Prediction Models to estimate crowdedness within Amsterdam

submitted in partial fulfilment for the degree of master of science

Don de Lange
11156414

master information studies
data science
faculty of science
university of amsterdam

June 21, 2019

	Internal Supervisor	External Supervisor
Title, Name	dhr. Noud de Kroon	Tom Knijff
Email	a.a.w.m.dekroon@uva.nl	t.knijff@amsterdam.nl
Affiliation	UvA	Gemeente Amsterdam

UvA

Gemeente Amsterdam

Prediction Models to estimate crowdedness within Amsterdam

Don de Lange
don.delange@student.uva.nl
11156414
Universiteit van Amsterdam
Amsterdam

ABSTRACT

Amsterdam is a crowded city, which increases the pressure on public services, public events and public spaces. Sensors were placed in the city to measure the crowdedness of pedestrians within small predetermined areas. This research aims to train prediction models that predict the crowdedness at and between predetermined locations within Amsterdam, at an hourly rate for each single day within a predetermined period.

In order to train the prediction models, historical data from the following sources was used; (i) public transport data from predetermined stations (ii) crowdedness counts from sensors located at the predetermined locations (iii) dates of events within Amsterdam.

The models used in this research were *Random Forest* and *XGBoost*, for both regression and classification. The regression models predicted the numerical pedestrian counts and the classification models predicted the crowdedness level of the pedestrians, based on the crowdedness distribution in the historical dataset.

With the predictions made at the sensor locations, the *XGBoost* models outperformed the *Random Forest* models.

As a method of testing the predictions between the known sensor locations, the models generalized the data of the known sensor locations to the unknown location. To evaluate the model performance, the data of a single sensor was removed during the training of the models and the crowdedness of that sensor was predicted during the evaluation phase. Which was repeated for each given sensor and the average performance results were evaluated.

With the generalizations to unknown sensor locations, the regression *XGBoost* models also outperformed the regression *Random Forest* models. But, the performance of the classification *XGBoost* and classification *Random Forest* was equal.

KEYWORDS

Crowdedness, Prediction, Regression, Classification, Random Forrest, XGBoost, Python

1 INTRODUCTION

Amsterdam is a crowded city due to an increasing number of tourists that visit the city each year and a growing number of residents. This leads to the following problems; (i) the public transport and Schiphol airport cannot handle the growing number of passengers effectively (ii) there is less parking space available (iii) there are increasingly more guests at public events, spaces, and places. In order to handle these problems, the municipality of Amsterdam aims to put gathered data to use.

The municipality measures the crowdedness with count cameras and Wi-Fi sensors. The count cameras count the number of pedestrians that walk through the predefined area of the camera, without saving personal information of the counted pedestrians. The Wi-Fi sensors count the number of devices of pedestrians that connect with the Wi-Fi sensors, without saving personal data from the connected devices. By only saving the necessary data, the sensors ensure that the privacy is upheld.

Currently, the pedestrian counts of the sensors are only used to identify historical trends of crowdedness, for use within the municipality. However, using the data to predict the crowdedness could be useful for a variety of reasons, such as: (i) the police could send additional patrols to highly crowded areas (ii) additional public transports could be planned for high rush hours.

The aim of this research is to predict the crowdedness at given sensor locations (count cameras and Wi-Fi sensors) with two different methods; first, the crowdedness is predicted at known sensor locations for unknown dates. And second, predictions are made at an unknown sensor location for known dates, by generalizing the known sensor location data to the unknown sensor location. The predictions are generated with *Random Forest* models and *XGBoost* models, at an hourly rate over a given period, in the following two formats; (i) regression, where the prediction is numerical number of pedestrians (ii) classification, where the prediction is a discrete level of crowdedness. The levels of crowdedness are based on the distribution of crowdedness counts present in the full dataset, by ordering and dividing these counts into quartiles. Each quartile is one level of crowdedness, leading to a total of 4 levels of crowdedness. All these steps can be formulated in the following research question:

How can a prediction of crowdedness within the city of Amsterdam be given, based on input from city-wide available data sources?

The *city-wide available data sources* used to make the predictions are the following; (i) public transport data from predefined stations (ii) event dates in near proximity to the prediction locations (iii) crowdedness sensor data, which is also used as ground truth.

The main challenges this research needs to overcome are the following; (i) combine and transform all the given data in an effective manner, which can be used by the models as input data (ii) optimize the model performance, by transforming the given data effectively and tuning the hyperparameters of the models (iii) construct an effective pipeline that can construct the needed features for a prediction at any given date and given location, with the use of the previously optimized models.

The remainder of the thesis is composed of six chapters. The first part discusses the scientific background of this research (section 2). The second part is concerned with the methodology of this research (section 3), focusing on Data (3.1), Models (3.4), Data exploration (3.2), Data transformation (3.3), Model construction (3.5), Generalization (3.6), and Prediction pipeline (3.7). The third part describes

the implementation of the methodology (section 4). The fourth part displays the model results of this research (section 5). The fifth part discusses the results and its implications (section 6). Finally, the sixth part concludes this research (section 7).

2 RELATED WORK

2.1 Crowdedness Prediction

This section describes previously constructed and evaluated prediction models that can predict the number of pedestrians within a predefined area.

Zhai et al. (2018) constructed a short-term passenger prediction model for buses. The model used data gathered from sensors in the bus to predict the number of passengers in the bus. The model returns the level of crowdedness, based on the difference between the maximum load and the current load of passengers. Ideally, passengers waiting for the bus could judge whether the wait would be worth it. Zhai et al. (2018) used real-time data from motion based sensors in the bus to predict the level of crowdedness. This research seeks to build upon their research by using previously gathered data from external sources to predict the crowdedness at the given sensors.

Wang et al. (2017) focused on developing a model and system for predicting foot traffic, by utilizing historical counts of pedestrians. These counts were gathered with sensors placed at multiple locations within the city. The aim of the model, was to predict the foot traffic in the city, given continuous streamed data gathered from the sensors. This research aims to construct a similar model, however, the model of this research will only make predictions solely based on historical data, not based on streaming data. Furthermore, this research uses public transport data next to the sensor data to predict the crowdedness. In addition, this research uses *Random Forrest* and *XGBoost* with numerical data to make predictions, while Wang et al. (2017) used a neural network with image based data to make predictions.

2.2 Models

According to Ho (1995), traditional derivation methods of decision do not generalize well to unseen data, due to lack of complexity. There are some methods, such as pruning, that can add complexity to the tree, but at the cost of accuracy on the training data. And adding a too high complexity means overfitting on the training data, thereby reducing performance on the test data. Ho (1995) proposed a method to construct tree-based classifiers that can arbitrarily expand their capacity, for an increase in both training and unseen data accuracy. This method was called *Random Forest* and is explained in more detail in section 3.4.3.

According to Chen et al. (2016), data-driven solutions are driven by effective models that capture the data complexities and scalable learning systems that are applicable on large datasets from a single machine. Chen et al. (2016) stated that *Gradient Boosting* models are already very effective, but lack scalability on large datasets. *XGBoost* offers a solution to this, by remaining as effective as *Gradient Boosting*, but offering scalability on large datasets with the use of minimal resources. This method is explained in more detail in section 3.4.4.

2.3 Regression Models

Random Forest (RF) is often used for regression problems. Zahedi et al. (2018) implemented RF due to its simplicity in tweaking parameters and good performance with nonlinear features. RF was used for predicting erosion in a comprehensive manner, based on historical data of erosion. For optimization, there are two main parameters that need to be tweaked. The first parameter is *Ntree*, which represents the number of trees to grow. The number should not be too small in order to ensure that every input experiment gets predicted at least a few times and not too big to prevent overfitting on the training data. The second parameter is *Mtry*, which represents the number of variables that are randomly sampled as candidates in each split. The paper used *tuneRF* to find the optimal value for *Mtry*. *TuneRF* is a function that starts with a default value (*number of variables* / 3) and searches for the optimal value with out-of-bag estimation. *TuneRF* was not used as an hyperparameter method. As this research seeks to apply a singular method of hyperparameter tuning for all the models and *TuneRF* was not applicable to the other models.

This research implemented RF due to its simplicity in implementing the model and optimizing its parameters, and its high performance with predictions based on historical data.

XGBoost (XGB) is also often used for regression problems, usually leading to a higher performance than RF. Nieto et al. (2018) used a *gradient boosted regression tree model* (GBRT) to predict cyanotoxin contents which is poisonous and poses a health threat in waters that are used for drinking or recreational purposes. This research used GBRT as it can obtain good predictions in highly nonlinear problems, similar to the problem within this research. GBRT offered good results due to high performance, robustness and simplicity. The XGB package was used to further increase the performance of the model.

Generally, XGB models outperform the previously discussed RF models (Chen and Guestrin, 2016).

2.4 Classification Models

RF is also often used for classification problems. Heydenrijk-Ottens et al. (2018) used RF to predict the passenger load of the subway at each station, based on the public transport card (OV kaart) that is used by a large portion of the passengers. Out of all the models used to make the prediction, RF offered the best performance.

This research also seeks to use RF to make classifications based on historical data.

Furthermore, XGB is often used for classification problems as well. Ding et al. (2016) applied a GBDT to short-term subway ridership prediction in order to capture the associations with the independent variables, which can be optimized with the use of various regularization methods, learning parameters and added tree complexity. In addition, the GBDT can give the features that led to the given prediction. This adds some explainability to the model.

This research uses a XGB model, due to having the same benefits as GBDT but with higher performance (Chen and Guestrin, 2016).

3 METHODOLOGY

In this section the used data and models are described, followed by a description of the prediction and generalization pipeline. All the processes and operations described here, were done in Python.

3.1 Data

Four datasets are used in this research, which are listed below in table 1.

3.1.1 General. *General* are features present in all the datasets: *Date* and *Hour* represent the given date and hour respectively, of the given measurements in the data. The *Date* is represented in the *YYYY-MM-DD* format. The *Hour* is represented in numbers from *0* till *2300*, where *0* refers to the slot *00:00 - 00:59* and *2300* refers to the slot *23:00 - 23:59*.

Weekday represents the number of the day in the week of the given date. With numbers ranging from *0* till *6*, where *0* is *Monday* and *6* is *Sunday*.

Is weekend is *1* when the weekday is in the weekend and *0* otherwise.

3.1.2 GVB. *GVB* contains public transport data. The stations in this dataset were selected because of their close proximity to the Red Light District. More specifically, the dataset contains the number of passenger that made a trip to or from each given station, on an hourly basis per day.

The features per station of the *GVB* dataset are as follows: *Station* represents the unique name of the given station.

Passenger Count represents the number of passengers that arrive at and depart from the given station, respectively. These passenger counts are made on an hourly basis, per day.

Longitude represents the longitude of the given station.

Latitude represents the latitude of the given station.

3.1.3 CMSA. *CMSA* contains the crowdedness count of the sensors placed in the Red Light District in Amsterdam. Each sensor ID in the dataset contains the counts of all the sensors in the given street zone. Each sensor can be a (i) count camera, that counts the number of pedestrians in a predetermined area on an hourly basis (ii) Wi-Fi sensor, counts the number of devices that connect with the sensor on an hourly basis. The sensors were combined under a single sensor ID to increase the prediction accuracy. So the crowdedness counts of each sensor ID are the summation of all the pedestrians and devices in the given street zone and are used as ground truths in this research. The street zone sensor IDs will be referred to as sensors in the rest of this thesis.

The features per sensor of the *CMSA* dataset, are as follows: *Sensor* refers to the unique sensor ID of a street zone.

Crowdedness Count refers to the number of pedestrians and devices in a single street zone that were counted in the sensors street zone, on an hourly basis, per day.

Longitude refers to the Longitude of a centralized point in the sensor zone.

Latitude refers to the Latitude of a centralized point in the sensor zone.

3.1.4 Amsterdam Events. The *Amsterdam Events* dataset contains the dates and locations of events within Amsterdam. This indicates which days are special, so the model does not generalize the outliers of the crowdedness counts in the *CMSA* dataset. Only events that are in proximity of the Red Light District are used in this research. The area of usable events was chosen arbitrarily.

The features per date of the *Amsterdam Events* dataset, are as follows: *Is event* is *1* if there is an event on the given data and *0* otherwise.

Longitude refers to the Longitude of the given event.

Latitude refers to the Latitude of the given event.

Table 1: Overview Data Datasets

Data Source	Features
General	Date
	Hour
	Weekday
	Is weekend
GVB	Station
	Passenger Count
	Longitude
	Latitude
CMSA	Sensor
	Crowdedness Count
	Longitude
	Latitude
Amsterdam Event Data	Is event
	Longitude
	Latitude

3.2 Data Exploration

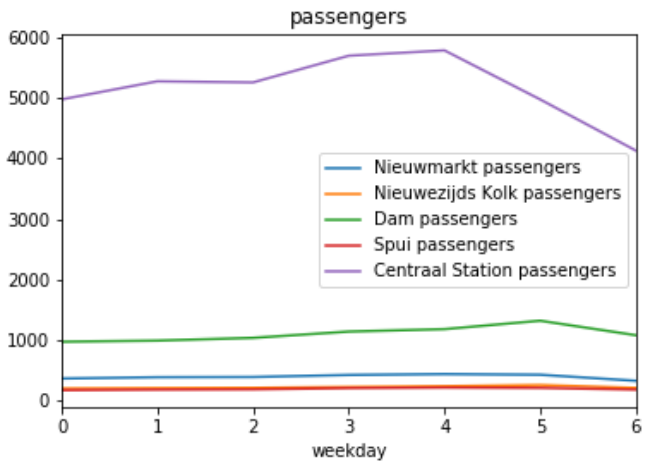
In this section the data used in this research are explored and visualized.

First, the passenger counts (see figure 1) visualize the mean number of passengers of each station per day. The figure shows that most passengers arrive at *Centraal Station* and *Dam Station*. In addition, this figure shows that *Dam*, *Nieuwezijds*, *Niewmarkt* and *Spui* have little variation in the passenger counts. While *Centraal Station* shows more variation in the counts from day to day. This may be due to the higher number of subway and tram lines that are taking into account with *Centraal Station*. So special events in Amsterdam that attract a higher number of tourists, will have a higher impact on *Centraal Station* than on the other stations.

Second, the *CMSA* contained a number of missing values in the period *March 2018* till *June 2018*. This was due to failure of the sensor to properly register the counts made for some hours of the day. Due to time constraints, this research made the assumption that all the crowdedness counts in the dataset were true.

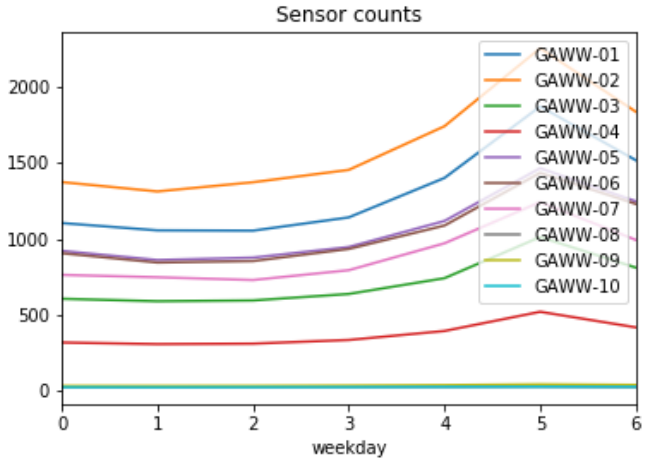
Third, the sensor counts (see figure 2) visualize mean crowdedness counts of each sensor per day. The figure shows that sensors *GAWW-02* and *GAWW-01* have the highest counts, while *GAWW-04* the lowest. The reasons for the relatively high differences in counts, is unclear. The counts reached the highest point on Saturday and the lowest point on Tuesday. The counts of sensors *GAWW-08*,

Figure 1: Station Passenger Counts



GAWW-09, and *GAWW-10* are constant for each date. So, these sensors were removed from this research.

Figure 2: Sensor Counts

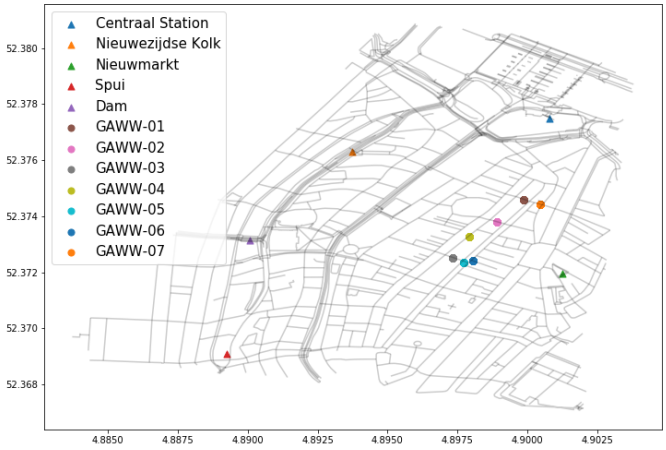


And fourth, the locations of the sensors and stations are visualized in a map (see figure 3). The map shows that the sensors are located in close proximity to each other within the Red Light District. While the stations are located further away from each other and the sensors.

3.3 Data Transformation

For the full dataset, the *CMSA*, *GVB*, and *Events* datasets were combined (section 3.1). The following data transformations were implemented on the full dataset to increase the performance of the

Figure 3: Map



models. The data transformations were tested on the training set. The results of these tests are shown in the Appendix (section 7)

The first transformation was making time circular[1]. Otherwise, the model would not be aware that *23:00* on the previous date and *01:00* on the current date are only 2 hours apart, not 22 hours and one day. Transforming the time to a circular format improved the model performance significantly. For this transformation the following two formulas are implemented (see formula 1 and 2):

$$x_{\cos} = \cos\left(\frac{2 \times \pi \times x}{\max(x)}\right) \tag{1}$$

$$x_{\sin} = \sin\left(\frac{2 \times \pi \times x}{\max(x)}\right) \tag{2}$$

In the given formulas, x stands for the current time in the specific time type that needs to be made circular. So time consisted of a cosine (see formula 1) and sinus (see formula 2) number.

The second transformation was encoding the longitude and latitude of the sensors and stations with standardization. As the coordinates were not given in a spatial dimension, the amount of distance between the sensors and stations was not clear. This made it more difficult for the model to correctly assess the impact of those small differences. Scaling the longitude/latitude increased the performance by a small margin.

Two different scalars were considered for this research, based on their simplicity in implementation. The first scalar was the *MinMax Scalar*[2], which ranges the values between 0 and 1. The second scalar was the *Standard Scalar*[3], which assumes that the data are normally distributed around 0, with a standard deviation of 1. All the data was scaled accordingly. However, there was no difference in performance between the two scalars. As the differences between the longitude and latitude values were too small for a scalar to outperform the other. But the standard scalar is more adapt in handling outliers, which is needed for the generalization to an

[1]Blog: Encoding Cyclical Features - 24hour Time
[2]SKlearn Model MinMax Scalar
[3]SKlearn Model Standard Scalar

unseen data point (see section 3.6). So the *Standard Scalar* was selected.

The formula of the *Standard Scalar* is as follows (see formula 3):

$$z = \frac{x - \mu}{\sigma} \tag{3}$$

In the given formula x is the current longitude/latitude, μ is the mean of the current longitude/latitude, and σ is the standard deviation from the mean.

The third transformation was calculating the distance between a given sensor and all the stations. The problem was that solely encoding the longitude/latitude only showed that those features contained different values, but failed to define the actual distance between the points on a longitude by latitude scale. To further increase the performance of the models, the distance between the data points was calculated. Several options were tested. First, simply the difference between the points was used. This was still too small for any significant changes in the model performances. Second, the distance was calculated with the use of a pairwise kernel. As pairwise kernels transport the data to a higher dimension and calculate the actual distance between the co-ordinates of the points. There were three types of pairwise kernels tested, due to their simple implementation. The first type was a *RBF Kernel*[4]. This computed the squared euclidean distance between the two co-ordinates and showed a small increase in performance. The second type was a *Polynomial Kernel*[5], which presented the similarity between two co-ordinates. This kernel significantly decreased the performance of the regression models and did not change the performance of the classification models. And the third type was a *Linear Kernel*[6], which computed the distance between the co-ordinates in the higher dimension. This was the simplest kernel en had no influence on the performance of the models. So calculating the distance between the scaled co-ordinates with an *RBF Kernel* showed the highest increase in performance. The *RBF Kernel* is defined by the following formula (see formula 4):

$$K(x, y) = \exp(-\gamma ||\mathbf{x} - \mathbf{y}||^2) \tag{4}$$

The given formula computes the Gaussian RBF kernel between the scaled latitude and longitude of a given sensor and all the stations. The different variables are defined as follows; (i) x is the scaled latitude (ii) y is the scaled longitude (iii) $gamma = 1$ (iv) $exp = 2.7182\ldots$. So, the stations with the lowest distance to the sensors had the highest influence on the predictions. The calculated weights are visualized in the Appendix (Appendix, figure 6).

And fourth, a possible transformation on passenger counts was tested, but did not yield any significant improvements. So the passenger data remained unchanged. The following three methods were tested. The first method was removing the passenger counts from the dataset, but this only decreased the performance of the models. The second method was scaling the passenger counts with the *Standard Scalar*, but this also decreased the performance of the models. And the third method was multiplying the weight of the previous transformation with the passenger counts, but this showed no significant increase in the performance.

[4]SKlearn Model RBF Kernel
[5]SKlearn Model Polynomial Kernel
[6]SKlearn Model Linear Kernel

3.4 Models

The *Random Forest* (RF) models were implemented from SKlearn[7] and the *XGBoost* (XGB) models were implemented from the XGBoost Docs[8], with an API from SKlearn. The benefit of the SKlearn API is an increase in the ease hyperparameter tuning.

Regression models predict the real valued crowdedness number. For the evaluation, the true numerical crowdedness count values were used.

Classification models predict one of four classes. The ranges of the classes were based on the quartiles of the all the numerical crowdedness counts in the full dataset. The first class *Level 1* has all the counts below the 25% qaurtile. The second class *Level 2* has all the counts between the 25% and 50% quartile. The third class *Level 3* has all the counts that are between the 50% and 75% quartile. And the fourth class *Level 4* has all the counts that are above the 75% quartile. For the evaluation, the classes of the true numerical crowdedness count values were used.

3.4.1 Linear Regression. *Linear Regression*[9] is a model that assumes that the output is a linear function of the input features (see formula 5). In order to find the optimal set of parameters for the model, the error between the predicted output and the true output is minimized. The *Linear Regression* model functioned as a baseline due to its simplicity and high performance.

$$\widehat{y} = \beta_0 + \beta_1 x_i 1 + \ldots + \beta_p x_i p \tag{5}$$

In the given formula y is the predicted crowdedness, β_0 is the intercept term, and $\beta_p x_i p$ is each of the input features used to predict y.

3.4.2 Majority Classification. As a baseline for the classification models, all the instances were classified as the majority class[10]. Majority classification is a widely used baseline for classification, as it returns a high performance without having learned the data complexities.

3.4.3 Random Forest. *Random Forest* (RF) is an ensemble machine-learning algorithm, which builds a group of weak learners of decision trees to form a strong learner, thus improving the capacity of the model. Each individual decision tree in the forest works with a subset of all the given features. Thereby reducing the complexities that would occur if all the features were combined into one tree and thus improving the generalizability of the model. As a prediction, the average prediction over all forests is taken. RF was implemented due to the following; (i) good performance with multiple features (ii) ability to handle non-linear data (iii) robustness to noise (iv) tuning simplicity (v) opportunity for efficient parallel processing (Zhu et al., 2018). The performance of RF can be easily optimized due to its simplicity in optimizing parameters (Zahedi et al., 2018).

Implementing RF as a regression[11] model results in real valued numbers. At each node, the regression model fits the target value to each independent variable. At each independent variable, the data are split into several data points. For each split point, a value

[7]scikit-learn: Machine Learning Modules
[8]Documentation website of XGBoost model
[9]SKlearn Model Linear Regression
[10]SKlearn Model Dummy Classifier
[11]SKlearn model Random Forest Regressor

is predicted and the squared error between this prediction and the true value is calculated. The variable that gave the least amount of error is selected for the node. This process is recursively continued until all the data is covered (Breiman, 2001).

With classification[12], the outcome of RF is a single class. RF grows many classification trees, that each classify an object from a vector. These classifications are gathered of each tree and the majority class from these classifications is given as outcome (Breiman, 2001).

3.4.4 XGBoost. Has the ideal combination of high performance and short processing time[13].

eXtreme Gradient Boosting (XGB) is a package that improved upon gradient boosting. In order to explain gradient boosting, boosting is first defined. Boosting is a method to turn a group of weak learners into a single strong learner. With boosting, each model is build sequentially on the previous model by minimizing the errors of the previous model and maximizing the influence of high performing models.

Gradient boosting differs from boosting, as gradient boosting uses gradient descend to minimize the errors of sequential models. In other words, gradient boosting seeks to minimize the loss function of the sequential models. One major advantage of optimizing a loss function, is that the loss function is defined by the user. So there is more control over the costs the model has to optimize.

XGB further improved on gradient boosting with the following features: first, XGB uses parallelized tree building with the use of all available CPU cores. Thereby making XGB more scalable. Second, XGB performs tree-pruning on a depth first basis, which converges faster than breadth-first. Third, XGB optimizes hardware usage by cache awareness and out-of-core computing. Cache awareness is reached by storing gradient statistics of each thread in internal buffers, thus freeing up cache space. Out-of-core computing is useful for large datasets that do not fit into memory, by optimizing disk usage. Fourth, XGB penalizes complex models with both L1 (LASSO) and L2 (Ridge) regularization terms, thereby reducing overfitting on the training data. Fifth, XGB is efficient in the handling of missing data values by learning tree branch directions for missing values during training. And sixth, XGB has a built-in cross-validation method at each iteration. Thereby finding the number of boosting iterations for each model on its own.

For regression[14], XGB predicts the real number of pedestrians at each of the given co-ordinates. Gradient boosting has a high performance, is robust and simple in its implementation and optimization (Nieto et al., 2018).

For classification[15], XGB predicts the crowdedness level at each of the given co-ordinates. Gradient Boosting can incorporate different types of predictor features, fit complex nonlinear relationships, and handle the multicollinearity effect of the features with high accuracy (Ding et al., 2016).

[12]SKlearn model Random Forest Classifier
[13]XGBoost Algorithm: Long May She Reign
[14]XGBoost Regressor model SKlearn API
[15]XGBoost Classifier model SKlearn API

3.5 Model Construction

After the full input dataset was constructed, the prediction models could be constructed. This process consisted of the following three steps.

3.5.1 Train and Evaluation split. The first step was splitting the dataset on days into a training and evaluation set. Splitting the dataset on hours would mean that the hourly data from the same day resided in both the training and evaluation set. This would lead to the models being evaluated on seen data, thus returning a too high performance. 80% of all the days were in the training set and the remaining 20% in the evaluation set.

3.5.2 Train Models. The second step was training each model, with the following operations: The first operation was tuning the hyperparameters of the given model on the training data, with *Random Search Cross Validation algorithm*[16] (RS). The hyperparameters were predefined in a dict, containing a grid of hyperparameters to tune. The grid was defined in the format *Parameter Name: Parameter Grid*, with *Parameter Name* referring to the ID of the parameter and *Parameter Grid* to the range of values defined in the grid. RS arbitrarily selects a predetermined number of subsamples from the predefined grid. The subsample of parameters is implemented into the model, which is trained and tested with cross-validation. Unlike hyperparameter tuning on the entire grid, there is no guarantee that the optimal subsample of hyperparameters will be found. The hyperparameters were tuned with the training data.

The second operation was training the model with *Cross-Validation*, using the hyperparameters selected with *Hyperparameter tuning*. As cross validation method, *KFold*[17] was used. *KFold* randomly splits the full training data further into train and test data n times. Each iteration the trained model was tested and the results over all the iterations were averaged, and returned. The advantage of this method, is that there is less chance of the model returning sub-optimal results due to an arbitrary ideal training and test data combination.

The regression models were tested with the following two metrics; the first metric was the R^2 score, which represents the proportion of the variance in the dependent variable that is predictable from the independent variable(s) (see formula 6). High explained variance means that the model understand the complexities within the data distribution, thereby having a high performance on unseen data. The variables in the formula can be defined as follows; (i) x represents the predicted values (ii) y represents the true values (iii) T represents the number of instances in the data. The R^2 score is generally between *0* and *1*, indicating that the regression line of the model represents none or all of the data, respectively. But the R^2 score can also be negative, which indicates that the model is a very poor fit to the data.

$$R^2 = \left(\frac{T(\sum xy) - (\sum x)(\sum y)}{\sqrt{(T\sum x^2 - (\sum x)^2)(T\sum y^2 - (\sum y)^2)}}\right)^2 \tag{6}$$

The second metric was the *Root Mean Squared Error* (RMSE). RMSE calculates the difference between the estimated number of people at a certain point and the actual number of people at that

[16]Random Search Cross Validation - SKlearn
[17]SKlearn Model KFold

given point (see formula 7). The model can over- or underestimate the values. The variables of the formula can be defined as follows; (i) $\widehat{y_t}$ represents the predicted value (ii) y_t represents the true value (iii) t represent the current instance (iv) T represents the total number of instances in the data. The outcome of RMSE can only be positive, as the all the results are squared. Generally, the closer the RMSE is to 0, the better the performance of the model.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(\widehat{y_t} - y_t)^2}{T}} \tag{7}$$

With the training of the regression models, the models seek to optimize the R^2 score. As the R^2 is a good indicator of how well the model fits the data.

The classification models were tested with four metrics. The variables in the formulas are defined as follows: (i) TP refers to *True Positive*, which means that the model correctly predicted a positive class for an instance (ii) TN refers to *True Negative*, which means that the model correctly predicted a negative class for an instance (iii) FP refers to *False Positive*, which means that the model incorrectly predicted a positive class for an instance (iv) FN refers to *False Negative*, which means that the model incorrectly predicted a negative class for an instance (v) $Total$ represents the total number of instances.

The following four metrics were used; the first metric was *Accuracy*, which gives the percentage of correctly classified instances (see formula 8). Even though this is a very intuitive metric, it is also extremely biased with unbalanced data. As just predicting the majority class can give a very high accuracy, even though the model has not correctly learned from the data.

The second metric was *Precision*, which indicates per class how many instances were correctly classified in the given class (see formula 9).

The third metric was *Recall*, which indicates how many instances of each class were correctly classified as the given class (see formula 10).

And the fourth metric was the *F1 Score*, which conveys the balance between the precision and recall score (see formula 11).

There is a trade-off between *Precision* and *Recall*. A high *Precision* usually means that the model focuses on only classifying items if its absolutely sure, thus reducing the recall. While a high *Recall* usually means that items will be easier classified as a member of a class, even though this may not be very accurate, thus reducing the precision of the classification. The higher the F1 score, the better the balance between the two metrics[18]. Thus, *F1 Score* is a good measure if balance between *Precision* and *Recall* is needed and if there is class imbalance present.

With hyperparameter tuning (section 3.5) the classification models seek to maximize the *F1 score*, as this offers the optimal balance between *Precision* and *Recall*.

$$Accuracy = \frac{TP + TN}{Total} \tag{8}$$

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

[18]F1 Score Blog

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{11}$$

3.5.3 Evaluate Models. The third step was evaluating the accuracy of the models on the remaining unseen data. The generated predictions are evaluated with the same metrics as described in the previous step.

3.6 Generalization

It was also possible to predict the crowdedness at an unknown sensor location, with the use of the data of the known sensor locations. The models use the learned data complexities from known sensor locations to generalize the prediction to the unknown sensor location.

For evaluation purposes the model training and evaluation step was altered, as the full dataset was now split on sensors instead of dates. For the training phase all the data of a single sensor were removed and the models were trained on the data of the remaining sensors. As evaluation the models had to predict the crowdedness counts of the previously removed sensor on all the dates in the data set.

The predicted crowdedness counts of each date present in the training set, were evaluated with the true crowdedness counts of the removed sensor.

3.7 Prediction Pipeline

For the municipality of Amsterdam, a pipeline was designed that could generate new unknown input data, based on existing past data (see Appendix 7). The pipeline combines all the previously used methods into a single pipeline.

3.7.1 Input. The input of the pipeline consisted of the prediction period, data point of prediction, the relevant stations, and the trained model that generates the predictions.

3.7.2 Dataset Generation. The dataset needed for the model to generate a prediction, was constructed as follows: first, from the given time period, the following could be derived; (i) date(s) (ii) Hours (iii) circular time of the given date(s) and hours (iv) weekday number(s) of the date(s) (v) whether the given weekday(s) are in the weekend.

Second, from the given data point of prediction the prediction longitude and latitude could be derived.

Third, from the given relevant stations, the latitude, longitude and the mean number of passengers of the given day of the day could be derived.

3.7.3 Dataset Transformation. The dataset needed for prediction was transformed as follows: first, the latitude and longitude of the given data point and each station was scaled.

And second, the distance between the latitude and longitude of the given data point and each station was calculated with the*RBF Kernel*.

3.7.4 Prediction. The predetermined trained model was used to generate predictions with the constructed and transformed input data.

4 IMPLEMENTATION

The methodology as described in section 3 was implemented as follows.

4.1 Prediction

For the normal prediction the data of the period starting from *11 March 2018* and ending on *24 March 2019* from datasets *GVB*, *CMSA* and *Events* is used to construct the full dataset. From the *GVB* data, the station name, the passenger counts, latitude and longitude from the following stations is used;

- *Nieuwezijdse Kolk*
- *Nieuwmarkt*
- *Spui*
- *Dam*
- *Centraal Station*

.

From the *CMSA* data, the sensor ID, crowdedness count, longitude, and latitude of the following sensors is used;

- *GAWW-01*
- *GAWW-02*
- *GAWW-03*
- *GAWW-04*
- *GAWW-05*
- *GAWW-06*
- *GAWW-07*

From the *Event* data, the dates of events are used to mark special dates as events.

The given data was used as input to train and evaluate the following models;

- *Random Forest Regressor*
- *Random Forest Classifier*
- *XGBoost Regressor*
- *XGBoost Classifier*

The predictions of the dates *23 March 2019* and *24 March 2019* of each model was plotted against the true values.

4.2 Generalization

For the prediction of a generalized data point the same data period starting from *11 March 2018* till *25 March 2019* was used with the same *GVB*, *CMSA* and *Event* data as above. The process as described in section 3.6 was done for each sensor. So the models were generalized seven times to each sensor and the average result of these generalizations was given as outcome.

- *GAWW-01*
- *GAWW-02*
- *GAWW-03*
- *GAWW-04*
- *GAWW-05*
- *GAWW-06*
- *GAWW-07*

5 RESULTS

In this section the results of the models are displayed. For the classification results, the *Precision*, *Recall*, and *F1 Score* are given per class label.

5.1 Prediction Results

Table 2: Overall Results Regression

	Training		**Evaluation**	
Model	R^2	**RMSE**	R^2	**RMSE**
LR	-	2051.64	57.7%	654.1
RFG	83.3%	414.53	83.3%	411.27
XGBR	**83.8%**	**408.73**	**85.2%**	**387.28**

The prediction regression results in table 2 show that the *XGBoost Regressor* (XGBR) outperformed the *Random Forest Regressor* (RFG) during the training and evaluation phase, with both *RMSE* and R^2 score. In addition, the results during the evaluation are higher than during the training.

Table 3: Overall Results Classification

Training							
Model	**Accuracy**	**Precision**		**Recall**		**F1 Score**	
Majority	24.1%	-		-		-	
RFC	84.3%	1	90.7%	1	86.6%	1	88.6%
		2	84.5%	2	84.5%	2	84.4%
		3	78.1%	3	78.1%	3	78.1%
		4	84.3%	4	88%	4	86.1%
XGBC	**84.9%**	1	92.4%	1	87.3%	1	89.3%
		2	85.5%	2	84.7%	2	85.1%
		3	78.3%	3	79.2%	3	78.8%
		4	84.9%	4	88.3%	4	86.5%
Evaluation							
Model	**Accuracy**	**Precision**		**Recall**		**F1 Score**	
Majority	24.1%	-		-		-	
RFC	84.4%	1	87.8%	1	87%	1	87.4%
		2	84.8%	2	85.4%	2	85.1%
		3	80%	3	78.8%	3	79.4%
		4	84.8%	4	86.5%	4	85.7%
XGBC	**85.8%**	1	89.9%	1	88.6%	1	89.3%
		2	86.5%	2	86.3%	2	86.4%
		3	81.1%	3	80.5%	3	80.8%
		4	85.8%	4	87.9%	4	86.9%

The prediction classification results in table 3 show that the *XGBoost Classifier* outperformed the *Random Forest Classifier* during both the training and evaluation phase. In addition, both models performed the worst with label 3 and the best with label 1.

5.2 Generalization

The generalization regression results in table 4 show that the *XGBoost Regressor* (XGBR) outperformed the *Random Forest Regressor*

Table 4: Generalization Results Regression

	Training		Evaluation	
Model	R^2	**RMSE**	R^2	**RMSE**
LR	-	1023.95	58.3%	655.97
RFG	82.8%	421.17	84.4%	400.95
XGBR	**84.1%**	**404.25**	**85.5%**	**386.1**

(RFG) in both the training and evaluation phase. The results of the models are less optimal on predicting crowdedness on unknown sensor locations than on known sensor locations. Similar to the prediction results, the performances are higher during the evaluation phase than during the training phase.

Table 5: Generalization Results Classification

Training							
Model	**Accuracy**	**Precision**		**Recall**		**F1 Score**	
Majority	24%	-		-		-	
RFC	**84.2%**	1	90%	1	86.2%	1	88.1%
		2	84.4%	2	84.4%	2	84.4%
		3	77.8%	3	78%	3	77.9%
		4	84.1%	4	87.3%	4	85.7%
XGBC	**84.2%**	1	91.6%	1	85.6%	1	88.5%
		2	84.3%	2	83.4%	2	83.8%
		3	76.7%	3	77.5%	3	77.1%
		4	82.7%	4	88.1%	4	85.3%
Evaluation							
Model	**Accuracy**	**Precision**		**Recall**		**F1 Score**	
Majority	24.2%	-		-		-	
RFC	**84.2%**	1	89.6%	1	87.3%	1	88.4%
		2	84.5%	2	82.9%	2	83.7%
		3	77.6%	3	79.5%	3	78.5%
		4	85.4%	4	87.4%	4	86.4%
XGBC	**84.2%**	1	91.1%	1	87.3%	1	89.2%
		2	85.5%	2	82%	2	83.7%
		3	76.1%	3	78.9%	3	77.4%
		4	84.3%	4	88.5%	4	86.3%

The generalization classification results in table 5 show that the performance of both the *XGBoost Classifier* (XGBC) and the *Random Forest Classifier* (RFC) were identical. Similar as during the prediction results, the evaluation phase had higher results than the training phase.

5.3 Prediction Plots

The plots in figure 5 visualize the crowdedness predictions made over a span of two days, with unseen data. The plots display the following trend; (i) the day starts with a high crowdedness (ii) the crowdedness quickly decreases to an absolute low around 5AM (iii) the crowdedness gradually increases till 5PM and stays there

Figure 4: Prediction Plot Regression

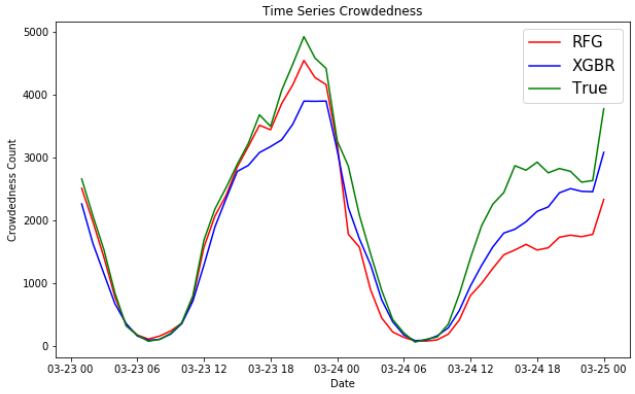
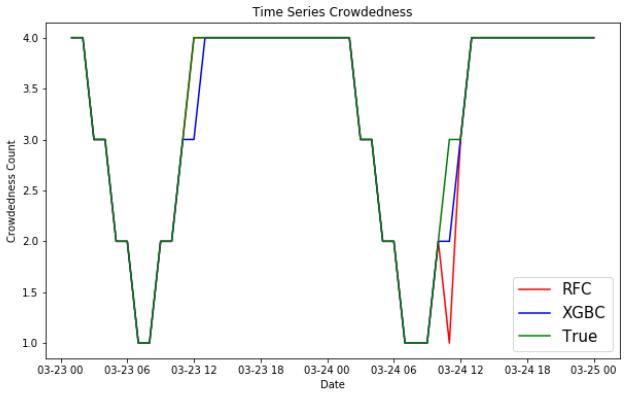


Figure 5: Prediction Plot Classification



6 DISCUSSION

In this section, the results of the previous section are discussed. In addition, the limitations of this research are discussed and ideas for future work is given.

6.1 Prediction

The prediction results indicate the following:

First, the *XGBoost* (XGB) models outperformed the *Random Forest* (RF) models in both regression and classification. This confirms the findings discussed in *Related Work* (2), which stated that XGB models would give the most optimal performance.

And second, the models performed better on the evaluation data than on the training data. This may be due to the model being unable to correctly predict the outliers of the data, which have a higher presence in the training data.

6.2 Generalization

The generalization results indicate the following:

First, the performance of generalization predictions on the unknown data points is less optimal than the predictions made on known data points. This may be due to lack of sensor co-ordinates to gain knowledge about the data complexities from.

Second, with regression, the XGBoost (XGB) model outperformed the Random Forest (RF) model. This confirms the research which stated that the XGB models would give the most optimal performance

And third, with classification, the XGB model had the same performance as the RF model. This contrary to the claims that XGB would return the most optimal performance. The XGB model may underperform due to lack of data. For the individual unknown sensor locations, the prediction performances of the XGB models varied less than the prediction performances of the RF models.

6.3 Prediction Plots

The prediction plots indicate that the *XGBoost* (XGB) models follow the actual trend of the true crowdedness more accurately than the *Random Forest* (RF) models. While the RF models seem to have fitted more to the training data than the XGB models, as RF has more variation in the predictions.

6.4 Limitations

This research has a number of limitations; first, the spatial dimension of the sensors was not explored, as this fell outside of the scope of this research. This is due to combining the Wi-Fi and count cameras of a single area under one name. So, the usefulness of the number sensors was not explored and the optimal number of sensors could not be derived.

Second, this research failed to explain which features have a high importance in predicting the crowdedness of the predefined areas due to lack of time.

And third, the crowdedness counts used as ground truths have not been validated. So it is possible that this affected the performance of the models.

6.5 Future Work and Recommendations

This research could be used as a basis for the the following future research; first, research could be done in the effectiveness of each of the sensors and whether they are all needed, or whether less sensors would offer the same effectiveness.

Second, research could be done in making hourly predictions based on real-time data. The models in this research work well with historical data, with the major drawback that much processing time and power is required to train the models. With the switch to real-time data, a more efficient method to generate predictions needs to formulated.

Third, this same research could be generalized to a larger area, or done on an entirely new area. And see whether this returns a similar performance.

And fourth, the designed pipeline of this research could be implemented and tested.

7 CONCLUSION

The question this research sought to answer was the following: *How can a prediction of the level of crowdedness within the city of Amsterdam be given, based on input from city-wide available data sources?*

A prediction of the level of crowdedness can be given in two different forms, at the given sensor locations. First, a regression model can be trained to predict the actual number of people that will be in at the sensor at an hourly rate. Second, a classification model can be trained to predict the level of crowdedness at the sensor locations. The level is based on the average pedestrian counts near the sensors.

The influence of the *GVB* data on the *CMSA* crowdedness counts is based on the distance between the co-ordinates of each *CMSA* sensor and all the co-ordinates of the *GVB* stations.

The prediction models returned effective results of crowdedness that represented the true values fairly accurately. In addition, the generalization to unknown sensor points produces similar, albeit less optimal results of crowdedness.

The *XGBoost* models outperformed the *Random Forest* models in both prediction and generalization to unknown sensor locations.

ACKNOWLEDGMENTS

The writer of this research would like to thank his supervisor, Noud de Kroon, for the helpful guidance and support he gave. He would also like to thank the municipality of Amsterdam for the support they offered during the research. In addition, he would like to thank his parents for the great support they offered during his entire school career.

REFERENCES

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

Ding, C., Wang, D., Ma, X., and Li, H. (2016). Predicting short-term subway ridership and prioritizing its influential factors using gradient boosting decision trees. *Sustainability*, 8(11):1100.

Heydenrijk-Ottens, L., Degeler, V., Luo, D., van Oort, N., and van Lint, J. (2018). Supervised learning: Predicting passenger load in public transport.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

Nieto, P. J. G., García-Gonzalo, E., Lasheras, F. S., Fernández, J. R. A., Muñiz, C. D., and de Cos Juez, F. J. (2018). Cyanotoxin level prediction in a reservoir using gradient boosted regression trees: a case study. *Environmental Science and Pollution Research*, 25(23):22658–22671.

Wang, X., Liono, J., Mcintosh, W., and Salim, F. (2017). Predicting the city foot traffic with pedestrian sensor data.

Zahedi, P., Parvandeh, S., Asgharpour, A., McLaury, B. S., Shirazi, S. A., and McKinney, B. A. (2018). Random forest regression prediction of solid particle erosion in elbows. *Powder Technology*, 338:983–992.

Zhai, H., Cui, L., Nie, Y., Xu, X., and Zhang, W. (2018). A comprehensive comparative analysis of the basic theory of the short term bus passenger flow prediction. *Symmetry*, 10(9):369.

Zhu, X., Du, X., Kerich, M., Lohoff, F. W., and Momenan, R. (2018). Random forest based classification of alcohol dependence patients and healthy controls using resting state mri. *Neuroscience letters*, 676:27–33.

A VARIABLES FULL DATASET

Full Dataset		
Name	**Description**	**Type**
Date	Date current measurements	Timestamp
Weekday	Weekday of current *Date*	Int
Is weekend	1 if *Weekday* is in weekend, 0 otherwise	Int
Hour	Hour current *Date*	Int
Is event	1 if there is an event on current *Date*, 0 otherwise	Int
Cos and Sin Time	Circular form of current *Date* and *Hour*	Float
Sensor	Unique ID sensor	String
Sensor Latitude	Latitude of each *Sensor*. Scaled and Actual	Float
Sensor Longitude	Longitude of each *Sensor*. Scaled and Actual	Float
Station Latitude	Latitude of each station. Scaled and Actual	Float
Station Longitude	Longitude of each station. Scaled and Actual	Float
Station Passengers	The number of arrivals summed with the number of departures	Int
Station Weight	The RBF Kernel between the station en the given *Sensor*	Float
CrowdednessCount	Number of pedestrians and devices counted in sensor area. Ground truth of this research	Int

B DATA TRANSFORMATION TESTS

B.1 Regression Models

Table 6: Regression: Test Circular Time

Test Features	Model	R^2	RMSE
Numerical Time	LR	29.2%	856.8
	RFG	64%	609.78
	XGBR	59.4%	648.96
Circular Time	LR	49.3%	725
	RFG	84.6%	397.1
	XGBR	77.9%	478.2

Table 7: Regression: Test Scalar

Test Features	Model	R^2	RMSE
No Scalar	LR	49.3%	725
	RFG	84.6%	397.1
	XGBR	77.9%	478.2
Standard Scalar	LR	50.2%	719.14
	RFG	84.6%	396.5
	XGBR	77.9%	478.2
MinMax Scalar	LR	50%	720
	RFG	84.7%	396.08
	XGBR	77.9%	478.2

Table 8: Regression: Test Distance

Test Features	Model	R^2	RMSE
No distance	LR	50.2%	719.14
	RFG	84.6%	396.5
	XGBR	77.9%	478.2
Difference	LR	48%	731.45
	RFG	84.6%	396.08
	XGBR	76.7%	490.6
RBK Kernel	LR	-	-
	RFG	84.6%	396.6
	XGBR	78.4%	472.5
Polynomial Kernel	LR	-	-
	RFG	84.6%	396.2
	XGBR	77.9%	478.2
Linear Kernel	LR	-	-
	RFG	84.7%	396.12
	XGBR	78.3%	478.2

Table 9: Regression: Test Passengers

Test Features	Model	R^2	RMSE
Passengers	LR	-	-
	RFG	84.6%	396.6
	XGBR	78.4%	472.5
No Passenger Data	LR	-	-
	RFG	84.7%	395.98
	XGBR	77.3%	484.2
Passengers multiplied with weight	LR	-%	-
	RFG	84.4%	399.62
	XGBR	78.4%	472.22
Passengers Scaled	LR	-%	-
	RFG	84.4%	399.62
	XGBR	78.4%	472.5

B.2 Classification Models

Table 10: Classification: Test Circular Time

Numerical Time					
Model	**Accuracy**	**Precision**		**Recall**	
RFC	71.2%	1	77.9%	1	77.2%
		2	71.4%	2	72.4%
		3	62.1%	3	61.4%
		4	73.2%	4	73.8%

XGBC	68.5%	1	77.8%	1	70.8%
		2	67.3%	2	69.8%
		3	61.2%	3	54.5%
		4	68.3%	4	79.2%
Circular Time					
Model	**Accuracy**	**Precision**		**Recall**	
RFC	83.7%	1	90.6%	1	88.2%
		2	83.4%	2	83.6%
		3	76.5%	3	77.6%
		4	84.6%	4	85.3%
XGBC	78.3%	1	87.4%	1	82.4%
		2	74.2%	2	79.5%
		3	71.8%	3	65%
		4	79.8%	4	86.2%

Table 11: Classification: Test Scalar

No Scalar					
Model	**Accuracy**	**Precision**		**Recall**	
RFC	83.7%	1	90.6%	1	88.2%
		2	83.4%	2	83.6%
		3	76.5%	3	77.6%
		4	84.6%	4	85.3%
XGBC	78.3%	1	87.4%	1	82.4%
		2	74.2%	2	79.5%
		3	71.8%	3	65%
		4	79.8%	4	86.2%
StandardScalar					
Model	**Accuracy**	**Precision**		**Recall**	
RFC	83.5%	1	90.9%	1	88.6%
		2	83.4%	2	83.3%
		3	76%	3	77.9%
		4	84.1%	4	85%
XGBC	78.3%	1	87.4%	1	82.4%
		2	74.2%	2	79.5%
		3	71.8%	3	65%
		4	79.8%	4	86.2%
MinMax Scalar					
Model	**Accuracy**	**Precision**		**Recall**	
RFC	83.5%	1	90.8%	1	88.2%
		2	83.4%	2	83.9%
		3	76%	3	77.6%
		4	84%	4	85%
XGBC	78.3%	1	87.4%	1	82.4%
		2	74.2%	2	79.5%
		3	71.2%	3	65%
		4	79.8%	4	86.2%

Table 12: Classification: Test Distance

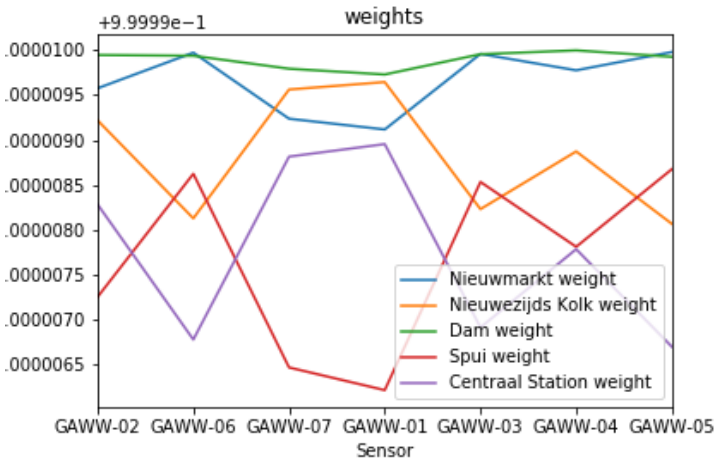
No Distance			
Model	**Accuracy**	**Precision**	**Recall**
RFC	83.5%	1	90.9%	1	88.6%
		2	83.4%	2	83.3%
		3	76%	3	77.9%
		4	84.1%	4	85%
XGBC	78.3%	1	87.4%	1	82.4%
		2	74.2%	2	79.5%
		3	71.8%	3	65%
		4	79.8%	4	86.2%
Difference					
Model	**Accuracy**	**Precision**		**Recall**	
RFC	83.55%	1	91.1%	1	87.7%
		2	83.4%	2	83.3%
		3	76.1%	3	77.9%
		4	84.5%	4	85.3%
XGBC	78.3%	1	90.7%	1	88.4%
		2	74.2%	2	79.5%
		3	71.2%	3	65%
		4	84%	4	85.4%
RBF Kernel					
Model	**Accuracy**	**Precision**		**Recall**	
RFC	84.2%	1	91.2%	1	88.5%
		2	83.8%	2	84%
		3	77.1%	3	78.4%
		4	85.1%	4	85.8%
XGBC	79%	1	88.4%	1	81.9%
		2	75.9%	2	79.9%
		3	72.6%	3	67.2%
		4	79.6%	4	86.4%
Polynomial Kernel					
Model	**Accuracy**	**Precision**		**Recall**	
RFC	84.1%	1	90.9%	1	88.6%
		2	83.4%	2	83.7%
		3	77.2%	3	78.3%
		4	85.1%	4	85.7%
XGBC	78.3%	1	87.4%	1	82.4%
		2	74.2%	2	79.5%
		3	71.8%	3	65%
		4	79.8%	4	86.2%
Linear Kernel					
Model	**Accuracy**	**Precision**		**Recall**	
RFC	84.1%	1	90.9%	1	88.6%
		2	83.4%	2	83.7%
		3	77.2%	3	78.3%
		4	85.1%	4	85.7%
XGBC	78.3%	1	87.4%	1	82.4%
		2	74.2%	2	79.5%
		3	71.8%	3	65%
		4	79.8%	4	86.2%

Table 13: Classification: Test Passengers

Passengers			
Model	**Accuracy**	**Precision**	**Recall**

RFC	84.2%	1	91.2%	1	88.5%
		2	83.8%	2	84%
		3	77.1%	3	78.4%
		4	85.1%	4	85.8%
XGBC	79%	1	88.4%	1	81.9%
		2	75.9%	2	79.9%
		3	72.6%	3	67.2%
		4	79.6%	4	86.4%
No Passenger Data					
Model	**Accuracy**	**Precision**		**Recall**	
RFC	84%	1	90.9%	1	90.1%
		2	83.3%	2	83.8%
		3	76.9%	3	77.2%
		4	85.3%	4	85.1%
XGBC	78.5%	1	88.9%	1	81.8%
		2	76%	2	78.3%
		3	71.2%	3	67.7%
		4	78.6%	4	86%
Passengers multiplied with weight					
Model	**Accuracy**	**Precision**		**Recall**	
RFC	82.9%	1	90.3%	1	87.1%
		2	82%	2	82.3%
		3	75.6%	3	77%
		4	84.1%	4	85.2%
XGBC	79%	1	88.4%	1	81.9%
		2	75.3%	2	78.7%
		3	71.3%	3	66.8%
		4	79.2%	4	86%
Passengers Scaled					
Model	**Accuracy**	**Precision**		**Recall**	
RFC	83.8%	1	90.9%	1	88.2%
		2	83.3%	2	83.6%
		3	76.7%	3	77.9%
		4	84.8%	4	85.6%
XGBC	79%	1	88.4%	1	81.9%
		2	75.9%	2	78.9%
		3	72.6%	3	67.9%
		4	79.6%	4	86.4%

Figure 6: Station Weights



B.3 Plots

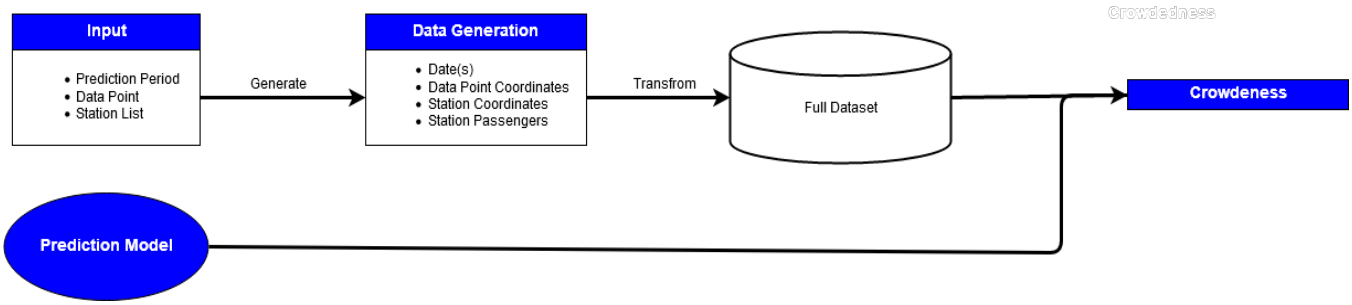


Figure 7: Prediction Pipeline