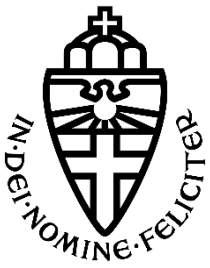
Radboud University



ARTIFICIAL INTELLIGENCE

Adapting and Employing Smart City Sensor Data for Strategic Planning

A thesis for the degree of Bachelor of Science in Artificial Intelligence

Author:

Domantas GIRŽADAS

(s1008829)

Internal supervisor:

Pim HASELAGER, PhD

(Radboud University, Donders Institute)

External supervisors:

Mariska BAARTMAN

Sjoerd DIKKERBOOM, MSc

Paul GEURTS, B

Jasper MEEKES, MSc

Arjen VERHULST, BA

(City Council of Nijmegen)

JULY 2020

Abstract

Today, more and more cities are adopting the ‘smart city’ trend by deploying new smart devices, trackers and sensors in public spaces. They gather data about the city life dynamics in order to make city planning and maintenance more efficient and effective. However, currently, most smart systems are heavily data-driven and require vast amounts of information for training before they can deliver any useful insights. The focus of this study is one of the straight-forward applications for pedestrian traffic data – building a prediction model (by using pedestrian traffic data from the city of Nijmegen (the Netherlands)). The aim is to explore different prediction methods: multi-layer perceptron, Gaussian process and support vector regression models, compared to an averaging-based baseline model and find one that performs the best with only a year or less of training data. Then, attempt to improve the applicability of that model further with the conversion of single value prediction to a prediction range as well as applying spatial interpolation to gain insight about unobserved areas in the city. The results show that a simple averaging-based model performs the best, given a low complexity version of the problem (only 168 possible value combinations for the input variables), which highlights the importance of problem analysis, while a described attempt of radial basis function interpolation of spatially sparse observations (predictions), resulting in only very high-level insights, shows how impactful problem representation is for the results of the system.

Table of contents

Introduction

The concept of ‘smart city’ is growing more and more popular. Cities are implementing vast numbers of different sensors and devices: from induction loops (for traffic light control) and electronic terminals (e.g. parking meters) to complex systems, such as pedestrian-tracking smart cameras. They help to utilise the power of the Internet of Things and fuel a data-driven approach to city planning.

The task of choosing which devices to use, however, is not trivial. There are many factors to evaluate and balance between. Today, one of the most concerning trade-off problems for smart systems is sacrificing user privacy for improved performance and/or personalised experience [1], [2]. Unsurprisingly, this problem becomes even more concerning when it comes to monitoring public spaces (e.g. tracking/counting pedestrians on city streets or inside shops). Some of the commercially available devices built for this task, have not been designed with privacy as the first priority. These devices perform with high accuracy at the cost of privacy of tracked subjects (e.g. Wi-Fi tracking systems that track and store subjects’ unique Wi-Fi MAC addresses [3], which has direct links to identifying individuals in public). That is why it is very important to evaluate each device option carefully.

The city of Nijmegen has recently expanded their toolkit with 21 computer vision-based smart sensors by Numina [4] and deployed them around the downtown area. This company has been chosen because of their “Intelligence without surveillance” philosophy [5]. Their sensor is capable of anonymously discriminating between pedestrians, cyclists, cars, trucks and busses in the field of view, while still doing it with high accuracy. This metric is measured separately for different observation sites, but as stated by Ilan Goodman (CTO of Numina), “<…> zone counts are usually accurate to within less than 5% [error rate].”. The anonymity is ensured by converting the obtained images into object data by using edge processing-based classification algorithm onboard the sensor itself and deleting the images immediately after they are processed. An example of ‘object data’ and more details can be found in the Numina Privacy Policy [5]. By using these sensors, the City Council of Nijmegen seeks to gain insight into the change in numbers of the previously mentioned traffic participants throughout any specific period. This kind of insight can aid the strategic planning of the city (e.g. highlighting overcrowded areas to induce better crowd distribution methods) or provide potential client estimations for the businesses of the city to help them with planning and scheduling.

Problem description

At the moment, the most popular methods, used for generating forecasts or gaining deeper-level understanding (e.g. finding patterns/trends in the data or extracting relations between observed variables), are based on machine learning or other heavily data-driven algorithms, which require vast amounts of training data to deliver any useful insights. However, since the 'smart city' trend is relatively young – a lot of cities are still only considering the possible smart device integrations – in a lot of cases, there are not much data to work with. Also, investing in gathering data for multiple years before getting any value back from it, is a risk not many local governments are willing to take. So, it is important to look into applications for relatively low amounts of data and evaluate their possibilities.

The problem mentioned above gave rise to the first research question of this thesis:

1. ***Given a low amount of available data, what is the best approach for modelling pedestrian traffic trends? What is the performance of this model and what are its possible applications?***

Many of these previously mentioned smart devices and trackers also have a very small functional range (e.g. narrow field of view on smart cameras) and as a result, provide a limited amount of insight. In the specific case of Numina sensors in Nijmegen, these data are presented as separate traffic participant count graphs, which depict the change of these numbers at the location of each sensor. While this knowledge is quite useful for specific locations, it does not provide any general insight about the traffic dynamics of the city, which would be very useful for strategic planning. A straightforward way of tackling this problem is increasing the sensor density by installing more of them. However, if software solutions are overlooked, the available tools are not utilised maximally, which means that the resources invested into their production and integration are not spent efficiently. That is why it is necessary to consider, test and evaluate a range of possible methods that would help us gain more insight from the same data. So, the second research question of this work is as follows:

2. ***How to interpolate spatially sparse pedestrian count (prediction or observed) data to get estimations for unobserved locations and visualise spatial trends of city traffic?***

Literature review

There are numerous different approaches in the field of traffic modelling and prediction. Most of the work seems to be focussed on short-term traffic forecasting [6]–[8] or look directly into higher-level effects, such as travel time [9], [10]. Siddiquee and Hoque [11] have used an Artificial Neural Network (ANN) to capture hourly traffic data patterns with promising results (Mean Absolute Error (MAE) of 12.67%). This approach seems to have some potential in this work as well. However, the lack of available training data might strongly affect the results of this kind of model.

Since the values, that are to be modelled, are simple (integers, representing the number of pedestrians in a certain time period), possible model choices are not limited to any specific field of research. For example, predicting the amount of bus riders is a very similar problem to pedestrian count prediction, as both values of interest are of the same format (single number, representing the count) and are usually mostly dependent on the 'time' or 'date' variables. Bhattacharya et al. [12] found that using a Gaussian Process Regression model works well for predicting bus ridership, based on time variables. So, this type of model might also work in certain cases of pedestrian traffic prediction. The types of models mentioned above (ANN, GPR) as well as a Support Vector Regression (SVR) model (one of the most popular machine learning algorithms) will be applied for the pedestrian count prediction problem and their performance will be compared to a baseline – an average count model. The hypothesis is that the results strongly depend on the complexity of the problem version – the number of input variables, the amount of possible values for these inputs and the complexity of the trends in the data. Because of the previously mentioned lack of available training data, the problem will possibly need to be simplified to a point where the complex machine learning models would not show much better performance than the baseline.

Regarding the interpolation of spatially sparse data, this problem is the most prominent in the field of Geographic Information Systems (GIS). Here, a wide range of interpolation methods is used to estimate both physical and socioeconomic phenomena [13]: elevations, climatic phenomena, soil pH [14], population densities, etc. Narrowing the scope down to spatial interpolation for traffic estimation, there is noticeably more attention towards graphical approaches (nodes and edges representing the street network – example from the City Council of Enschede [15] in Figure 1), such as the approach by Liebig et al. [16]. However, in this work, only Radial Basis Function interpolation (example in Figure 2) will be discussed, as it

was found to perform well by Zandi et al. [14], and the results of this kind of approach for pedestrian count estimation will be evaluated. The hypothesis for this part of the thesis is that the chosen methods (e.g. the representation complexity of the interpolation space) will have high impact on the results. The more detailed the input – the more precise and realistic the output.

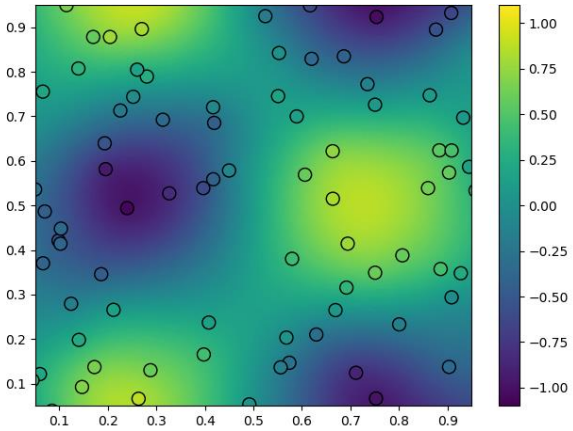


Figure 2: Example of RBF interpolation. Source: https://rbf.readthedocs.io/en/latest/

Figure 1: Example of a graphical representation. Source: [15]

Description of the data

As mentioned before, data from Numina sensors [4] for both the prediction and the spatial interpolation will be used for model training and testing. The format of these data is simple – the Numina API [17] allows the user to make a request for the number of any type of traffic participants (out of the five available categories: pedestrians, cyclists, cars, vans and busses), that were detected within any time period by a certain sensor. So, these numbers are just integers, representing the number of traffic participants within some period of time. More details about the data used for this thesis specifically can be found in the 'Data pre-processing' section.

The smart sensors have not been installed in Nijmegen simultaneously. The first one started operating early January of 2019, after which, new ones have been installed within different time intervals. This means that the amount of available data (after data clean-up and filtering described in 'Data pre-processing' section) differs for each location in the city, as seen in Appendix A: Data. One thing that might seem unexpected in this graph is the number of observed locations – 42, whereas it was mentioned that there are only 21 active sensors in the city so far. This is made possible with a feature in Numina API called 'behavior zones'.

Behavior zones

Computer vision-based smart systems are state-of-the-art. They grant computers the ability to gain a high-level understanding of images and videos. Recently, this field of research has gained a substantial amount of attention. However, the performance of these systems is not perfect and Numina sensors are not an exception. J. Ding [18] explains that there is quite a bit of room for improvement in terms of accurate pedestrian tracking. There are multiple steps in the process where problems may arise – errors in *detection*, *classification*, as well as *tracking*, may lead to poor performance of the device. One of the highlighted problems is a 'Track Break Error' where the track of an object moving through the field of view is mistakenly broken (Figure 3; see the gap in the green line on the right).

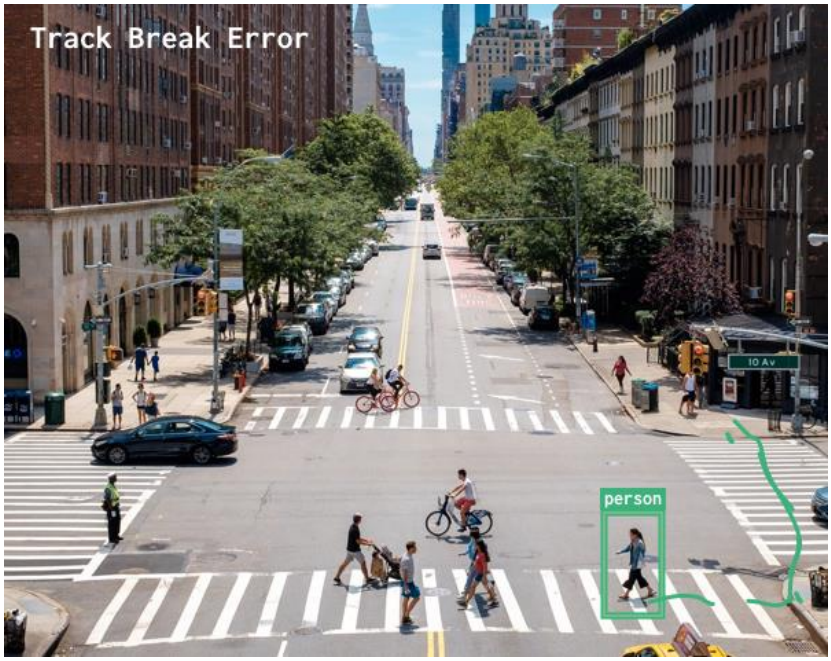


Figure 3: Track Break Error. Source: [18]

This can be caused by false-negative classifications or simply by objects moving behind obstructions (e.g. a person walking behind a billboard). The classification is completely anonymous – classified objects are only represented as a box with a label. So, if there is a noticeable gap in the tracked path, an algorithm that parses paths from detected points, struggles to combine the broken track, which results in two separate paths (which counts as two separate objects) being parsed. This means that if a Track Break Error happens within the sensor's field of view, one object is counted multiple times, which results in overcounting.

To tackle this problem, the developers of Numina recommended to use the 'behavior zone' feature of their API. This feature allows the user to make custom tracking zones within the

sensor's field of view. By reducing the size of this area, the chance of a broken track is reduced to a minimum and a better count accuracy is achieved. In addition, this feature allows to separate the sensor's field of view into multiple observation zones (e.g. if a sensor is observing an intersection, it is possible to get separate observations for each path/road going in/out of the said intersection, as illustrated by an example in Figure 4).

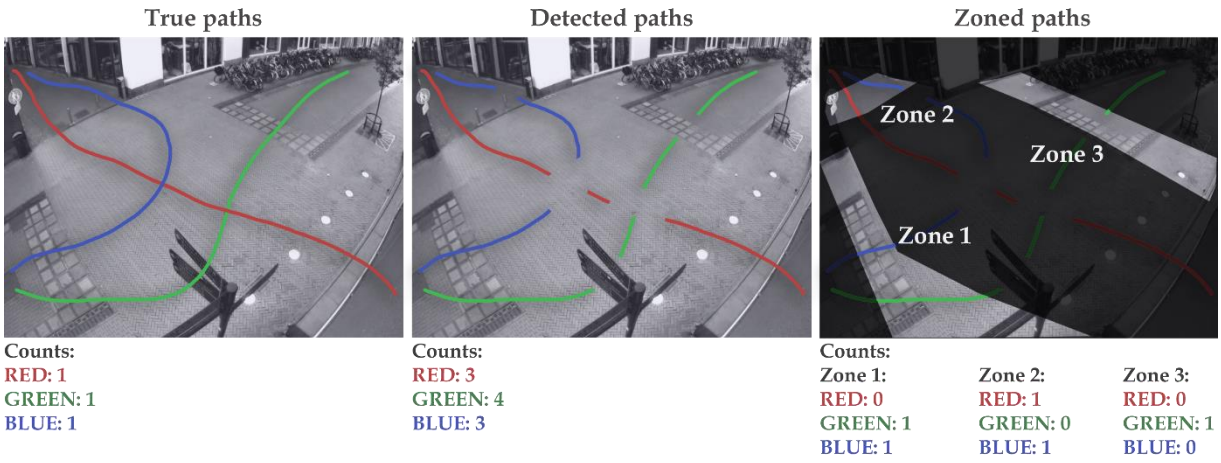


Figure 4: Effect of behavior zones. Source: Adapted from Numina API.

Approaches and methods

In order to make use of the data at hand, some initial analyses and pre-processing steps are necessary. A visual analysis of the data provides general insight about the trends present in the dataset. One important observation to make, while investigating the graphs of observations (Figure 5), is that individual locations have unique patterns of change. This means that every location will need a separate model to capture these unique trends.

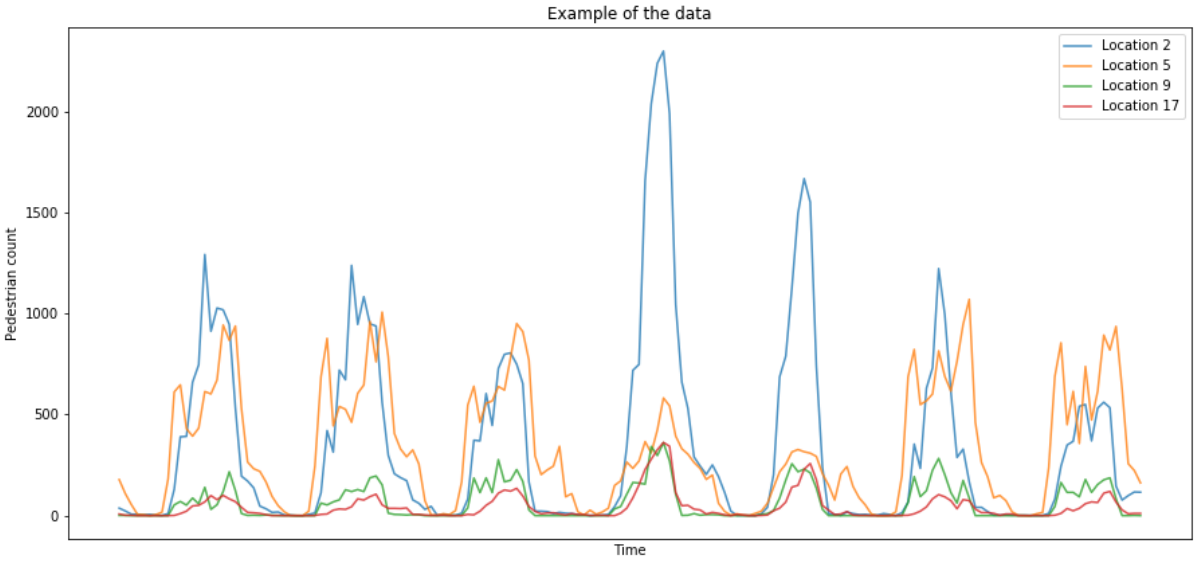


Figure 5: Example of the available data

As mentioned before, different locations do not have the same amount of valid observations in the present dataset (Appendix A: Data). This has a high impact on the choices for the pedestrian count prediction models:

1. Since by far not all locations contain data from all of the months, including a 'month' independent variable in the model is not a valid option. This would leave big gaps in the training sets, which would result in poor model performance.
2. Because of the previous point, using an independent variable 'day', becomes sub-optimal as well. The 'day' variable alone would only be able to capture monthly trends. However, as commonly known, traffic (pedestrian or not) usually does not follow monthly trends at all, but rather daily, weekly and yearly ones instead.

Because of these reasons, the prediction models will be trained on two independent variables:

- Day of the week (i.e. Monday - Sunday) – the model captures weekly trends, based on this variable;
- Hour of the day (i.e. 0-23) – the model captures daily trends, based on this variable.

These variables are not affected by the low amount of available training data anymore, as there is a low number of possible value combinations for these variables ($7 * 24 = 168$), even a low amount of data can fill all of the input combinations up with many observed values for the output.

Data pre-processing

Structured and clean data is important for achieving good results with a machine learning model. First step to obtaining it was to pull the data from the Numina API in a structured way. As explained in the previous section, ‘day of the week’ and ‘hour’ variables have been chosen for prediction. This meant that it was necessary to gather observations that come from timeframes that are at most one hour long. Otherwise, it would not be possible to directly match observations with the input variables. So, the observations, pulled from the Numina API, were aggregations of detected pedestrians within a time frame of one hour (e.g. Number of pedestrians in location 0 between 10:00:00 and 11:00:00 – example in Figure 6).

```
['2020-05-11T10:00:00' 366]
['2020-05-11T11:00:00' 398]
['2020-05-11T12:00:00' 379]
['2020-05-11T13:00:00' 425]
['2020-05-11T14:00:00' 380]
['2020-05-11T15:00:00' 320]
['2020-05-11T16:00:00' 115]
```

Figure 6: Example data

Again, different locations had different amounts of available data. This meant that when all of the data was pulled from the same interval (January 2019 – January 2020), certain locations had a lot of invalid observations with value ‘0’ from the time points when the sensors that track these locations were not installed yet (e.g. location 17 - Figure 7). To tackle this problem, any observations that had the value ‘0’ continuously for 12 hours or longer were discarded from the dataset. An example of the results can be seen in Figure 8.

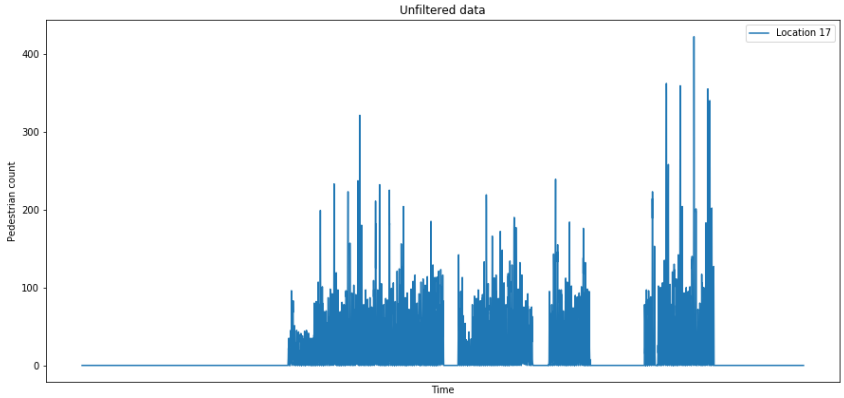


Figure 7: Unfiltered data from location 17

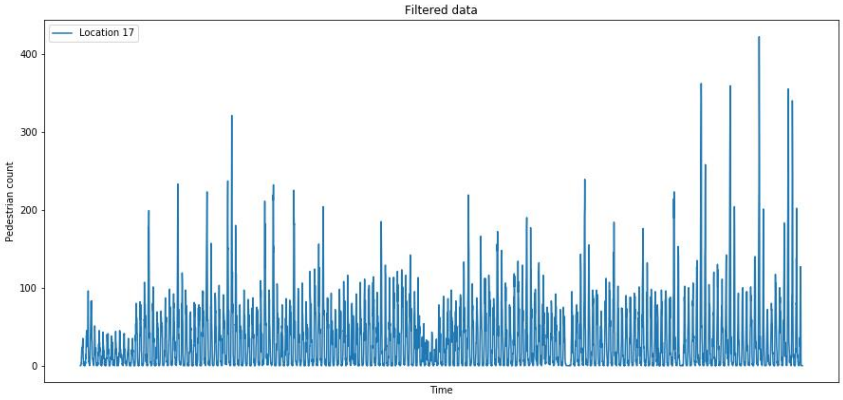


Figure 8: Filtered data from location 17

Another important step is outlier detection and removal. This step is crucial for reducing the influence of unobserved variables on the data. In this case, there was one obvious outlier in the pedestrian traffic data – the event of Four Days Marches (high peak in Figure 9). To avoid negative influence of this outlier on the prediction models, the interval where this obvious peak is present, has been removed from the dataset (result in Figure 10).

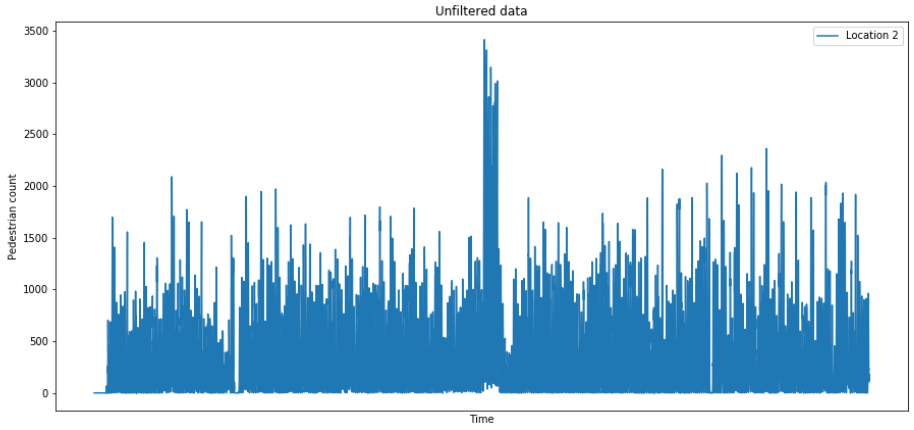


Figure 9: Four Days Marches unfiltered

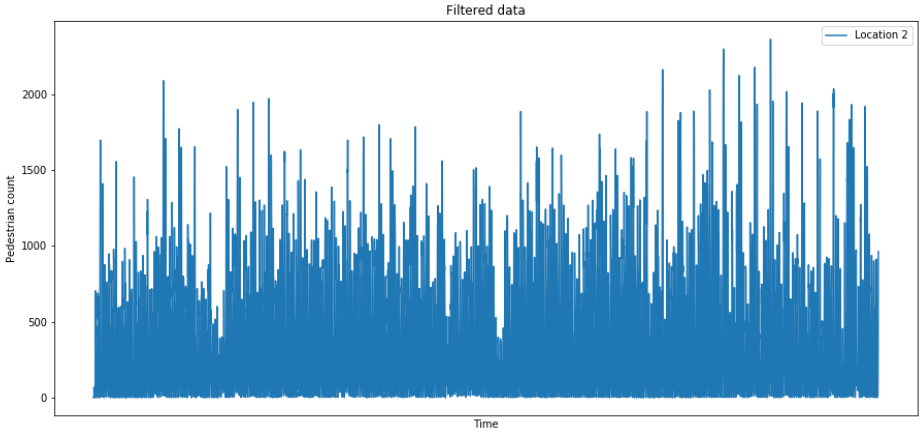


Figure 10: Four Days Marches filtered

Pedestrian count prediction models

As suggested and explained in the 'Literature review' section, in this work, the following machine learning models were built and trained for pedestrian count prediction (Table 1):

1. Multi-Layer Perceptron model (an artificial neural network), inspired by Siddiquee and Hoque [11]. Three different model architectures have been tested:
    - MLP1 has one hidden layer with 24 nodes – directly inspired by the model used by Siddiquee and Hoque [11].
    - MLP2 has one hidden layer with 168 nodes – inspired by the total number of possible value combinations for the input variables of the model.
    - MLP3 has two hidden layers with 168 nodes each – similar inspiration as for MLP2, but with a possible multi-layer effect in mind.
2. Gaussian Process Regression model, inspired by Bhattacharya et al. [12];
    - The kernel chosen for this model has been inspired by the kernel used in the example code, for the scikit-learn library [19].
3. Support Vector Regression model.
    - The values for hyperparameters C and ε have been computed by using formulas, suggested by Cherkassky and Ma [20].
    - The optimal value for γ has been found by scanning through a range of values and evaluating the mean absolute error as well as the prediction curves themselves. The MAE values can be seen in Figure 11. However, even though the error values were better for $\gamma > 0.05$, with these γ values, the prediction curves ended up not reaching the minimum value (0) as accurately, as lower values of γ (Figure 12 and Figure 13). This results in value overestimation for around half of the possible input combinations. As stated by my supervisors at the City Council of Nijmegen, value underestimation is preferred to overestimation. So, the value of γ was left at 0.05

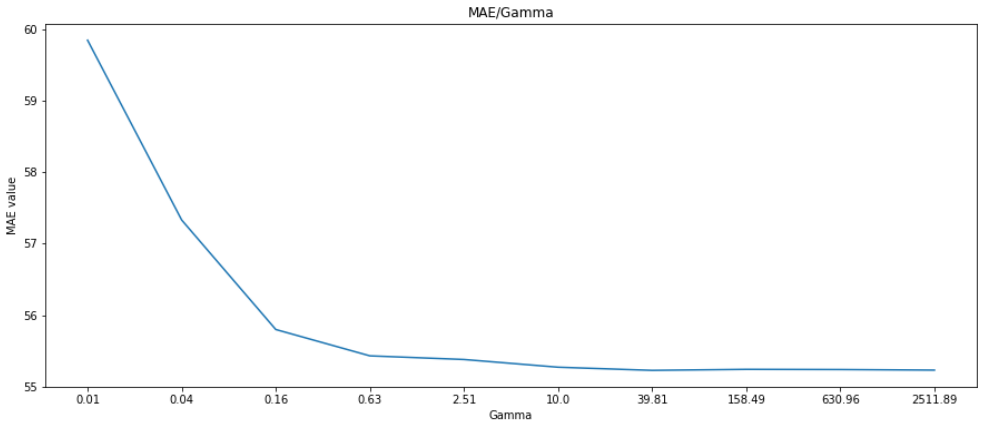


Figure 11: MAE/Gamma values for the SVR model

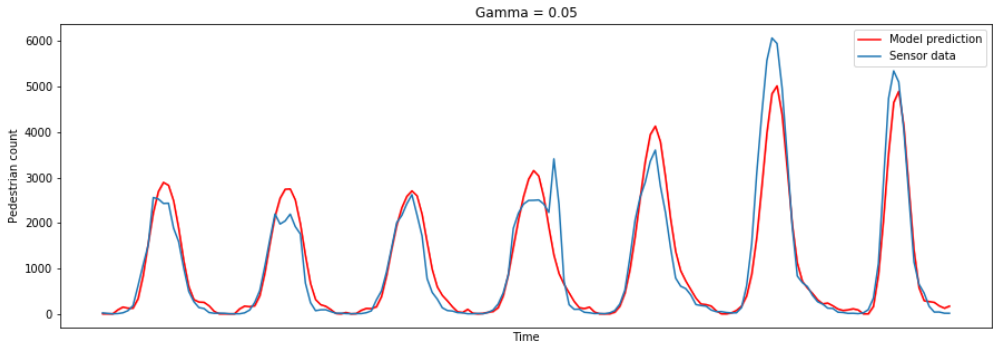


Figure 12: Gamma = 0.05.

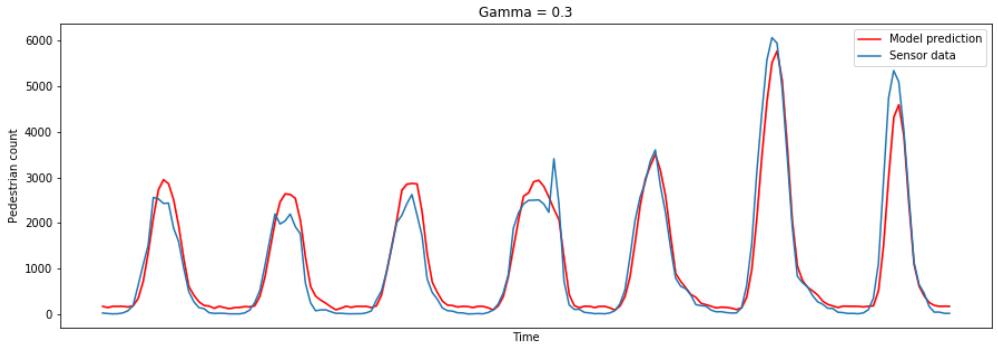


Figure 13: Gamma = 0.3.

Table 1: Prediction model hyperparameters

Model type	**Chosen hyperparameters**		
Multi-Layer Perceptron	**MLP1**	**Layers**: 1 Hidden layer – 24 nodes	All other parameters set
	MLP2	**Layers**: 1 Hidden layer – 168 nodes	
	MLP3	**Layers**: 2 Hidden layers – 168 nodes each.	

Gaussian Process Regression	**Kernel**: C(1.0, (1e-3, 1e3)) * RBF(10.0, (1e-3, 1e3)) (inspired by the example [19]), **Optimiser restarts**: 5	to 'scikit-learn' default values.
Support Vector Regression	**Kernel**: Radial Basis Function, $\boldsymbol{C} = \max(\lvert\bar{y} + 3\sigma_y\rvert, \lvert\bar{y} - 3\sigma_y\rvert)$, where $\bar{y}$ and σ_y are the mean and the standard deviation of the y values of training data [20]. $\boldsymbol{\varepsilon} = 3\sigma\sqrt{\frac{\ln n}{n}}$, where σ and n are the standard deviation and the amount of y values in the training data [20]. $\boldsymbol{\gamma} = 0.05$.	

The implementation of these models has been made easy by the 'scikit-learn' library for python [21]. This library contains classes for all three model types as well as methods for their evaluation (Mean Absolute Error and R^2 metric calculations).

In order to appropriately evaluate the performance of these models, a baseline model is necessary. Since the machine learning models are trained on 'day of the week' and 'hour' variables as input, a good baseline model would be one that aggregates data for each value of these variables and gives the mean as the result (e.g. a prediction for 'Wednesday, 11:00' is the mean value of all data points that come from Wednesday, 11:00 in the whole data set of each location).

After the best-performing model has been chosen (details in the 'Outcomes and results' section), it has been upgraded further with incorporation of an estimated yearly trend into the prediction process and prediction value conversion into prediction intervals:

1. The yearly trend scaling has been applied, as the original predictions are only based on 'day of the week' and 'hour' variables. This means that a prediction for a Wednesday, 10:00 in January is the same as a prediction for Wednesday, 10:00 in July. This is quite unrealistic. So, the predictions were scaled on a yearly trend, which has been estimated as follows:
    a. As there is not enough available data for individual location trend estimation (many locations do not have any data from multiple months), monthly averages

(over all locations) for hourly counts, have been collected. This way, a yearly trend that is general to the whole city is still extracted.

b. Scaling estimates, based on discrete monthly values is not a good option, as scaling predictions based on the month value would mean that there would be drastic value jumps between estimations of different months. For example, a prediction for January 31st would be quite different from the prediction for February 1st, as the 'January' scaling factor is applied to January 31st and 'February' one to February 1st - see the difference in bar heights for 'Jan' and 'Feb' in Figure 14. That is why a smoothened trend curve is needed. It was computed by using Radial Basis Function interpolation (implementation from 'SciPy' library for python [22]), applied on normalised (divided by the overall mean) monthly values (Figure 14).

c. By using this RBF trend estimation, a unique scaling factor was extracted for every day of the year (orange line in Figure 14).

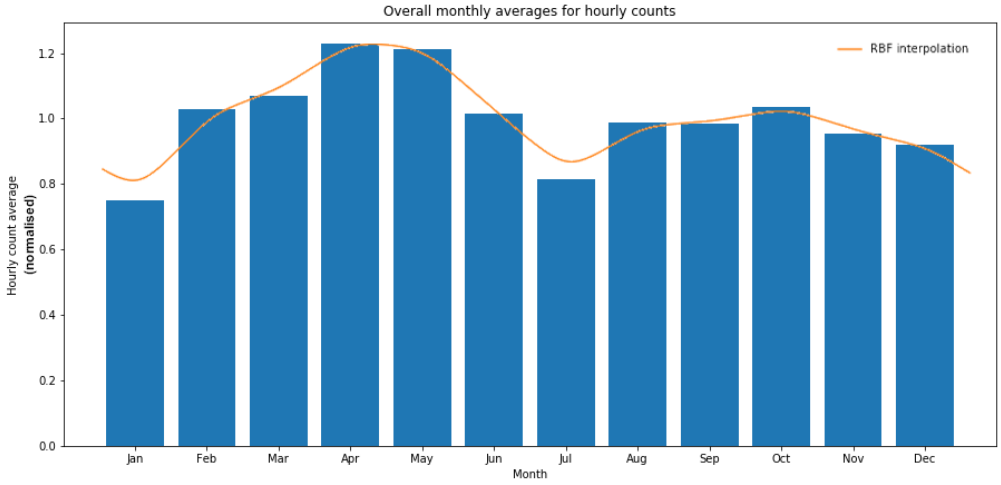


Figure 14: Smoothened RBF monthly trend.

2. Pedestrian traffic counts are very dependent on a lot of different factors (weather, local events, etc.), so a static model (a model which does not change the predictions, based on recent observations) would most likely struggle to capture these fluctuant values. To tackle this problem, a single value prediction can be converted into a prediction range. For the prediction interval of the estimation, each prediction is extended by one standard deviation of the data (specific to the day of the week and hour of the prediction), in both directions. This results in the following confidence interval for the prediction:

$$PI = \left[Prediction_{wd,h} - SD_{wd,h}, Prediction_{wd,h} + SD_{wd,h}\right],$$

where: *wd* – day of the week,

h – hour of the day.

This interval, together with the initial prediction is presented as the result of the model after applying this method. This means that the predictions are presented in a different format (i.e. [*min, pred, max*] array), which requires a different error evaluation method:

- When evaluating the predictions of this model, if the correct value is between the predicted *min* and *max* values, the error of the prediction is considered to be 0
- If the correct value is outside of the predicted interval, the error is considered to be the absolute error between the actual value and the predicted (*pred*) value.

Spatial interpolation model

In order to make a model that makes estimations for unobserved locations on the map, the following system has been built:

1. Generating a grid of custom resolution that will be used as a representation of the area of interest. For example, a 50x50 grid over the downtown area of Nijmegen.
2. Determining which grid cells represent observed locations (mock-up in Figure 15).
3. Setting the values of those cells to the observed (or predicted) value.
4. Estimating the values of unobserved grid cells by using two-dimensional Radial Basis Function (from the SciPy library [22]).
5. Overlaying the fully-estimated grid with an image that blocks non-street estimations.
    - This approach was suggested by Paul Geurts (City Council of Nijmegen) and showed better readability than overlaying the full interpolated grid over a map background (example in Figure 16).

This type of system can be used to interpolate different types of spatially sparse numeric observations, a few of them will be discussed in the sections ‘Outcomes and results’ and ‘Applications’.

Figure 15: Mock-up for representation of spatially sparse data.
Source: Adapted from https://www.openstreetmap.org/.

Figure 16: 'Interpolated grid on top of the map' example.

Outcomes and results

Once the models have been built and trained, they were tested on a validation set of the data – randomly selected observations that the models were not trained on.

Pedestrian count prediction model

The Mean Absolute Error and R^2 values for all of the models seem to lie within very close and mostly overlapping intervals (Figure 17). Nevertheless, both the highest R^2 value mean of 0.722 and the lowest MAE value mean of 45.854 (Table 2) belong to the predictions, made by the baseline model.

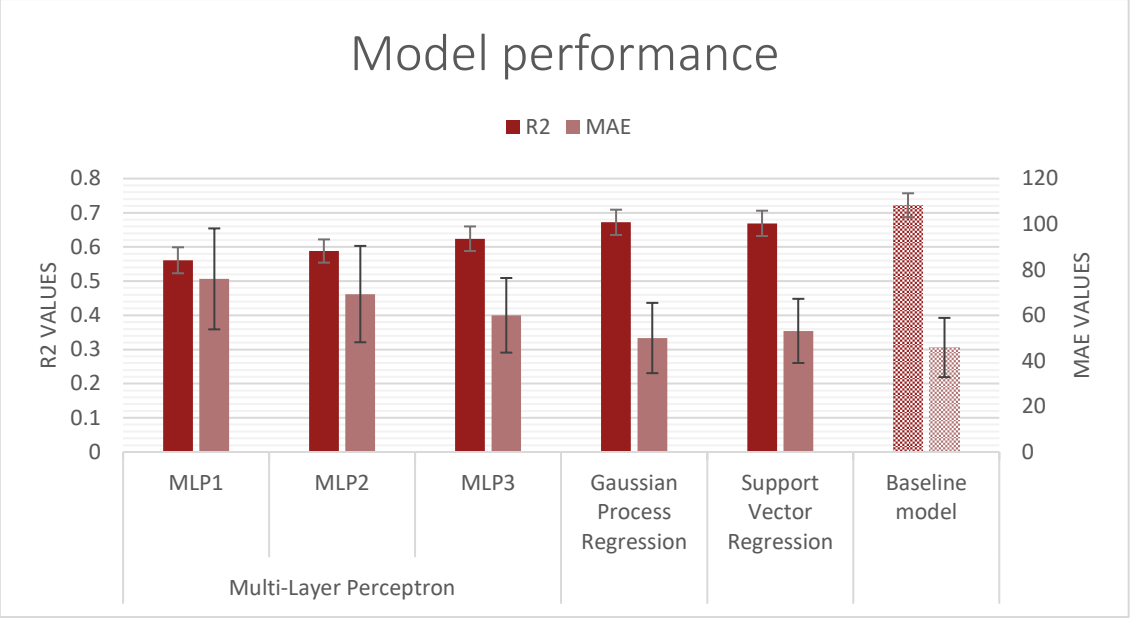


Figure 17: Model performance results,
("R2" – R^2, "MAE" – Mean Absolute Error)

Upon further inspection, it has been found that this model struggles to capture pedestrian counts during peak hours (around 12:00 – 16:00), which can be seen in Figure 19 (only comparing the two lines). Apart from that, no noticeable flaws have been found in the predictions of the averaging-based model.

Based on these results, the best model out of the tested ones is the baseline model. This means that the further improvements (yearly trend scaling and prediction interval inclusion) have been applied on this model. The additional upgrades have shown the following results (Figure 18 and Table 3):

1. Yearly trend-based daily prediction scaling only showed slight statistical improvements over the initial baseline model. The MAE has been decreased by only 1.67% and the R^2 value stayed nearly the same with only a 0.55% increase.
2. Converting the prediction into an interval seemed to have a high impact on the results. While the R^2 value did not change, the Mean Absolute Error has decreased by 45.44%. The prediction intervals have different widths at different times of the day and days of the week (as seen in Figure 19):
    - The width of the interval is high during the times when there is high variability in the pedestrian count (e.g. the number of people in the streets at 12:00-15:00 differs a lot, based on multiple unobserved factors, such as weather, holidays, etc.)

- The width of the interval is smaller for the predictions at times when there is not much variability (e.g. the number of people in the streets of Nijmegen at night does not depend on as many unobserved factors as at day, hence the precision of the prediction is higher).

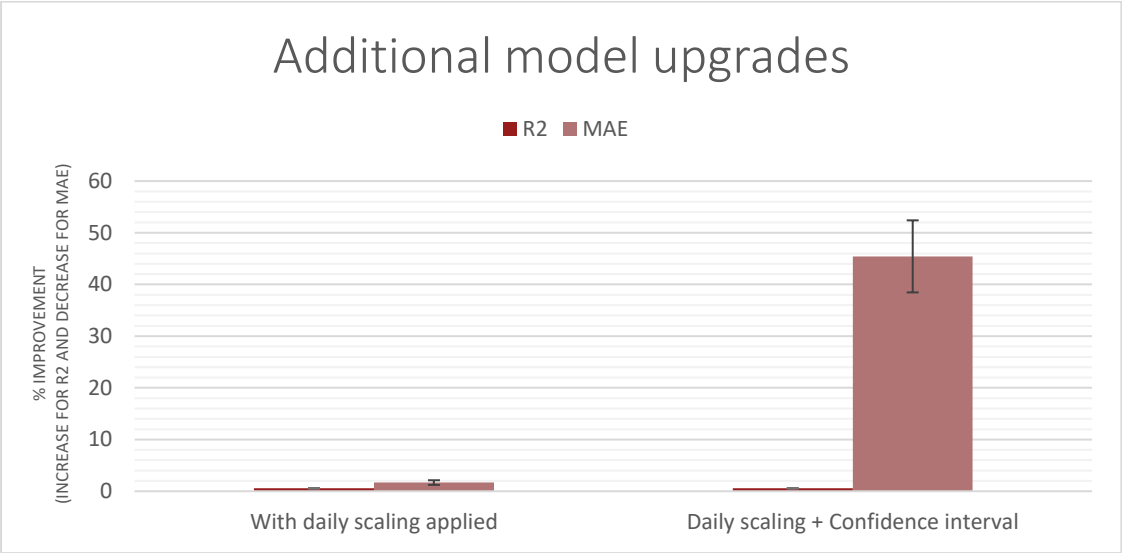


Figure 18: Additional model upgrades

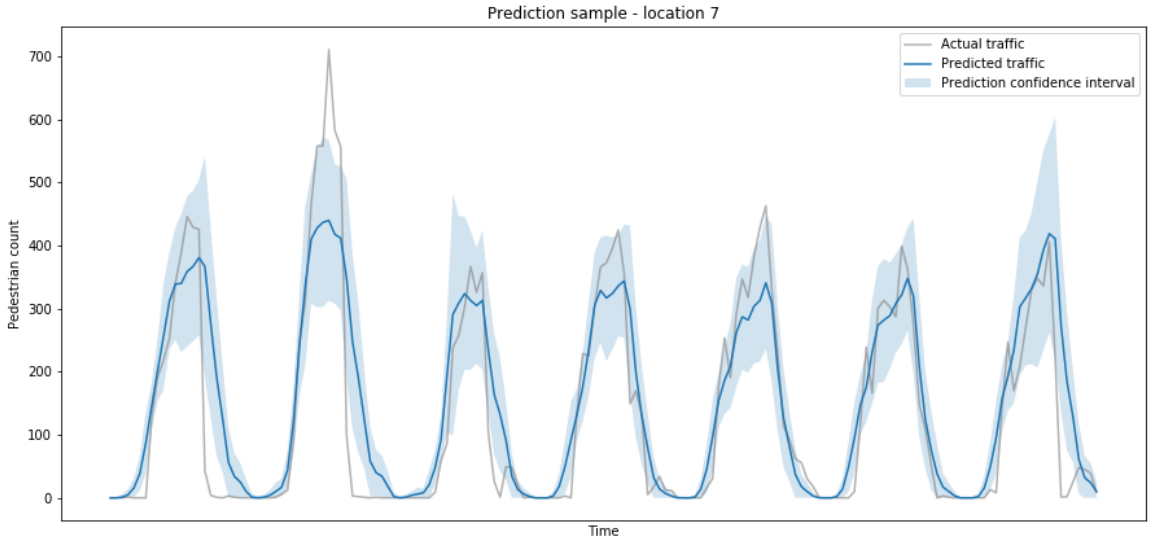


Figure 19: Prediction sample from location 7

Spatial interpolation model

For demonstration purposes, the interpolation grid resolution has been set to 1600x950, as it matches the aspect ratio and size of the overlay map image (Figure 29).

The results of the simple (grid) representation approach highly depend on all of the variables that the system contains: radial basis function parameters (*ε, smoothing factor*), grid resolution,

number of observed cells as well as the locations of these cells. However, based on a qualitative visual analysis (minimising the negative effects described below), the following RBF parameters have been chosen.

$$\varepsilon = 3;$$

$$\textit{smoothing factor} = 5.$$

These values have been chosen, as they do not seem to show the following problems:

1. If the ε value is too low, the interpolated values change rapidly from location to location (one-dimensional example in Figure 21), which is rather unrealistic in most cases.
2. If the ε value is too high, then the interpolation values do not represent the trends well anymore (one-dimensional example in Figure 20).
3. It is also important to find the right *smoothing factor*. If the value is too low – the interpolation values will have rapid changes and be prone to overestimation (Figure 23). Also, the highly location-specific nature of the data would cause problems with a low smoothing factor. Namely, if two very close locations have drastically different observations, that would cause large overestimations or underestimations. Whereas if the smoothing factor is too high, interesting trends get lost (Figure 22).

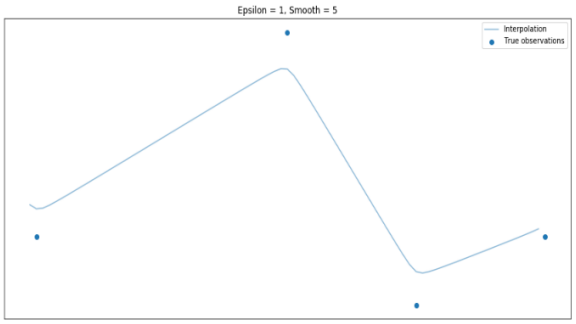


Figure 21: ε value is too low.

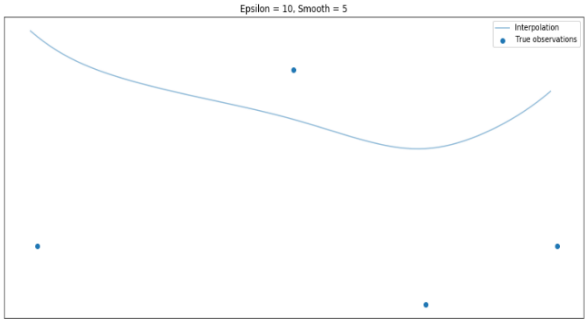


Figure 20: ε value too high.

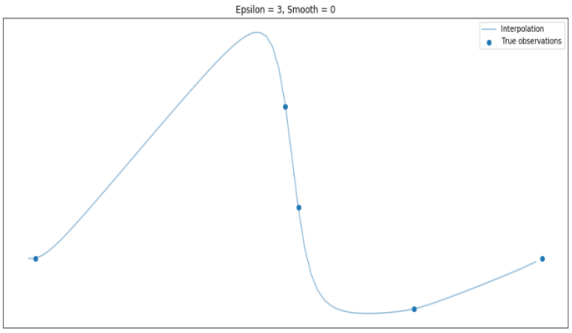


Figure 23: smoothing factor too low.

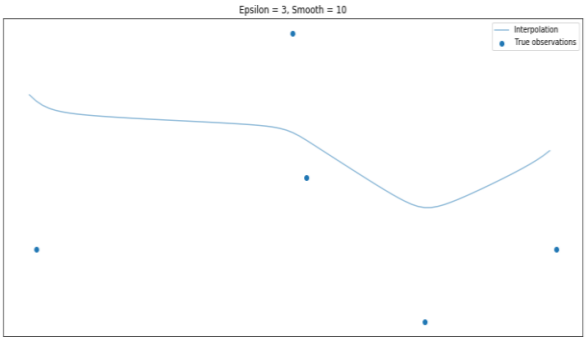


Figure 22: smoothing factor too high.

By using the chosen values, the final images, showing the spatially interpolated values of the predictions, were generated (example image of the pedestrian count prediction for 2019-05-04, 13:00-14:00 shown in Figure 24).

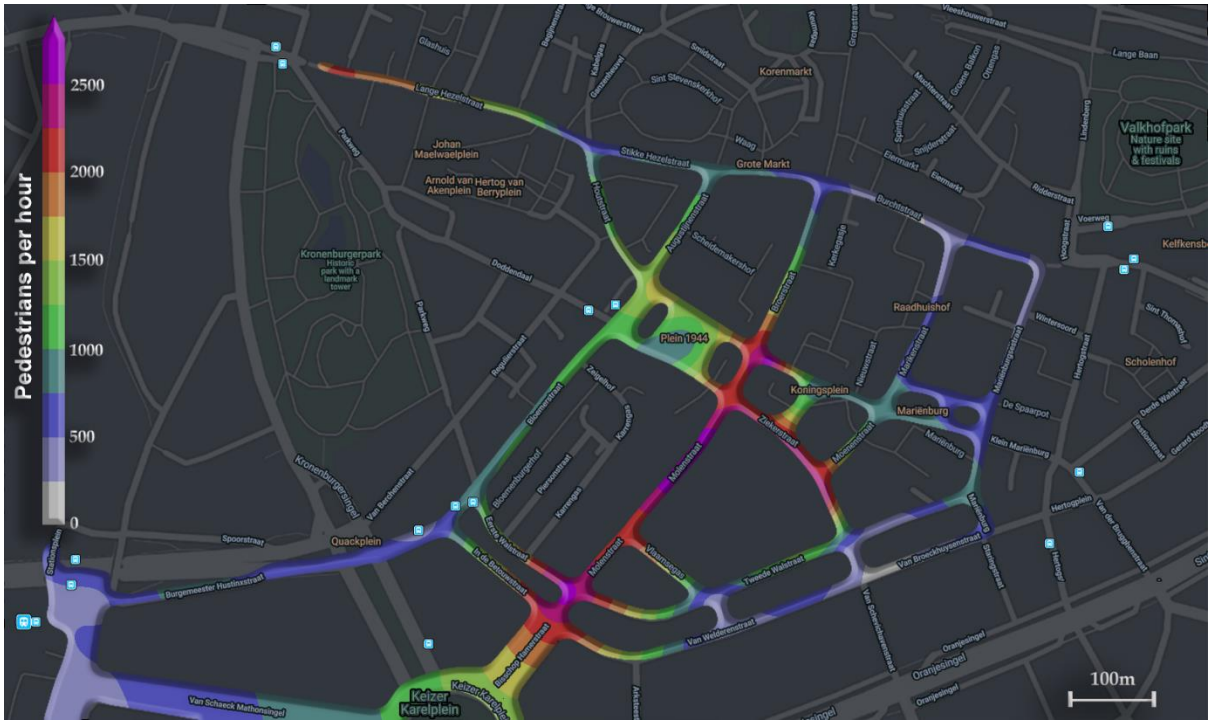


Figure 24: Spatially interpolated and overlaid prediction image.

Discussion

The pedestrian count prediction performance results are quite interesting (Figure 17 and Table 2). Firstly, the performance of the baseline (averaging) model is very impressive. This model has an average R^2 value of 0.722, which means that it explains 72.2% of the variance in the data by only taking 'day of the week' and 'hour' variables into account. Knowing that pedestrian count data is directly linked to the highly variable crowd dynamics in the city, that depend on a lot of different unobserved or even unobservable factors, an average MAE measure lower than 50 is a very good result. It means that these models are, on average, only 50 pedestrians off from the real count. This error is almost negligible in most of the possible applications for a predictive model of this type (discussed in Applications).

Meanwhile, other, complex models failed to perform better than this simple baseline model. At first glance it might seem surprising, but the simplicity of the problem at hand (simple input format, only 168 possible input combinations and rather basic input-output relations) explains the poorer complex model performance, as overcomplicating problems may introduce errors as a result of finding patterns where there are none as well as failing to converge on an optimum, because of the large number of tuned parameters. These results highlight the fact that machine learning is a very data-dependent field and explain the high academic, as well as

commercial interest in reducing the amount of data, necessary for model training. Most of the people and companies that want to integrate machine learning into their pipelines are hoping for quick and easy profit gain. However, they tend to underestimate the amount of training data necessary for getting any benefit from these models. So, it is necessary to explore methods that would reduce this amount and help machine learning gain more traction in commercial, as well as public sector applications. At the same time, the result draws attention towards the importance of in-depth analysis of the problem at hand before deciding on possible solutions, especially in data-driven fields, where 'machine learning' and 'AI' are often being used as buzz words and these models are often applied just for the sake of marketing. Of course, once the problem grows larger and more complex (e.g. introducing weather measurements as independent variables), the number of possible input combinations (possibly one of the most important factors to take into account) grows at an exponential rate. Quickly, the simple averaging model would become too difficult to manage and possibly perform worse, while complex models would gain an edge in performance, as well as maintainability. However, the specific details of this effect are not analysed in this work.

Regarding the additional model upgrades, in retrospect, the results were expectable. The fact that applying an average yearly trend to these predictions did not show a significant impact is most likely due to the uniqueness of these trends for each location. The crowdedness of some locations is highly dependent on the time of year (e.g. locations near water or around ice cream parlours gain a lot of popularity during summer), while others might not be affected by this factor at all. Meanwhile, the significant error decrease after extending the prediction with a confidence interval was to be expected as well, as the prediction range eliminates most of the errors that are due to the variability in the data throughout the day/week. One important thing to point out about the single value to range conversion is the "precision vs. error" trade-off. It is possible to tweak the standard deviation multiplier of this conversion (the default is one standard deviation). By increasing this value, one can improve the mean absolute error performance of the model, as the prediction range would get wider. However, this means that these predictions lose precision. Namely, one can increase the multiplier value by a lot and reach a MAE value close or equal to 0, but at that point, the predictions are close to "0-3000", which does not provide any useful (precise) insight into the actual trends of traffic in the city (example with a multiplier of 5 in Figure 25). So, it depends on the precision of predictions that a specific application requires and a mean absolute error that this application is able to deal with.

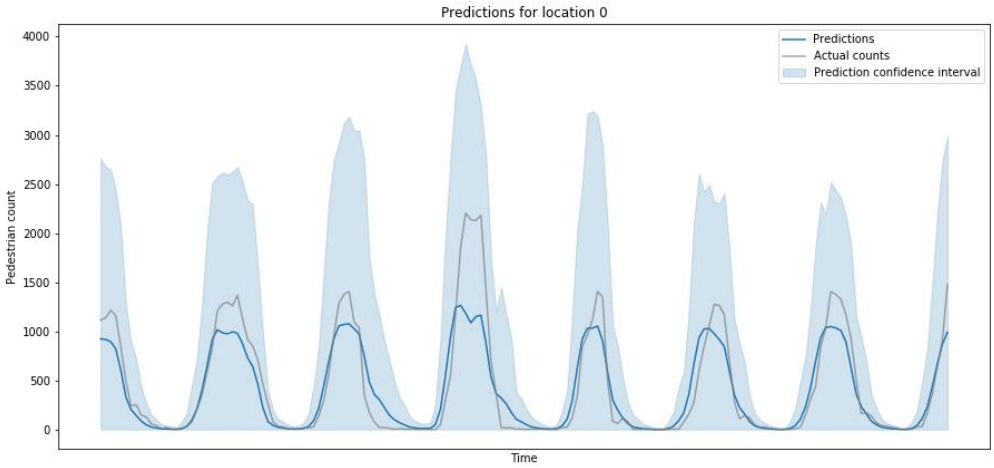


Figure 25: Predictions with 5 SD range.

With regards to the spatial interpolation model, as it heavily depends on a lot of factors that it contains, it does not provide a very rigid tool for unobserved location estimation. The model provides a very comprehensible colour-coded map, which visualises the general spatial trends in the city (e.g. pedestrian traffic hotspots), but does not generate any detailed estimations with acceptable accuracy. This is a result of the very simple problem representation (grid) that has been chosen. With this representation, the model does not contain any information about the street network, which means that the results are accurate only on a very high level. So, it is recommended to use this RBF interpolation model as a visualisation tool for values in the observed locations rather than as an estimation model for the unobserved ones. If a detailed estimation model is required, it can be achieved by incorporating information about the street network into the model, which can be done by making a custom kernel for the interpolation, or simply by switching from a grid spatial representation to a graph, where edges represent the streets (or sections of them).

Applications

The final models can be used for a wide range of applications. Only a few examples will be given and discussed.

The pedestrian count prediction model can technically be used for any purpose that requires or makes use of pedestrian counts and their forecasts. A basic use case would be simply predicting the number of pedestrians in specific locations in order for businesses to estimate the number of customers, which would provide useful information that helps with scheduling and production planning. A more complex application for the prediction model is to use it as a

baseline when analysing the effect of any kind of event on the pedestrian traffic. For example, it is possible to determine the impact of government restrictions, which started in March 2020 and were meant to combat the spread of the novel coronavirus (COVID-19) (Figure 26).

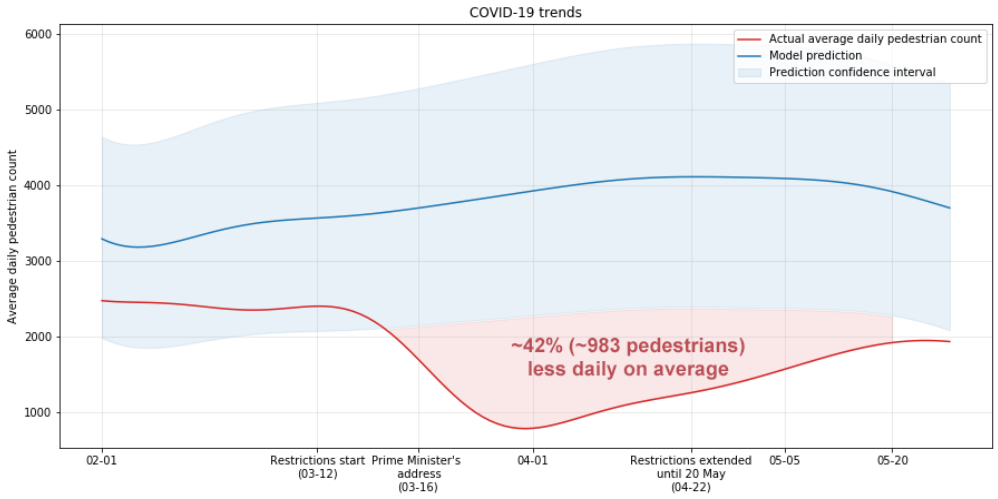


Figure 26: Coronavirus (COVID-19) restriction effects.

In the result of this analysis, we can see that the pedestrian count was already on the low end at the start of February (possibly because of the news about the spread of the virus in certain regions of China). We can also see that once the official government restrictions started, the pedestrian traffic decreased rapidly (red line after 03-12 in Figure 26), dropped below the predicted minimum and stayed around 42% (on average) below it throughout the restriction period. These types of analyses allow the local government to evaluate the impact of certain events as well as the effectiveness of introducing or lifting certain restrictions.

Using this model as a baseline is beneficial in terms of generalisation. One could use the data from the previous year for this type of analysis, however, that is not always possible as that long ago, the location of interest might not have been monitored yet. Even if the previous year data are available, real observations are quite noisy and affected by a lot of factors, which are not taken into account for the analyses, such as weather or local events (e.g. comparing data from 2019 and 2020 might show a significant difference purely because the weather was better/worse in 2019 than in 2020). By using a model that gives an average estimation, one can reduce the impact of these unobserved variables on their analyses.

As mentioned before, the interpolation model can be used to estimate any kind of numeric variable. For example, to find out which areas were affected by COVID-19 the most, it is

possible to interpolate the observed impact evaluations (the relative difference between the average hourly count before and after the government restrictions) (Figure 27).

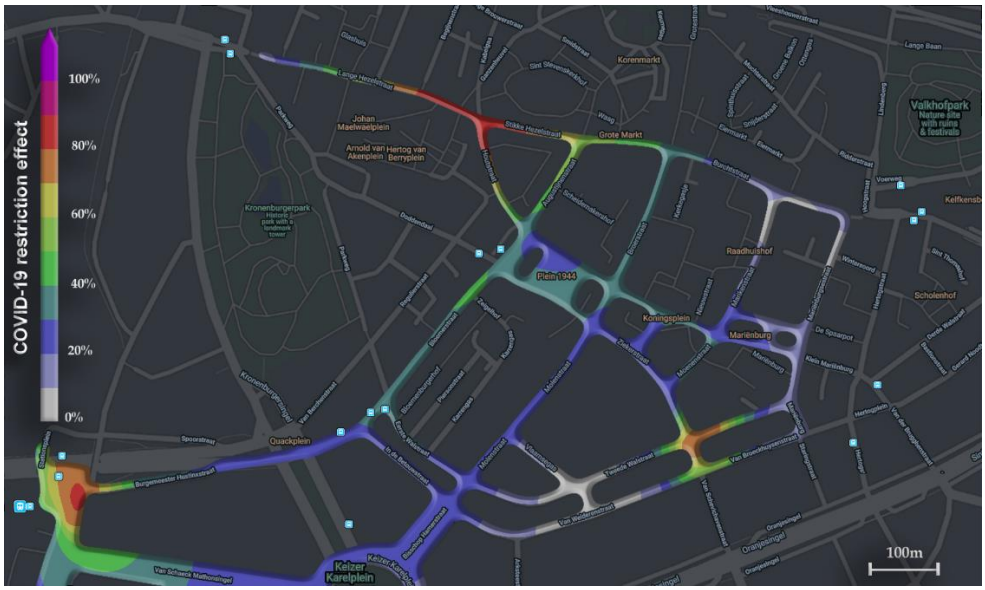


Figure 27: COVID-19 restriction effects mapped.

This result shows which areas had the most drastic decrease in pedestrian traffic in the city (percentage on the scale show the percentage decrease during the restrictions). While this kind of visualisation provides useful information for the local government about the areas that have been affected the most or the locations that require additional effort to reduce the pedestrian traffic, it has to be taken with a grain of salt. The relative differences are exactly what the name suggests – relative. Even if the locations that were very crowded before the restrictions, show a large relative decrease (e.g. the north-north-west side of the map labelled red in Figure 27) , it does not mean that the resulting pedestrian count is low enough for social distancing. While areas that were already suitable for social distancing, might not show any relative decrease at all, but actually will not require any additional effort to reduce the number of pedestrians any more (e.g. grey area in the south side of the map in Figure 27) . So, it is important to treat this type of result as a relative impact evaluation rather than an absolute location quality metric.

Conclusion

The aim of this thesis was to answer the following questions:

1. ***Given a low amount of available data, what is the best approach for modelling pedestrian traffic trends? What is the performance of this model and what are its possible applications?***

2. ***How to interpolate spatially sparse pedestrian count (prediction or observed) data to get estimations for unobserved locations and visualise spatial trends of city traffic?***

After multi-layer perceptron, gaussian process regressor, support vector regressor models have been build, trained, tested and compared to a baseline averaging model, it was found that it is indeed possible to model pedestrian traffic with high accuracy (MAE lower than 50 – the predictions are less than 50 pedestrians off on average), even without having a vast amount of available data. However, the prediction model results highlight the importance of problem analysis, as the simple baseline model showed better results than the more complex machine learning models. This predictive model can be used for a wide range of applications that make use of pedestrian count data, such as customer number prediction for business planning or analysing the effect of certain events in the city.

In this work, radial basis function spatial (2D) interpolation has been tested. It provides a good visualisation of very high-level spatial trends in the city, but does not have the necessary street network knowledge to show detailed unobserved location estimates with acceptable precision. This model can be used to visualise the general spatial distribution of any type of measurement in the city. However, in order to accurately estimate values that are influenced by the city environment (e.g. pedestrian traffic is limited by the street network, obstacles, etc.), more complex models, which incorporate these variables into account (e.g. a graph representation of the street network), are recommended.

Project repository

The final version of the complete pedestrian count prediction and interpolation system, built for this thesis is available on GitHub: https://github.com/dgirzadas/Pulse-of-the-City [23]. It is implemented in Python 3 and published under the MIT license.

References

[1] O. J. Postma and M. Brokke, “Personalisation in practice: The proven effects of personalisation,” *J. Database Mark. Cust. Strateg. Manag.*, vol. 9, no. 2, pp. 137–142, 2002.

[2] J. E. Phelps, G. D’Souza, and G. J. Nowak, “Antecedents and consequences of consumer privacy concerns: An empirical investigation,” *J. Interact. Mark.*, vol. 15, no. 4, pp. 2–17, 2001.

[3] “Autoriteit Persoonsgegevens,” *Dutch DPA investigates WiFi tracking in and around*

shops, 2015. [Online]. Available: https://autoriteitpersoonsgegevens.nl/en/news/dutch-dpa-investigates-wifi-tracking-and-around-shops. [Accessed: 08-May-2020].

[4] Numina, "Numina | Know Your Streets." [Online]. Available: https://numina.co/. [Accessed: 05-Mar-2020].

[5] Numina, "Numina.co," *Our Privacy Philosophy*, 2019. [Online]. Available: https://numina.co/our-privacy-principles/. [Accessed: 08-May-2020].

[6] W. Zheng, D.-H. Lee, and Q. Shi, "Short-term freeway traffic flow prediction: Bayesian combined neural network approach," *J. Transp. Eng.*, vol. 132, no. 2, pp. 114–121, 2006.

[7] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2016, pp. 324–328.

[8] M. Castro-Neto, Y.-S. Jeong, M.-K. Jeong, and L. D. Han, "Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions," *Expert Syst. Appl.*, vol. 36, no. 3, Part 2, pp. 6164–6173, 2009.

[9] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 4, pp. 276–281, 2004.

[10] J. Kwon, B. Coifman, and P. Bickel, "Day-to-day travel-time trends and travel-time prediction from loop-detector data," *Transp. Res. Rec.*, vol. 1717, no. 1, pp. 120–129, 2000.

[11] M. S. A. Siddiquee and S. Hoque, "Predicting the daily traffic volume from hourly traffic data using artificial neural network," *Neural Netw. World*, vol. 27, no. 3, p. 283, 2017.

[12] S. Bhattacharya, S. Phithakkitnukoon, P. Nurmi, A. Klami, M. Veloso, and C. Bento, "Gaussian process-based predictive modeling for bus ridership," in *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, 2013, pp. 1189–1198.

[13] L. Mitas and H. Mitasova, "Spatial interpolation," *Geogr. Inf. Syst. Princ. Tech. Manag. Appl.*, vol. 1, no. 2, 1999.

[14] S. Zandi, A. Ghobakhlou, and P. Sallis, "Evaluation of spatial interpolation techniques for mapping soil pH," 2011.

[15] "Binnenstadsmonitor Enschede." [Online]. Available: https://www.binnenstadsmonitorenschede.nl/druktebeeld-straten. [Accessed: 04-Jun-2020].

[16] T. Liebig, Z. Xu, M. May, and S. Wrobel, "Pedestrian quantity estimation with trajectory patterns," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2012, pp. 629–643.

[17] Numina, "Numina | API." [Online]. Available: https://numina.co/api/.

[18] J. Ding, "Defining Accuracy for Street-Level Mobility Data," *Defining Accuracy for Street-Level Mobility Data*, 2020. [Online]. Available: https://medium.com/numina/defining-accuracy-for-street-level-mobility-data-895c1da45986. [Accessed: 23-Apr-2020].

[19] "Gaussian Processes regression: basic introductory example." [Online]. Available: https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpr_noisy_targets.html#sphx-glr-auto-examples-gaussian-process-plot-gpr-noisy-targets-py. [Accessed: 19-May-

2020].

[20] V. Cherkassky and Y. Ma, “Practical selection of SVM parameters and noise estimation for SVM regression,” *Neural networks*, vol. 17, no. 1, pp. 113–126, 2004.

[21] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[22] P. Virtanen *et al.*, “SciPy 1.0: fundamental algorithms for scientific computing in Python,” *Nat. Methods*, vol. 17, no. 3, pp. 261–272, 2020.

[23] D. Giržadas, “Pulse of the City,” 2020. [Online]. Available: https://github.com/dgirzadas/Pulse-of-the-City. [Accessed: 01-Jul-2020].

Appendix

Appendix A: Data

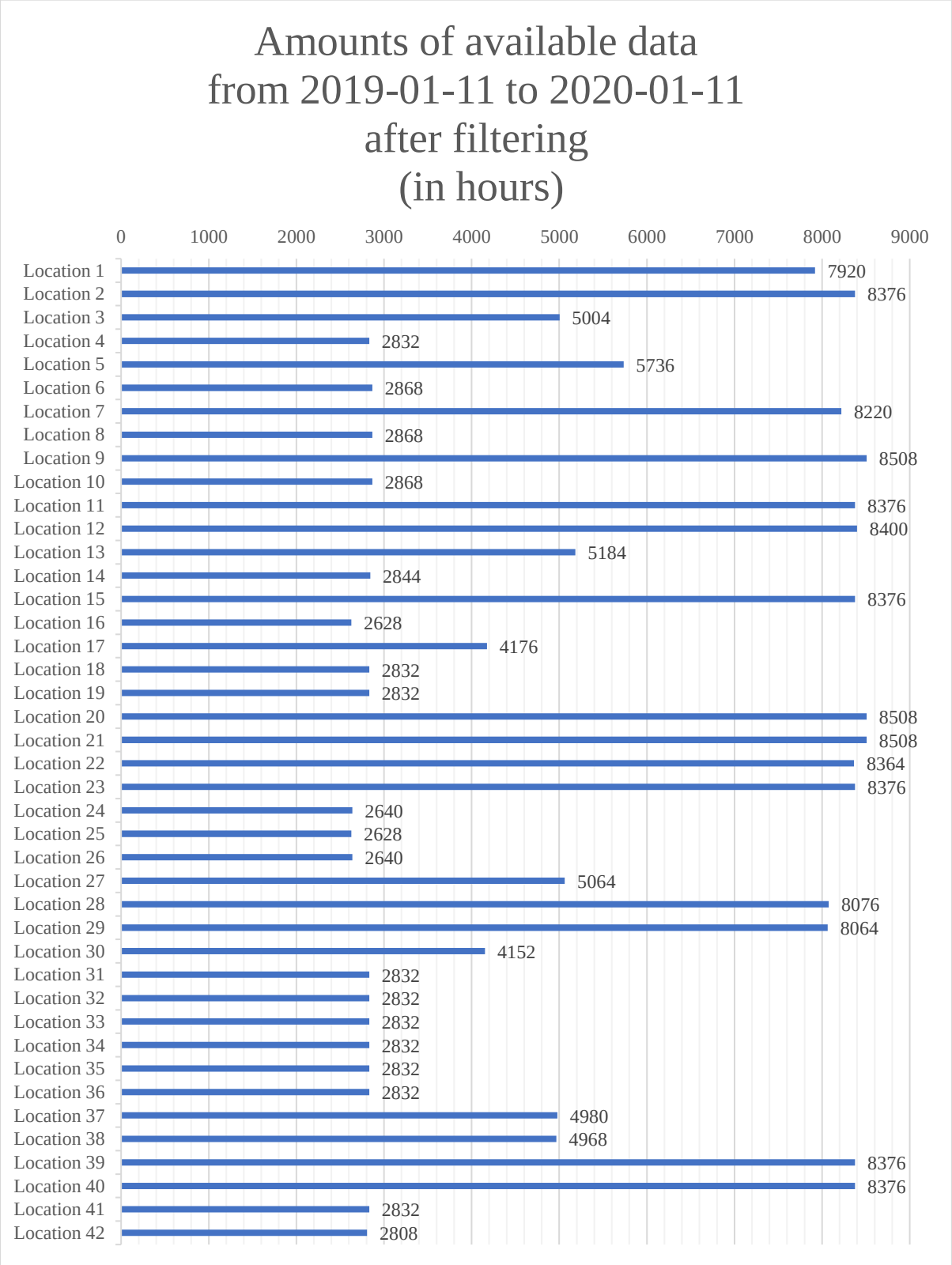


Figure 28: Amounts of available data after filtering

Appendix B: Model performance

Table 2:Model performance

Model	Average R^2 value	Average MAE value
MLP1	0.561	75.975
MLP2	0.588	69.309
MLP3	0.624	59.993
Gaussian Process Regression	0.672	50.027
Support Vector Regression	0.669	53.149
Baseline model	**0.722**	**45.854**

Table 3: Additional model upgrades

Model type	Average R^2 increase (%)	Average MAE decrease (%)
With daily scaling applied	0.55401662	1.674881145
Daily scaling + Confidence interval	0.55401662	45.43551271

Appendix C: Map Overlay

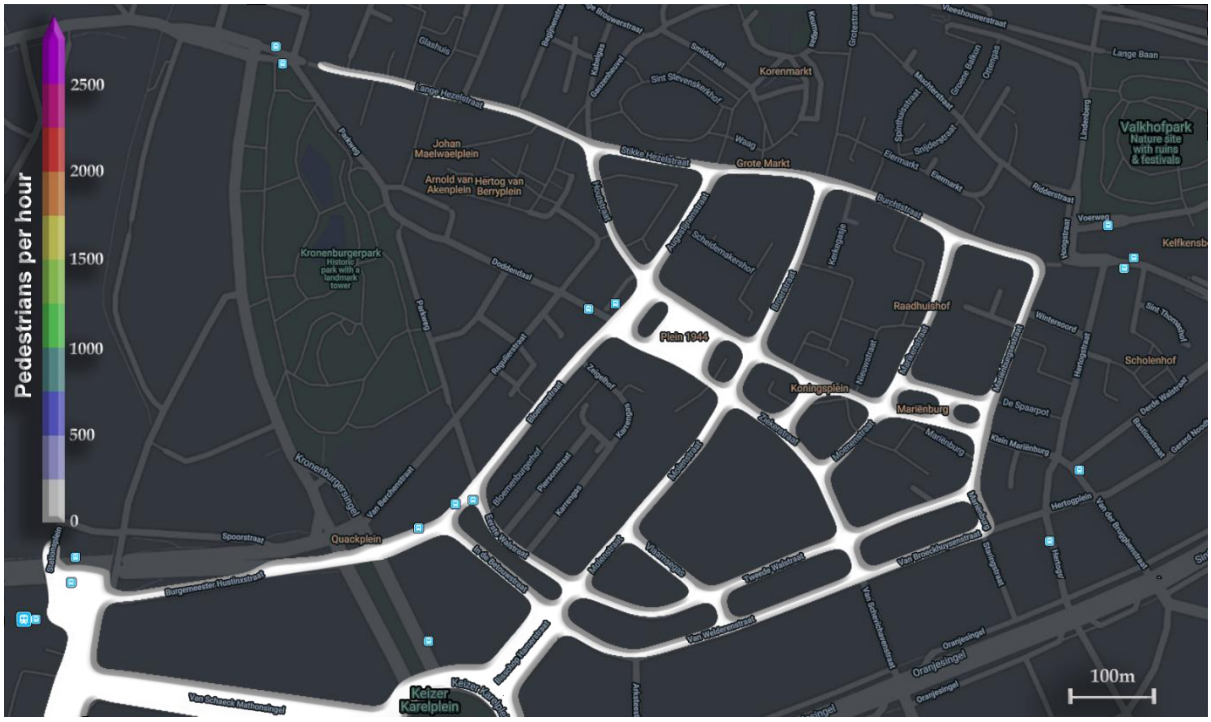


Figure 29: Map overlay. Source: Adapted from mapstyle.withgoogle.com