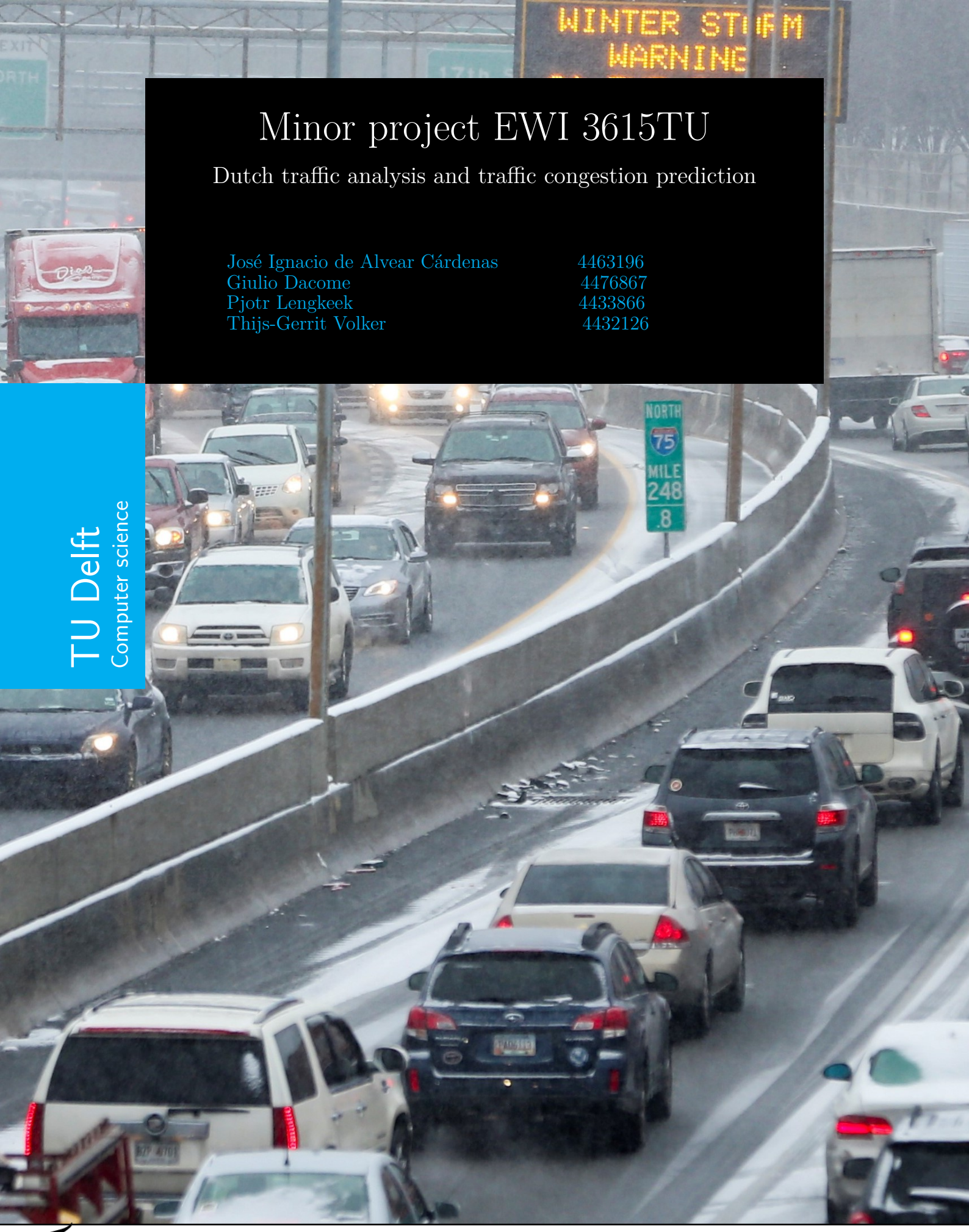

Minor project EWI 3615TU

Dutch traffic analysis and traffic congestion prediction

José Ignacio de Alvear Cárdenas	4463196
Giulio Dacome	4476867
Pjotr Lengkeek	4433866
Thijs-Gerrit Volker	4432126

TU Delft
Computer science

TU Delft
Computer Science

January 15, 2018

Minor project EWI 3615TU: Dutch traffic analysis and traffic jam prediction

Minor project Software Design and Application 2017-2018

José Ignacio de Alvear Cárdenas	4463196
Giulio Dacome	4476867
Pjotr Lengkeek	4433866
Thijs-Gerrit Volker	4432126

The link to the group's GitHub repository is the following:
https://github.com/plengkeek/Minor-Project-Group-2
Course responsible lecturer: Dr. Georgios Gousios

Contents

1. Introduction

In a densely populated country such as The Netherlands, it is the case that a large number of people needs to commute daily for a wide variety of reason. In particular, according to the *Transport and Mobility* report of 2016, compiled by CBS Nederlands [1], commuters mainly choose the car as their mean of transportation (either as driver or passenger): more than half of the total distance travelled by Dutch daily travellers is, in fact, covered by car. It is a well-known fact that traffic at rush-hour in the regions of Utrecht, Noord- and Zuid-Holland is extremely heavy, due to the high number of commuters on the roads [1]. Due to traffic flow exceeding the network capacity in the aforementioned areas of the network, this necessarily results in the generation of tailbacks and traffic congestion. Hence, it is convenient to have a tool which enables the drivers to identify the most critical sections of the network (in terms of traffic) so to, if necessary, select an alternative route to get to their destination.

This report is meant to explain to the reader the decisions and the process that the group followed in order to develop the model mentioned in the previous paragraph. To achieve this goal, it was decided to make use of the data generated by the incredibly large array of sensors spread throughout the network (more that 60,000 of them) and collected by the *Nationale Databank Wegverkeergegevens* (NDW). The group was able to get hold of one year of traffic data, and continuously download and store new data throughout the duration of the project. The data was then processed and fed to machine learning and statistical models, so to extrapolate relevant trends and construct a prediction and analysis.

The report is structured in six sections. The second chapter provides a clear statement of the objectives of this project, together with data sources and tools used by the group. The third chapter provides insight on the elements of traffic theory used to develop the models and how the measurement on the Dutch network are structured. The fourth and fifth chapter explain how the data was actually collected and processed, respectively. In particular, chapter five illustrates how the statistical and prediction models were developed. Important to notice is that the report does not contain a separate section for the results. This decision was taken due to the fact that every section contains partial results.

2. Initial organization

2.1 Objectives

For this project, two milestones have been defined with the aim of applying Big Data processing and machine learning/artificial intelligence techniques to traffic and weather data.

First of all, the team will perform an analysis on the Dutch traffic data and the weather information collected. The goal of the analysis is to identify which areas in the Dutch road network are the most dangerous ones, based on the numbers of accidents reported in each of them. Within that frame, three tasks have been defined:

1. **Filtering**: the data will be collected, filtered and structured in order to facilitate its manipulation and reduce the memory storage required.
2. **Statistics**: with the data collected, statistical conclusions will be drawn. Both datasets, the traffic and weather data, will be merge bearing in mind time synchronization. During this stage multiple graphs and tables will be generated.
3. **Results**: conclusions will be drawn and documented in a report with a thorough explanation of the visuals created.

Secondly, the group will apply artificial intelligence techniques (such as an Artificial Neural Network) in order to generate a model that could predict future traffic jams. Furthermore, a trend-based model is applied to the output of the machine learning algorithm to understand how the situation will evolve. The ANN will be trained with historical data and the group aims at implementing the final model in a web service that is fed with live data.

2.2 Data

For this assignment, the team requires historical and live data of the Dutch traffic and the weather conditions on the road.

For the first one, the group retrieves the information from the Nationale Databank Wegverkeergegevens (NDW); the data is collected in an XML format. More insight is provided in chapter 3.

With respect to the second information required, the weather data is collected from the Koninklijk Nederlands Meteorologisch Instituut (KNMI) in a text format.

Besides the previous data banks, the schedule of the yearly Dutch holidays will be also included since it will help in the prediction of traffic jams of the AI model.

2.3 Tools

The following software tools will be used in order to carry out the project successfully:

1. **Python**: programming language key for the processing of the data.
2. **Spark**: open-source cluster-computing framework that will be extensively used for the processing of the data. The Spark software also provides us with a set of components, such as the Spark Streaming, the MLLib and the GraphX; which will help with the data analysis and the creation of the artificial intelligent model.
3. **Keras**: Python API for Tensor Flow: a library for artificial neural networks.
4. **GitHub**: development platform to ease the collaboration of the team members to the project's code.
5. **Stack**: online server for the storage of high amounts of data.

3. Traffic Theory

The purpose of this chapter is to provide an overview of the traffic measurement systems in the Netherlands. Furthermore, this chapter contains a recap of the main theoretical concepts used to complete the project.

3.1 Traffic Theory

Multiple theories describing the behaviour of vehicular traffic and generation of a congestion are available. The group decided to make use of *Kerner's three-phase traffic theory*, developed by Boris Kerner between 1996 and 2002. The main point of this theory is that a vehicular flow can be classified in two main categories:

1. **Free flow**: characterized by a region of positive slope on the density-flow rate graph, as illustrated in figure 3.1. Physically, this corresponds to a situation in which the addition of vehicles in the stream does not affect its average speed over time. As the flow rate increases, though, the traffic approaches a critical state, the overcoming of which leads into the *congested* state. By inspection of figure 3.1, it is possible to notice how the transition is fairly sharp: the addition of only a few vehicles causes transition into the congested zone.
2. **Congested**: this classification identifies a state in which any increment of density will cause a decrease in flow rate, and hence a reduction in the average speed of the flow. This particular condition can be identified by measurement points with negative correlation in the flow-density graph and, as can be appreciated from figure 3.1, the data is more sparse. This phase can be further into two sub-categories:
   - **Synchronized flow**: the speed of the flow drops significantly, but there is no major effect on the traffic flow yet.
   - **Wide moving jam**: significant drop in the average speed of the vehicles in the flow.

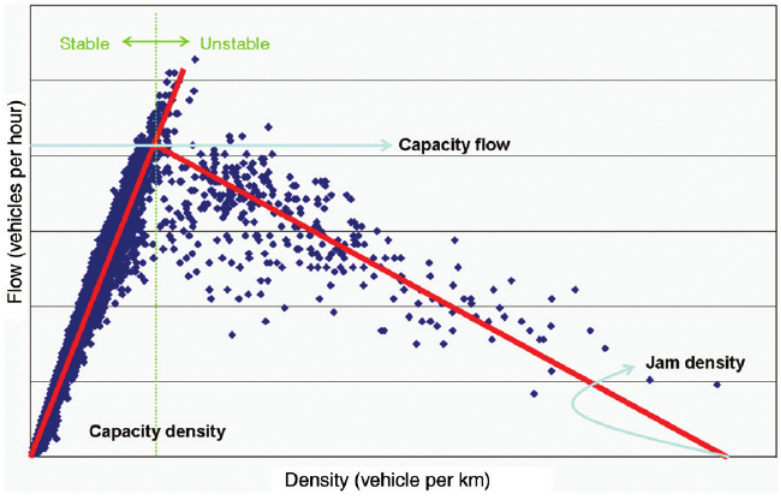

Figure 3.1: Example of a density-flow graph

The figure above clearly explains the dynamics of traffic in terms flow versus density. However, it does not show how the average speed evolves in time, which is a more understandable measure of how the dynamics evolve. Considering this aspect, a plot such as the one in figure 3.2 can be generated. The top part of it shows a so-called *traffic wave* propagating through time and space. Instead, the bottom part shows a sections of the three-dimensional plot at a specific location, with time as running variable.

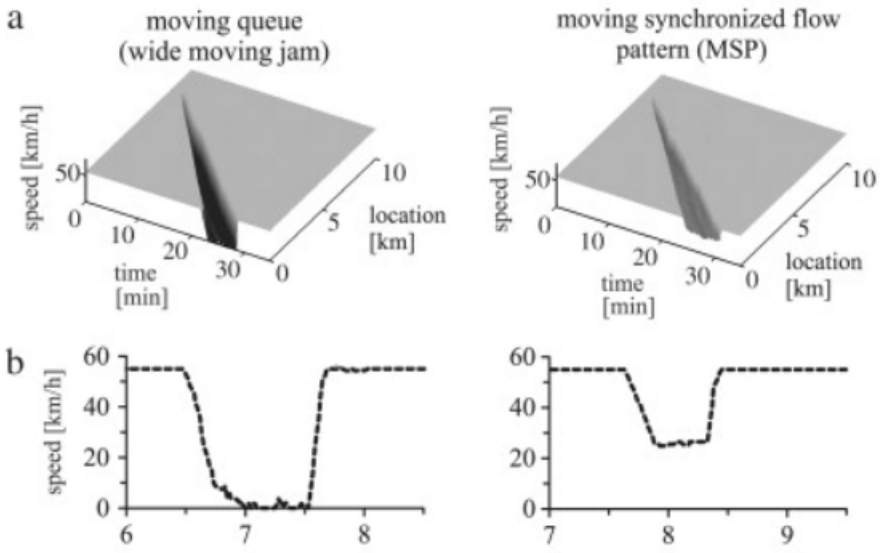

Figure 3.2: Plot of traffic speed versus time and location

Figure 3.2 is equivalent to figure 3.1. In fact, it can be seen how the traffic suddenly transitions from a free-flow state (high average speed) to a congested situation (drop in average speeds).

3.2 The Dutch measurement system

Traffic information, together with information regarding the measurement system installed on the network was retrieved from the NDW, the *National Data Warehouse for Traffic Information* [2]. The sensor array consists of approximately 60,000 sensors spread throughout the Dutch roads and highways. The majority of the sensing instruments are located on highways.

The NDW provides a file containing the information of the sensors; including its location with longitude and latitude coordinates (e.g. 52.1223,4.49208), its location within the Dutch road network (e.g. N206 km 8.024) and the amount of lanes from which it takes the measurements (e.g. 3 lanes). The interpretation of the value is explained in the a file provided by the NDW [3]. Within the scope of each sensor, there is a varying number of indices or sensor measurements. Each of them measures "TrafficFlow" (traffic flow in a given measurement period) or "TrafficSpeed" (average traffic speed in the same time interval).

Besides that, the sensor measurements can be classified in two groups: those that only take into account for their computations vehicles of certain length and those that take into account any type of vehicle. For this project, only the sensor measurements that used all the vehicles were considered, since the information regarding the vehicles' length does not provide new useful insight.

Historical data is also provided as files containing the speed and traffic flow measured by each of the sensors in the time span of 1 minute. As with the sensors' file, each of the sensors contain multiple indices that store the information classified by lanes and vehicle size.

Finally, the NDW also provides live data regarding the accidents occurred on the Dutch traffic network; including location, time stamp and type of accident.

3.3 Analysis of the data

Given that all the files were obtained in XML format, a Python code had to be devised such that the useful fields of each file were collected in a TXT format for further analysis.

As explained in section 3.2, only the sensor measurements that contain information about all type of vehicles were used, so a filter had to be applied to the sensor file. The output was a text file containing the location, position within the Dutch traffic network and number of lanes of each of the sensors. Besides that, the file also contained the information of the sensor measurements that took into account any type of vehicle.

The file of the sensors provided the information of which sensor measurements had to be read and what kind of information they contained (traffic flow or traffic speed). Once reading the generated text file, two dictionaries were created within the Python code: both with the id of the sensors as keys, one with the indices of the sensor measurements that contained traffic flow information as values and the other with the indices of the sensor measurements that contained

traffic speed information as values.

The "TrafficSpeed" files were also retrieved as XML documents, so when converting them to TXT the following fields were obtained separated by a semicolon (;): time of measurement, name of the sensor and the information of each sensor measurement. In order to have a consistent format in the text file for further analysis, each sensor measurement had 2 numbers: the first one was for traffic flow and the second one was for traffic speed. Given that the sensor measurements contain either traffic flow or traffic speed, there will be always one of the two numbers for which there is no information. If that is the case, -2 is filled in where the information is missing.

With the devised dictionaries, the correct sensor measurements from the "TrafficSpeed" text file were read. The objective was to compute an average speed for each of the sensors in the Dutch network. Given that the sensors had multiple measurements depending on the lane, a weighted average had to be computed using the traffic flow as weight. As a result, the traffic flow in each of the lanes was divided by the total amount of vehicles between all the lanes; the percentage of vehicles per lane was computed. The resulting weights were multiplied by their corresponding traffic speeds and the outcomes were summed.

$$V_{avg} = \frac{\sum_{l=1}^{n} V_l \cdot TF_l}{\sum_{l=1}^{n} TF_l} \tag{3.1}$$

Equation 3.1 shows how the average speed for one sensor was computed. l represents the lane and n the total amount of lanes for that sensor. V_l is the traffic speed for that lane and TF_l is the traffic flow for that same lane.

In order to have a visual understanding of the data retrieved, the results for a single minute were plotted in Google Maps, as can be observed in Figure 3.3. This was done by writing an HTML file with Python, allowing the user to customize the plotting by means of predefined variables. The colour scheme (red-blue-green) was selected such that red symbolizes low speed (danger for the current research since it may represent a traffic jam) and green represents high speed. Blue is the colour used for transition.

Figure 3.3: Map of the Netherlands with speed measurements.

4. Data Collection

4.1 Historical Data

Historical data could be retrieved with the NDW open data service. This service comes in the form of a webform. In this webform it is possible to specify the days and type of data. The service comes with some limitations. Every request can have a length of 7 days with max 3 request per ip-address. In addiction, the download speed of every request is limited to $300kB/s$. Every day with datatype intensity and speeds is approximately $1.2GB$ of data. With these limitations it is possible to estimate the amount of time needed to collect all the data. $(1.2GB * 365days/0.9 * 10^{-3}GB/s)/3600/24 \approx 6days$ Now it is clear that 6 days of continuous supervision of a human is not feasible and that the retrieval of data had to be automatized.

A script was made in Python to automatically fill the webform and download/upload the correct files. Below is the pseudo-code of the script:

```
open webbrowser;
for each day from 01-01-2017 to current day do
    for traveltime and travelspeed do
        load website;
        load webform;
        save captcha;
        solve captcha;
        fill webform;
        submit webform;
        while link is not clickable do
            wait 1 minute;
        end
        download file;
        upload file to STACK;
    end
end
```

Algorithm 1: NDW Webbot

The NDW uses CAPTCHAS to protect themselves against DDOS attacks. This meant that the script had to be able to solve these CAPTCHAS. The automatic solving of the CAPTCHAS was done with help of an external service called 2captcha. This external service is a paid service where you can send CAPTCHAS with use of an API and receive the answer.

4.2 Live Data

Live data could be retrieved from a FTP server that NDW is hosting. The files hosted on the FTP server are updated every minute. Another script was made to download the traveltime, travelspeed and incident data every minute and upload it to STACK. The incident data was also processed directly and uploaded to STACK. Below is the pseudo-code of the script:

```
connect to ftp;
while True do
    for file in files_to_download do
        download file;
        extract file;
        process file;
        upload file to STACK;
    end
    wait 1 minute
end
```

Algorithm 2: FTP Script

5. Data Processing and Results

This chapter is meant to provide the reader insight on how the gathered data was processed and fed into machine learning and statistical models in order to produce relevant results.

5.1 Road Safety Statistics

In this section, the reader is provided insight on how the data was processed in order to understand in which areas of the Dutch road network the most crashes occur.

As mentioned in section 3.2, the Dutch National Database provides data not only about the traffic's main dynamic variables (i.e. traffic speed and flow rate), but also about accidents recorded on the network. The data are given in XML format and provide the following information.

- **ID**: sequential identification code of the situation.
- **Time stamp**: the date and time when the situation was reported.
- **Location**: the coordinates of the accidents (latitude, longitude).
- **Type of accident**: either "broken down vehicle" or "other".

A python script had to be written that would convert the XML files into CSV files, which take python less effort to read compared to the original format. Next, in order to identify the most critical regions in the network, the group made use of a clustering algorithm called *K-means*. This algorithm enables grouping of data points in a high-dimensional vector space, given an initial data set and number of clusters to generate. A brief overview of the working principle of this algorithm is provided below, taken from A. Stevino's *Introduction to K-means Clustering* [4].

1. **Input**: in order to run, the algorithm requires the following items.
   (a) Multidimensional data set; in the present case, the algorithm operates in a two-dimensional domain (latitude,longitude).
   (b) Number of clusters the user wants the algorithm to generate.
2. **Generate initial centroids**: centroids are the centres of the clusters. The initialization is random within the vector space.
3. **Assign each data point to the closest centroid**: the assignment is executed by computation of the Euclidean distance (equation 5.1) of each point to the centroids.

$$d(p,q) = \sqrt{\sum_{n=1}^{N} (p_i - q_i)^2} \tag{5.1}$$

4. **Cluster generation**: the point will belong to the cluster having, as centroid, the closest centroid, as computed with equation 5.1.
5. **Compute new centroid**: once all points are assigned to a cluster, all centroids are recomputed as the average of the coordinates of all points belonging to the same cluster.
6. **Loop**: steps 2 through 5 are repeated until there is no change in the centroids anymore.

The result of this was then visualized on the map, to highlight the most critical areas of the network. The clusters are represented with circles centred at the coordinates of the centroid, and have a diameter which is proportional to the number of points contained in each group. The optimal number of clusters the system should generate was determined by trial-and-error: visual inspection of the outcome on the map and subsequent manual modification of the number of groupings to create. Still, as it can be seen in the Figure 5.1, some clusters are located in an area where almost no roads are present. This is due to the fact that some accidents were recorded in neighbouring areas and have a high degree of spread; the coordinates of their average point is therefore located in very low-density areas. However, the system is indeed able to precisely identify the areas where the most situations are reported: on the roads close to Amsterdam, The Hague, Rotterdam and Utrecht. Due to the high population density, these areas are the ones where the most road traffic is concentrated. Thus, it makes sense that the most road accidents reports come from these regions.

Figure 5.1: In red the accidents in five days in the Dutch road network and in blue the location of the centroids with their size proportional to the amount of points they enclose.

5.2 Prediction of Congestion

In this section the reader can find details about how the group developed a model to predict the presence of a traffic congestion in the proximity of each valid sensor.

5.2.1 Data Preparation

As stated in section5.1, XML files from the NDW were converted to CSV files for easier use within Python. However, these CSV still only contain the raw measurements and need to be combined with data from the KNMI to train the model which is described in section 5.2.2.

Obviously, the traffic camera's do not register it's weather conditions on site. Therefore the data from the KNMI needs to be linked to those camera's. Because the weather stations are not close to the traffic sensors, linear interpolation between stations was used to get the weather information at the location of each sensor. This is done by selecting the three closest weather stations for every traffic camera and computing the weighted average of the weather measurements based on distance from the camera. In this way, it is taken into account that weather is localized and tries to compensate for it.

Furthermore, the machine learning model in section 5.2.2 is designed to output a single speed value on a specific location. Therefore, to train the model, a single value needs to be inserted as a target. However, the CSV files provide speed and traffic flow values for every lane. To obtain a single value, a weighted average is computed. This is done following equation 3.1 in section 3.3.

5.2.2 Machine Learning

The goal of the machine learning is to predict if there will be a traffic jam in the near future. The assumption is made that the data is correlated. The correlation lies between the traffic's main dynamic variables (i.e. traffic speed and flow rate) and the running variables (location, time and weather). In order to predict the future behaviour of traffic, this correlation has to be found. Due to the extremely high number of points and complexity of the fitting model, a neural network is the best possible algorithm that one can select.

An initial intent to execute a machine learning model on the data was attempted with a standard multi-layer perceptron architecture. This was achieved by making use of pre-implemented models from the SparkML library. The network was implemented with one hidden layer having neurons with the ReLu function as their activation function. This model showed a descending learning curve (meaning the error was eventually converging to zero), but the performance was not as desired. The problem was quickly identified to be in the data set itself. It can be observed that the data is sequential and time-dependent. This means that feed-forward neural networks, such as Multi-Layer Perceptrons, will not suit the problem at hand as they are designed for linearly separable time-invariant problems. It is for this reason that, after some investigation, it was decided to switch to the use of a Recurrent Neural Network. In particular, a Long Short-Term Memory (LSTM) network was used. These networks produce one output per processed time step, which is taken as input for the next one. These networks are designed to work with sequential time dependent data.

5.2.3 Statistical Modelling

As stated in section 5.2.2, the machine learning algorithm gives, as output, the predicted average speed for each processed sensor. However, at this stage, the system is not able to tell what type of traffic flow (i.e. free or congested flow) the data represents yet. In order to achieve this goal, a model was designed, based on trends extrapolated from the data. Before going into the details of the model, one has to remember what the final output needs to be:

- Prediction of whether or not a traffic jam will be present at the designated location.
- In the case of a positive outcome, the prediction shall include when the congestion will form.
- A measure of the error on the aforementioned prediction.

In order to detect the presence of a congestion, it was decided to compare the predicted speed with a certain threshold, taken as percentage of the average speed recorded for each sensor over a period of two weeks, each of which located in a different season of the year in order to avoid landing on data outliers (bad weather, start of the holiday season, end of the holiday season) that do not represent the whole data set. To achieve this, a python script was written, which iterated through the data files generated in the two selected weeks, calculating the average of all weighted averages of every sensor, computed with equation 3.1. For many data points, velocity values were inconsistent. Therefore, they were eliminated from the data.
For the determination of a reliable threshold, it was decided to look into statistical data of reported speeds in traffic jams. In particular, a document was found containing data about traffic congestion occurring in the Ile-de-France (the Paris region) in France [5]. From there, it was possible to understand that the average velocity of vehicles in congested traffic is approximately 40% of the velocity recorded during free flow. One limitation of this model is that it was recorded mainly for tailbacks generating due to traffic overflowing the bottleneck capacity (i.e. rush-hour traffic). Therefore, it does not reflect the queuing dynamics of congested traffic whose cause is, for example, an accident or another sudden and unpredictable event. However, according to the U.S. Department of Transportation [6], the majority of the recorded traffic jams are generated due to traffic demand exceeding the bottleneck capacity.

It is now necessary to briefly digress on how the probability of formation of a tailback is calculated. It was decided to consider four data points: the ones corresponding to instants in time 12 to 15 minutes in the future: the last three data points given by the machine learning algorithm. The probability of occurrence of a congestion ($P(jam)$) was decided to be the product of the *coefficient of determination* (R^2) of the line fitting the three data points and a time-dependent probability: a value depending on how far in time the congestion is predicted to occur (P_1). This last factor is formally defined as:

$$P_1 = (T_{max} - T_{forecast})/T_{max} \tag{5.2}$$

being T_{max} a time instant beyond which the prediction is considered completely unreliable (i.e. $P(jam) = 0$, in the present case it equals fifteen minutes) and $T_{forecast}$ the time instant at which the model is predicting the congestion to occur. Therefore, the probability of occurrence of a

traffic jam is formally defined to be the following:

$$P(jam) = P_1 \cdot R^2 \tag{5.3}$$

Therefore, this model sees three possible use scenarios, explained in detail below.

1. **The output of the machine learning algorithm shows a speed which is already below the congestion threshold.** In this case, the model will estimate the presence of a traffic jam with a probability of 100%.

2. **The three data points are above the threshold, and show a monotonously descending trend.** In this case, the scenario is further split in two possibilities.

   (a) *The model is predicting that a jam will form more than fifteen minutes beyond the output of the machine learning.* In this case, the time-dependent probability will be null, and no traffic jam is predicted in the considered time interval (30 minutes in the future from the present instant).

   (b) *The model is predicting that a jam will form within the next fifteen minutes.* The forecast time is valid. Hence, the model will signal the presence of a congestion at the given moment in the future, and compute its probability of occurrence with equation 5.3.

3. **The three data points are above the threshold, and show a monotonously ascending or ambiguous trend.** If the three points show an ascending trend or no trend at all, then the model predicts that no traffic jam will form at the designated location.

Once the estimation was completed, it was decided, as for previous sections of the current project, to visualize the outcome on the map. This time, the colours refer to the probability of formation of a jam at that specific location. The colour scheme (red-blue-green) goes from red (maximum probability of congestion) to green (lowest probability). The result of this (run on a randomly generated file) is shown in figure 5.2.

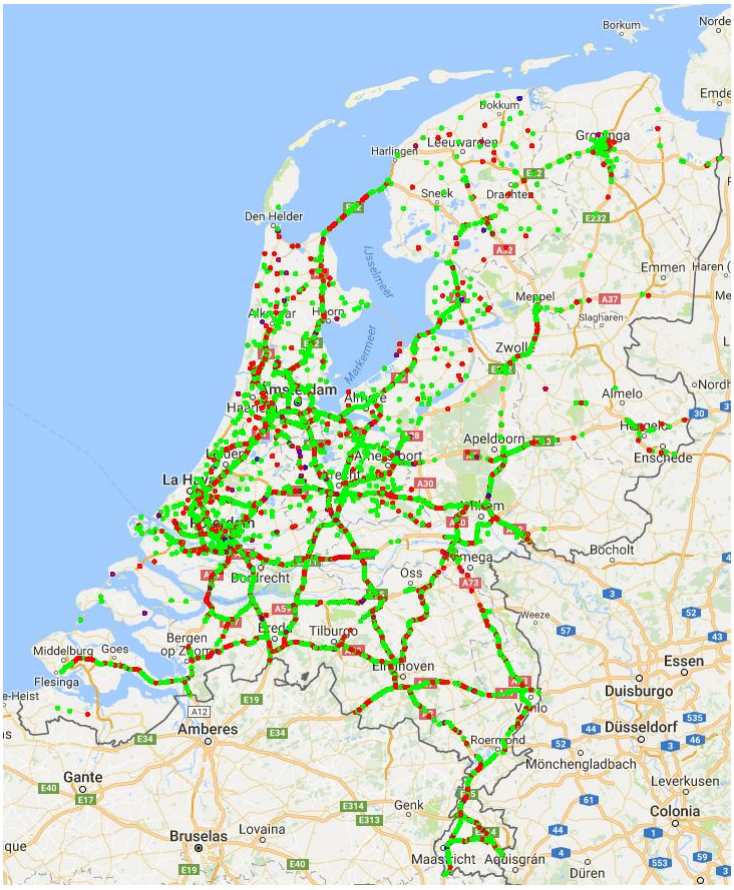

Figure 5.2: Map showing congestion probabilities

5.3 Stream processing

The previous sections of this chapter have described the development and implementation of the various parts of the system. However, at the current stage, all parts are not able to cooperate

yet. It is therefore required to design a system that fuses all parts together into one, coherent system. In order to accomplish this, it was decided to make use of multi-threading techniques. This allows to organize the execution of the program such that the output of each phase is taken as input for the following one. The current system requires three threads, described in detail below.

1. **Web Access**. The first phase consists of accessing the NDW website in order to download all relevant data from it. This is accomplished by making use of an FTP protocol, as explained in section 4.2.

2. **Data-frame**. This intermediate is needed in order to prepare data for feeding into the trained machine learning model. In this step, the data generated by step 1 is converted to an array.

3. **Machine Learning**. Once the relevant data has been downloaded, it should be fed into the trained model to produce the prediction for the next fifteen into the future.

4. **Trend-based model**. Finally, the aforementioned prediction is fed into the trend-based model developed as explained in section 5.2.3 and plotted on the map.

It is necessary to mention that all parts were connected through queues. For the present case, it was decided to generate two different queues: one to connect the first and second threads, and one to connect the second and third threads.

6. Conclusion

To finalize the design and the implementation of the system described in the previous chapters of the current report, this chapter provides a summary of the decisions taken and the procedures followed by the group to complete the process, together with an overview of the most meaningful results.

The goal of the project was to perform mainly two tasks: perform a statistical analysis of road traffic accidents within the Dutch network and develop a model that would predict, given a certain day, time and weather conditions, the occurrence of a traffic congestion at a given location on the network. All relevant data for both applications was downloaded from the NDW. To achieve the first goal, data concerning accident reports was processed and clustered using the K-means algorithm to identify the most critical regions of the network. For the second, historical traffic and weather data (taken from the KNMI) were used in order to train an artificial neural network, which would output predicted traffic flow and speed in the next fifteen mintues. To that prediction, the group then applied a trend-based model to better classify the output into a specific traffic state (free flow or congestion).

The group managed to accomplish the objectives set in the first place, it is the case that some parts could have been engineered in a more efficient way. For instance, conversion of the XML files to CSV might have been done using the make module of UNIX, and UNIX itself could have been used to great extent, in general, for the processing and analysis of data. Furthermore, in the context of Spark, the group could have tried to configure a cluster to expedite the computation.

Finally, the group wishes to mention what the possible future developments are of the system that developed throughout the project. In particular, the software could be made interactive, such that it would provide drivers with traffic information in real time, and suggest alternative routing. Furthermore, it could be tuned such that the output of the machine learning corresponds to the optimal speed the driver should maintain in order to avoid congestion. This last computation can also be optimized by making use of other computational intelligence techniques, such as particle swarm optimization.

Bibliography

[1] C. Netherlands, "Transport mobility 2016," 2016.

[2] NDW. (2017) Database. [Online]. Available: http://www.ndw.nu/pagina/en/78/database/

[3] T. Delissen, "Datex ii dutch profile," 2015. [Online]. Available: http://www.ndw.nu/pagina/en/20/brochures/

[4] A. Trevino. (2016) Introduction to k-means clustering. [Online]. Available: https://www.datascience.com/blog/k-means-clustering

[5] V-trafic, "L'etat du trafic en ile-de-france," 2014.

[6] F. H. Administration, "Traffic congestion and reliability: Trends and advanced strategies for congestion mitigation," 2017.