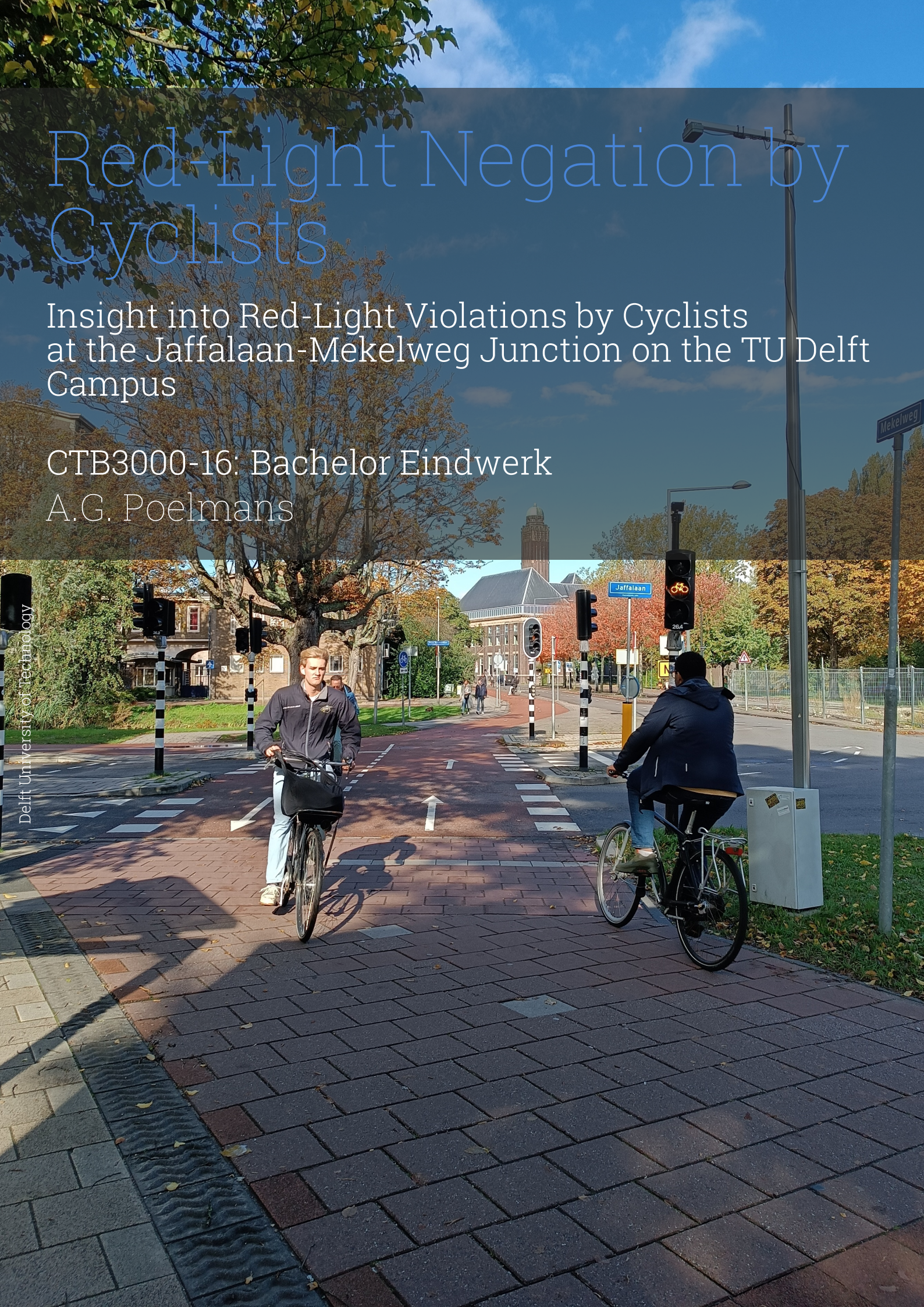
Red-Light Negation by Cyclists

Insight into Red-Light Violations by Cyclists at the Jaffalaan-Mekelweg Junction on the TU Delft Campus

CTB3000-16: Bachelor Eindwerk

A.G. Poelmans



Delft University of Technology

Red-Light Negation by Cyclists

Insight into Red-Light Violations by Cyclists at the Jaffalaan-Mekelweg Junction on the TU Delft Campus

by

A.G. Poelmans

Student Name	Student Number
A.G. Poelmans	5173361

Supervisor/Examiner:	Dr. Y. Yuan
Second examiner:	Dr. S. Sharif Azadeh
Second supervisor:	Dr.ir. K. Kavta
Project Duration:	September, 2023 - October, 2023
Faculty:	Faculty of Civil Engineering and Geosciences, Delft

Cover: Cyclists Crossing the Jaffalaan by A.G. Poelmans

TU Delft

Preface

In front of you lies the report 'Red-Light Negation by Cyclists'. This report is my final work to obtain the degree Bachelor of Science in Civil Engineering at the Delft University of Technology. This thesis was carried out at the domain of Transport and Planning of the faculty of Civil Engineering and Geosciences.

This report might interest anyone who would like to gain a better understanding in what factors may influence cyclists to violate traffic signals and how red-light running behaviour can subsequently be predicted using linear regression analysis.

I would like to express my gratitude to my supervisors Dr. Y. Yuan and Dr.ir. K. Kavta for their guidance during this bachelor thesis project. In addition, I would like to thank Dr. S. Sharif Azadeh for examining this thesis. Also, I would like to thank Dr.ir A.M. Salomons and Dr.ir. W. Daamen for providing me the topic and data for this project. Lastly, I would like to thank my fellow students Daniël Rijnders, Tieme Van Hijum and Arend-Jan Timmermans for giving weekly feedback and helping me during this project.

A.G. Poelmans
Delft, October 2023

Summary

Cycling plays a crucial role in the daily commute of students and staff at the TU Delft campus, where many individuals rely on bicycles for transportation. The Jaffalaan-Mekelweg intersection, situated on the campus, is used by a large portion of university commuters on a daily basis.

It has been known for a while that this intersection is often subject to red-light negation by cyclists, which can result in unsafe situations. The fact that red-light violations occur regularly at this intersection makes it important to better understand the reasons why cyclists ignore traffic lights and what factors possibly influence them.

This research aims to find relationship between the frequency of red-light violations and various meteorological and traffic-related factors, leading to the following research question. Can a multiple linear regression model predict red-light violations by cyclists based on rain, wind speed, temperature, the frequency of passing vehicles and the frequency of red-lights at the Jaffalaan-Mekelweg intersection?

Data is gathered from the Royal Dutch Weather Institute (KNMI) and from the traffic light it self. Simple linear regression models are used to verify if the relationships between the predictor variables and the response variable are statistically significant. Finally, a multiple linear regression (MLR) model is built using machine learning to predict red-light violations based on the influence of predicting variables.

After finding hourly values of red-light violations and processing the data from the independent variables, the linearity between the response and predictor variables is assessed. The weather data show no linear relation with red-light running behaviour. Therefore the weather is not included in the MLR. On the other hand, the frequency of passing vehicles and the frequency of red-lights do show some linear relation. The resulting MLR model, based on the remaining predictor variables, is able to account for only 22.3% of the variance in red-light violations. This means 77.7% of the variance in red-light negations remains unaccounted for. Indicating that more predictor variables are needed to improve the MLR model. Despite the poor performance of the MLR model, machine learning is used to produce predictions based on the model. A part of the data is used to train the new model. The remaining part is used to test it. The results are visualised in Figure 4.16. It can be seen that the model tends to over predict the number of hourly red-light violations.

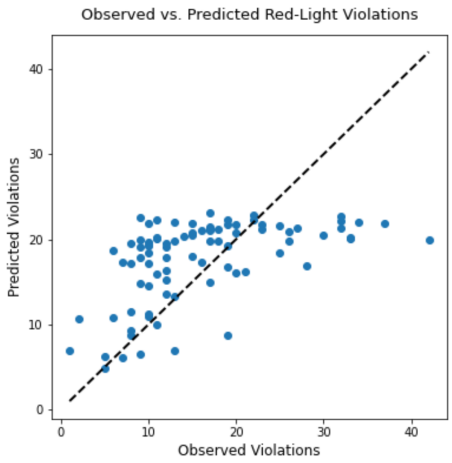


Figure 1: MLRM Scikit-learn results

Due to limited and inaccurate predicting powers of the model for red-light violations, no tangible conclusions can be drawn from it. More predictor variables, and data recorded over a longer period of time are needed to build a model that can accurately predict red-light violations.

Contents

1 Introduction

1.1. Problem Description

Many students and staff members cycle daily to and from the TU Delft campus, resulting in large cycling flows on the cycle paths. Predominantly during morning and afternoon peak hours and during lunch, bicycle lanes can become crowded. The Jaffalaan-Mekelweg intersection is located at the north side of the campus. All cyclists coming from or going to the train station or the city must pass this intersection. As a result, the Jaffalaan-Mekelweg is one of the busiest cyclist intersection in Delft and is therefore of great interest regarding cyclist safety.

In general, it is important to ensure that busy traffic intersections are safe because traffic incidents with cyclist remain a common phenomenon in The Netherlands. In 2022, 291 cyclists passed away after a traffic incident. That is 84 more victims than the year before and is in fact the highest number of casualties since 1996, the year registration was started (CBS, 2023). Also, in 2022, there were 79 more cyclist casualties caused by a crash with a car, van, bus, lorry or other vehicle than in 2021 (CBS, 2023). Implying that improving safety on bicycle lanes continues to be an important issue to society.

Besides that, it is fair to assume the traffic flows on the bicycle lanes of the TU Delft campus have increased over the last couple of years. According to data from the TU Delft, the total student population has increased from 23,604 students in December 2017 to 27,080 students in December 2022 (Delft, 2023). In the same period, the total number of staff employed by the TU Delft has increased from 5,210 to 6,647 and the total PhD population has increased from 2,719 to 3,144 (Delft, 2023). Increasing traffic flows on bicycle lanes will lead to an increased probability of accidents, underlining the importance of improving cyclists' safety on campus.

It has been known for some time that at this intersection not all cyclists comply with the traffic lights, which may result in unsafe situations and accidents. Research into the motivations of cyclist to ignore traffic signals can be of great importance to find ways to reduce traffic light violations.

Reducing the number of traffic violations at the Jaffalaan-Mekelweg intersection will be beneficial to many parties. First of all, the direct users of the intersection are important stakeholders. Cyclists will benefit from a safer traffic situation as the probability of accidents and injuries will decrease. In addition, drivers of motorised vehicles have interest in an improved traffic situation as the risk of collision with a red-light negating cyclist will become smaller. Secondly, traffic flow of motorised vehicles will run smoother if red-light violations are less frequent. Besides the direct users, the municipality and the TU Delft are important stakeholders as well. Both parties have interest in a safer intersection. The municipality is in general responsible for providing safe bicycle lanes and intersections. Also, the TU Delft is involved because the intersection is on the university campus and many of its daily users are students and faculty. Having safe infrastructure for students and faculty is understandably an important matter to the TU Delft.

1.2. Research Objective

This research aims to predict red-light violations at the Jaffalaan-Mekelweg intersection based on factors that may influence red-light running behaviour. As a result of the relatively short time period allotted for this project, only existing data is used for analysis. This means only publicly available weather data, data from the intersection and provided cyclist trajectory data can be used to discover factors of influence.

In this report the following research question will be answered. Can a multiple linear regression model predict red-light violations by cyclists based on rain, wind speed, temperature, the frequency of passing vehicles and the frequency of red-lights at the Jaffalaan-Mekelweg intersection? The research question will be answered with the aid of the following sub-questions:

1. How can the frequency of red-light infringements at the Jaffalaan-Mekelweg intersection be determined?
2. How can the predictor variables be preprocessed to allow for linear regression analysis?
3. Do linear relations exist between the predictor variables and red-light violations at the Jaffalaan-Mekelweg intersection?
4. To what extent can each of the predictor variables independently predict red-light violations at the Jaffalaan-Mekelweg intersection?

The research will be performed by analysing and quantifying data on traffic-light cycles, trajectory data of crossing cyclists and meteorological data. Subsequently, a multiple linear regression model will be performed. Results from the regression model will be visualised and interpreted.

The trajectory and traffic light data were by provided by Dr.ir. W. Daamen from the Transport & Planning department of the faculty of Civil Engineering and Geosciences, TU Delft. Trajectory data is only available for a period of roughly a month during the winter of 2021 and only cyclist coming from campus (south to north) were registered. Therefore, this thesis will be limited to analysing cyclists coming from the campus for this period of time.

1.3. Structure Description

The report of this Bachelor thesis is structured as follows. In Chapter 2 the literature review is presented. In this chapter the influence of various factors on red-light running is explained. Chapter 3 describes how the data was collected and processed, including the method on finding red-light violations. In addition, the method of the linear regression analysis is illustrated. Subsequently, in Chapter 4 the resulting red-light violations and the results of the linear regression analysis are demonstrated. Chapter 5 contains the discussion on the methodology and results of this research. Finally, in Chapter 6 conclusions from the results are drawn and recommendations are given. Also, possibilities for further research are mentioned in this chapter. In the appendices Python DataFrames and Python code is provided, accompanied with detailed descriptions.

2

Literature Review

Red-light running poses significant risks to cyclists and other traffic users, leading to an increased likelihood of collisions, injuries, and fatalities. Consequently, understanding the influential factors that contribute to this behaviour is crucial for designing effective countermeasures and to enhance overall traffic safety.

In this chapter an introductory examination is performed of influential factors associated with red-light running behaviour. Through literature research, we aim to provide valuable insights into the underlying causes and motivations that drive individuals to disobey red signals.

After conducting the literature research, a literature gap is found. In section 2.2 is described what this thesis aims to research and why this is deemed desirable and needful.

2.1. Influential Factors in Red-Light Running Behaviour

2.1.1. Demographic factors: gender, age, education and income

The demographic profile of the cyclists crossing the Jaffalaan-Mekelweg intersection can be useful to understand red-light running behaviour at this junction. For this thesis, four demographic factors are researched: gender, age, education and income. According to a study on red-light running by cyclists at five intersections in The Hague, men were 1.32 times more likely to run a red-light (Van der Meel, 2013). Another study, conducted in Beijing, China, also found men to be more likely to negate red traffic signals. From video observations it followed that 53% of men, and 43% of women violated a red light (Wu et al., 2012).

From the same study conducted in The Hague, followed that age also plays an important role in the likelihood of violating a red-light. From gathered observations in the study followed that 34.7% of young people (aged below 20) ignored a red-light. Of adults (20-65) and elderly (65+), 24.3% and 21.6% ran a red-light respectively (Van der Meel, 2013).

A study based on a national survey in Australia showed some interesting results regarding red-light negation and the level of education. It was found that the percentage of cyclists complying with traffic signals decreased as the level of education increased. Of cyclists with just secondary education, 66.8% is compliant. Of cyclists with a university degree or higher degree, 62.5% and 59.4% are compliant respectively (Johnson et al., 2013).

The same Australian survey also asked respondents about their income. However, the resulting percentages of compliance are less straightforward as for education levels. This is probably due to the lowest two income groups having relatively few respondents compared to the higher two income groups. Of the two higher income groups, 64.0% of the respondents with an income between $40.000-$99.999 obeys traffic signals. Whereas just 61.6% of respondents with an income over $100.000 obeys traffic lights (Johnson et al., 2013).

2.1.2. Socio-Psychological factors

Besides demographic factors, socio-psychological factors also play a role in red-light running behaviour. For this research we focus on the following socio-psychological factors: risk-perception, haste, cycling experience and herd behaviour.

Regarding psychological factors that influence red-light violating behaviour, risk-perception can be named as a factor of influence. In existing literature often a distinction is made between three groups of cyclists: risk-taking behaviour, opportunistic behaviour and law-obeying behaviour (Pai and Jou, 2014). They defined risk-taking behaviour as violating a red-light signal without stopping (but possibly slowing down). Pai and Jou (2014) classified 859 (6.9%) of in total 12,447 observations as risk-taking behaviour. They also found that cyclists who wore helmets, used bicycle lights and wore protective clothing or gloves consistently stopped more often at a red-light.

Not many studies have been publicised on the relation between haste and red-light negation by cyclists. Some literature is available on the relation between red-light negation and pedestrians. Two survey-based studies from the eighties state that haste is a significant factor for pedestrians to determine whether they ignore a traffic light or not. In two studies haste was named 19.9% and 36.1% by pedestrian as a reason to ignore a red-light (Van der Meel, 2013). From a survey-based research on Australian cyclists, 3.0% of the cyclist who indicated they run red-lights stated ‘when in a hurry’ as a reason (Johnson et al., 2013). Red-light violations in relation to waiting time might indicate how haste influences red-light running. In their research, Pai and Jou (2014) looked at signal violations categorised by waiting time. However, they did not obtain unambiguous results. They observed the most signal violations (9.12%) for a waiting between 46 and 99 seconds. After that comes 1-30 seconds with 7.77% and finally, 31-45 seconds with 5.98% violating a red-light. Making it difficult to draw conclusions from waiting time.

The influence of cycling experience on red-light running can be categorised into two groups. Namely, the years of experience an individual has with riding bikes. And secondly, the years of experience an individual has with a certain traffic situation (Van der Meel, 2013). It is likely that experienced cyclists are more inclined to violate traffic signals because they are more confident in their cycling abilities and they can asses a traffic situation more easily. This assumption is backed by Johnson et al. (2013). They found that individuals who cycle more (in this case 100+ km/week) are less likely to comply with traffic signals (60.7%) than people who cycle less than 100 km/week (64.9%).

Herd behaviour among cyclists at a traffic light occurs when cyclists arrive at a red traffic light and there are already people waiting. In this case they are more likely to stop as well. In the earlier mentioned research on red-light running in The Hague, it was found that the percentages of cyclists ignoring red-light are consequently lower (37.4% vs. 62.6%) when there are already people waiting at the traffic light (Van der Meel, 2013). A study on social influence on cyclists behaviour at traffic lights in Bologna, Italy found that when more cyclists were are already waiting at the intersection, more cyclists who approached the intersection stopped as well. When zero cyclists were waiting, 30.3% of approaching cyclists stopped. When cyclists in groups of 1, 2-4 or 5+ were already waiting, 34.8%, 42.3% and 62.4% stopped respectively (Fraboni et al., 2018). It must be added that this research does not take into account the phenomenon where cyclists who are waiting block the bicycle lane and force arriving cyclists to stop as well.

Herd behaviour among cyclists can also work vice versa. This occurs when one cyclist violates a red-light and other cyclists follow suit. The same study on social influence on cyclists behaviour at traffic lights in Bologna found that 27% of cyclists followed another cyclists whilst violating a red-light (Fraboni et al., 2018).

The phenomenon of herd behaviour is corroborated by a study performed on e-bike riders and cyclists in China. A logistic regression analysis was performed from which followed that the probability of a cyclist running a red-light increases when there were fewer cyclist waiting at the traffic signal, and when other cyclists were already crossing during red (Wu et al., 2012).

2.1.3. Meteorological factors

Meteorological factors can possibly influence red-light running behaviour. During adverse weather, cyclists are exposed to weather elements like cold temperatures, rain, snow and wind. If one or more bad-

weather elements are present, the desire of cyclists to quickly arrive at their destination may increase and therefore may influence cyclists' behaviour at traffic signals. However, studies on meteorological factors do not show an unambiguous picture. One study performed on cyclists' crossing behaviour in Taoyuan County, Taiwan showed that risk-taking crossing behaviour decreased from 10.3% during fine weather to 5.1% during adverse weather, i.e. rain. Opportunistic crossing behaviour slightly increased from 9.6% during fine weather to 11.7% during adverse weather. Overall, law-obeying behaviour increased during adverse weather (83.3%) in respect to during fine weather (80.1%) (Pai & Jou, 2014).

2.1.4. Temporal factors

Does red-light ignorance vary during the day, or during the week? Is red-light running behaviour influenced by the difference in traffic flows during peak and off-peak hours?

Pai and Jou (2014) found a significant decrease in red-light violations during peak hours. In total, 88.3% of cyclists obeyed traffic lights during peak hours (7-9;17-19), in respect to 83.9% during non-peak hours. Furthermore, they observed that the portion of signal-obeying cyclists was slightly larger during weekdays (90.2%) in comparison to weekends (87.0%).

2.1.5. Traffic Situation

Another factor that could influence traffic signal violations is the layout of the intersection. Are some intersection designs more prone to red-light violations? And how do traffic light cycles influence red-light running behaviour?

The occurrence of red-light running by cyclists is likely to be partially determined by the type of road that is crossed. Pai and Jou (2014) found that cyclists are more likely to ignore a red signal when only crossing a pedestrian lane. They observed that cyclists who only crossed a pedestrian lane were significantly less likely to obey the traffic signal (65.1%) in comparison to cyclists who (also) needed to cross motorised traffic lanes (83.7%).

Speed limits are likely to be influential on the ignorance of traffic lights by cyclists. One would assume that roads with a higher speed limit will lead to a reduction of red-light infringements as crossing becomes more risky. However, looking at existing literature does not support this hypothesis unambiguously. Pai and Jou (2014) looked at roads with speed limits of 50 and 60 km/h and found increases of risk-taking and opportunistic crossing behaviour of 72% and 76% respectively. They attributed this to "less law enforcement less dense traffic on roadways with speed limit of 60 km/h" (Pai and Jou, 2014, p.191-198).

Traffic volume may be directly related to the frequency of red-light running. It is fair to assume that low traffic volumes will lead to an increase in red-light violations as the probability that an approaching cyclists encounters an empty intersection increases. This is confirmed by the findings of Pai and Jou (2014). They found that low traffic volumes (<15 motorised vehicles per minute) lead to increases in risk-taking and opportunistic crossing behaviour. 69% and 53% respectively, whilst law-obeying crossing behaviour decreases with 27%.

2.1.6. Type of Bicycle

Many different types of bicycles exists and characteristics of different types of bicycles can vary greatly. Especially the speed of bikes can vary enormously between regular bicycles and racing bicycles or e-bikes. An extensive study performed in Germany looked at the rates of red-violations between users on regular bicycles, e-bikes and speed e-bikes by analysing video material. An overall signal violation percentage of 18% was found. However, no substantial difference between the three types of bikes could be noticed (Schleinitz et al., 2019). An observation study in China however, did find a significant difference in red-light negations between bicyclists and e-bike users. According to Wu et al. (2012), 49% of regular cyclists showed traffic signal obeying behaviour. Whereas only 37% of e-bike riders showed traffic signal obeying behaviour.

2.2. Literature Gap

In section 2.1 many factors that influence red-light running behaviour have been named and researched in existing literature. In all of those studies, two main types of research can be distinguished. Observational studies, and survey based studies. The observational studies monitored intersections, registered cyclists's crossing behaviour and inventoried various cyclists's attributions, sometimes by means of cameras. The survey based studies asked respondents about their cycling and crossing behaviour as well as personal attributes that cannot be (easily) determined in observational studies. For example, age, income, cycling experience etc.

During the literature study for this thesis, only one study was found among the observational studies that attempted to predict red-light running behaviour by means of regression analysis. This study was performed by Wu et al. (2012) in Beijing, China and also included a large portion of e-bike riders.

Our research will therefore aim to build a predicting model based on linear regression analysis that will hopefully provide new insights into red-light negating behaviour of bicyclists in The Netherlands. It must be added that, due to the short period of time allowed for this research, no observations can be gathered at the intersection. For that reason, only limited, existing data of crossing cyclists is used for this research. This means the number of influencing factors that can be used to predict red-light running behaviour will be limited.

3

Methodology

In this chapter, the methodology of the research is explained. In Section 3.1.1 the data used for this research is described and is explained how the data was obtained. Next, in Section 3.2, a preliminary data analysis is performed in preparation for the linear regression model. This section focuses on the different methods of finding red-light violations. Also, the processing of the other variables is explained. Section 3.3 elaborates on the linear regression analysis. Finally, in Section 3.4, the methodology chapter is summarised.

3.1. Data Clarification

In this section is a description is given on every data set that has been used for this research. In detail is explained how the data used for this research was obtained, what the data looks like and in what file or format the data was provided.

3.1.1. Data Collection

Traffic light data and cyclist trajectory data used for this thesis were provided by Dr.ir. W. Daamen, faculty of the Transport & Planning Section at the Faculty of Civil Engineering and Geosciences, TU Delft. The cyclists' trajectory data is purely anonymous, has only been used for research purposes and was never shared with third parties. The meteorological data has been obtained from the publicly accessible weather database of the Royal Netherlands Meteorological Institute (KNMI).

3.1.2. Trajectory Data

The trajectory data has been obtained from a smart sensor installed at the Jaffalaan-Mekelweg intersection. Trajectory data from only one smart sensor was available. The data provided covered the period from January 20, 2021, to February 22, 2021. The smart sensor in question was installed on the campus-side of the intersection, recording cyclists coming from the campus (south to north), crossing the Jaffalaan before they continue straight (parallel to Mekelweg) or turn left (parallel to Jaffalaan). An overview of the intersection can bee seen in Figure 3.1.

The data set was provided in MAT-file (MATLAB) format and consists of date, time and coordinates assigned to unique ID's. For the given time period, 816.569 trajectory coordinates of 36.502 unique ID's have been recorded.

Figure 3.1: Jaffalaan-Mekelweg Interesection

3.1.3. Vlog Data

The data from the traffic light was available in Vlog-format. The Vlog files that were provided cover the exact same time frame as the trajectory data. The Vlog files can be accessed and visualised using CuteView software (Figure 3.2). The Vlog data contains data on the phase of the traffic light for motorised vehicles, cyclists and pedestrians. Furthermore, data from several detection cables on the road and bicycle lanes near the intersection is available. For the entire duration of the given time frame, the data shows the colour of the four traffic lights. Two for motorised vehicles (02 and 08), one for cyclists (26) and one for pedestrians (36). In addition, it can be seen which detection cables were activated at what time and for how long they were activated.

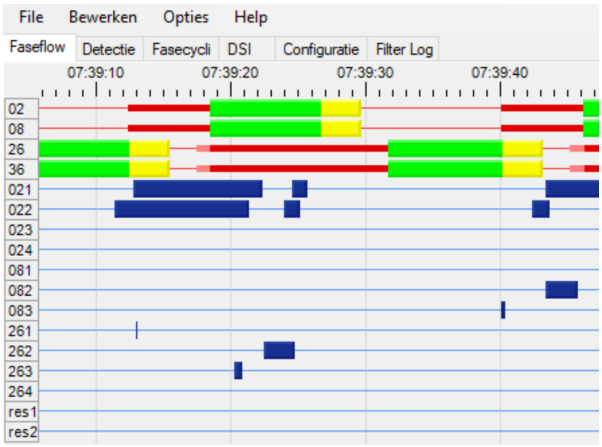


Figure 3.2: A random vlog sample visualised in CuteView

From the Vlog data the green times, i.e. the times when the traffic light for cyclists showed green, have been previously calculated for a different research project using MATLAB code and have been exported into a MAT-file. The original code can be found in Appendix C.2. In this research, the MAT-file with green times has been subsequently used to calculate the red times. In section 3.2.1 is precisely explained how the red times have been obtained.

3.1.4. Meteorological Data

Meteorological data has been obtained from the Royal Netherlands Meteorological Institute (KNMI). For the city of Delft only daily values of precipitation are available. Because daily weather data is not accurate enough for this research, hourly weather data from three surrounding weather stations is used. Hourly values of precipitation, wind speed and temperature were downloaded for the time period between 20-01-2021 and 22-02-2022. In order to find hourly weather for Delft the weighted average of all three weather stations was calculated. The process of finding weighted meteorological data by means of triangular interpolation is explained in section 3.2.6.

3.2. Data Analysis

In this section of the methodology chapter, the method on how the provided data has been systematically analysed, processed and visualised, is described. This process of finding the hourly values of red-light violations is divided into four parts: calculation of red time intervals, calculation of times of crossing, finding the red-light violations and a discussion on the used method. Lastly, the weather data and the traffic data is analysed.

3.2.1. Calculation of Red Time Intervals

The first step into gaining insight in the occurrence of red-light violation is to determine when the traffic light for cyclists was showing red. As explained in chapter 3.1.3, the green times had already been calculated for a previous BEP by Janneke Wind (Wind, 2022). The green times data set comprises of one matrix consisting of four columns. The four columns represent data for: date, cycle count, start time of green cycle (in decimal seconds) and end time of green cycle (in decimal seconds). The start time of each red cycle was calculated by adding exactly three seconds to end green time. The three seconds represent the three seconds of yellow light between the end of each green period en the start of each red period. From the Vlog data followed that the yellow time is exactly three seconds for each cycle. Finally, the end time of each red cycle is set equal to the start of the green time of the next cycle, as there is no yellow light when a signal turns from red to green in The Netherlands. All time values are in seconds. A sample of the resulting DataFrame can be seen in Appendix A. The Python code of this process can be found in Appendix C.1.

3.2.2. Calculation of Times of Crossing

In the next section, an explanation is given on how the red-light negating cyclists have been found. The first step to calculate the number of red-light violations is to find out what the exact time of crossing is of each passing cyclist. The crossing time of a cyclist is defined as the moment in time a cyclist passes the stopping line. In order to determine the crossing time of each cyclist, two possible methods of approach have been identified, given the provided data. The first method addresses how the trajectory data could be used to find crossing times. The second method explains how the crossing times were found using data from the detection cables coming from the Vlog files. Finally, a motivation for the chosen method is given.

1. Method 1: Determine each cyclist's crossing time based on their respective trajectory data.
2. Method 2: Determine cyclists' crossing times based on data from the detection cables of the traffic light.

For this research both methods have been attempted. In the following section both methods are illustrated in broad terms. A detailed, step-by-step explanation accompanied with the used Python code can be found in Appendix C.

Method 1: Crossing Times Based on Trajectory Data

The first method looks at the cyclists' trajectory data. The trajectory data was recorded by a smart sensor that was installed on a pole and placed next to the bicycle lane. The smart sensor recorded cyclists on a section of the bicycle lane south of the intersection going south to north. For each recorded cyclist, the smart sensor registered a series of x- and y-coordinates together with a date and timestamp.

When the first analysis of the trajectory data was being performed, some issues came to light. The first problem is that there was no reference available of how the x- and y-coordinates of the trajectories relate to any real world coordinate system. In other words, it can only be seen how the individual trajectories relate to each other but not how they relate to the real world situation.

One way to find red-light violating cyclists using this method is, for example, to identify cyclists who have at some point stopped based on their trajectory. If other cyclists have (at a constant speed) passed these stopping cyclists, they could be identified as red-light violators. However, this method leaves much room for errors. For example:

- This method will not allow us to find violating cyclists if no other cyclists are present who have stopped.
- No exact crossing time can be extracted which makes further analysis very difficult.
- The smart sensor usually stopped recording between 17:00 and 18:00. This means between the time of final recording and 20:00, when the traffic signal goes on stand-by, no cyclist trajectories are recorded.

The trajectory data was plotted to give a general idea of what the trajectories data looks like. The trajectory plot can be seen in Figure 3.3. It must be added that the trajectories are manually fitted. The real scale and extent of the trajectories may be different. The process of plotting the trajectories, including Python code, can be seen in Appendix C.3.

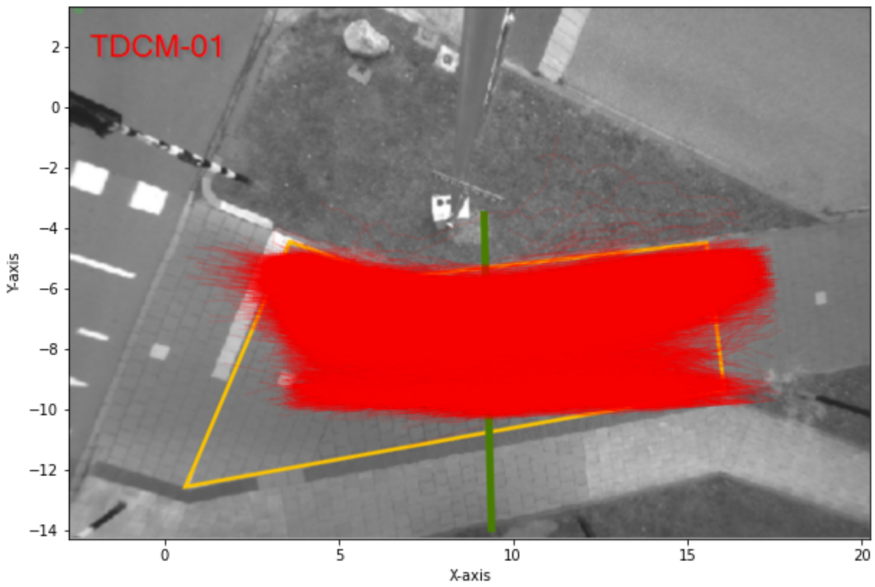


Figure 3.3: Manually fitted trajectories of cyclists

Method 2: Crossing Times Based on Detection Cable Data

The second method to find accurate times of crossing uses the activation times of a detection loop that is installed underneath the bicycle lane at the intersection. The detection cable is in the shape of a parallelogram where the two edges perpendicular to the cycling direction are roughly 60 cm apart

(Figure 3.4). The times when the detection loop is activated are recorded and can be visualised using CuteView software. The time between the start and end of each activation is represented by the blue bars in CuteView, see Figure 3.2. The detection cable of the bicycle lane going south to north is number '263'. The method to find the number of red-light violations using data from the detection cable is illustrated in the remainder of this section and in subsequent section 3.2.3. The general steps of this method are listed below. A detailed, step-by-step explanation of the process of finding red-light violations using method 2, accompanied with Python code, can be found in Appendix C.4.

1. Export detection times from CuteView.
2. Combine detection times of all dates into one DataFrame.
3. Drop all long detection periods.
4. Calculate time to stopping line and add to time of crossing.
5. Check for every time of crossing if it falls into any of the red-light intervals during that day.

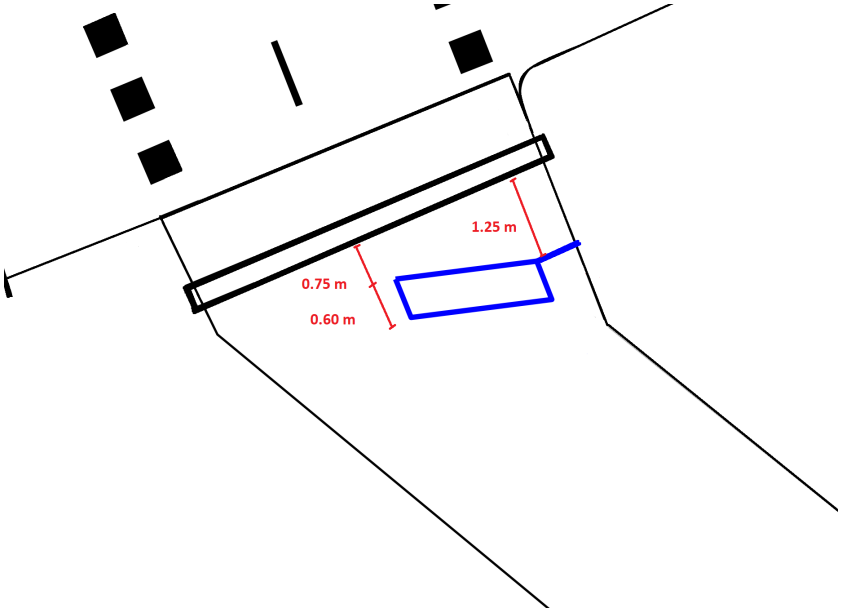


Figure 3.4: Intersection Detection Cable

In order to find the times of crossing using the data from the detection cables the following method was used. First, the detection times were exported from CuteView to Excel. Next, the Excel files were modified into a workable format and combined into one DataFrame using Python.

After that, the length of the activation time is considered. Due to the limitations of the information the detection cable provides, only cyclists with a short detection time, i.e. the time between the start and end of the detection period, can be safely considered to be crossing or 'not-stopping' cyclists. During longer activation periods, individual cyclists can no longer be identified. Also, for longer activation periods it is impossible to accurately determine times of crossing as people have to accelerate from zero speed and acceleration can vary greatly between cyclists. Because of these reasons, longer activation periods are excluded.

In order to find which cyclists can be identified as not-stopping crossers we must determine a maximum activation period for which we can safely say only one cyclist crossed. From technical drawings of the intersection follows that the distance between the edges of the detection loop in cycling direction is roughly 60 cm (Figure 3.4).

In this case the speed of a cyclist is determined by dividing the distance between the detection cables (0.60 m) by the difference in detection time (detection period). A minimum speed of 2 m/s is chosen because for any lower speed it is more difficult to assume a cyclist who passed the detection loop actually went on and crossed the intersection. Any higher speed would result in too many activation

periods to be removed because detection times are recorded in deciseconds. This means only cyclists with crossing times of 0.1, 0.2 and 0.3 are included in the final DataFrame (see equation 3.1).

The calculation for the maximum detection period, based on the chosen minimum speed for a cyclist is as follows:

$$t = \frac{s}{v} = \frac{0.6}{2} = 0.3s \tag{3.1}$$

As can be seen in Figure 3.4, there is some distance between the second detection cable and the stopping line. The time it takes to cross this distance must be taken into account. The distance lies between 0.75 m and 1.25 m, depending on where exactly the cyclist crosses. An average of 1 m is taken. Based on the activation time and thus speed of each cyclist, the time between the second edge of the detection cable and the stopping line can be calculated using a simple formula. The results can be seen in Table 3.1.

$$t = s * v \tag{3.2}$$

Length Detection Period [s]	Speed [m/s]	Time to Stopping Line [s]
0.3	2	0.5
0.2	3	0.33
0.1	6	0.17

Table 3.1: Added times per speed

The time it takes for each cyclist to reach the stopping line are added to the DataFrame. The DataFrame is now complete and can be seen in Appendix A.

3.2.3. Finding Red-Light Violations

Having obtained the crossing times in subsection 3.2.2, we can combine these results with the traffic light data calculated in subsection 3.2.1.

All registered times of crossings were passed through all red-time intervals of their respective date. A time of crossing is registered as a red-light violation if the time of crossing falls into any of the red-time intervals. All violations with date and time are stored.

3.2.4. Red-Light Violations Discussion

After performing the steps of section 3.2.2 and 3.2.3, the number of red-light violations is found. It must be stressed that the found number of violations is only a minimum number due to limitations of the traffic light data and assumptions that were made. The imperfections of the traffic light negation numbers found using the detection cable data are mentioned below:

- Only short detection periods can be identified as crossing cyclists. Cyclists who crossed when at least one other cyclist was activating the detection loop are not counted.
- Cyclist who crossed the intersection while (illegally) using the side of the bicycle lane of the opposite direction are not counted as they not cycled over the detection cables.
- The calculated time of crossing might be slightly off due to assumptions made to calculate the time from the detection loop to the stopping line. Therefore, some cyclist who crossed close to the start or end of a red-time interval may be falsely identified as a red-light violator.

However, it must be added that the occurrence of the phenomenon mentioned in the first bullet point is less frequent as one would initially think. When other cyclists stop at the traffic light they often block the lane forcing arriving cyclists to stop as well. This phenomenon is also mentioned in the research of Fraboni et al. (2018). Secondly, the probability of falsely identifying a cyclist as a red-light violator (third bullet point) is likely to be just as high as falsely identifying a cyclist passing during a green light. Therefore, these effects likely cancel each other out and thus can be considered to be negligible.

In addition, method 2 allows us to find a part of the red-light runners that cannot be identified using the trajectory data. The smart sensor stops recording between 17:00 and 18:00. However, the traffic light does not go on stand-by until 20:00. This means the cable detection data provides an extra number of crossing cyclists each day that are missing in the trajectory data.

3.2.5. Red-Light Violations Validation

As described in section 3.2.2, Method 2 only allows us to find a part of the red-light runners because cyclists who may have crossed during a longer detection period are not accounted for. In order to attempt to validate the found number of crossing cyclists with Method 2 (only not-stopping cyclists), a comparison is made between the number of crossing cyclists found by both methods. Bar plots of four arbitrary days are displayed in Figure 3.5, divided into five minute intervals for the hour 12:00-12:59.

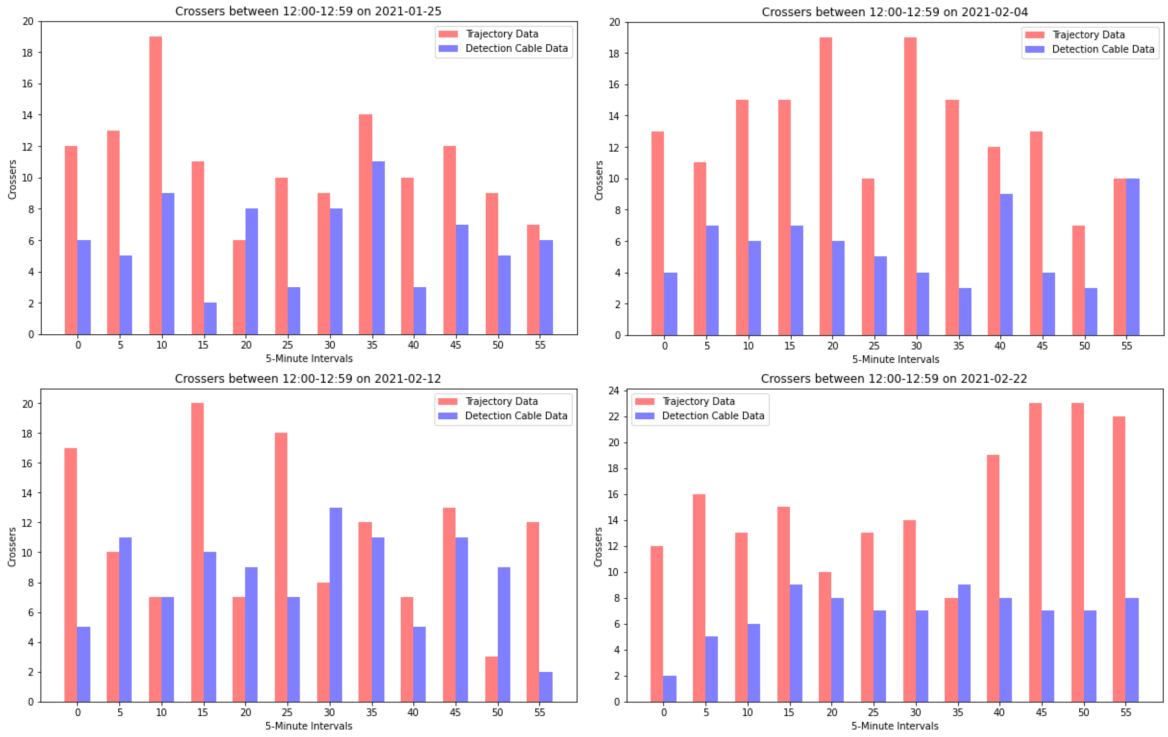


Figure 3.5: Crossing cyclists between 12:00-12:59 on four arbitrary days, found by Method 1 and Method 2

After studying the graphs, two interesting conclusions can be drawn. First, no clear pattern can be noticed between both methods. Ideally, the values found with the detection cable data would be roughly be a fixed percentage of the values from the trajectory data. This would have strongly solidified the choice for Method 2. This is not the case however. As a matter of fact, some five-minute intervals indicate a larger number of crossing cyclists found by means of detection cable data than with the trajectory data. This should be by no means possible. However, the five minute intervals in question have been manually verified and no discrepancies were found. This implicates that the smart sensor has failed to record all passing cyclists.

To conclude, the choice for Method 2 cannot be corroborated with the trajectory data from Method 1.

Nevertheless, Method 2 remains the more accurate method due to the fact that not all passing cyclists have been recorded and stored in the trajectory data.

3.2.6. Weather Data Analysis

As mentioned in section 3.1.4, hourly weather data is not available for the city of Delft. To obtain reasonably accurate weather data, the data of three surrounding weather stations has been averaged using triangular interpolation. Hourly weather data was obtained from three surrounding weather stations: Hoek van Holland (17 km away), Voorschoten (16 km away) and Rotterdam-The Hague Airport (7 km away). See image 3.6. The values of the middle point of the triangle, i.e. the intersection in Delft are calculated by using the following formula, obtained from Tiberius et al. (2022).

$$\widehat{z_0} = \frac{A_{0jk}}{A_{ijk}} y_i + \frac{A_{i0k}}{A_{ijk}} y_j + \frac{A_{ij0}}{A_{ijk}} y_k \tag{3.3}$$

Where A_{ijk} denotes the area of triangle with vertices i, j, k.

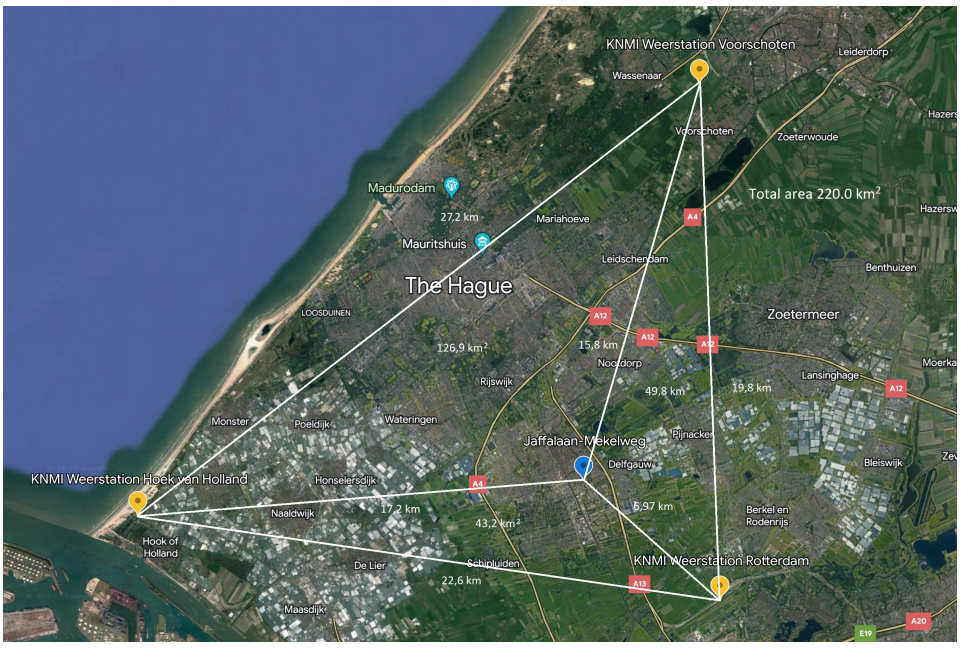


Figure 3.6: These lengths and areas have been used to perform triangular interpolation

After triangular interpolation the results have been stored in a DataFrame, which can be seen in Appendix A.

3.2.7. Traffic Data Analysis

Vehicle Frequency

The hourly numbers of passing vehicles have been obtained from the Vlog data. The activation times of the detection loops for vehicles have been registered. The detection loops in question are labeled

21, 22, and 81 in CuteView (Figure 3.2). There are three different detection loops for vehicles at this intersection because the intersection can be approached from three directions. Data from the detection cables was exported from CuteView to Excel. Python was used to arrange the data in the same format as the weather data and the red-light violations. Also, the number of detections of all three detection loops are combined for each hour. The Python code for processing the traffic data can be found in Appendix B.2. The resulting DataFrame can be seen in Appendix A.

Traffic Light Cycle Frequency

Also, the hourly numbers of completed traffic light cycles for cyclists have been obtained from the V-log data using the same method as for the vehicle frequency. The Python code for the process of finding the number of traffic light cycles per hour can be found in Appendix B.2. The resulting DataFrame can be seen in Appendix A.

3.3. Linear Regression

3.3.1. Dependent and independent variables

In order to determine linear relationships one must define what the dependent variable and what the independent variables are. In this case the dependent variable is the number of hourly red-light violations at the Jaffalaan-Mekelweg intersection calculated in chapter 3.2.3. The independent variables are the variables that have influence on the dependent variable, i.e. the red-light running. In this research the following variables are evaluated: hourly values of wind speed [m/s], temperature [C°], rain [mm], passing vehicles and red traffic lights (or traffic light cycles).

3.3.2. R^2-value

R^2 is value that describes the level of accuracy of the fitted model and is defined by Field (2000) as:

$$R^2 = \frac{|SS_R - SS_T|}{SS_T} \tag{3.4}$$

Where:

- SS_T is the sum of the squared residuals of the most basic model, i.e. the mean as horizontal line.
- SS_R is the sum of the squared residuals of the most fitted model, i.e. the linear regression line.

R^2 indicates the level of improvement of how well the model can predict the outcome variables compared to the most basic model. An R^2 value of 0 would mean the model has not improved at all. An R^2 value nearing 1 would imply the model has improved tremendously.

3.3.3. F-Ratio

According to Field (2000), ”F-ratio is a measure of how much the model has improved the prediction of the outcome compared to the level of inaccuracy of the model”.

$$F = \frac{MS_M}{MS_R} \tag{3.5}$$

Where:

- MS_M is the mean squares for the model.
- MS_R is the residual mean squares.

A good model will result in improved predictions by the model, which means MS_M will be large. The difference between the model and the observed data points will be small if the model is good, i.e. a small MS_R. In summary, a good model must have a large value for the F-ratio, at least higher than one (Field, 2000).

3.3.4. Linear Relationships

After completing the steps of 3.2.3 and 3.2.7 an attempt can be made to find linear relationships between the dependent variable (red-light violations) and the independent variables (rain, wind, temperature, cars per hour and traffic light cycles per hour). Scatter plots are used to give an initial insight into the relationship between the number of violations per hour and the different independent variables. Also the Pearson correlation coefficient is calculated. The resulting scatter plots and Pearson coefficients can be seen in section 4.2.

3.3.5. Simple Linear Regression Model

In order to evaluate the strength of the linear relation between the dependent variable and the independent variables a Simple Linear Regression (SLR) Model will be built using Statsmodels. Statsmodels is a Python module that can be used for "many different statistical models, as well as for conducting statistical tests, and statistical data exploration" (Statsmodels-0.14.0, 2023). Statsmodels outputs many statistical measures like R-squared, adjusted R-squared, F-statistic and many more.

With just a few lines of code, Statsmodels uses the method of Ordinary Least Squares (OLS) to fit a linear regression line. It can output a statistical summary of the results, which can be subsequently interpreted. Also, Statsmodels can output the intercept and slope values allowing the user to plot the regression line. SLR models are built for all the chosen variables.

A significance level of $p<0.05$ is chosen because this level of significance is commonly accepted and the statistical analysis of this report is not based on previous research.

Summaries of the results of the OLS Models can be found in section 4.3, including the values of R^2 and F-statistic. The corresponding Python code can be seen in Appendix D.2.

3.3.6. Multiple Linear Regression Model

Hypotheses

Hypothesis testing can be used to determine if there is enough evidence in the data set to draw conclusions. If there is enough statistical significance, the null hypothesis can be rejected and thus the alternative hypothesis can be accepted (Field, 2000). The null hypothesis (h0) and alternative hypothesis (ha) for this research are as follows:

- H0: Rain, wind speed, temperature, the frequency of passing vehicles and completed traffic light cycles are not related to red-light violations at the Jaffalaan-Mekelweg intersection.
- Ha: At least one of the independent variables is a significant predictor of red-light violations at the Jaffalaan-Mekelweg intersection.

The results of the hypothesis testing can be found in section 4.4.6.

Choosing an Entry Method

When constructing a multiple linear regression model it is important to consider the order in which the independent variables are entered. Field (2000) describes three main methods of order of entry. The

first method is hierarchical or blockwise entry. In this case the order is determined by the experimenter. However, the variables that are selected based on previous research should be entered first. After that, new variables can be entered (Field, 2000). This thesis is not based on previous work. Therefore this method is not chosen.

Another method of entry order is called the stepwise method. Using this method the computer chooses the order of entry based on which variable can explain the not yet explained remainder of variance of the outcome variable best. This method is especially useful when many predictor variables are available (Field, 2000). In this research only five predictor variables are considered. Therefore, this method is not chosen.

The third method is called forced entry. Using this method all predictor variables are entered at the same time. It is crucial that only predictor values are chosen that can predict a large part of the variance of the outcome variable (Field, 2000). The forced entry method is chosen for this research because in section 3.3.5 the relationship between the individual independent variables and the outcome variable is proven.

Building the Model

First, Statsmodels is used to build the MLR model in the same way as for the SLR models. The results of which can be found in section 4.4. Again, a significance level of $p<0.05$ is chosen. Next, a prediction model is made using train test split from Scikit-learn. With train test split the examiner can use one part of the data to train the model. The remaining part of the data set is used to test the model. The data is divided randomly (scikit-learn.org, 2023). For this MLR model, 70% of the data is used to train the regression model. The other 30% is used to the test the model. The results of the MLR model, built with train test split from Scikit-learn can be seen in section 4.4.5.

3.4. Summary

To summarise, the following steps have been defined in the methodology for this research. First, the origin of the used data is clarified. After that, an elaborate description was given on which method is used to find hourly values of crossing cyclists, why this method is appropriate and how values for red-light negation have been subsequently found. Lastly the steps of the linear regression analysis were demonstrated.

4

Results

Chapter 4 contains the results of the data analysis for this research. In section 4.1.1 the found numbers for crossing cyclists and red-light violators are demonstrated. Section 4.2 elaborates on the relations between the dependent and independent variables. Lastly, section 4.3 and 4.4 contain the results of the SLR and MLR models.

4.1. Data Results

This section gives the results of the two different methods defined to find numbers of daily crossing cyclists. The daily number of crossing cyclist is displayed in bar plots. Also the daily number of red-light violations and percentage of red-light violations are displayed by means of a bar plot in 4.1.2. The horizontal dashed red lines indicate the means of of daily crossings and red-light violations over the given period.

4.1.1. Crossing Cyclists

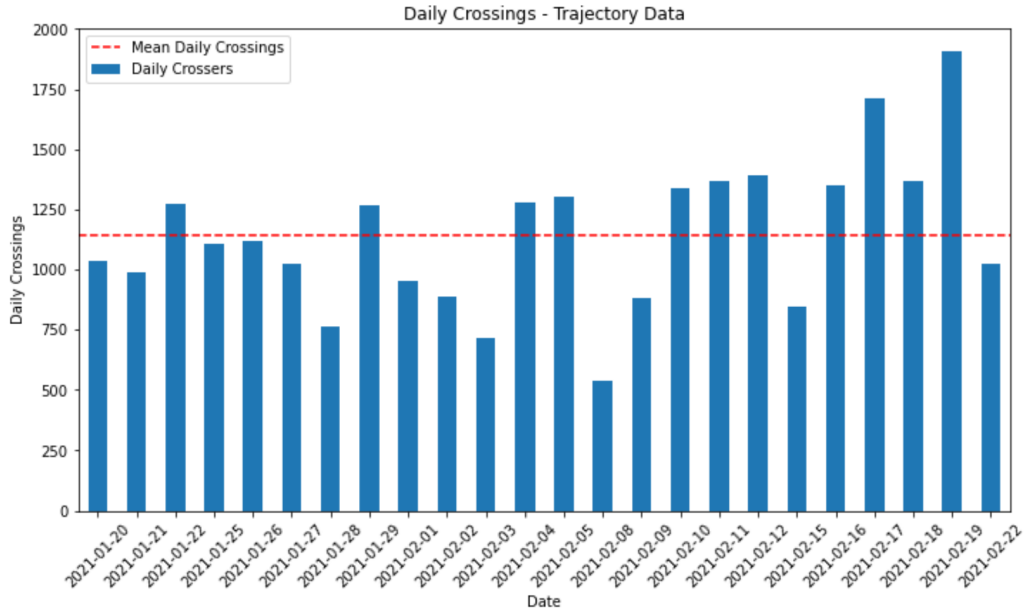


Figure 4.1: Number of crossings per date, using trajectory data

- In total 27.431 cyclists have been registered by the smart sensor during weekdays between 2021-01-20 and 2021-02-22 by use of the trajectory data.
- The mean daily crossers for this period is 1143.

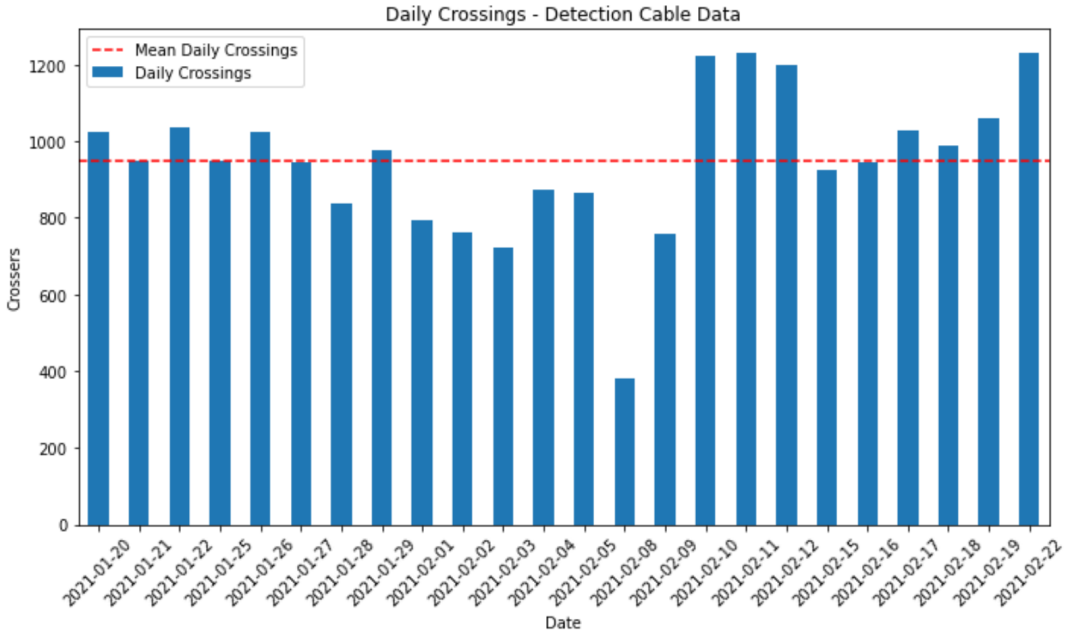


Figure 4.2: Number of crossings per date, using detection cable data

- In total 21.045 cyclists have been registered during weekdays between 2021-01-20 and 2021-02-22 using the detection cable data.
- The mean daily crossers for this period is 947.

4.1.2. Red-Light Violations

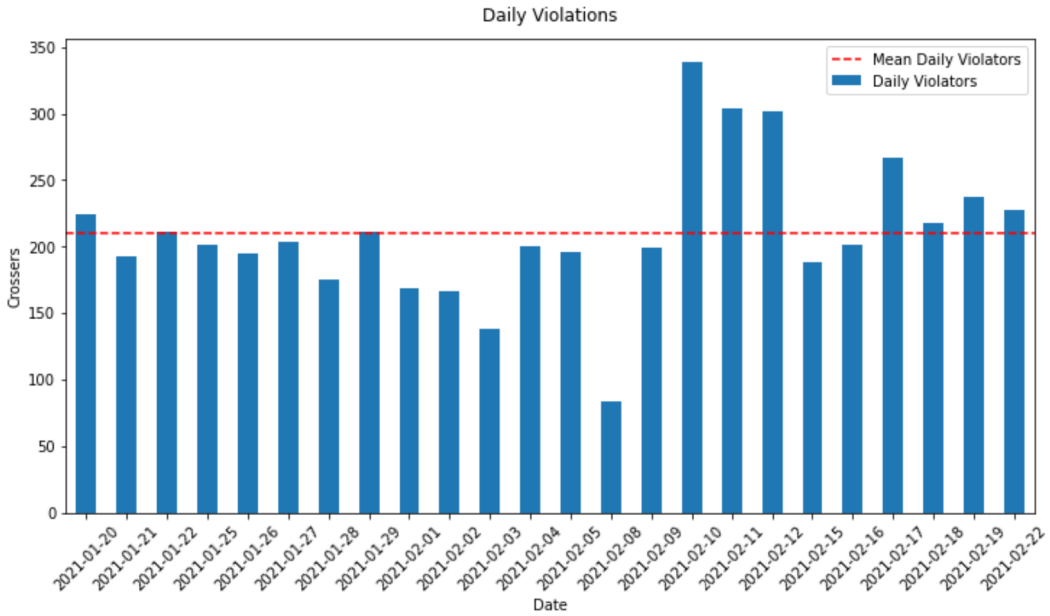


Figure 4.3: Number of red-light violations per date

- A minimum of 4686 cyclists violated a red-light signal during the given period.
- This means 4668 out of 21.045 'not-stopping' cyclists (22.2%) violated a red signal.
- This means 4668 out of 27.431 trajectory cyclists (17.0%) violated a red signal.
- The mean number of daily red-light violations is 210.

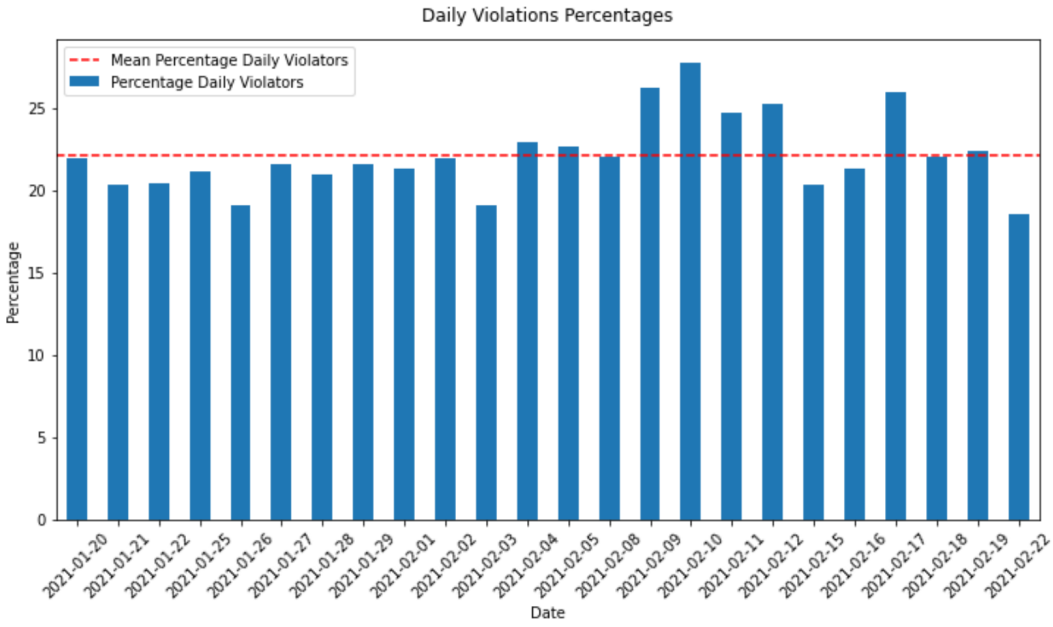


Figure 4.4: Percentage of red-light violations per date

- Noticeable is the fairly consistent percentage of daily violators ($\pm$ 22%).

4.2. Linear Relationships

This subsection shows the relationships between the hourly red-light violations and the individual independent variables by means of scatter plots and Pearson correlation coefficients. All variables are stored in a DataFrame, of which the values for the first day of the period can be found in Appendix A.

Table 4.1 shows the Pearson correlation coefficient for each independent variable. A perfect positive correlation is indicated with 1. A perfect negative correlation is indicated with -1. 0 means there is no correlation at all (Kent-State-University, 2023).

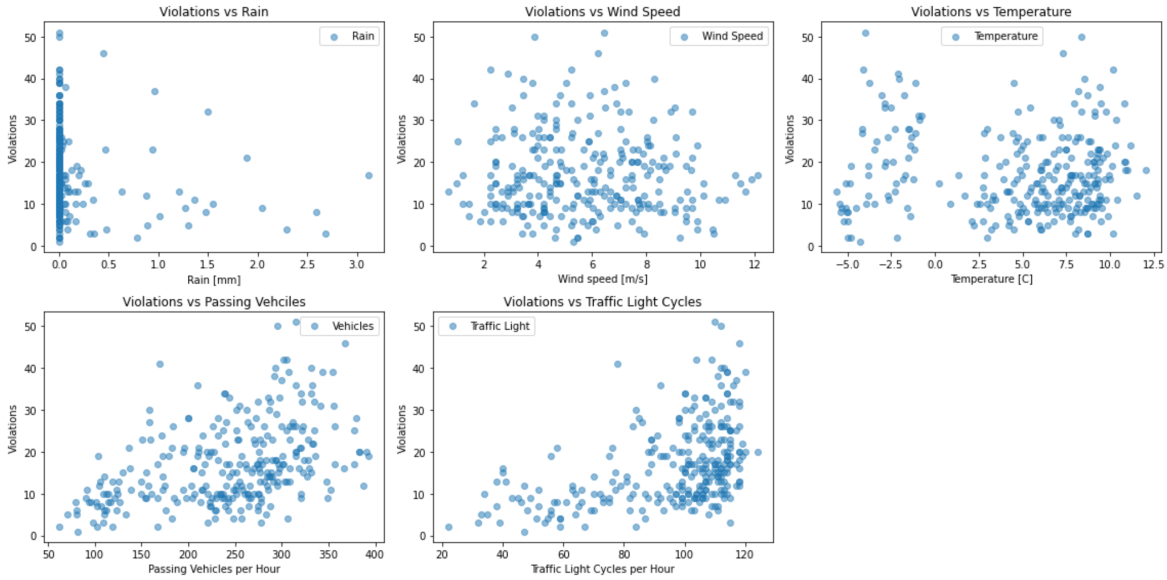


Figure 4.5: Scatter plots of red-light violations v. predictor variables

Independent Variable	Pearson Correlation Coefficient
Rain	-0.14
Wind Speed	-0.04
Temperature	-0.05
Hourly Passing Vehicles	0.45
Hourly Traffic Light Cycles	0.46

Table 4.1: Pearson correlations coefficients

It can be noted that regarding the weather data there is no linear relationship shown between the three independent weather variables and the red-light violations. This is supported by the low values of the Pearson correlation coefficient. The scatter plot of the rain data looks distorted due to the fact that for many hourly intervals no rainfall was recorded during the time period. The non-zero data points show some negative linear relationship. However, there are not enough non-zero data points available to allow for further research. The scatter plots for temperature and wind speed do not show any form of linear relation. Therefore, rain, temperature and wind speed will not be used for linear regression analysis.

Unlike the weather data, the independent variables for passing vehicles and traffic light cycles do have somewhat of a linear relationship with the dependent variable. In both scatter plots a positive linear relation with the number of red-light violations per hour can be observed. Albeit not a very strong linear relations, the Pearson correlation coefficients are 0.45 and 0.46 respectively. In other words, a higher frequency of passing vehicles suggests an increased number of hourly red-light violations. Similarly, more completed traffic light cycles mean more violations per hour.

In conclusion, the hourly passing of vehicles and the hourly number of completed traffic light cycles will be used for the SLR model and the subsequent MLR model.

4.3. Simple Linear Regression Model Results

4.3.1. Passing of Vehicles

Ordinary Least Squares Regression Result Summary

OLS Regression Results

Dep. Variable:	Violation Count	R-squared:	0.201
Model:	OLS	Adj. R-squared:	0.198
Method:	Least Squares	F-statistic:	70.96
Date:	Sat, 14 Oct 2023	Prob (F-statistic):	1.88e-15
Time:	13:59:50	Log-Likelihood:	-1014.8
No. Observations:	284	AIC:	2034.
Df Residuals:	282	BIC:	2041.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.0966	1.726	1.794	0.074	-0.302	6.495
Sum	0.0573	0.007	8.424	0.000	0.044	0.071

Figure 4.6: Regression results with constant: red-right violations v. passing of vehicles

From the result summary of the OLS Model follows that the model that R^2 is 0.201. This means the variance of hourly red-light violations for 20.1 % is explained by the hourly passing of vehicles. This percentage is not extremely high, but it is significant enough to include this independent variable in the MLR Model. However, the adjusted R^2 is rather close to the R^2 value. This means adding more variables will not improve the model significantly (Field, 2000). An F-ratio of 70.96 and the associated p-value of near zero suggests the model overall is statistically significant. In other words, the independent variable has a significant effect on the dependent variable. Looking at the coefficients, one can see that the constant has a coefficient of 3.0966 with a p-value of 0.074. Which is not statistically significant at a significance level of 0.05. However, the p-value of the predictor value is zero. Therefore, the independent variable is statistically significant. In this case it means that for each unit increase of passing vehicles the number of hourly red-light violations increases with 0.0573.

Another way a linear regression model can be interpreted is by looking at the residuals distribution. Preferably, the residual histogram is normally distributed meaning that the model equally over- and underestimates its predictions. Looking at the residuals histogram of the passing vehicles independent variable (Figure 4.7), one can immediately see the histogram skews somewhat to the left and is centered slight left of zero. This would indicate the model tends to overestimate the dependent variable resulting in a systematic bias in the model's predictions (Field, 2000). In addition, some outliers can be seen. Outliers can indicate that the model is not adequately capturing certain data points (Field, 2000).

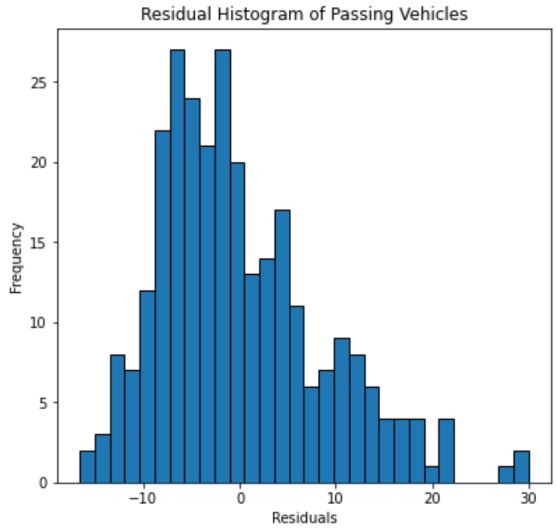


Figure 4.7: Residual Histogram of Passing Vehicles

In summary, the model has some statistical significance as indicated by the low p-value for the F-statistic and the coefficient for hourly passing vehicles. However, the low R-squared value suggests that the model may not explain a large portion of the variance in the number of red-light violations. Also, there is some bias in the model. For the purpose of this research, the model is deemed good enough to enter the predictor varibale in the MLR Model. The regression line, fitted through the data points can be seen in Figure 4.8.

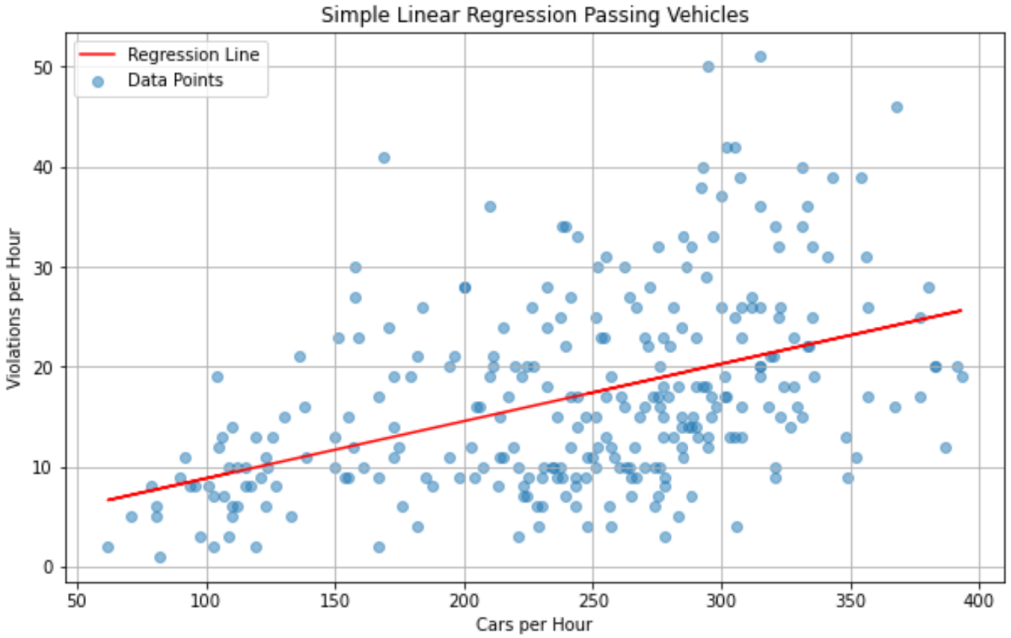


Figure 4.8: Linear regression with Statsmodels

4.3.2. Traffic Light Cycles

OLS Regression Results

Dep. Variable:	Violation Count	R-squared:	0.209
Model:	OLS	Adj. R-squared:	0.206
Method:	Least Squares	F-statistic:	74.54
Date:	Sat, 14 Oct 2023	Prob (F-statistic):	4.44e-16
Time:	13:59:50	Log-Likelihood:	-1013.4
No. Observations:	284	AIC:	2031.
Df Residuals:	282	BIC:	2038.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.2107	2.281	-0.969	0.333	-6.701	2.279
Traffic Light	0.2004	0.023	8.634	0.000	0.155	0.246

Figure 4.9: Regression results red-right violations v. traffic light cycles

Roughly the same conclusions can be drawn from this OLS Regression Results Summary. The R^2 value indicates that the number of completed traffic light cycles can for 20.9% explain the variance in hourly red-light violations. Again, Adjusted R^2 is almost equal to R^2, indicating that adding this independent variable will most likely not improve the MLR Model significantly. However, the F-statistic and its probability show that the model is statistically significant. The constant or intercept of this SLRM is negative and has a probability much larger (0.33) than 0.05. This is obviously wrong because the number of red-light violations cannot be negative.

Again looking at the residual histogram of the residuals from the traffic light cycles SLRM, one can see the histogram skews to the left. Once more, indicating that the model tends to overestimate the dependent variable. Also, some outliers are present. Most likely occurring due the inaccuracy of this model.

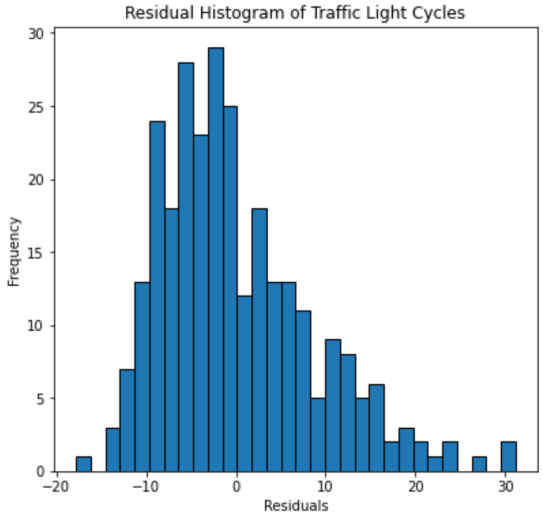


Figure 4.10: Residual Histogram of Traffic Light Cycles

The same conclusion can be drawn for this independent variable as for the independent variable of the passing vehicles. Therefore, the hourly number of traffic light cycles will be included in the MLRM. The fitted regression line can be seen in Figure 4.11.

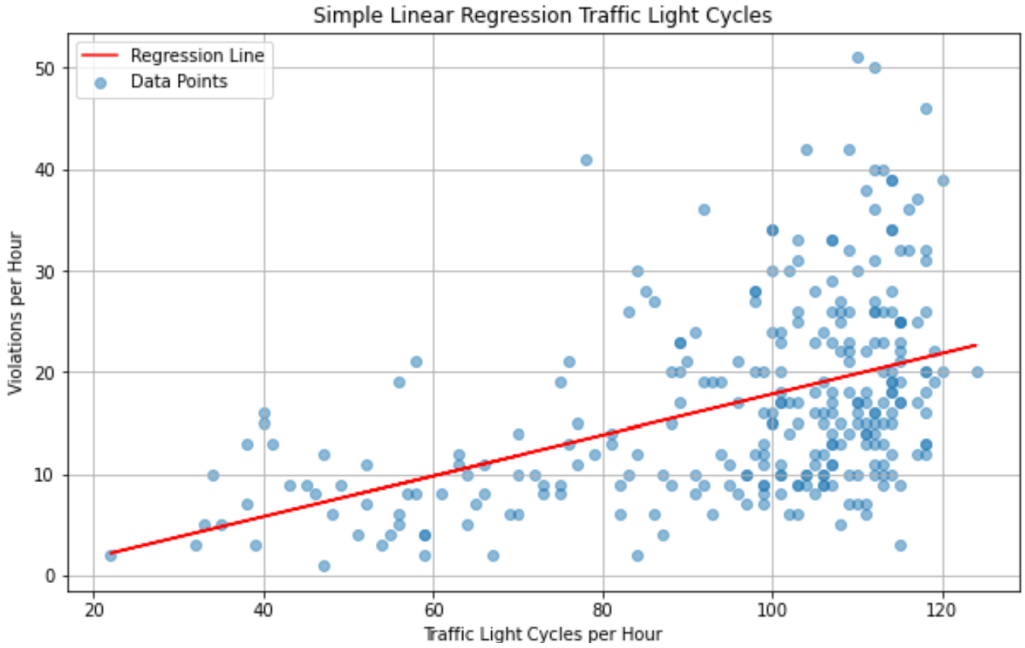


Figure 4.11: Linear regression with Statsmodels

4.4. Multiple Linear Regression Model Results

In Figure 4.12, the summary of the results of the multiple linear regression model is displayed. In section 4.4.1 the performance of the model will be analysed. In section 4.4.2 it will be checked if the model can be applied to other data sets.

OLS Regression Results

Dep. Variable:	Violation Count	R-squared:	0.223
Model:	OLS	Adj. R-squared:	0.217
Method:	Least Squares	F-statistic:	40.26
Date:	Fri, 13 Oct 2023	Prob (F-statistic):	4.23e-16
Time:	17:38:27	Log-Likelihood:	-1010.9
No. Observations:	284	AIC:	2028.
Df Residuals:	281	BIC:	2039.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.2477	2.306	-0.541	0.589	-5.788	3.292
Sum	0.0278	0.012	2.221	0.027	0.003	0.052
Traffic Light	0.1200	0.043	2.799	0.005	0.036	0.204

Omnibus:	34.713	Durbin-Watson:	1.026
Prob(Omnibus):	0.000	Jarque-Bera (JB):	43.697
Skew:	0.897	Prob(JB):	3.25e-10
Kurtosis:	3.688	Cond. No.	1.24e+03

Figure 4.12: Multiple Linear regression with Statsmodels

4.4.1. Does the Model Fit Well?

The result summary produces an R^2 value of 0.223. This means only 22.3% of the variance in the dependent variable (Red-Light Violations) is explained by the independent variables in the MLR model. However, this is a slight increase compared to the R^2 of the SLR models individually (0.201 and 0.209). See section 4.3.

The p-values associated with the predictor variables are both below 0.05, with ”Sum” (Passing Vehicles) at 0.027 and ”Traffic Light” at 0.005. This suggests that both independent variables are statistically significant in predicting the dependent variable. Also, the F-statistic is 40.26, and the associated p-value is very close to zero (4.23e-16). This indicates that the model, as a whole, is statistically significant.

The Durbin-Watson statistic is close to 1.0 (1.026), which suggests that there may not be a significant autocorrelation in the residuals (Glen, 2020).

Normality of Residuals

From the histogram of the residuals in Figure 4.13 can be deducted that again the model tends to overestimate the dependent variable because the histogram skews to the left. Also, the mean of the distribution lies slightly left of zero.

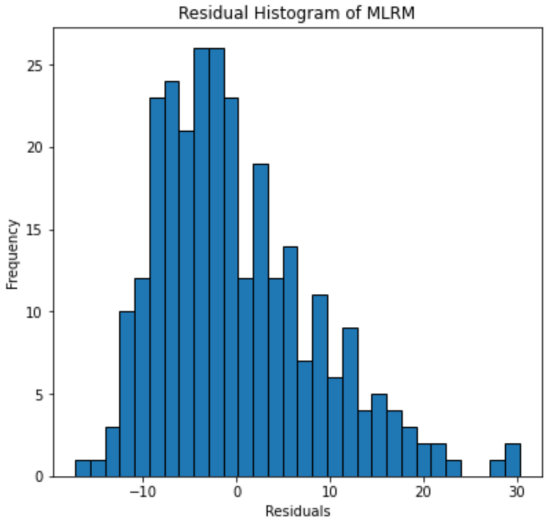


Figure 4.13: MLRM Residuals Histogram

Homoscedasticity of Residuals

The homoscedasticity of the residuals is analysed by looking at the residuals scatter plot in Figure 4.14. Ideally, a residual scatter plot should have no clear pattern. This would suggest homoscedasticity of residuals (Taylor, 2022). In the residuals scatter plot of the MLRM a funnel or cone shape can be detected. This suggests heteroscedasticity of the residuals. In other words, the spread of the residuals changes as you move along the values of the independent variables (Taylor, 2022). Overall, heteroscedasticity does not create bias. However, it makes the results of a regression analysis less trustworthy (Taylor, 2022).

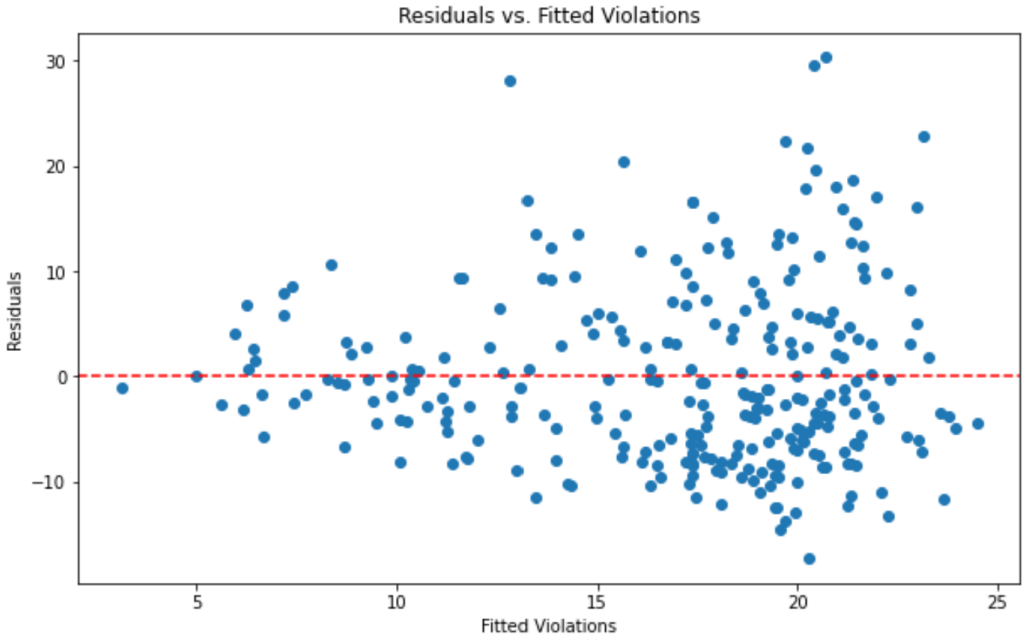


Figure 4.14: MLRM Residuals Plot

4.4.2. Can the Model Be Generalised?

4.4.3. Sample Size

As a rule of thumb, literature suggests the sample size should be at least 10-20 times the number of independent variables (Field, 2000). With only two independent variables in the model and 284 data points, the size of the data set should be more than sufficient. However, the measured effects of the predictor variables on the response variable are rather low. A smaller effect size usually requires a larger sample (Field, 2000). Therefore, this model could benefit from a larger sample size.

4.4.4. Multicollinearity

”Multicollinearity exists when there is a strong correlation between two or more predictors in a regression model” (Field, 2000). When independent variables are strongly correlated, it becomes challenging to distinguish the individual effects of each variable on the dependent variable. The coefficients of the correlated variables may become unstable and difficult to interpret because they depend on each other. From a simple scatter plot (Figure 4.15) it can immediately seen the two predictor variables have strong linear relationship. This strong linear relation can be explained by the fact that the traffic lights on the Jaffalaan-Mekelweg intersection are demand controlled. In other words, when more cars pass the intersection, more often the traffic signal for cyclists will show red. This high level of co-linearity significantly reduces the predicting power of the MLR model.

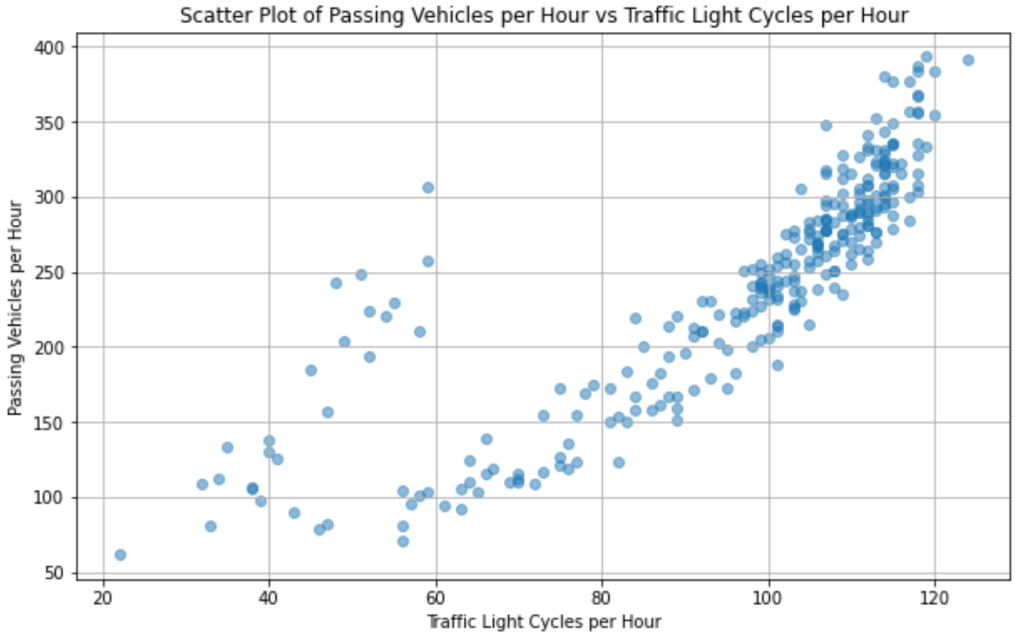


Figure 4.15: Scatter plot of the two independent variables

4.4.5. MLRM Testing

Despite aforementioned limitation of this MLRM, the model has been used to predict red-light violations using Sckit-learn, described in 3.3.6. In Figure 4.16 can be seen that the predicted violations do not accurately match the observed values. Ideally, the predicted and observed values follow the dashed line and there is an equal amount of scatter points left and right of the dashed line. In the case of our MLR model, it can be noticed that more scatter points are located left of the dashed line. This means the model generally over predicts the number of hourly traffic light negations.

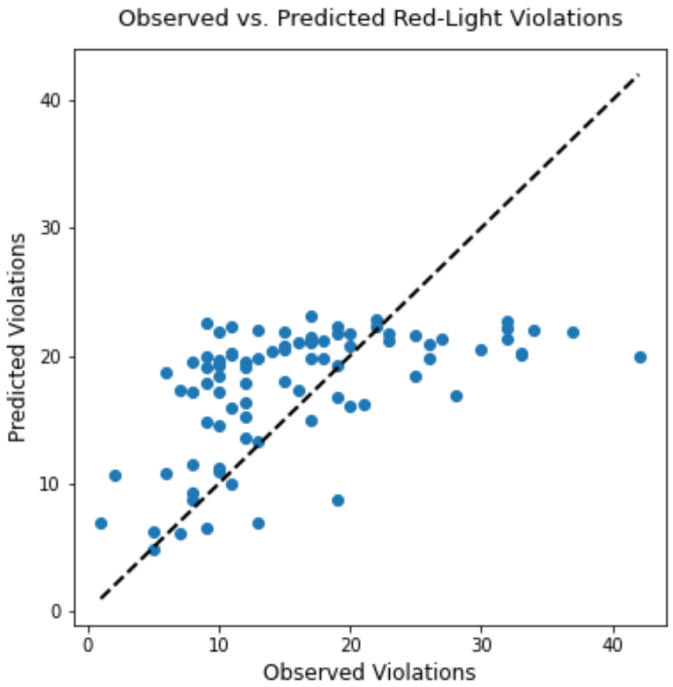


Figure 4.16: MLRM Scikit-learn results

4.4.6. MLRM Conclusions

Coefficient Estimates

In the MLR model the coefficients represent the change in the dependent variable associated with a one-unit change in the corresponding independent variable, while holding the other variable constant. The first step to check if the null hypothesis (h0) can be rejected, is to verify if the coefficient estimates are not equal or close to zero. From the result summary follows that the coefficients are 0.0278 and 0.1200, for passing vehicles and traffic light cycles respectively.

P-Values

The p-values associated with the predictor variables are 0.027 and 0.005. Both p-values are smaller than the chosen significance level of $p < 0.05$. This suggests that both variables are significant predictors of red-light violations.

Hypothesis Testing

Because the coefficient estimates and the associated p-values of both predictor variables meet the given requirements, the null hypothesis can be safely rejected and the alternative hypothesis can be accepted.

Final Conclusions

The null hypothesis is rejected, meaning the frequency of passing vehicles and completed traffic light cycles have a significant influence on the number of red-light violations. However, due to the lack of more predictor variables and the presence of co-linearity between the used predictor variables, the MLR model cannot accurately predict hourly red-light violations.

5 Discussion

In this chapter the limitations of the research are discussed. Three main points of discussion are considered. First the used method of finding hourly values of red-light violations is mentioned. After that, a brief analysis is given on the amount of data and the types of data that have been used in this research.

5.1. Method of Finding Red-Light Violations

In section 3.2 the method of finding hourly values of red-light violations by cyclists on the Jaffalaan-Mekelweg intersection is precisely demonstrated. In this section is explained that the the time of crossing of each cyclist is determined by looking at the detection times of the detection loop that lies underneath the bicycle lane. However, it was not possible to identify all individual cyclists. During longer activation periods of the detection cable, individual cyclists are not registered and were therefore dropped and thus not included in following parts of the data analysis. As a result, an unknown number of potential red-light runners is not included in the research.

In order to improve the preciseness and validity of this research, a method should be formulated to include the trajectory data of the cyclists and combine this data with the traffic light data from the intersection. Not only will this method prevent possible red-light runners to be missed. It will also allow for cross-reference, which can help validate both data sets. Cross-reference would make the foundation of this research much stronger and therefore improve the validation of potential results.

Lastly, the trajectory data can provide more information on individual cyclists. For example, the variance in speed over the trajectory. Obtaining more data on crossing cyclists can be used to identify more predictor variables, which will further improve the research. More on the number of predictor variables is mentioned in section 5.3.

5.2. Amount of Data

Another way to improve the results of this research could be to input more data into the model. For this thesis, data has been studied of the intersection and cyclists for a period of 35 days. 10 of which were weekend days, which could not be included in the research because the traffic light is inactivated on weekends. This is means only 25 days remained to be studied, which is rather limited. Especially considering the fact that this period took place in the winter of 2021. During this period, multiple Covid 19 restrictions were in place, including a curfew active from 23-01-2021 (Rijksoverheid, 2023). It is very likely that as a result less people crossed the intersection because many facilities on campus were closed. This limited the number of daily crossers, which subsequently complicated the data research.

Besides the number of daily crossers, also the weather data was less significant due to the short period of registration. The weather data registered during is rather homogeneous, as can be seen in

the scatter plots in section 4.2. In other words, the weather did not vary enough to be able to identify patterns indicating that weather might influence cyclists's red-light running behaviour. It is safe to say that studying weather data registered over a whole year will most likely increase the probability to find relations between adverse weather for cyclists and traffic signal violations.

5.3. Number of Predictor Variables

In order to build a model that can accurately predict red-light running behaviour it is quintessential to identify and gather data of more predictor variables. In this report only two predictor variables have been identified that correlate with the response variable and have been used for the model. Moreover, these two predictor variables have a high level of collinearity. This reduced the predicting power of the model greatly.

In order to end up with a model that can make accurate predictions on red-light negation numbers, more predictor variables should be identified. Especially predictor variables of different fields could be useful. Factors of influence like age, gender, cycling experience, type of bike etc. could be used to gain better understanding in the behaviour of red-light runners. However, it must be stressed that it is rather difficult to combine survey data like socio-psychological and demographic data with observational data, for example cyclist trajectories.

5.4. Summary

To summarize, a method must be found to incorporate the cyclists trajectory data in finding the number of red-light violations. In addition, data should be gathered of more different predictor variables and over a longer period of time. This way the research can be greatly improved and possibly a model can be built to accurately predict red-light runners.

6

Conclusions and Recommendations

Chapter 6 provides the conclusions and recommendations of this thesis. Section 6.1 contains the findings of research. In section 6.2, recommendations are provided based on the the results of this research. Section 6.3 lists some points that explain in which ways this thesis can be used for further research. Finally, a summary of this chapter is given in section 6.4

6.1. Findings

The aim of this research is to investigate if a multiple linear regression model can be built that can predict hourly values of red-light violations based on various predictor variables. In this research two groups of independent variables have been included in the research. First, adverse weather conditions are researched. To be more precise: hourly values of precipitation, wind speed and temperature. Secondly, data from the intersection is used. Namely, the hourly number of vehicles passing the intersection and the hourly number of red-lights. The first step of building the model is to find out if there exists any relationship between the predictor variables and red-light violations. From scatter plots displaying correlations between the adverse weather variables and hourly traffic light violations, no linear relations could be identified. In the case of rain, there were many hourly intervals where no precipitation was registered. This makes the precipitation data unusable for further analysis. The other two meteorological factors, temperature and wind speed, both do not show any correlation with the frequency of red-light violations. As a result, none of the meteorological factors have been incorporated in the regression models.

In case of the predictor variables that were extracted from the intersection data, some linear relation could be identified. Both traffic data variables show a significant (positive) linear relation with the response variable. 20.1% of the variance in hourly red-light negation can be explained by the number of passing vehicles. For the variable of hourly red-lights, 20.9% of the variance in red-light violations can be explained. Both predictor variables are statistically significant at a significance level of $p<0.05$. In conclusion, both variables can for a small part explain the variance in red-light violations

Finally, the two remaining predictor variables are combined into a multiple linear regression model (MLRM). From the MLRM follows that the two predictor variables combined can explain for 22.3% of the variance in red-light violations. This is only a slight increase from the simple linear regression models. The small increase can be explained by the high level co-linearity between the predictor variables. Despite the limited predicting power of the variables, a MLRM using machine learning is used to predict red-light violations. The model uses 70% of the data to train itself in predicting red-light violations. The other 30% is used to test the model. The resulting scatter plot shows that the model generally overestimates the number of red-light violations.

In conclusion, the model produces statistically significant results. However, due to the small number of predictor variables, the predicting powers of the model remain rather limited. As a result, no real conclusions can be drawn from this model.

6.2. Recommendations

The goal of this research was to predict red-light negation behaviour based on various predicting factors. Unfortunately no substantial conclusions could have been drawn from this research. As a result, no recommendations can be made to possibly reduce the number of red-light violations by managing one or more of the predictor variables. However, this research could provide a starting point for further research. In the next section is explained in what ways the research can be improved. After applying these improvement measures, the model can perhaps make accurate predictions. These results can subsequently be used to present useful recommendations.

6.3. Further Research

Trajectory Data

To advance this research, a crucial step is the integration of trajectory data into the analysis. The trajectory data can provide more information on passing cyclists. For example, their speed at the time of crossing. Secondly, it will lead to more accurate violation numbers than the current method used. Incorporating the trajectory data will improve any future regression model and provide more accurate predictions of violation numbers.

When employing trajectory data of cyclists in further research two main factors must be considered. First, the exact location of the origin of the local coordinate system of the trajectories must be able to be placed in the real world. This will allow the researcher to accurately determine the exact time of crossing of each cyclist. Secondly, the trajectory data should cover the exact time range of the traffic light data. As a result, the time of crossing for all cyclists can be checked for possible a red-light violation and no possible red-light violators are missed.

Time Period

Another way to improve this research is to gather data over a longer time period. A longer time period will not only provide more data to work with. It can perhaps also allow us to find correlations between violation numbers and weather data. Weather data will likely vary more over a longer time period, possibly allowing us to see correlations between weather conditions and red-light running.

Predictor Variables

Lastly, more predictor variables will definitely improve any regression model. Many different factors of influence have been identified in previous research. Data of more possible predictor variables should be gathered in order to make better predictions.

6.4. Summary

In conclusion, this study aimed to predict red-light violations using various predictor variables. Weather conditions did not show significant correlations, while data from the intersection revealed positive relationships with violations. A machine learning model, though statistically significant, had limited predictive power. Further research could incorporate trajectory data, extend the period of research, and include more predictor variables to enhance predictions.

References

CBS. (2023). *Cbs - meer verkeersdoden in 2022, vooral fietsende 75-plussers vaker slachtoffer*. Retrieved September 12, 2023, from https://www.cbs.nl/nl-nl/nieuws/2023/16/meer-verkeersdoden-in-2022-vooral-fietsende-75-plussers-vaker-slachtoffer

Delft, T. (2023). *Tu delft - feiten en cijfers*. Retrieved September 12, 2023, from https://www.tudelft.nl/over-tu-delft/feiten-en-cijfers

Field, A. (2000). *Discovering statistics using spss* (3th ed.). SAGE Publications.

Fraboni, F., Marín Puchades, V., De Angelis, M., Pietrantoni, L., & Prati, G. (2018). Red-light running behavior of cyclists in italy: An observational study. *Accident Analysis Prevention*, *120*, 219–232. https://doi.org/https://doi.org/10.1016/j.aap.2018.08.013

Glen, S. (2020). *Durbin watson test test statistic*. Retrieved October 15, 2023, from https://www.statisticshowto.com/durbin-watson-test-coefficient

Johnson, M., Charlton, J., Oxley, J., & Newstead, S. (2013). Why do cyclists infringe at red lights? an investigation of australian cyclists' reasons for red light infringement. *Accident Analysis Prevention*, *50*, 840–847.

Kent-State-University. (2023). *Spss tutorials: Pearson correlation*. Retrieved October 21, 2023, from https://libguides.library.kent.edu/SPSS/PearsonCorr

Pai, C., & Jou, R. (2014). Cyclists' red-light running behaviours: An examination of risk-taking, opportunistic, and law-obeying behaviours. *Elsevier*, *62*, 191–198.

Rijksoverheid. (2023). *Coronavirus tijdlijn - januari 2021: Invoering avondklok en start vaccinatie*. Retrieved October 17, 2023, from https://www.rijksoverheid.nl/onderwerpen/coronavirus-tijdlijn/januari-2021-invoering-avondklok-en-start-vaccinatie

Schleinitz, K., Petzoldt, T., Kröling, S., Gehlert, T., & Mach, S. (2019). (e-)cyclists running the red light – the influence of bicycle type and infrastructure characteristics on red light violations. *Accident Analysis Prevention*, *122*, 99–107. https://doi.org/https://doi.org/10.1016/j.aap.2018.10.002

scikit-learn.org. (2023). *Sklearn.linearmodel.linearregression*. Retrieved October 8, 2023, from https://scikitlearn.org/stable/modules/generated/sklearn.linearmodel.LinearRegression.html

Statsmodels-0.14.0. (2023). *Statsmodels.org*. Retrieved October 12, 2023, from https://www.statsmodels.org/stable/index.html

Taylor, M. (2022). *Homoscedasticity and heteroscedasticity*. Retrieved October 15, 2023, from https://www.vexpower.com/brief/homoskedasticity

Tiberius, C., Van de Marel, H., Reudink, R., & Van Leijen, F. (2022). *Surveying and mapping*. TU Delft OPEN.

Van der Meel, E. (2013). Red light running by cyclists.

Wind, J. (2022). Cyclist push button behaviour at signalized intersections.

Wu, C., Yao, L., & Zhang, K. (2012). The red-light running behavior of electric bike riders and cyclists at urban intersections in china: An observational study [PTW + Cognitive impairment and Driving Safety]. *Accident Analysis Prevention*, *49*, 186–192. https://doi.org/https://doi.org/10.1016/j.aap.2011.06.001

A

DataFrames

Appendix A contains all DataFrames that have been referred to in this document.

	Date	Count	StartGreen	EndGreen	StartRed	EndRed	LengthRed
0	2021-01-20	1	0.0	27033.3	27036.3	27054.1	17.8
1	2021-01-20	2	27054.1	27060.4	27063.4	27079.6	16.2
2	2021-01-20	3	27079.6	27102.5	27105.5	27122.8	17.3
3	2021-01-20	4	27122.8	27141.7	27144.7	27159.5	14.8
4	2021-01-20	5	27159.5	27165.8	27168.8	27188.3	19.5
...	...	...	...	...	...	...	...
28470	2021-02-22	1185	71745.8	71772.1	71775.1	71789.3	14.2
28471	2021-02-22	1186	71789.3	71799.1	71802.1	71817.1	15.0
28472	2021-02-22	1187	71817.1	71831.5	71834.5	71848.8	14.3
28473	2021-02-22	1188	71848.8	71861.2	71864.2	71878.8	14.6
28474	2021-02-22	1189	71878.8	86399.0	86402.0	NaN	NaN

Figure A.1: Sample of start and end times of green- and red-light cycles

	Date	Start Detection [ds]	End Detection [ds]	End Detection	Detection Time [ds]	Time of Crossing [s]
0	2021-01-20	39692.0	39695.0	01:06:09	3.0	3970.00
1	2021-01-20	44331.0	44332.0	01:13:53	1.0	4433.37
2	2021-01-20	47110.0	47111.0	01:18:31	1.0	4711.27
3	2021-01-20	47472.0	47474.0	01:19:07	2.0	4747.73
4	2021-01-20	47782.0	47784.0	01:19:38	2.0	4778.73
...	...	...	...	...	...	...
22733	2021-02-22	854091.0	854094.0	23:43:29	3.0	85409.90
22734	2021-02-22	856440.0	856443.0	23:47:24	3.0	85644.80
22735	2021-02-22	856884.0	856887.0	23:48:08	3.0	85689.20
22736	2021-02-22	863499.0	863501.0	23:59:10	2.0	86350.43
22737	2021-02-22	863513.0	863515.0	23:59:11	2.0	86351.83

Figure A.2: Detection times and times of crossings

Date	Hour	Wind speed [m/s]	Temperature [C]	Rain [mm]	Hour	Wind speed [m/s]	Temperature [C]	Rain [mm]
2021-01-29	1	3.45	8.78	0.00	13	8.65	9.57	0.00
2021-01-29	2	4.25	9.04	0.05	14	8.65	9.14	0.00
2021-01-29	3	4.25	9.14	1.82	15	8.07	8.99	0.00
2021-01-29	4	5.48	9.37	1.89	16	7.87	8.89	0.00
2021-01-29	5	7.87	8.71	1.45	17	8.30	8.51	0.00
2021-01-29	6	8.10	8.58	1.23	18	9.07	7.58	0.00
2021-01-29	7	9.68	8.77	1.73	19	9.45	7.01	0.00
2021-01-29	8	10.48	8.70	0.35	20	8.45	6.26	0.00
2021-01-29	9	10.67	8.68	0.00	21	6.45	5.80	0.00
2021-01-29	10	9.45	9.03	0.00	22	4.65	5.21	0.00
2021-01-29	11	9.45	9.18	0.00	23	4.87	4.35	0.35
2021-01-29	12	9.22	9.47	0.00	24	4.90	3.16	0.19

Figure A.3: Hourly weather data for a random day, averaged with triangular interpolation

	Date	Hour	Violations	Wind speed [m/s]	Temperature [C]	Rain [mm]	Passing Vehicles	Traffic Light Cycles
0	2021-01-20	1	0	10.10	7.29	0.0	0	0
1	2021-01-20	2	0	9.52	7.48	0.0	1	0
2	2021-01-20	3	0	9.87	7.70	0.0	0	0
3	2021-01-20	4	0	10.10	7.76	0.0	0	0
4	2021-01-20	5	0	10.67	7.95	0.0	4	0
5	2021-01-20	6	0	10.10	8.12	0.0	31	0
6	2021-01-20	7	0	9.87	8.23	0.0	151	0
7	2021-01-20	8	4	9.87	8.17	0.0	248	51
8	2021-01-20	9	12	10.10	8.26	0.0	387	118
9	2021-01-20	10	15	10.10	8.52	0.0	247	100
10	2021-01-20	11	13	9.52	8.48	0.0	308	118
11	2021-01-20	12	9	9.68	8.63	0.0	243	99
12	2021-01-20	13	32	8.90	9.13	0.0	288	115
13	2021-01-20	14	18	8.71	9.31	0.0	293	114
14	2021-01-20	15	11	9.68	9.33	0.0	248	107
15	2021-01-20	16	21	9.68	9.25	0.0	320	115
16	2021-01-20	17	22	9.10	9.27	0.0	334	115
17	2021-01-20	18	32	9.68	9.78	0.0	322	116
18	2021-01-20	19	24	9.87	10.27	0.0	171	91
19	2021-01-20	20	11	10.90	9.79	0.0	123	77
20	2021-01-20	21	0	10.10	10.08	0.0	85	0
21	2021-01-20	22	0	10.45	10.34	0.0	88	0
22	2021-01-20	23	0	10.67	10.28	0.0	44	0
23	2021-01-20	24	0	10.90	10.78	0.0	38	0

Figure A.4: DataFrame with all dependent and independent variables for the first day.

B

Independent Variables Source Code

This appendix contains the Python code of data processing of the independent variables. Appendix B.1 shows the code for the processing of the weather data. In Appendix B.2 the for the processing of the traffic intersection data is shown.

B.1. Weather Data

```
# Construct data set with hourly meteorological and red-light violation data

# Load data for all three weather stations
hourly_data_voorschoten = np.loadtxt('hourly_data_voorschoten.txt', dtype='int', skiprows=4, delimiter=',')
hourly_data_hvh = np.loadtxt('hourly_data_hvh.txt', dtype='int', skiprows=4, delimiter=',')
hourly_data_rotterdam = np.loadtxt('hourly_data_rotterdam.txt', dtype='int', skiprows=4, delimiter=',')

# Create dataframes for all three weather stations
df_voorschoten = pd.DataFrame(hourly_data_voorschoten)
df_hvh = pd.DataFrame(hourly_data_hvh)
df_rotterdam = pd.DataFrame(hourly_data_rotterdam)

# Create empty lists for triangular interpolated data
hourly_windspeed = np.zeros(816)
hourly_temperature = np.zeros(816)
hourly_rain = np.zeros(816)

# Use a loop to interpolate and store hourly weather data
for i in range(816):
    hourly_windspeed[i] = (df_voorschoten.iloc[i,-3] * (43.2/220)) + (df_hvh.iloc[i,-3] * (49.8/220)) + (df_rotterdam.iloc[i,-3] * (126.9/220))
    hourly_temperature[i] = (df_voorschoten.iloc[i,-2] * (43.2/220)) + (df_hvh.iloc[i,-2] * (49.8/220)) + (df_rotterdam.iloc[i,-2] * (126.9/220))
    hourly_rain[i] = (df_voorschoten.iloc[i,-1] * (43.2/220)) + (df_hvh.iloc[i,-1] * (49.8/220)) + (df_rotterdam.iloc[i,-1] * (126.9/220))

# Create data frame with new, average values
df_model = df_voorschoten
df_model[1] = pd.to_datetime(df_model[1], format='%Y%m%d')
df_model.columns = ['Station', 'Date', 'Hour', 'Wind speed', 'Temperature', 'Rain']
df_model.set_index('Date', inplace=True)
df_model.drop('Station', axis=1, inplace=True)

df_model['Wind speed'] = hourly_windspeed
df_model['Temperature'] = hourly_temperature
df_model['Rain'] = hourly_rain

# Change all negative rain values to zero
df_model['Rain'] = df_model['Rain'].apply(lambda x: max(0, x))

```

```
# Set all values to the correct unit [m/s, C, mm]
df_model['Wind speed'] = df_model['Wind speed'].div(10)
df_model['Temperature'] = df_model['Temperature'].div(10)
df_model['Rain'] = df_model['Rain'].div(10)
df_model.columns = ['Hour', 'Wind speed [m/s]', 'Temperature [C]', 'Rain [mm]']
df_model = df_model.round(2)

df_model.reset_index(inplace=True)

df_model.to_csv('df_model.csv', index=False)
```

B.2. Traffic Data

```
# Passing vehicle count
# Read the Excel file
vehicle_data = pd.read_excel('traffic_light_sensor.xlsx')
df_vehicles = pd.DataFrame(vehicle_data)

# Remove last rows of dataframe
df_vehicles = df_vehicles[:-11]

# Initialize the 'date' column
df_vehicles['date'] = '2021-01-20'
# df_vehicles['date'] = pd.to_datetime(df_vehicles['date'])

date_list = ['2021-01-20', '2021-01-21', '2021-01-22', '2021-01-23', '2021-01-24', '
    2021-01-25', '2021-01-26', '2021-01-27', '2021-01-28', '2021-01-29',
    '2021-01-30', '2021-01-31', '2021-02-01', '2021-02-02', '2021-02-03', '2021-02-04', '
        2021-02-05', '2021-02-06', '2021-02-07', '2021-02-08',
    '2021-02-09', '2021-02-10', '2021-02-11', '2021-02-12', '2021-02-13', '2021-02-14', '
        2021-02-15', '2021-02-16', '2021-02-17', '2021-02-18',
    '2021-02-19', '2021-02-20', '2021-02-21', '2021-02-22']

date_count = 0

for i in range(len(df_vehicles)):
    if df_vehicles['periode'].iloc[i] == '00:00-00:05':
        date_count += 1

    df_vehicles.loc[i, 'date'] = date_list[date_count]

# Remove first two rows and reset index
df_vehicles = df_vehicles[2:]
df_vehicles = df_vehicles.reset_index(drop=True)

# Rename columns
df_vehicles.columns = ['periode', '21', '22', '81', 'date']

# Convert the 'periode' column to datetime format
df_vehicles['periode_start'] = pd.to_datetime(df_vehicles['periode'].str.split('-').str[0],
    format='%H:%M')
df_vehicles['hour'] = df_vehicles['periode_start'].dt.hour

# Group by 'date', 'hour', and aggregate the sum of '21', '22', and '81' columns
df_grouped = df_vehicles.groupby(['date', 'hour']).agg({
    '21': 'sum',
    '22': 'sum',
    '81': 'sum'
}).reset_index()

# Reset the index and rename the columns
df_grouped = df_grouped.rename(columns={'date': 'Date', 'hour': 'Hour', '21': 'Sum_21', '22':
     'Sum_22', '81': 'Sum_81'})

# Change hour column into 1-24 hour format
df_grouped['Hour'] = df_grouped['Hour'] + 1

# Insert row for first hour, first day
first_row = {'Date': '2021-01-20', 'Hour': 1, 'Sum_21': 0, 'Sum_22': 0, 'Sum_81': 0}
```

```
df_grouped = pd.concat([pd.DataFrame(first_row, index=[0]), df_grouped], ignore_index=True)
df_grouped['Date'] = pd.to_datetime(df_grouped['Date'])

# Sum hourly car presence
df_grouped['Sum'] = df_grouped.apply(lambda row: row['21'] + row['22'] + row['81'], axis=1)

# Remove weekend days
df_grouped = df_grouped[df_grouped['Date'].dt.weekday < 5]
df_grouped = df_grouped.reset_index(drop=True)

# df_grouped.to_csv('df_grouped.csv', index=False)

# Traffic light cycle count
# Read the Excel file
cycle_data = pd.read_excel('traffic_light_cycles.xlsx')
df_cycles = pd.DataFrame(cycle_data)

#Remove last rows that go past midnight
df_cycles = df_cycles[:-11]

# Initialize the 'date' column
df_cycles['date'] = '2021-01-20'

date_list = ['2021-01-20', '2021-01-21', '2021-01-22', '2021-01-23', '2021-01-24', '
    2021-01-25', '2021-01-26', '2021-01-27', '2021-01-28', '2021-01-29',
    '2021-01-30', '2021-01-31', '2021-02-01', '2021-02-02', '2021-02-03', '2021-02-04', '
        2021-02-05', '2021-02-06', '2021-02-07', '2021-02-08',
    '2021-02-09', '2021-02-10', '2021-02-11', '2021-02-12', '2021-02-13', '2021-02-14', '
        2021-02-15', '2021-02-16', '2021-02-17', '2021-02-18',
    '2021-02-19', '2021-02-20', '2021-02-21', '2021-02-22']

date_count = 0

for i in range(len(df_cycles)):
    if df_cycles['periode'].iloc[i] == '00:00-00:05':
        date_count += 1

    df_cycles.loc[i, 'date'] = date_list[date_count]

# Remove first two rows and reset index to start on full hour
df_cycles = df_cycles[2:]
df_cycles = df_cycles.reset_index(drop=True)

# Rename columns
df_cycles.columns = ['periode', 'traffic light', 'date']

# Convert the 'periode' column to datetime format
df_cycles['periode_start'] = pd.to_datetime(df_cycles['periode'].str.split('-').str[0],
    format='%H:%M')
df_cycles['hour'] = df_cycles['periode_start'].dt.hour

# Group by 'date', 'hour', and aggregate the sum of '21', '22', and '81' columns
df_grouped_cycles = df_cycles.groupby(['date', 'hour']).agg({'traffic light': 'sum'}).
    reset_index()

# Reset the index and rename the columns
df_grouped_cycles = df_grouped_cycles.rename(columns={'date': 'Date', 'hour': 'Hour', '
    traffic light': 'Traffic Light'})

# Change hour column into 1-24 hour format
df_grouped_cycles['Hour'] = df_grouped_cycles['Hour'] + 1

# Insert row for first hour, first day
first_row = {'Date': '2021-01-20', 'Hour': 1, 'Traffic Light': 0}
df_grouped_cycles = pd.concat([pd.DataFrame(first_row, index=[0]), df_grouped_cycles],
    ignore_index=True)
df_grouped_cycles['Date'] = pd.to_datetime(df_grouped_cycles['Date'])

# Remove weekend days
df_grouped_cycles = df_grouped_cycles[df_grouped_cycles['Date'].dt.weekday < 5]
df_grouped_cycles = df_grouped_cycles.reset_index(drop=True)
```

C

Red-Light Violations Source Code

Appendix C describes the steps that have been performed in order to find the number of red-light violations. The original Python code has been divided into several parts. Every part of code is preceded by a descriptive text. Appendix C is made up of five sections. The first two sections explain how the green and red time intervals have been extracted from the Vlog data. The third section describes the analysis of the trajectory data. Section Appendix C.4 illustrates the way the times of crossing of each cyclist were found. Finally, in Appendix C.5 is explained how the red-light violations were found.

C.1. Finding Red Time Intervals

To start, the green time intervals of the traffic light are loaded and converted from a MATLAb file into a Pandas DataFrame. The green-time intervals for this period have been previously calculated by a Bachelor End Work (BEP) student. The source code of this process can be found in Appendix C.2.

Next, the columns are renamed and the date columns is set as datetime. Then the start and end time of each red-light cycle is calculated. First, a StartRed column is created with all values set to zero. Then, the greentime values are converted from deciseconds to seconds. After that, the StartRed values are calculated by adding three seconds to the EndGreen times and stored in the new column. These three seconds are the exact length of each yellow-light period between the end of a green-light period and the start of a red-light period. The EndRed times are set equal to the start of the next green-light interval. Also, the lenght of each red-light cycle is calculated and stored. Finally, the first and last row of each date is removed because every day starts and end with the traffic signal on stand-by, i.e. flashing yellow.

```
# GreenTime Data

greentimes_matdata = scipy.io.loadmat('GreenTimes.mat')
greentimes_matvariable = greentimes_matdata['GreenTimes']

df_greentimes = pd.DataFrame(greentimes_matvariable)

df_greentimes[0] = pd.to_datetime(df_greentimes[0], format='%Y%m%d')
df_greentimes.columns = ['Date', 'Count', 'StartGreen', 'EndGreen']

# Calculate RedTime Data
# Create columns for start and end red times
df_greentimes = df_greentimes.assign(StartRed=0)

# Convert times to seconds
df_greentimes['StartGreen'] = df_greentimes['StartGreen'].div(10)
df_greentimes['EndGreen'] = df_greentimes['EndGreen'].div(10)

# Calculate red times
df_greentimes['StartRed'] = df_greentimes['EndGreen'] + 3
df_greentimes['EndRed'] = df_greentimes['StartGreen'].shift(-1)
df_greentimes['LengthRed'] = df_greentimes['EndRed'] - df_greentimes['StartRed']
```

```python

# Remove first and last row for every date due to traffic light being on stand-by
def remove_first_last_rows(group):
    return group.iloc[1:-1]

df_greentimes = df_greentimes.groupby('Date').apply(remove_first_last_rows)

# Reset the index of the modified groups
df_greentimes = df_greentimes.reset_index(drop=True)
```

C.2. Green Time Source Code

```
clear all; clc; close all; fclose all;
%------------------------------------------------------------------------
% This code is meant to identify green periods for the traffic light
% Jaffalaan
%------------------------------------------------------------------------
% Load file with start times
dirname = 'C:\Users\wdaamen\TrafficLight\Data\';
fname = 'startTimes.txt';
fileID = fopen([dirname fname],'r');
formatSpec = '%d %d %d %d %d';
sizeA = [5 Inf];
startTimes = fscanf(fileID,formatSpec,sizeA);
startTimes = startTimes';

GreenTimes =[];
%% Add a loop over all days
for day = 1:size(startTimes,1)
    day
    % Load files with vlog data
    fname = ['kr067_' num2str(startTimes(day,1)) '.txt'];
        fileID = fopen([dirname fname],'r');
        formatSpec = '%d %d';
        sizeA = [2 Inf];
        VRIdata = fscanf(fileID,formatSpec,sizeA);
        VRIdata = VRIdata';

        % Correct times
        correction = startTimes(startTimes(:,1) == str2double(fname(findstr(fname,'_')+1:
    findstr(fname,'.')-1)),2:5);
        VRIdata(:,1) = VRIdata(:,1) + correction(1)*36000 + correction(2)*600 + correction(3)
    *10 + correction(4);

        % Calculate internal green
        VRIdata(:,3) = mod(VRIdata(:,2),32);

        startG = sort(VRIdata(find(VRIdata(:,3)==1),1));
        endG = sort(VRIdata(find(VRIdata(:,3)==6),1));

        % Create day boundaries
        % day ends with orange so if necessary add last green
        if endG(end) > startG(end)
            startG(end+1) = endG(end)+1;
        end
        endG(end+1) = 23*36000 + 59*600 + 59*10; % Still green at midnight (orange)

        % day starts with orange
        if endG(1) < startG(1)
            startG = [0; startG];
        else
            startG(1) = 0; % First green at midnight (orange)
        end

        Rec = 0;
        % check for incomplete cycles
        if length(endG) == length(startG)
            Rec = sum(startG<endG);
            if Rec ~= length(endG)
```

```
                for c = 1:min(length(endG),length(startG))
                    if startG(c)>endG(c)
                        endG(c)=[];
                        tbr = abs(length(endG)-length(startG));
                        countR = countR+1;
                        if tbr == 0
                            break
                        end
                    elseif startG(c+1)<endG(c)
                        startG(c+1)=[];
                        tbr = abs(length(endG)-length(startG));
                        countR = countR+1;
                        if tbr == 0
                            break
                        end
                    end
                end
            end
        else
            countR = 0;
            tbr = abs(length(endG)-length(startG));
            for c = 1:min(length(endG),length(startG))
                if startG(c)>endG(c)
                    endG(c)=[];
                    tbr = abs(length(endG)-length(startG));
                    countR = countR+1;
                    if tbr == 0
                        break
                    end
                elseif startG(c+1)<endG(c)
                    startG(c+1)=[];
                    tbr = abs(length(endG)-length(startG));
                    countR = countR+1;
                    if tbr == 0
                        break
                    end
                end
            end

        end


        % Restructure time into desired format
        GreenTimes = [GreenTimes; startTimes(day,1).*ones(length(startG),1) (1:length(startG)
    )' startG endG];
end

%% Save green times
save('GreenTimes','GreenTimes')
```

C.3. Plotting Trajectory Data

The trajectory is loaded converted from MAT (MATLAB) to a Pandas DataFrame. The columns are renamed and a DateTime column is added. Lastly, unnecessary columns are dropped.

```
#Trajectory Data

trajectories_matdata = scipy.io.loadmat('LabeledRawlTrajectories.mat')
trajectories_matvariable = trajectories_matdata['Trajectories']

df_trajectories = pd.DataFrame(trajectories_matvariable)

df_trajectories.columns = ['ID', 'Year', 'Month', 'Day', 'Hour', 'Minute', 'Second', '
    Second_Count', '?', 'x', 'y']

df_trajectories['Date'] = pd.to_datetime(df_trajectories[['Year', 'Month', 'Day']])

```

```
df_trajectories = df_trajectories.loc[:, ['Date', 'Year', 'Month', 'Day', 'Hour', 'Minute', 'Second', 'Second_Count', 'ID', '?', 'x', 'y']]
df_trajectories.drop('?', axis=1, inplace=True)

display(df_trajectories)
```

Next, all trajectories are plotted onto the provided background image. The extent of the image is changed in order to fit the trajectories more or less accurately. Also, the trajectories are slightly rotated to fit better onto the path of the bicycle lane.

```
# Display trajectories

background_image = mpimg.imread('JaffalaanX.png')

fig, ax = plt.subplots(figsize=(10,6.8))
ax.imshow(background_image, extent=[-2.75, 20.25, -14.3, 3.35], aspect='auto', origin='upper')

ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')

angle_degrees = -5
angle_radians = math.radians(angle_degrees)

for i in range(36502):

    plot_trajectory = df_trajectories[df_trajectories['ID'] == i+1]

    x = plot_trajectory['x']
    y = plot_trajectory['y']

    x_rotated = x * math.cos(angle_radians) - y * math.sin(angle_radians)
    y_rotated = (x * math.sin(angle_radians) + y * math.cos(angle_radians)) + 0.2

    ax.plot(x_rotated, -y_rotated, color='red', alpha=0.1)

plt.show()
```

The daily crossings based on the trajectory data are plotted as follows.

```
# Display crossing cyclists per day - trajectory data

# Remove duplicates to end up with only unique ID's, obtain frequencies
df_unique_ID = df_trajectories.drop_duplicates(subset='ID', keep='first')

# Weekdays only
df_unique_ID_week = df_unique_ID[df_unique_ID['Date'].dt.weekday < 5]
df_unique_ID_frequency_week = df_unique_ID_week.groupby('Date').size().reset_index(name='Frequency')

# Some statistics
lowest_daily_crossings = df_unique_ID_frequency_week['Frequency'].min()
lowest_daily_crossings_row = df_unique_ID_frequency_week[df_unique_ID_frequency_week['Frequency'] == lowest_daily_crossings]
lowest_daily_crossings_date = lowest_daily_crossings_row['Date'].dt.date.values[0]

highest_daily_crossings = df_unique_ID_frequency_week['Frequency'].max()
highest_daily_crossings_row = df_unique_ID_frequency_week[df_unique_ID_frequency_week['Frequency'] == highest_daily_crossings]
highest_daily_crossings_date = highest_daily_crossings_row['Date'].dt.date.values[0]

mean_daily_crossings = df_unique_ID_frequency_week['Frequency'].mean()

plt.figure(figsize=(10,6))

daily_crossers = df_unique_ID_week.groupby('Date').size()
daily_crossers.plot(kind='bar', label='Daily Crossers')

plt.axhline(y=mean_daily_crossings, color='red', linestyle='--', label='Mean Daily Crossings')

```

```
date_labels = df_unique_ID_frequency_week['Date'].dt.strftime('%Y-%m-%d')
plt.xticks(range(len(date_labels)), date_labels, rotation=45)
plt.tight_layout()
plt.xlabel('Date')
plt.ylabel('Daily Crossings')
plt.legend();
```

C.4. Calculation of Time of Crossings

The detection times were exported from CuteView to Excel for every date in our time period except for weekend days, as the traffic light is on stand-by (flashing yellow signal) during weekends. Some dates consisted of multiple Vlog files. These Vlog files were then combined into one Excel file.

Then, the Excel files are read and converted to a Pandas DataFrame. Next, the one single time column of the Excel file is converted to an end- and a start-time column. Because the Vlog files of each date do not start at the same time, an additional file with starting times for each date was provided together with the rest of the data. These starting times are subsequently added to the detection times. For the dates that consist of multiple Vlog files the start times were calculated manually and were already added in the Excel file. After that, the end-time column is copied to a time column and converted from decimal-seconds into hh:mm:ss format. Also, a date column with respective date is added. Next, the rows with detection times that run after midnight are removed. Removing the columns poses no problem as the traffic light is on stand-by during those hours. Finally, the DataFrames with detection times are converted to as csv-file and subsequently stored.

```
# Start times

start_times = np.loadtxt('startTimes.txt')

# Convert start times to deci-seconds
start_times_ds = np.zeros(24)

for i in range(24):

    start_times_sec = (start_times[i,1] * 3600) + (start_times[i,2] * 60) + (start_times[i
        ,3]) + (start_times[i,4] / 10)
    start_times_ds[i] = start_times_sec * 10
```

```
# Detection cable 263 - an example for 2021-01-20

# Construct dataframe
sensor263 = pd.read_excel('Sensor_263\sensor263_2021-01-20.xlsx')
df_sensor = pd.DataFrame(sensor263)

# Create end time column
df_sensor['eindtijd'] = df_sensor['starttijd'].shift(-1)

# Remove first and last row
df_sensor = df_sensor.iloc[1:]
df_sensor = df_sensor.iloc[:-1]

# Rename columns
df_sensor.columns = ['Unnamed', 'starttijd', 'detector263', 'eindtijd']

# Drop unnecessary columns
df_sensor = df_sensor.drop('Unnamed', axis=1)
df_sensor = df_sensor.drop('detector263', axis=1)

# Add start time
df_sensor = df_sensor + start_times_ds[0]

# Remove all even index rows
even_index_mask = df_sensor.index % 2 == 0
df_sensor = df_sensor[~even_index_mask]
df_sensor = df_sensor.reset_index(drop=True)

```

```
# Convert to hh:mm:ss
def deciseconds_to_hh_mm_ss(deciseconds):
    seconds = deciseconds / 10
    hours = seconds // 3600
    minutes = (seconds % 3600) // 60
    seconds = (seconds % 3600) % 60

    return f'{int(hours):02d}:{int(minutes):02d}:{int(seconds):02d}'

df_sensor['Time'] = df_sensor['eindtijd'].apply(deciseconds_to_hh_mm_ss)

df_sensor['Date'] = '2021-01-20'

df_sensor_20_01 = df_sensor.iloc[:-12]

df_sensor_20_01.to_csv('df_sensor_20_01.csv', index=False)
```

A slightly different procedure was followed for the dates 2021-02-02 and 2021-02-18, because on these days the detection cable was activated at the moment registration of that day had started. In this case all uneven rows had to be removed instead of the even numbered rows. Also, the first row was not removed.

```
# Repeat process for 04-02 and 18-02

# Construct dataframe
sensor263 = pd.read_excel('Sensor_263\sensor263_2021-02-18.xlsx')   #Change date here
df_sensor = pd.DataFrame(sensor263)

# Create end time column
df_sensor['eindtijd'] = df_sensor['starttijd'].shift(-1)

# Remove last row
df_sensor = df_sensor.iloc[:-1]

# Rename columns
df_sensor.columns = ['starttijd', 'detector263', 'eindtijd']

# Drop unnecessary columns
df_sensor = df_sensor.drop('detector263', axis=1)

# Add start time
df_sensor = df_sensor + start_times_ds[21]   #Change index here

# Remove even number rows
uneven_index_mask = df_sensor.index % 2 != 0
df_sensor = df_sensor[~uneven_index_mask]
df_sensor = df_sensor.reset_index(drop=True)

# Convert to hh:mm:ss
def deciseconds_to_hh_mm_ss(deciseconds):
    seconds = deciseconds / 10
    hours = seconds // 3600
    minutes = (seconds % 3600) // 60
    seconds = (seconds % 3600) % 60

    return f'{int(hours):02d}:{int(minutes):02d}:{int(seconds):02d}'

df_sensor['Time'] = df_sensor['eindtijd'].apply(deciseconds_to_hh_mm_ss)

df_sensor['Date'] = '2021-02-18'   #Change date here

df_sensor_18_02 = df_sensor.iloc[:]   #Change date, delete all rows over 24:00:00

df_sensor_18_02.to_csv('df_sensor_18_02.csv', index=False)   #Rename date twice
```

For the next step, the csv-files of all dates are read and converted into a new DataFrame. All the single DataFrames are then stored in a list and subsequently concatenated into one large DataFrame, which is sorted by date and time. Now we are left with one dataframe for all dates consisting of the following columns: start-time (deciseconds), end-time (deciseconds), crossing time (hh:mm:ss) and date (yyyy-mm-dd).

```
# Regroup all dataframes into one dataframe
import glob

directory = 'df_sensor/'
file_pattern = '*.csv'
sensor_dataframes = []

csv_file_list = glob.glob(directory + file_pattern)

for csv_file in csv_file_list:
    sensor_df = pd.read_csv(csv_file)
    sensor_dataframes.append(sensor_df)

sensor_dataframe = pd.concat(sensor_dataframes, ignore_index=True)

sensor_dataframe = sensor_dataframe.sort_values(by=['Date', 'Time'])
sensor_dataframe = sensor_dataframe.reset_index(drop=True)
```

Now we have a DataFrame with all detection times, all rows with a too large detection time can be removed. Also the time it takes to reach the stopping line can be added.

```
# Filter out all rows with a too large time difference

sensor_dataframe['Time Difference'] = sensor_dataframe['eindtijd'] - sensor_dataframe['
    starttijd']
mask = sensor_dataframe['Time Difference'] > 3

df_filtered_sensor = sensor_dataframe[~mask]

df_filtered_sensor = df_filtered_sensor.copy()
df_filtered_sensor['Crossing Time [s]'] = df_filtered_sensor['eindtijd'].div(10)

# Add extra time to crossing times
df_filtered_sensor.loc[df_filtered_sensor['Time Difference'] == 1.0, 'Crossing Time [s]'] +=
    0.17
df_filtered_sensor.loc[df_filtered_sensor['Time Difference'] == 2.0, 'Crossing Time [s]'] +=
    0.33
df_filtered_sensor.loc[df_filtered_sensor['Time Difference'] == 3.0, 'Crossing Time [s]'] +=
    0.5

df_filtered_sensor = df_filtered_sensor.reset_index(drop=True)

df_filtered_sensor.columns = ['Start Time [ds]', 'End Time [ds]', 'Time', 'Date', 'Time
    Difference [ds]', 'Crossing Time [s]']
```

C.5. Calculation of Violations

First, a new column is added to the crossing times DataFrame of which all values are by default set to False. Subsequently, a function is applied to the DataFrame in which, for all crossing times, is checked if each crossing time falls into one of the red time intervals. If this is the case the False value is set to True. The resulting DataFrame is exported to csv-file, due to the long running time of the code. The final results of the red-light violation data can be seen in Chapter 4.

```
# Run times of crossing through the greentime data

# Create new column to store true/false value
df_filtered_sensor = df_filtered_sensor.copy()
df_filtered_sensor.loc[:, 'Within Red Interval'] = False

def check_within_interval(row):
    for index, interval_row in df_greentimes[df_greentimes['Date'] == row['Date']].iterrows()
        :
        if interval_row['StartRed'] <= row['Crossing Time [s]'] <= interval_row['EndRed']:
            return True
    return False

# Apply the function to update the 'Within_Red_Interval' column
```

```
df_filtered_sensor['Within Red Interval'] = df_filtered_sensor.apply(check_within_interval,
    axis=1)

df_filtered_sensor.to_csv('df_filtered_sensor.csv', index=False)

# Construct dataframe violators
violators = pd.read_csv('df_filtered_sensor.csv')
df_violators = pd.DataFrame(violators)
```

D

Linear Regression Source Code

Appendix D contains the code used for the linear regression analysis. In Appendix D.1 is explained how the hourly values of all dependent and independent variables are merged into one DataFrame. Appendix D.2 shows the code for the SLR models. Next, Appendix D.3 shows the code for the MLR model. This section includes both the Statsmodels and the Scikit-learn code.

D.1. Constructing the DataFrame

```
import numpy as np
import pandas as pd
import scipy.io
import math
from matplotlib import pyplot as plt

# Weather data
weather = pd.read_csv('df_model.csv')
df_weather = pd.DataFrame(weather)

df_weather['Date'] = pd.to_datetime(df_weather['Date'])

# Vehicle data
vehicles = pd.read_csv('df_grouped.csv')
df_cars = pd.DataFrame(vehicles)

df_cars['Date'] = pd.to_datetime(df_cars['Date'])

# Traffic light data
traffic_light = pd.read_csv('df_grouped_cycles.csv')
df_traffic_light = pd.DataFrame(traffic_light)

df_traffic_light['Date'] = pd.to_datetime(df_traffic_light['Date'])

# Add violation data to DataFRame
crossers = pd.read_csv('df_filtered_sensor.csv')
df_crossers = pd.DataFrame(crossers)

# Combine date and time columns into a single datetime column
df_crossers['Datetime'] = pd.to_datetime(df_crossers['Date'] + ' ' + df_crossers['Time'])

# Extract the date and hour
df_crossers['Date'] = df_crossers['Datetime'].dt.date
df_crossers['Hour'] = df_crossers['Datetime'].dt.hour

# Create a pivot table to count the violations
violators = df_crossers.pivot_table(index=['Date', 'Hour'], values='Within Red Interval',
    aggfunc='sum', fill_value=0)
violators.reset_index(inplace=True)
violators.rename(columns={'Within Red Interval': 'Violation Count'}, inplace=True)

```

```
# Create a date-hour grid for all hours from 1 to 24
date_range = pd.date_range(start=violators['Date'].min(), end=violators['Date'].max(), freq='H')
date_hour_grid = pd.DataFrame({'Date': date_range.date, 'Hour': date_range.hour})

# Merge the grid with the violation data and fill missing counts with zeros
violators = date_hour_grid.merge(violators, on=['Date', 'Hour'], how='left').fillna({'Violation Count': 0}).astype({'Violation Count': int})

# Change DataFrame to 1-24 hour format
violators['Hour'] = violators['Hour'] + 1
violators['Date'] = pd.to_datetime(violators['Date'])

# Merge dataframes
df_merged = pd.merge(df_weather, violators, on=['Date', 'Hour'])
df_merged = pd.merge(df_merged, df_cars, on=['Date', 'Hour'])
df_merged = pd.merge(df_merged, df_traffic_light, on=['Date', 'Hour'])

df_filtered = df_merged[df_merged['Violation Count'] != 0] #Remove hours without violations
```

D.2. Simple Linear Regression Model

```
# Statsmodels
import statsmodels.api as sm

# Vehicles

X_cars = df_filtered['Sum'] # independent variable
y_cars = df_filtered['Violation Count'] # dependent variable

X_cars = sm.add_constant(X_cars) # add constant

slr_model_cars = sm.OLS(y_cars, X_cars) # Ordinary Least Squares
slr_reg_cars = slr_model_cars.fit()

print(slr_reg_cars.summary())

intercept = slr_reg_cars.params[0]
slope = slr_reg_cars.params[1]
y_pred = slr_reg_cars.predict(X_cars)

plt.figure(figsize=(10,6))

plt.scatter(cars, violations, marker='o', alpha=0.5, label='Data Points')
plt.plot(X_cars['Sum'], y_pred, color='red', label='Regression Line')
plt.xlabel('Cars per Hour')
plt.ylabel('Violations per Hour')
plt.title('Simple Linear Regression Passing Vehicles')
plt.grid()
plt.legend(loc='upper left')

residuals = slr_reg_cars.resid

# Histogram of residuals
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
plt.hist(residuals, bins=30, edgecolor='k')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.title('Residual Histogram of Passing Vehicles')

# Traffic light cycles

X_tlc = df_filtered[['Traffic Light']] # independent variable
y_tlc = df_filtered['Violation Count'] # dependent variable

X_tlc = sm.add_constant(X_tlc) # add constant

slr_model_tlc = sm.OLS(y_tlc, X_tlc) # Ordinary Least Squares
```

```
slr_reg_tlc = slr_model_tlc.fit()

print(slr_reg_tlc.summary())

intercept_tlc = slr_reg_tlc.params[0]
slope_tlc = slr_reg_tlc.params[1]
y_pred_tlc = slr_reg_tlc.predict(X_tlc)

plt.figure(figsize=(10,6))

plt.scatter(traffic_light, violations, marker='o', alpha=0.5, label='Data Points')
plt.plot(X_tlc['Traffic Light'], y_pred_tlc, color='red', label='Regression Line')
plt.xlabel('Traffic Light Cycles per Hour')
plt.ylabel('Violations per Hour')
plt.title('Simple Linear Regression Traffic Light Cycles')
plt.grid()
plt.legend(loc='upper left')

residuals_tlc = slr_reg_tlc.resid

# Histogram of residuals
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
plt.hist(residuals_tlc, bins=30, edgecolor='k')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.title('Residual Histogram of Traffic Light Cycles')
```

D.3. Multiple Linear Regression Model

```
# MLRM

X1_var = df_filtered[['Sum','Traffic Light']] # independent variables
y_var = df_filtered['Violation Count'] # dependent variable

X1_var = sm.add_constant(X1_var)

mlr_model = sm.OLS(y_var, X1_var).fit()

print(mlr_model.summary())

plt.figure(figsize=(10,6))

fitted_values = mlr_model.fittedvalues
residuals = mlr_model.resid
plt.scatter(fitted_values, residuals)
plt.xlabel("Fitted Violations")
plt.ylabel("Residuals")
plt.title("Residuals vs. Fitted Violations")
plt.axhline(y=0, color='r', linestyle='--')
plt.show()

import seaborn as sns
import statsmodels.graphics.regressionplots as smg

# Create a scatterplot with a regression line
sns.lmplot(x='Violation Count', y='Sum', data=df_filtered, ci=None)
plt.title(f'Partial Regression Plot for Passing Vehicles')

sns.lmplot(x='Violation Count', y='Traffic Light', data=df_filtered, ci=None)
plt.title(f'Partial Regression Plot for Traffic Light Cycles')
plt.show()

residuals_mlr = mlr_model.resid

# Histogram of residuals
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
plt.hist(residuals_mlr, bins=30, edgecolor='k')
```

```
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.title('Residual Histogram of MLRM')

# MLRM Scikit-learn

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X_train, X_test, y_train, y_test = train_test_split(X1_var, y_var, test_size=0.3,
    random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

plt.figure(figsize=(6, 6))
plt.scatter(y_test, y_pred)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.xlabel('Observed Violations', fontsize=12,)
plt.ylabel('Predicted Violations', fontsize=12)
plt.title('Observed vs. Predicted Red-Light Violations', fontsize=13, y=1.02)
```