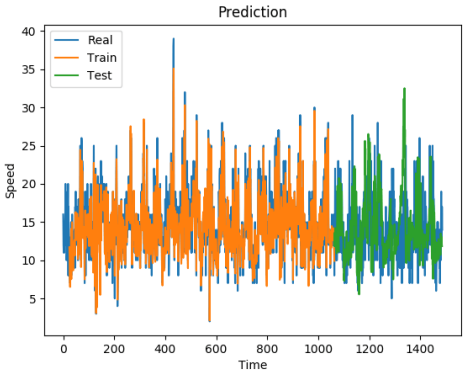

Figure A.45: Prediction using 25 minute time window, epochs: 500, test MAE: 3.891

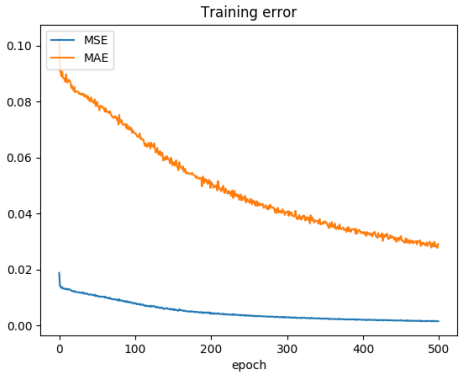

Figure A.46: Learning curve using 25 minute time window, epochs: 500, test MAE: 3.891

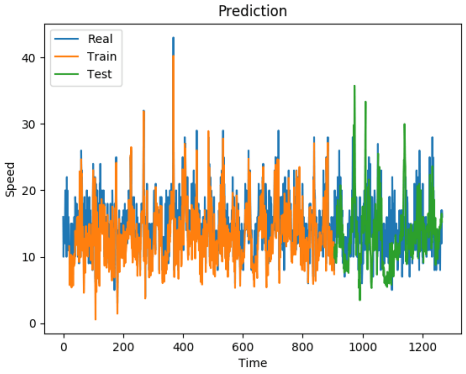

Figure A.47: Prediction using 30 minute time window, epochs: 500, test MAE: 4.295

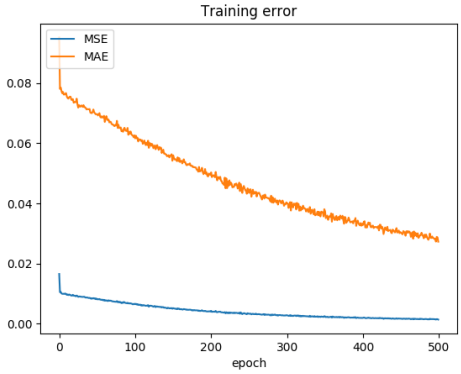

Figure A.48: Learning curve using 30 minute time window,epochs: 500, test MAE: 4.295

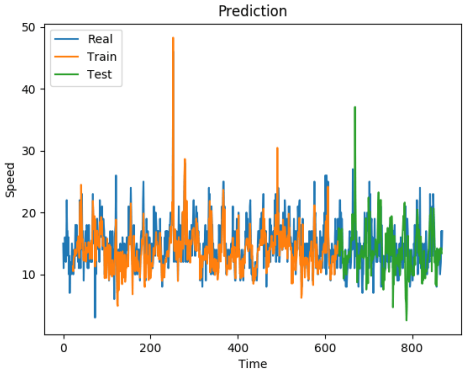

Figure A.49: Prediction using 45 minute time window, epochs: 500, test MAE: 3.288

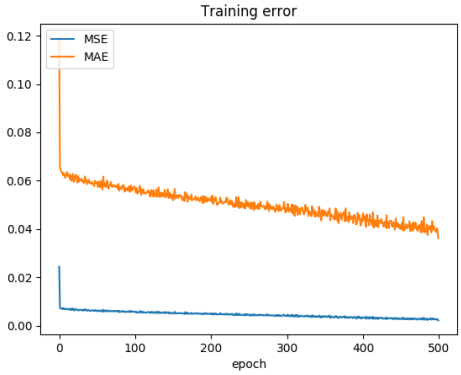

Figure A.50: Learning curve using 45 minute time window, epochs: 500, test MAE: 3.288

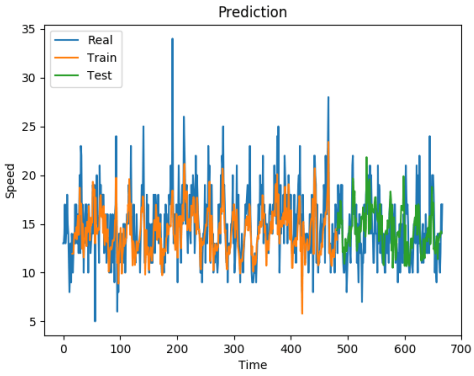

Figure A.51: Prediction using 60 minute time window, epochs: 500, test MAE: 2.588

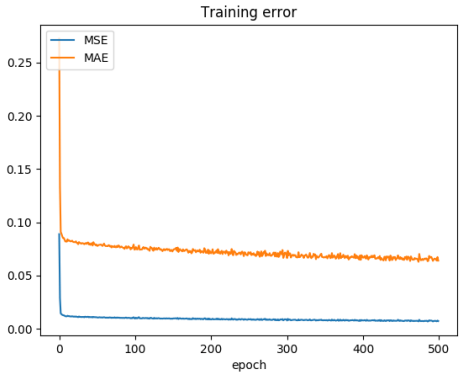

Figure A.52: Learning curve using 60 minute time window, epochs: 500, test MAE: 2.588

A.4 Results using supervised learning (Classification)

Models per prediction timestep

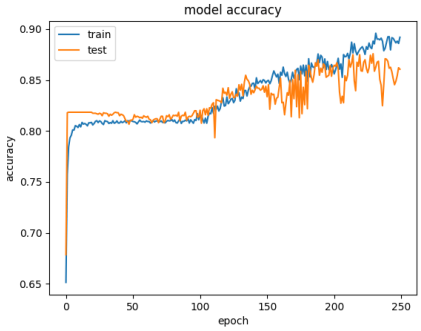

Figure A.53: Prediction using 15 minute time window, epochs:250 , look back:26, train size:70%, timestemps to predict: 1, test accuracy: 86%

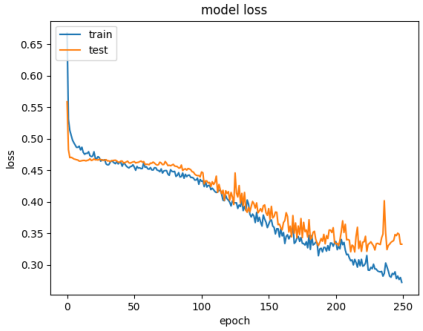

Figure A.54: Learning curve using 15 minute time window, epochs:250 , look back:26, train size:70%, timestemps to predict: 1, validation loss: 0.406

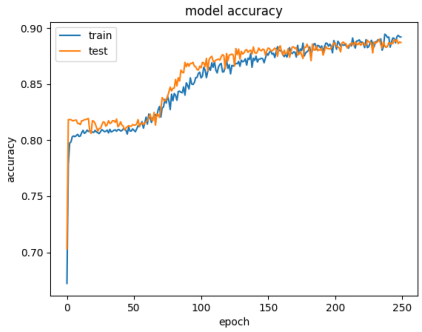

Figure A.55: Prediction using 15 minute time window, epochs:250, look back:6, train size:70%, timestemps to predict: 2, test accuracy: 86%

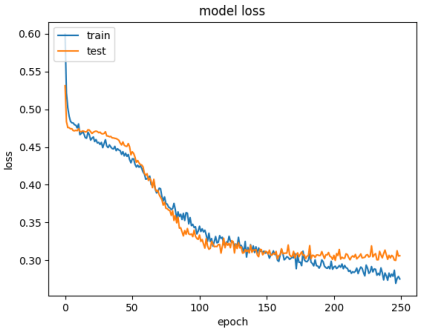

Figure A.56: Learning curve using 15 minute time window, epochs:250, look back:6, train size:85%, timestemps to predict: 2, validation loss: 0.355

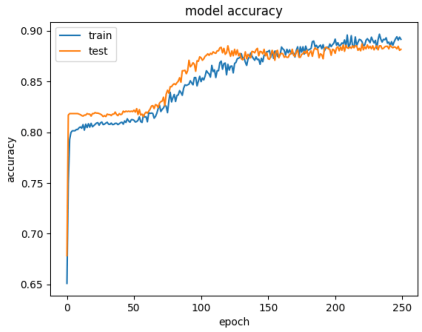

Figure A.57: Prediction using 15 minute time window, epochs:250, look back:6, train size:70%, timestemps to predict: 3, test accuracy: 85%

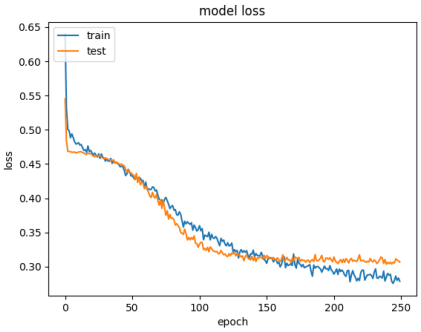

Figure A.58: Learning curve using 15 minute time window, epochs:250, look back:6, train size:85%, timestemps to predict: 3, validation loss: 0.351

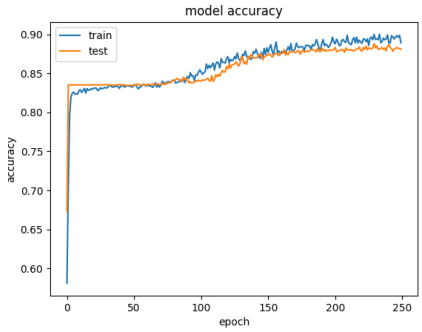

Figure A.59: Prediction using 20 minute time window, epochs:250, look back:5, train size:70%, timestemps to predict: 11, test accuracy: 85%

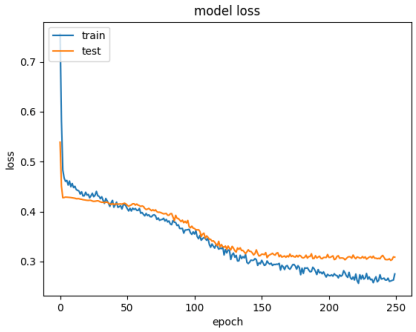

Figure A.60: Learning curve using 20 minute time window, epochs:250, look back:5, train size:70%, timestemps to predict: 11, validation loss: 0.354

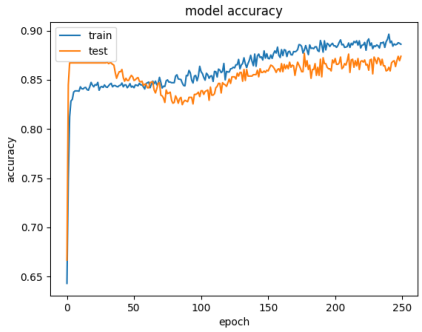

Figure A.61: Prediction using 25 minute time window, epochs:250, look back:8, train size:70%, timestemps to predict: 8, test accuracy: 85%

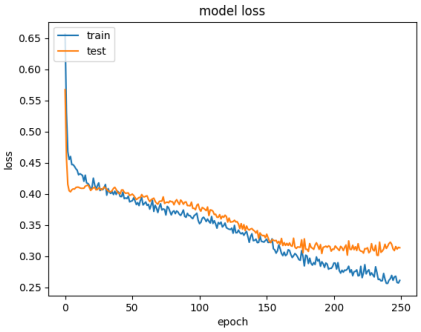

Figure A.62: Learning curve using 25 minute time window, epochs:250, look back:8, train size:70%, timestemps to predict: 8, validation loss: 0.357

Figure A.63: Prediction using 15 minute time window, epochs:250 , look back:6, train size:70%, timestemps to predict: 11, test accuracy: 85%

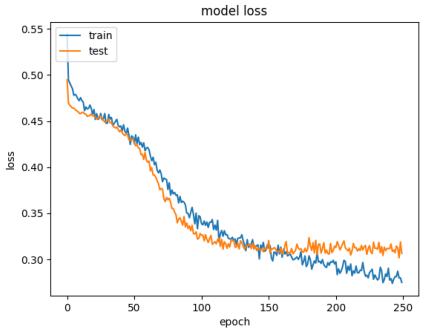

Figure A.64: Learning curve using 15 minute time window, epochs:250 , look back:6, train size:70%, timestemps to predict: 11, validation loss: 0.352

Bibliography

Abbaspour, Majid et al. (2015). “Hierarchal assessment of noise pollution in urban areas – A case study”. In: *Transportation Research Part D: Transport and Environment* 34, pp. 95 –103. issn: 1361-9209. doi: https://doi.org/10.1016/j.trd.2014.10.002. url: http://www.sciencedirect.com/science/article/pii/S1361920914001436.

Bastião, Luís et al. (2014). “Medical imaging archiving: A comparison between several NoSQL solutions”. In: *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pp. 65–68.

Bengio, Y., P. Simard, and P. Frasconi (1994). “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. issn: 1045-9227. doi: 10.1109/72.279181.

Box, G.E.P. and G.M Jenkins (1970). “Time Series Analysis: Forecasting and Control.” In: *Francisco: Holden-Day. (Revised edition published 1976)*.

Carrealtime (Digital age transportation). In:

Cheng, Xingyi et al. (2017). “DeepTransport: Learning Spatial-Temporal Dependency for Traffic Condition Forecasting”. In:

Dunne, S. and B. Ghosh (2013). “Weather Adaptive Traffic Prediction Using Neurowavelet Models”. In: *IEEE Transactions on Intelligent Transportation Systems* 14.1, pp. 370–379. issn: 1524-9050. doi: 10.1109/TITS.2012.2225049.

Epelbaum, Thomas et al. (2017). “Deep Learning applied to Road Traffic Speed forecasting”. In:

G. Polson, Nicholas and Vadim Sokolov (2017). “Deep learning for short-term traffic flow prediction”. In: 79, pp. 1–17.

Hennessy, Dwight A. and David L. Wiesenthal (1999). “Traffic congestion, driver stress, and driver aggression”. In: *Aggressive Behavior* 25.6, pp. 409–423. issn: 1098-2337. doi: 10.1002/(SICI)1098-2337(1999)25:6<409::AID-AB2>3.0.CO;2-0. url: http://dx.doi.org/10.1002/(SICI)1098-2337(1999)25:6<409::AID-AB2>3.0.CO;2-0.

Hochreiter, Sepp and Jürgen Schmidhuber (1997a). “Long Short-Term Memory”. In: *Neural Comput.* 9.8, pp. 1735–1780. issn: 0899-7667. doi: 10.1162/neco.1997.9.8.1735. url: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

— (1997b). “Long Short-Term Memory”. In: *Neural Comput.* 9.8, pp. 1735–1780. issn: 0899-7667. doi: 10.1162/neco.1997.9.8.1735. url: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hong, W. C. et al. (2010). “Seasonal adjustment in a SVR with chaotic simulated annealing algorithm traffic flow forecasting model”. In: pp. 560–565. doi: 10.1109/NABIC.2010.5716266.

Imagery, NIMA National and Mapping Agency (1984). “ECHNICAL REPORT 8350.2”. In: *Department of Defense World Geodetic System* 3.

INRIX (Global Traffic Scorecard). In:

Iokibe, T., N. Mochizuki, and T. Kimura (1993). "Traffic prediction method by fuzzy logic". In: *[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems*, 673–678 vol.2. doi: 10.1109/FUZZY.1993.327408.

Jia, Yuhan, Jianping Wu, and Ming Xu (2017). "Traffic Flow Prediction with Rainfall Impact Using a Deep Learning Method". In: 2017, pp. 1–10.

Jo, Youngtae and Inbum Jung (2014). "Analysis of Vehicle Detection with WSN-Based Ultrasonic Sensors". In: *Sensors* 14.8, pp. 14050–14069. issn: 1424-8220. doi: 10.3390/s140814050. url: http://www.mdpi.com/1424-8220/14/8/14050.

Kalman, R. E. (1960). "A New Approach to Linear Filtering and Prediction Problems," Transactions of ASME, Series D, Journal of Basic Engineering". In: 82.1. doi: 10.1115/1.3662552.

Keay, Kevin and Ian Simmonds (2005). "The association of rainfall and other weather variables with road traffic volume in Melbourne, Australia". In: *Accident Analysis and Prevention* 37.1, pp. 109 –124. issn: 0001-4575. doi: https://doi.org/10.1016/j.aap.2004.07.005. url: http://www.sciencedirect.com/science/article/pii/S0001457504000624.

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980. arXiv: 1412.6980. url: http://arxiv.org/abs/1412.6980.

Kleusberg, Alfred and Peter J. G. Teunissen (eds.) (1998). *GPS for Geodesy*. Springer. isbn: 978-3-642-72011-6. url: http://www.springer.com/la/book/9783642720130.

Lu, C. N., H. T. Wu, and S. Vemuri (1993). "Neural network based short term load forecasting". In: *IEEE Transactions on Power Systems* 8.1, pp. 336–342. issn: 0885-8950. doi: 10.1109/59.221223.

Luxen, Dennis and Christian Vetter (2011). "Real-time routing with OpenStreetMap data". In: GIS '11, pp. 513–516. doi: 10.1145/2093973.2094062. url: http://doi.acm.org/10.1145/2093973.2094062.

Pan, Bei et al. (2013). "Crowd Sensing of Traffic Anomalies Based on Human Mobility and Social Media". In: SIGSPATIAL'13, pp. 344–353. doi: 10.1145/2525314.2525343. url: http://doi.acm.org/10.1145/2525314.2525343.

Prof. Dr. Bernhard Hofmann-Wellenhof Dr. Herbert Lichtenegger, Dr. James Collins (auth.) (1993). *Global Positioning System: Theory and Practice*. Springer Vienna. isbn: 978-3-211-82477-1,978-3-7091-3293-7. url: http://gen.lib.rus.ec/book/index.php?md5=fe3a28d2673b37dcec216ba7a874930a.

Schmidhuber, Jürgen (1990). "Learning Algorithms for Networks with Internal and External Feedback". In:

Song, Xuan, Hiroshi Kanasugi, and Ryosuke Shibasaki (2016). "Deeptransport: Prediction and Simulation of Human Mobility and Transportation Mode at a Citywide Level". In: IJCAI'16, pp. 2618–2624. url: http://dl.acm.org/citation.cfm?id=3060832.3060987.

Srivastava, Nitish et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15, pp. 1929–1958. url: http://jmlr.org/papers/v15/srivastava14a.html.

WEF (Cities are tackling traffic). In:

Willard, Brandon (2014). "Real-time On and Off Road GPS Tracking". In:

Yanminsun, Andrew Wong, and Mohamed S. Kamel (2011). "Classification of imbalanced data: a review". In: 23.

Yi, Hongsuk, HeeJin Jung, and Sanghoon Bae (2017). “Deep Neural Networks for traffic flow prediction”. In: pp. 328–331. doi: 10.1109/BIGCOMP.2017.7881687.

Zeng, Dehuai et al. (2008). “Short Term Traffic Flow Prediction Using Hybrid ARIMA and ANN Models”. In: PEITS ’08, pp. 621–625. doi: 10.1109/PEITS.2008.135. url: http://dx.doi.org/10.1109/PEITS.2008.135.

Zheng, Yu (2015). “Trajectory Data Mining: An Overview”. In: *ACM Trans. Intell. Syst. Technol.* 6.3, 29:1–29:41. issn: 2157-6904. doi: 10.1145/2743025. url: http://doi.acm.org/10.1145/2743025.