Crowd Forecasting based on Counting Cameras in Amsterdam's Red Light District

Intelligent Use of Public Space, Gemeente Amsterdam

Paula Sorolla Bayod, Laura Hilhorst,
Emiel Steegh, Aiden Ta, Chieling Yueh
Data System Projects 2021
University of Amsterdam

ABSTRACT

Amsterdam, with its dense population and high level of tourism, is a city that benefits from effective crowd management. This project focuses on assisting municipality aid workers to deal with pedestrian overcrowdedness in the Red Light District. We do so by implementing statistical and machine learning forecasting methods, and visualising the results in a dashboard. We find that our implemented LSTM outperforms the current model that the City of Amsterdam uses for predicting pedestrian crowds with a 7-day forecasting horizon. Additionally, our forecasts are shown in an application that allows municipality employees to turn this information into actionable strategies for crowd management. These two factors make a promising proof of concept for a crowd management tool for the City of Amsterdam to make the city liveable, accessible and safe.

KEYWORDS

Crowd Forecasting, Long-Short Term Memory, XGBoost, Time Series Forecasting, Deep Learning

1 INTRODUCTION

Amsterdam is a very crowded city: it's densely populated and it endures a lot of pressure from tourism. Having an accurate crowd prediction system is of great importance to traffic control and public safety [1]. Therefore, early detection of overcrowdedness has advantages for both residents of the city and the municipality. This project aims to facilitate that by creating a forecasting model for pedestrian crowdedness, which is wrapped in an interactive user interface to be used by municipal employees. This solution aids the City in managing crowds, increasing the city's accessibility and safety.

The geographical scope of this project is limited to the pedestrian sensors located in Amsterdam's Red Light District. The Red Light District is an area of interest for the City of Amsterdam because it is one of the city's most crowded locations and because its spatial features make it dangerous for crowds. Narrow quays and bridges create a risk of falling into the canal when it is too crowded, and violence and chaos may ensue due to excessive crowdedness. The ability to detect excessive and anomalous crowdedness enables the City of Amsterdam to take measures to prevent and solve these situations.

To achieve this goal, we create a long short-term memory (LSTM) recurrent neural network model to predict the crowdedness at three locations in the Red Light District: Korte Niezel, Oudekennissteeg and Oudezijds Voorburgwal (see figure 1). The model has a forecasting horizon of seven days. In addition to this model, we use external data sources such as public transport and hotel occupancy as additional predictors for the model. This state-of-the-art technique outperformed the model currently used by the City of Amsterdam with a 62% accuracy improvement.

This prediction of crowdedness is then used to identify unusual crowds and present that information to the user (the City of Amsterdam) in an interactive application called PLACE[1]. The application features various ways to view crowdedness predictions, as well as a rich source of crowd-related historical data.

2 RELATED WORK

Crowd forecasting. A crowd can be defined as a large gathering of people in one physical location, at the same time, not necessarily sharing the same goal or interest [2]. In the last decades, many studies have been published containing overviews of common crowd management practices [3, 4]. Prior work on crowd forecasting has been done using various methods [5–7].

In [5], they forecast the crowd flow based on cellular network and used a convolution neural network which they compared to traditional time series methods, such as ARIMA. Moreover, studies have been done on crowd forecasting based on WiFi data, such as [6] which used ARIMA and [7] which used LSTMs.

[1]Demonstration video of PLACE: https://youtu.be/D6vy8emBC1o

Figure 1: Locations of the three sensors at Amsterdam's Red Light District.

XGBoost in time series. XGBoost stands for Extreme Gradient Boosting and is a specific implementation of gradient boosting that was initially developed at the University of Washington [8]. XGBoost has proven to perform exceptionally in time series forecasting with its efficient computing time, and memory resource usage [8, 9].

XGBoost has been studied on the task of forecasting sales volume [10]. In [10], they compared XGBoost against traditional models, such as ARIMA, and outperformed them. Moreover, XGBoost is used to forecast cases of hemorrhagic fever with renal syndrome (HFRS) disease, in which XGBoost was compared to ARIMA [11]. Their study showed that XGBoost was better in forecasting the cases than ARIMA. XGBoost shows potential by outperforming traditional time series models, and therefore we include XGBoost in our paper.

LSTM in Time Series. Apart from the statistical and more classical machine learning methods, neural networks are a strong contender in time series forecasting problems [12]. Recurrent Neural Networks (RNNs) are a specific category of neural networks, where the output of the network from the previous time step is provided as an input in the subsequent time step [13]. A variation of RNN is the Long Short-Term Memory (LSTM) Network, which is a popular time series forecasting algorithm [14]. Unlike standard feed-forward neural networks, LSTMs have feedback connections [14].

Prior work is done on LSTMs in forecasting [15–18]. LSTMs significantly improved the prediction accuracy in cellular networks compared to ARIMA [15]. Moreover, variations of LSTM has been studied in financial time series forecasting [16]. In [17], they compared LSTM, XGBoost and ARIMA in time series forecasting in which LSTM outperformed the other models. Furthermore, LSTMs outperformed XGBoost on the task of predicting COVID-19 transmissions in the United States [18]. LSTMs are currently state-of-the-art in time series forecasting and, therefore, are included in our paper.

3 METHODS

Data

We use the *Crowd Monitoring Systeem Amsterdam* (CMSA) dataset collected by the City of Amsterdam. The CMSA dataset consists of the number of pedestrians passing specific sensors located in Amsterdam, measured at an interval of fifteen minutes. As previously mentioned, we focus on three locations inside the Red Light District, namely Korte Niezel, Oudekennissteeg, and Oudezijds Voorburgwal. The CMSA dataset contains information about the date and time, sensor name/location and the number of pedestrians passing that sensor at that time point.

Furthermore, we use a variety of external datasets.

- GVB: This dataset contains check-in and check-out information at two bus stops: Dam and Nieuwmarkt. Both are close to the Red Light District. The data is collected by GVB, a Dutch public transport company. The check-in and check-out counts are recorded every hour, from 01/09/2020 to 15/12/2021.
- KNMI: A wide range of weather-related information about temperature, humidity, sun, rain and more. This data is collected by the KNMI, a Dutch national weather forecasting service, and is collected hourly from 01/09/2020 to 15/12/2021.
- COVID-19: Daily COVID-19 stringency index data which indicates how strict the governmental corona measures in The Netherlands were that day [19]. Moreover, the dataset contains daily COVID-19 cases, hospitalisations and deaths in Amsterdam [20]. All COVID-19 related data is measured from 01/09/2020 to 15/12/2021.
- Holidays and vacation: The annual vacations and holidays of the Netherlands during the period from 01/09/2020 to 15/12/2021.
- Airport passengers: The daily number of passengers arriving at and departing from Schiphol Amsterdam [21], measured from 01/09/2020 to 15/12/2021.
- Hotel: The number of hotel overnights and hotel guests in Amsterdam, measured monthly, measured from 01/09/2020 to 15/12/2021. [22]

We split the aggregated dataset into three parts: a train, validation and test set, as shown in table 1. We train the models on the train set, validate our hyperparameter tuning using the validation set, and use the test set to evaluate model performance.

Table 1: Train, validation, and test set

	Start	End	No. samples
Train	01/09/2020 00:00:00	01/12/2021 11:45:00	43824
Validation	01/12/2021 12:00:00	08/12/2021 11:45:00	672
Test	08/12/2021 12:00:00	15/12/2021 11:45:00	672

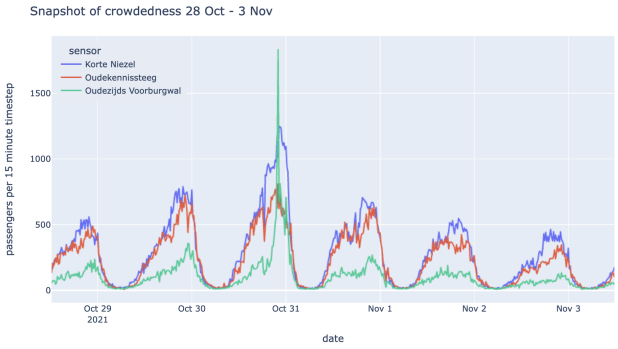


Figure 2: Snapshot of pedestrian crowd of the three locations from 01/12/2021 to 08/12/2021.

Exploratory data analysis

We first investigate the crowd trend over the whole period. Figure 2 shows the pedestrian counts captured at the three sensor locations from 01/12/2021 to 08/12/2021, which indicates seasonality in the time series. The daily crowdedness has a lower peak around 5 in the afternoon and a higher peak around 9-10 in the evening.

In addition, the average crowds fluctuate between days of the week and between months. Figure 3 indicates that the average crowd is higher during the weekend, especially on Saturday, which is not surprising. Likewise, figure 4 shows that the average crowds are the highest in August.

We then investigate the correlation between the crowd and other features extracted from the external datasets by correlation plots and line graphs. As an example of our analysis, figure 5 shows that the crowd trend over time aligns with the COVID-19 stringency index, which directly impacts residents' daily movement and commuting. An extensive correlation heat map between our target value (CMSA data) and our features is shown in appendix A.

Data preprocessing

In this section, we discuss data cleaning, handling null values, and feature engineering.

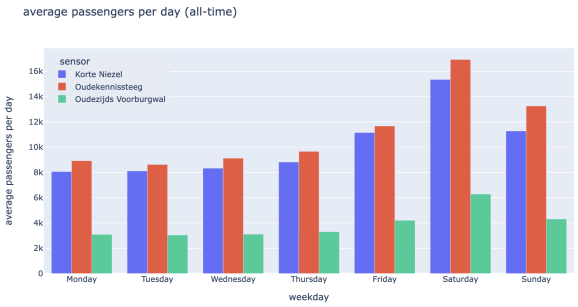


Figure 3: Average crowd grouped by days of week over the whole time period.

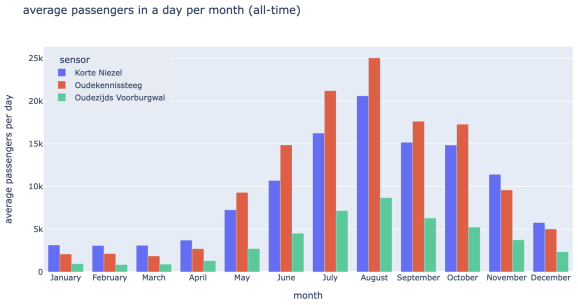


Figure 4: Average crowd grouped by months over the whole time period.

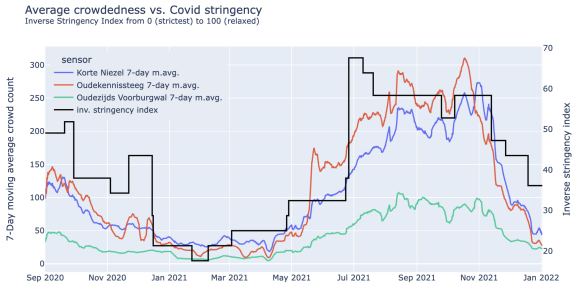


Figure 5: Average crowd trend (7-day moving average crowd count) versus inverse COVID-19 stringency index from 0 (strictest) to 100 (relaxed).

Data cleaning. We conduct data cleaning on the CMSA and GVB datasets in the following steps. First, we remove the duplicate data points. Second, missing time points are added to the dataset and are given a *null* value. Third, we fill in missing values by replacing them with the average of seven and fourteen days before. For example, if 24/10/2021 00:00:00 is missing, then we take the average value of 17/10/2021 00:00:00 and 10/10/2021 00:00:00 to fill in the missing value.

Handling the missing data for the baseline is done differently. Our baseline is a replicate of the current model used by the City of Amsterdam. Therefore, we follow, for our baseline, the data cleaning steps as done by the City of Amsterdam, in which they filled all missing values with zeros.

The other datasets require little data cleaning since they do not contain duplicates. However, we reshape all datasets to follow a 15-minute interval and range from 01/09/2020 to 15/12/2021. Since we have daily, hourly and monthly datasets, we forward fill in the missing values. For example, if the dataset is measured daily, then the rows of the same date have the same values. Likewise, if the data is monthly, the same month's rows have the same values. In cases like hourly rain measures, we divide the millimetres over the quarters of an hour.

Feature engineering. For the XGBoost model, we perform extensive feature engineering. The features are date, weather, passenger check-in and check-out at Nieuwmarkt and the Dam, COVID-19, hotel overnights in Amsterdam and most importantly, past crowd count (i.e., lag features of CMSA). To arrive at these explanatory variables, we conduct a step-by-step data processing as follows:

(1) Extracting time features: *year, month, day, day of week, hour, minute* from the *datetime* column.
(2) Introducing lag features based on the same time point seven days ago. For example, lag features of data point *09/10/2021 11:45:00* include the pedestrian count at *02/10/2021 11:45:00*, at *02/10/2021 12:00:00* (15 minutes later), at *02/10/2021 12:45:00* (one hour later), etc. By doing so, we can avoid having missing lag values in the test set when the forecasting horizon is one week.
(3) Introducing statistical features: average pedestrian count grouped by hour, days of the week, and month. Our crowd data has significant fluctuation in the average crowd count across different hours in the day, days of the week, and months of the year, which motivates our choice in adding these features.
(4) Introducing lag features of the passenger check-in and check-out count at close-by GVB stops, namely *Dam* and *Nieuwmarkt*. A lag of one week in this case, meaning that the feature values are equal to the counts at the same time point of the week before.
(5) Introducing additional features from other external data, including dummy features to represent holidays and vacations, numerical features reflecting COVID-19 stringency, daily cases, hospitalisations, and deaths, 19 numerical features about daily weather and numerical features about the number of daily hotel guests and overnight stays.

Statistical methods

In time series forecasting, it is important to use simple forecasting methods to evaluate how well our more complex methods perform against the simple alternatives [23]. We use the following four methods as benchmark models: *mean*, *naive*, *seasonal naive* (*sNaive*) and *drift*.

The *mean* method uses the average of the historical data for the forecast of all future values [24]. The *naive* method sets all forecasts to the value of the last observation of the historical data [24]. The *sNaive* method is similar to the naive method; however, it takes the seasonal behaviour into account [24]. The sNaive method sets the forecast equal to the last observed value from the same season of the year. The *drift* method is another variation of the naive method [24]. The drift is the amount of change over time or the average change seen in the historical data. Therefore, the drift method allows the forecasts to increase or decrease over time.

XGBoost

Gradient boosting is an efficient machine learning technique for various classification and regression problems, based on the idea of creating an accurate learner by combining many so-called "weak learners" [25]. It has also been used for time series forecasting tasks [26]. Given that the time series dataset is properly transformed into a supervised learning problem, by introducing lag features, gradient boosting algorithms have been demonstrated to be a reliable and efficient problem solver in time series forecasting competitions [26].

The City of Amsterdam has implemented the extreme gradient boosting (XGBoost) algorithm for crowd forecasting. XGBoost is a machine learning model that combines many intentionally crippled decision trees to create a more robust result. We implement the same algorithm as our baseline and aim to improve its performance through feature engineering and data enrichment. Compared to other gradient boosting methods, XGBoost displays the following advantages: (1) effectively minimise the loss function by adding weak learners; (2) be able to prevent overfitting; (3) parallel and distributed calculation reduce running time and memory consumption [27]. These advantages motivate our choice for the gradient boosting algorithm.

The crowd dataset is transformed into a supervised learning problem by introducing lag features of, for example, pedestrian count (CMSA), GVB check-in and check-out passenger count as input variables (as mentioned in section 3). We then fit the training set using all the previously created features and conduct hyperparameter tuning on the validation set.

Table 2: Hyperparameters of XGBoost

Hyperparameters	Result
Gamma	8.27
L2 regularizer	0.41
L1 regularizer	164
Maximum tree depth	14
Minimum child weight	10
Subsample ratio	0.84

Hyperparameters. XGBoost consists of a wide range of hyperparameters that can significantly impact the learning process of the model. We mainly focus on regularisation and tree structure parameters, which directly affect how the model prevents overfitting and builds optimal tree structures. We tune the hyperparameters using a Bayesian optimisation algorithm called *Hyperopt*, in which parameter search ranges are specified based on the references from standard practices [28]. The mean squared error (MSE) is used as the objective function. The results of the hyperparameters search are shown in table 2.

Long Short-Term Memory

We implement a Long Short-Term Memory (LSTM) network for our crowd forecasting. LSTMs are a type of Recurrent Neural Networks (RNNs) [29]. RNNs are valid with sequential data since it uses the internal memory to maintain information about the previous input [29]. However, a fundamental problem with RNNs is that they cannot capture long-term dependencies in a sequence, which is solved in LSTMs [30]. Since LSTMs are capable of learning long-term dependencies, they are useful for time series data [29].

LSTMs capture important features from inputs and preserve this information over a long sequence [31]. Furthermore, LSTMs learn what information is worth removing or preserving based on the weight values assigned to the information during training [31]. An LSTM consists of three gates - forget, input and output gates [7, 31, 32]:

- The forget gate decides whether information is preserved or removed.
- The input gate determines whether new information is added into the memory.
- The output gate decides whether the existing values in the cell will contribute to the output.

We implement LSTM using *TensorFlow* [33]. The model consists of an LSTM layer and a dense layer. Based on the trial and error approach, the model uses CMSA data of the corresponding sensor, check-in Dam, check-in Nieuwmarkt and hotel overstays. Furthermore, to give the LSTM a sense of time and cyclical patterns, we use the sine and cosine transforms of the CMSA data to obtain the cyclical patterns. To determine which frequencies are important, we use the Fast Fourier Transform. Figure 6 shows the Fast Fourier Transform of the sensor Korte Niezel and the chosen frequencies for our LSTM. The input data is normalised using a simple average, and the model predictions are inverse normalised.

Furthermore, our LSTM uses 96 time steps (24 hours) to predict 96 time steps ahead. A visualisation of this can be seen in figure 7.

Table 3: Hyperparameters of LSTM

Parameters	Value
Batch size	18
Epochs	20
Optimiser	ADAM
Learning rate	1e-2
Loss function	MSE
Neurons or cells	32
Hidden layers	1
Drop out	0.2

Table 4: Hyperparameters of EarlyStopping callback

Parameters	Value
Monitor	Validation MSE
Patience	2
Mode	Minimise
Restore best weights	True

Hyperparameters. All hyperparameters for our LSTM are chosen based on the trial and error approach. The chosen hyperparameters for all three sensors are shown in table 3.

To prevent overfitting in the LSTM models, we add a drop out of 0.2 to the LSTM layer. Moreover, we use early stopping using the callback *EarlyStopping* from *TensorFlow*. The *EarlyStopping* callback stops the training when the monitored metric has stopped improving [34]. Table 4 shows the chosen hyperparameters.

Furthermore, we reduce the learning rate when the validation loss stops improving using *TensorFlow's* callback *ReduceLROnPlateau* [35]. By doing so, the model tries to prevent overshooting at the minimum. In table 5, we show the chosen hyperparameters.

Evaluation

We use three metrics in this paper to assess the performance of our models: mean squared error (MSE), root mean square error (RMSE) and mean absolute error (MAE). These metrics

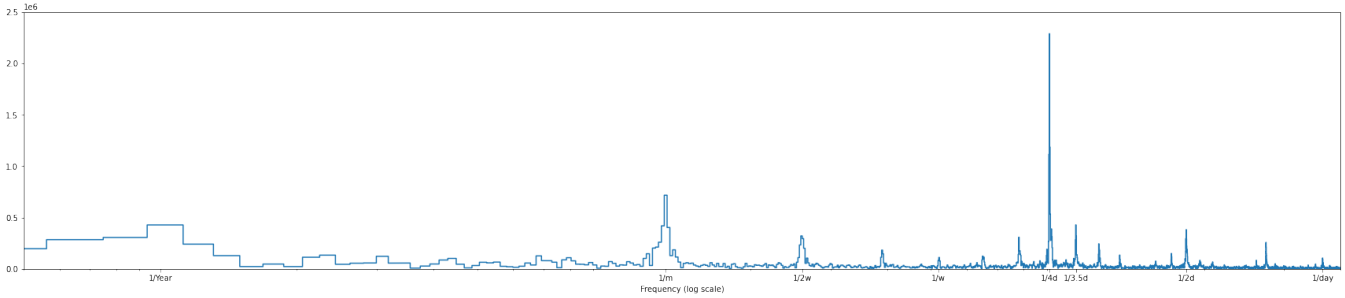


Figure 6: Fast Fourier transform of sensor Korte Niezel over time. The peaks in the graph show important frequencies. The labels on the x-axis show the used frequencies that are used in the LSTM models.

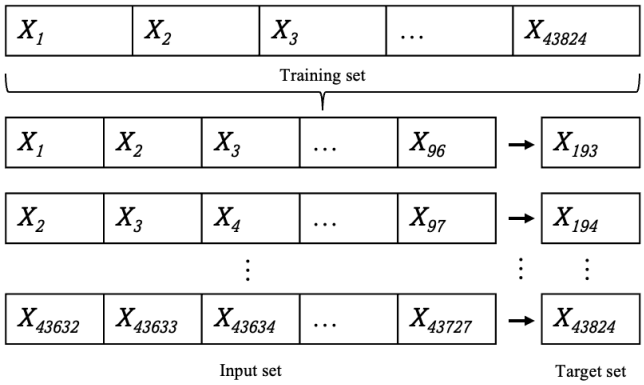


Figure 7: Insight of LSTM training. Training data consists of 43824 samples. The LSTM uses 96 samples to predict 96 samples ahead. In other words, LSTM uses 24 hours to predict 24 hours ahead.

Table 5: Hyperparameters of ReduceLROnPlateau callback

Parameters	Value
Monitor	Validation MSE
Factor	0.2
Patience	1
Verbose	0
Mode	Minimise
Minimum delta	0.0001
Cooldown	0
Minimum learning rate	0

are commonly used in evaluating time series forecasting models [24].

We use the MSE as the validation loss function for LSTM and objective function for XGBoost. MSE is the average of the squared difference between the predicted and actual values and is formulated as follows:

$$MSE = \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}, \tag{1}$$

in which $\hat{y}_i$ is the predicted value and y_i the target value. RMSE is the square root of the average of the squared difference between the predicted and actual values, i.e., the square root of MSE, and is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}, \tag{2}$$

in which $\hat{y}_i$ is the predicted value and y_i the target value. A disadvantage of RMSE is that it is sensitive to outliers since it gives greater importance to large errors [36]. Therefore, we also use the MAE as a performance metric since it is more robust to outliers [36]. MAE is the average of the absolute difference between predicted and actual values, and is calculated as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|, \tag{3}$$

in which $\hat{y}_i$ is the predicted value and y_i the target value.

4 RESULTS

This section presents the results of our models for the three chosen sensors at the Red Light District, namely Korte Niezel, Oudekennissteeg and Oudezijds Voorburgwal. As our baseline model, we implemented the same XGBoost model as the City of Amsterdam implemented. It is common practice to implement simple statistical methods, i.e., benchmark methods, as mentioned in section 3. Furthermore, we implemented an XGBoost model in which we enriched the dataset with additional features. Lastly, we implemented an LSTM model.

Our best model is chosen based on the average improvements of the RMSE as compared to the baseline model. The average improvement is calculated by averaging the improvement in RMSE compared to the baseline over all three sensors.

Table 6: Forecasting Performance of the models per sensor on test set.

	Korte Niezel		Oudekennissteeg		Oudezijds Voorburgwal	
Model	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*
XGBoost*	90.33	72.37	98.48	76.98	25.93	21.46
Mean	96.87	77.68	109.33	93.59	27.92	22.35
Naive	95.89	75.48	118.88	105.88	46.55	42.57
sNaive	49.29	23.94	46.00	**21.43**	16.65	9.72
Drift	95.89	75.48	119.00	106.02	46.81	42.83
XGBoost	49.84	25.96	49.10	30.23	16.61	10.22
LSTM	**29.16**	**18.72**	**31.68**	22.20	**13.08**	**8.86**

* Baseline model, which is the current model of the City of Amsterdam.

Table 6 shows the results of the forecasting performances of the models per sensor. The simple seasonal naive method outperforms the model currently implemented by the City of Amsterdam, with 45% improvement on average in terms of RMSE. Our implemented XGBoost with additional features and hyperparameter tuning improved the baseline RMSE by 44% on average. Hence, our XGBoost did not do better than the seasonal naive method.

Furthermore, table 6 shows that LSTM outperforms the baseline and seasonal naive method. LSTM performs 62% better (RMSE) on average than the baseline model, whereas it performs on average 32% better in terms of RMSE than the seasonal naive method. For sensor Oudekennissteeg, the MAE is slightly lower when using the seasonal naive method, however, the difference with the LSTM is relatively small.

Figure 8 shows the comparison of the predictions between our LSTM and seasonal naive model and the actual values. As our evaluation metrics showed, the LSTM captures the crowdedness better than the seasonal naive method. However, in general, we notice that both models cannot fully capture the high peak in crowdedness on the 11th of December. We see that LSTM captures the high peak better than the seasonal naive model and that it seems to have learned that the 11th of December is a more crowded day than the other days in that week.

5 INTERFACE

Functional requirements and user experience

As mentioned before, the predictions are the basis of the final product: an application used to display crowdedness forecasts and, as such, empower the user with information. To effectively produce a dashboard that serves as a working platform for the team of the City of Amsterdam, we compiled user personas, functional requirements and non-functional requirements. This allowed us to gain an insight into the user of the platform and their needs and wants.

Users. The application's end-users are municipal employees, specifically those responsible for crowd intervention and traffic control. They are educated and have a lot of domain knowledge. They want to use the information to make informed decisions about crowd management measures before the area reaches its safe capacity.

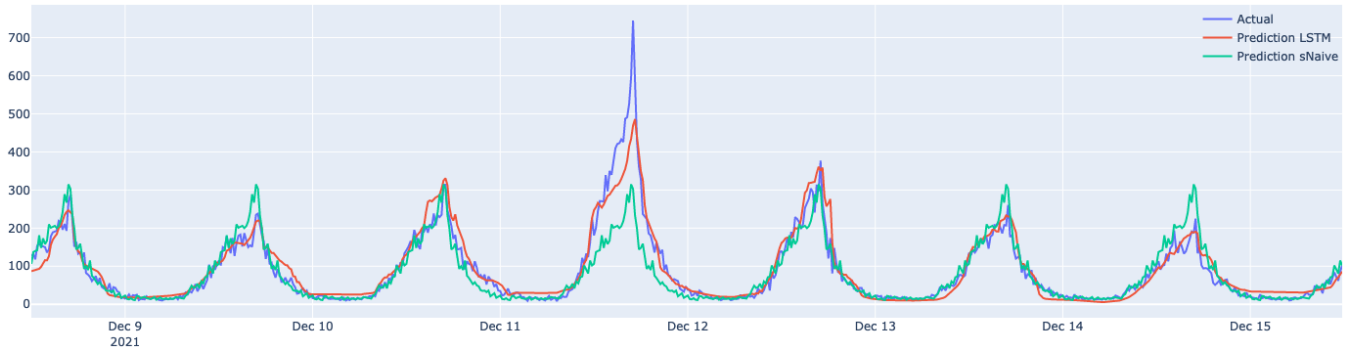
Functional and Nonfunctional requirements. Before designing the interface, we defined functional and nonfunctional requirements [37]. Doing so allowed us to gain a more complete understanding of the problem and its domain, which results in a product that is well thought-out and that provides a solution to the underlying conundrum: helping the City manage overcrowdedness. The main functional and non-functional requests are listed in tables 7 and 8.

Interface design

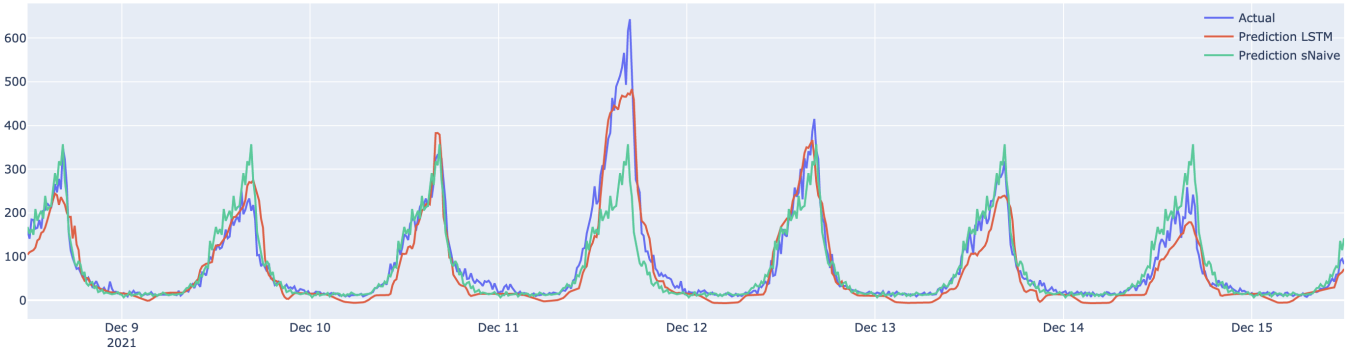
The tool is developed as an interactive web application following a dashboard style. The app is developed using the open-source app framework *Streamlit*, which uses Python language and is commonly used to create data science and machine learning applications in a short time.

The application consists of four tabs. The homepage is presented in a dashboard format. Crowdedness predictions can be viewed in three different ways, serving three distinct purposes: The default view comes in the form of a map, which allows the user to view crowdedness at various locations in one glance like a snapshot of the city (see figure 9a). Switching from the map view to the graph view presents the user with another interactive visualisation, this time in the form of a line graph containing the predicted progression of crowdedness over the selected sensors (see figure 9b). Users can add and remove sensors from the plot as they please and change the prediction horizon. Lastly, the right-hand side of the page contains a list of predicted crowded and overcrowded moments in the next seven days.

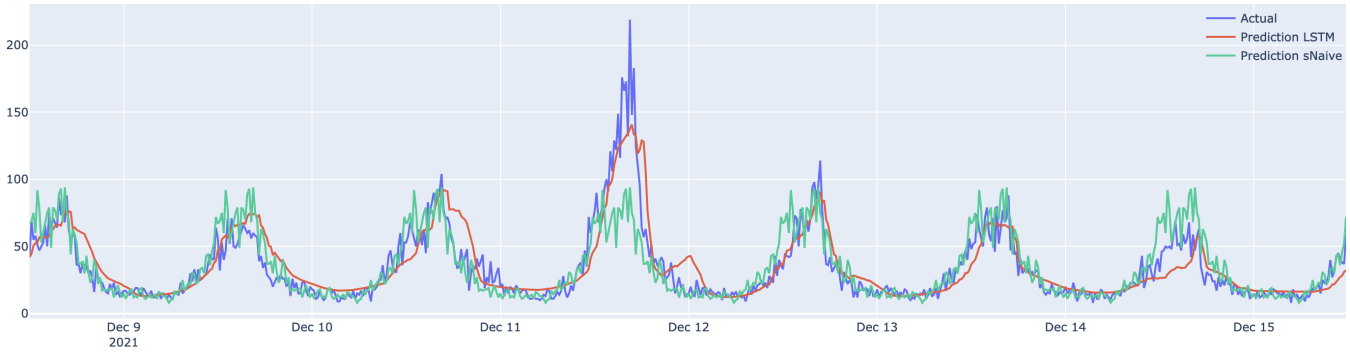
The next tab collects all the information regarding crowdedness in the city, which can be helpful to users to understand



(a) Sensor Korte Niezel



(b) Sensor Oudekennissteeg



(c) Oudezijds Voorburgwal

Figure 8: Comparison of predictions between LSTM, sNaive model and the actual values.

Table 7: Non-functional requirements for the crowd prediction tool

Requirement	Description
NFR-1	The product shall be easily accessible for data consulting.
NFR-1	The system shall allow integration of other data sources, such as weather data.
NFR3	The tool shall have low learnability. It shall guide the user.
NFR-4	The tool shall be implemented in a way that it is easily scalable. and extendable to further application.
NFR-5	The data shall be easy to visualise with filtering and sorting capabilities.

Table 8: Functional requirements for the crowd prediction tool

Requirement	Description
FR-1	The user should be able to identify the crowdedness of each location compared to the established thresholds.
FR-2	The user should be able to change the prediction horizon graph view (from 1 to 7 days ahead).
FR-3	The user should be able to see the prediction for a particular interval on the predicted horizon.
FR-4	The locations with real time overcrowdedness should be identified according its relevance compared to the stablished thresholds.
FR-5	The user should be able to visualize historical data from the CMSA dataset.
FR-6	The user should be able to export historical data from external datasets given a time interval.
FR-7	The user should be able to visualise separate historical graphs plotted next to the CSMA graph for additional comparison.
FR-8	The days and hours for the prediction showed in the map should be able to be changed with intuitive sliders.

future behaviour by taking a look at past situations. This page features plots based on several external datasets: in addition to the CSMA data, we've added public transport, weather and COVID-19 stringency index visualisations (see figure 9c).

The third page allows the user to download the dataset used in the predictive model. A preprocessed, clean, readily available dataset could be helpful in future projects. The user can retrieve the sensors and external datasets of the user's choice in CSV format with the customised export (see figure 9d).

Finally, an about page gives a backstory to the product and presents the team behind it.

6 USER VALIDATION

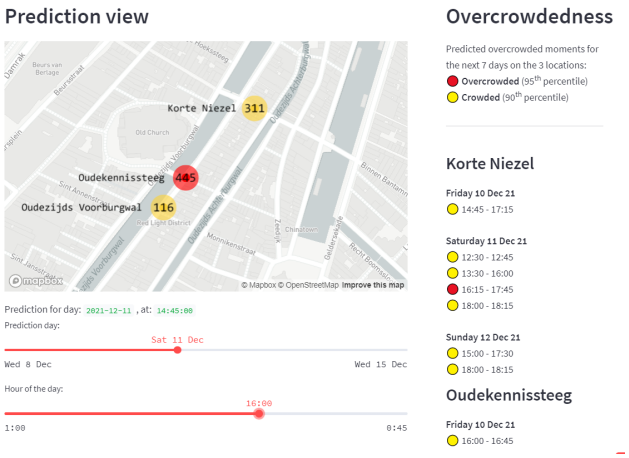
This project is conducted in collaboration with the City of Amsterdam's Data Science department, mainly guided by Boen Groothoff and Vishruth Krishnan. For validation purposes, we maintained contact with the stakeholders every week. We organised meetings with the stakeholders in which we discussed the concepts, user requirements, product design and the execution of the project. Based on the requirements defined by the stakeholders at the ideation stage of the project, i.e., to look into the solution for the pedestrian crowd in the Red Light District area, we then developed the product step-by-step to achieve these requirements. The stakeholders' requests and inputs were taken into account during the execution of the project. At the end of the production stage, the project received positive feedback from the stakeholders regarding the product completeness.

A potential downside of the application, however, is the lack of real-time input data in the training process of the currently implemented model, which makes it unready to be deployed into production. However, the stakeholders' interest were not primarily focused on this aspect, but rather to improve the accuracy performance of the predictive model. In this case, the product is shown to meet the stakeholder's expectation successfully.

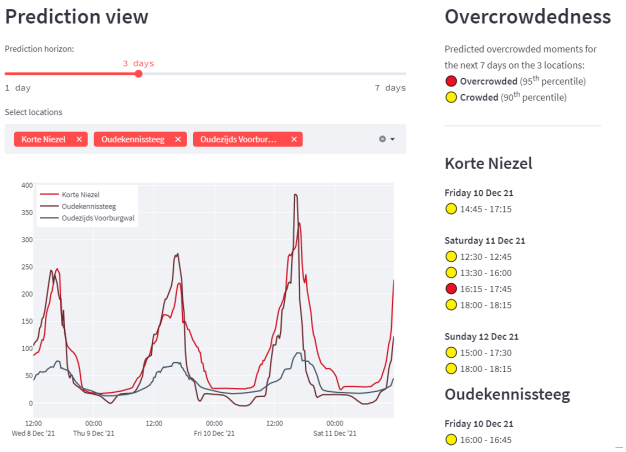
7 DISCUSSION

We found that the seasonal naive method, a simple statistical method, performed exceptionally well on the forecasting task, not unsurprisingly since the CMSA data is strongly seasonal. It performs well because it manages to roughly replicate the daily pattern of the data but fails to capture its finer patterns. The method is very cheap from a computational standpoint, but luckily even the more complex models like the LSTM are not a problem on a small dataset like this one. Because the model does not reflect any differences across days, the municipality should stay away from it, the nice scores mean very little.

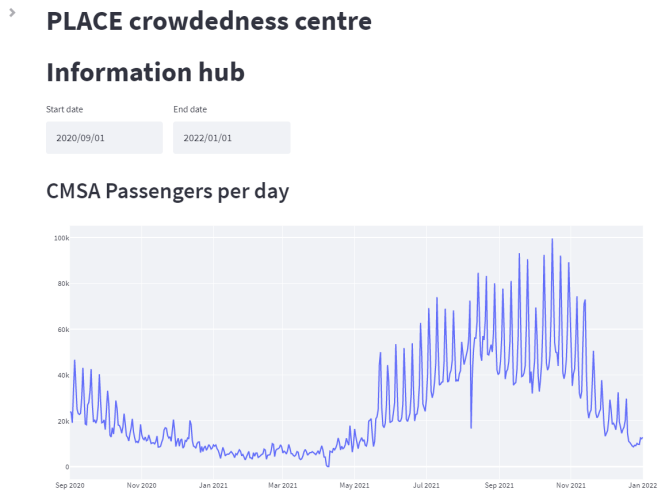
Moreover, our XGBoost model performs slightly worse than seasonal naive. This is in contrast with some competition results, in which XGBoost outperforms statistical methods in time series forecasting tasks [38–40]. However, it is important to note that our time series data is much smaller than those in the competitions, and that the performance of XGBoost greatly depends on the size of training set [38]. In future work, other approaches in introducing lag features should be tested, since the XGBoost model accuracy for time series forecasting tasks greatly depends on what and how the lagged values are utilised [38, 39]. One of the suggested approaches is *recursive modelling*, in which predictions generated for a given day will be used as lag features for the following days. This method is shown in some competition results to perform well in time series forecasting tasks, especially when the forecasting horizon is relatively large [38];



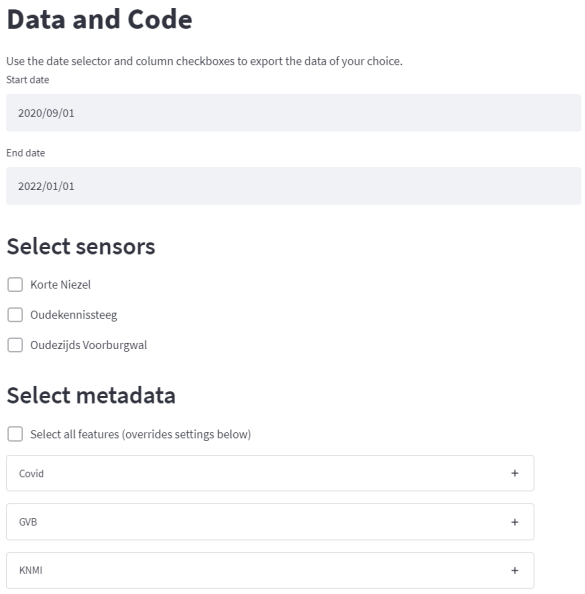
(a) Home page - Map view



(b) Home page - Graph view



(c) Information hub page



(d) Data and Code page

Figure 9: Application interface screenshots

however, the model might not be stable when a large prediction error is made on one day, which in turn deteriorate the predictive performance for the following days.

Our LSTM outperformed the currently implemented XGBoost of the City of Amsterdam, which we used as our baseline. We chose the hyperparameters of LSTM based on trial and error approach. To further improve the model, one could tune the hyperparameters in a more sophisticated way by doing, for example, a grid search. Moreover, we used a simple LSTM model, as in it only uses one LSTM layer. However, in [7], they compared different variations of LSTMs, in which a convolutional LSTM (ConvLSTM) gave the best results on forecasting crowdedness based on WiFi sensors. Hence, future work could implement a ConvLSTM model for the forecasting task of the CMSA data.

While our LSTM does seem to capture that some days are more crowded than other days, it still does not entirely capture the high peaks of crowdedness. Since the high peaks are of importance for the City of Amsterdam, future work could focus on a model that fully captures the high peaks of crowdedness.

Overall, the application is easy to use, and the functional and non-functional requirements have been met. However,

we did notice that Streamlit has limited options for multi-page applications and interface customisation in general. For a production-grade application, it would be better to use more custom techniques than Streamlit. Additionally, it was not possible to use real-time data for this project.

If the application as we built needs to go into production, a few things need to happen: We start with real-time data ingestion from the sensors and external data sources. The data needs to be cleaned, versioned and stored. Models need to be retrained every so often; in the case of an LSTM with our currently small dataset around monthly might be a good starting point, but this needs investigation. The model would need live validation on the incoming data reflected in a dashboard. Preferably, if the validation accuracy falls under a certain threshold, this also triggers the model's retraining. And lastly, the model needs to generate a seven-day forecast every day and supply that to the front-end. Figure 10 illustrates the pipeline we designed to make this work. the pipeline also provided a loose structure to our programming approach.

8 CONCLUSION

The focus of this project was to develop a solution to aid municipal employees in taking decisions regarding pedestrian crowdedness. We focused on the Red Light District because of its unique pedestrian situation and because of the presence of CMSA sensors in this area. Our LSTM model outperforms the City's baseline XGBoost model, meaning that our predictions are more accurate than those used previously. Additionally, we developed a prototype for an interactive dashboard to visualise these predictions, which contributes to aiding the decision-makers. Lastly, having a rich source of historical crowd-related information may be of use to other cases within the City of Amsterdam or help to improve this experiment's reproducibility.

While the results of this project are very much a proof of concept and not a deployable product, they seem promising. With more resources dedicated, our solution could be of great added value to the City of Amsterdam's crowdedness management strategy.

REFERENCES

[1] S. Wray, "Why the city of amsterdam developed its own crowd monitoring technology - itu hub," 2021.
[2] N. Wijermans, "Understanding crowd behaviour," *University of Groningen, Groningen*, 2011.
[3] W. Challenger, W. Clegg, and A. Robinson, "Understanding crowd behaviours: Guidance and lessons identified," *UK Cabinet Office*, pp. 11–13, 2009.
[4] H. (Health and S. Executive), *Managing Crowds Safely, A Guide for Organisers at Events and Venues.* 2014.
[5] Y. Zhao, X. Miao, J. Li, and X. Ding, "Urban crowd flow forecasting based on cellular network," *ACM International Conference Proceeding Series*, 5 2019.
[6] J. F. Determe, U. Singh, F. Horlin, and P. D. Doncker, "Forecasting crowd counts with wi-fi systems: Univariate, non-seasonal models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 6407–6419, 10 2021.
[7] U. Singh, J. F. Determe, F. Horlin, and P. D. Doncker, "Crowd forecasting based on wifi sensors and lstm neural networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, pp. 6121–6131, 9 2020.
[8] H. Zheng, J. Yuan, and L. Chen, "Short-term load forecasting using emd-lstm neural networks with a xgboost algorithm for feature importance evaluation," *Energies*, vol. 10, no. 8, 2017.
[9] B. M. Pavlyshenko, "Linear, machine learning and probabilistic approaches for time series analysis," in *2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP)*, pp. 377–381, IEEE, 2016.
[10] W. Ping, Z. Min, L. Zhang, W. Bian, W. Qu, L. Tuo, and Y. Wang, "Time series forecast of sales volume based on xgboost you may also like re-layout of product placement in retail industry to minimize order picking time with group technology method c wahyudin, s rahmawati and n shafanah-the effect of labor transfer based on panel data on farmers' per capita consumption of commodity energy time series forecast of sales volume based on xgboost," *Journal of Physics: Conference Series*, vol. 1873, p. 12067, 2021.
[11] C. X. Lv, S. Y. An, B. J. Qiao, and W. Wu, "Time series analysis of hemorrhagic fever with renal syndrome in mainland china by using an xgboost forecasting model," *BMC Infectious Diseases*, vol. 21, pp. 1–13, 12 2021.
[12] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice (3rd ed).* Monash University, Australia, 2021.
[13] J. Brownlee, "Multi-step lstm time series forecasting models for power usage."
[14] D. Karmiani, R. Kazi, A. Nambisan, A. Shah, and V. Kamble, "Comparison of predictive algorithms: Backpropagation, svm, lstm and kalman filter for stock market," *Proceedings - 2019 Amity International Conference on Artificial Intelligence, AICAI 2019*, pp. 228–234, 4 2019.
[15] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," *Proceedings - IEEE INFOCOM*, 10 2017.
[16] J. Cao, Z. Li, and J. Li, "Financial time series forecasting model based on ceemdan and lstm," *Physica A: Statistical Mechanics and its Applications*, vol. 519, pp. 127–139, 2019.
[17] I. Paliari, A. Karanikola, and S. Kotsiantis, "A comparison of the optimized lstm, xgboost and arima in time series forecasting," Institute of Electrical and Electronics Engineers Inc., 7 2021.
[18] J. Luo, Z. Zhang, Y. Fu, and F. Rao, "Time series prediction of covid-19 transmission in america using lstm and xgboost algorithms," *Results in Physics*, vol. 27, p. 104462, 8 2021.
[19] OXCGRT, "Oxford covid-19 government response tracker," 2020.
[20] esri Nederland, "Covid-19 - historische gegevens rivm (vlakken)," 2022.
[21] CBS, "Statline - luchtvaart; maandcijfers nederlandse luchthavens van nationaal belang," 2022.
[22] CBS, "Statline - hotels; gasten, overnachtingen, woonland, regio," 2022.
[23] R. Hyndman, "A forecast ensemble benchmark," 2018.
[24] R. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice.* Australia: OTexts, 3rd ed., 2021.
[25] S. B. Taieb and R. J. Hyndman, "A gradient boosting approach to the kaggle load forecasting competition," *International journal of forecasting*, vol. 30, no. 2, pp. 382–394, 2014.
[26] M. Assaad, R. Boné, and H. Cardot, "A new boosting algorithm for improved time-series forecasting with recurrent neural networks," *Information Fusion*, vol. 9, no. 1, pp. 41–55, 2008.

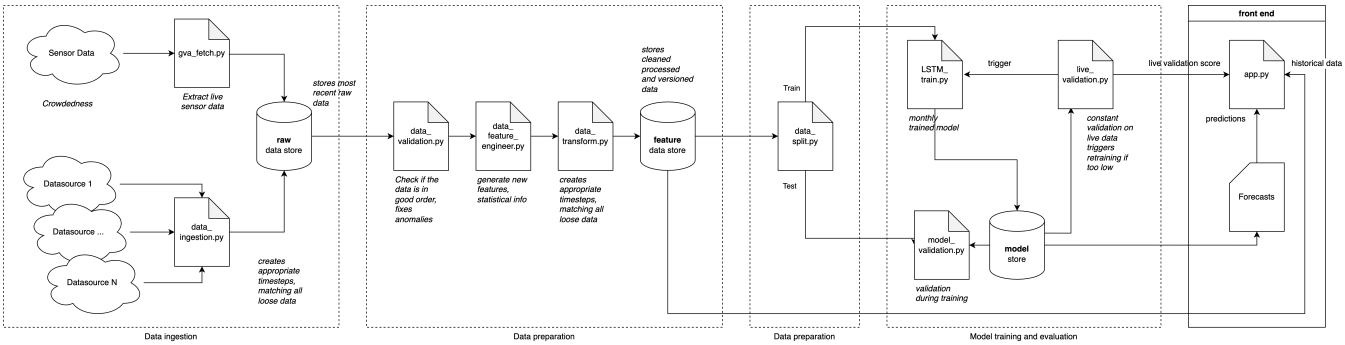


Figure 10: Pipeline for a production grade version of the application.

[27] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, *et al.*, "Xgboost: extreme gradient boosting," *R package version 0.4-2*, vol. 1, no. 4, pp. 1–4, 2015.

[28] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.

[29] F. Lazzeri, *Machine Learning for Time Series Forecasting with Python*. John Wiley & Sons Inc, 2021.

[30] F. Lazzeri, "3 reasons to add deep learning to your time series toolkit," 2019.

[31] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "The performance of lstm and bilstm in forecasting time series," in *2019 IEEE International Conference on Big Data (Big Data)*, pp. 3285–3292, IEEE, 2019.

[32] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of arima and lstm in forecasting time series," pp. 1394–1401, Institute of Electrical and Electronics Engineers Inc., 1 2019.

[33] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[34] TensorFlow, "tf.keras.callbacks.earlystopping | tensorflow core v2.7.0," 2021.

[35] TensorFlow, "tf.keras.callbacks.reducelronplateau | tensorflow core v2.7.0," 2021.

[36] N. Vandeput, "Forecast kpi: Rmse, mae, mape & bias | towards data science," 2019.

[37] L. Chung and J. C. S. d. Prado Leite, "On non-functional requirements in software engineering," in *Conceptual modeling: Foundations and applications*, pp. 363–379, Springer, 2009.

[38] E. Madrid and N. Antonio, "Short-term electricity load forecasting with machine learning," *Information (Switzerland)*, vol. 12, pp. 1–21, 01 2021.

[39] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "M5 accuracy competition: Results, findings, and conclusions," *International journal of forecasting*, 2022.

[40] S. Ma and R. Fildes, "The performance of the global bottom-up approach in the m5 accuracy competition: A robustness check," *International Journal of Forecasting*, 2021.

A APPENDIX

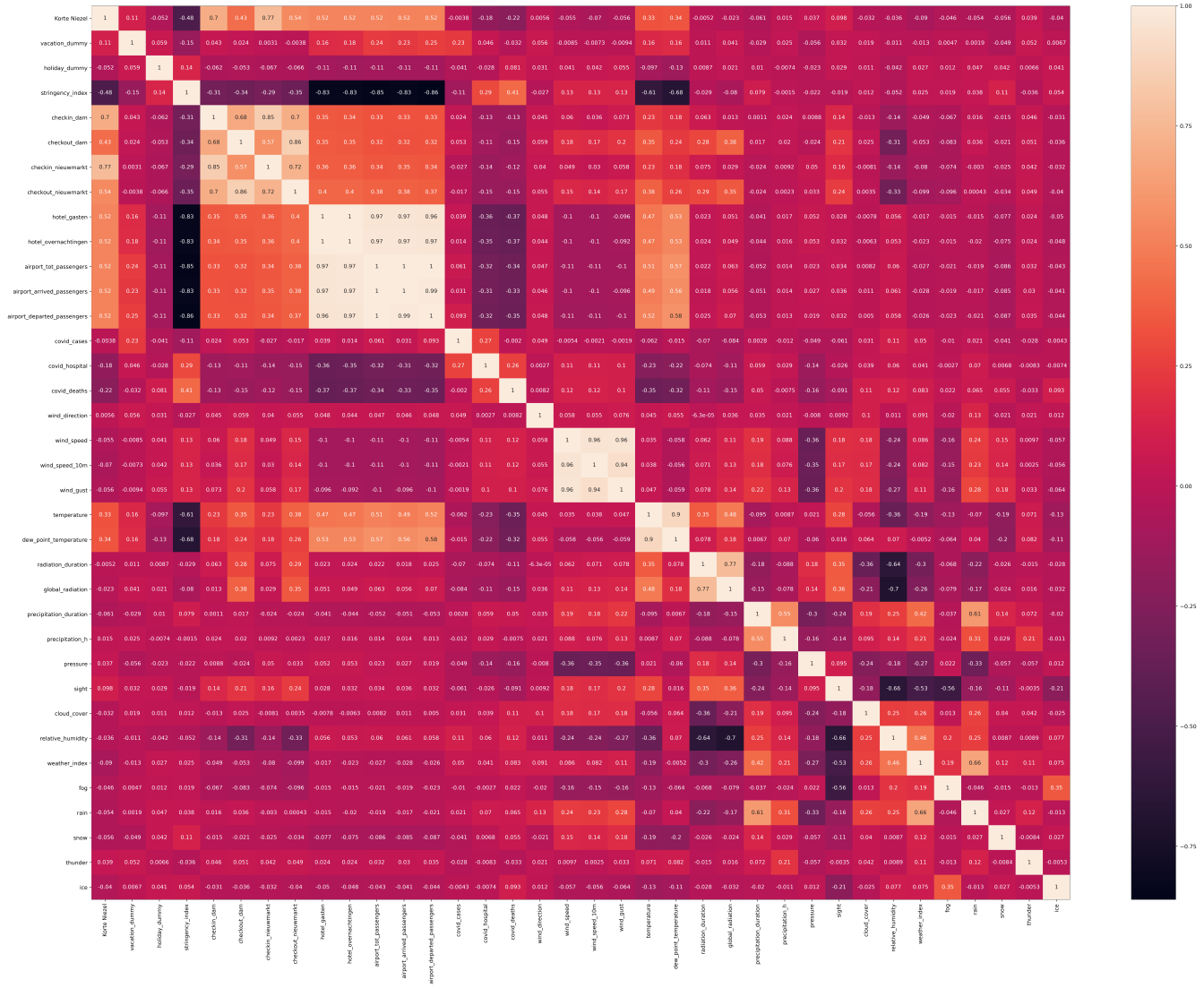


Figure 11: Correlation heat map of sensor Korte Niezel (target value) and features.