University of Amsterdam

Leren en beslissen

AI#34

SEH accident prediction for Stokhos

Authors:
Niels Sombekke - *1268573*
Erencan Tatar - *12681598*
Luka Wong Chung - *12676926*
Ian Ronk *12664804*

January 31, 2021

Contents

1 Abstract

It is impossible to precisely know where accidents will happen in the future. But what can be done is make predictions, using historical data. This is the exact thing the Stokhos challenge is about. Stokhos assists local ambulance departments to predict where and when accidents will occur and how much resources, in the form of personnel and ambulance vehicles are needed in the future. This challenge aims to support this endeavor, by creating a model, capable of finding cyclical patterns in historic data. The historical data spans two years (15 May 2018 - 10 May 2020), consisting of 84456 rows and contains all ambulance rides, with their respective time of arrival, position of accident and type of ride. The goal of this challenge is to give the best estimate possible to the question, whether there is a statistical significant pattern, over different time spans, in a year. Used methods to answer this question are cyclical regression methods and random forest models and LSTMs. These methods initially showed that there is a significant cyclical pattern, but that, to reach a higher accuracy, there is more data necessary to create a better model. The final mean absolute error on the prediction model, using an LSTM model is 10, but using extra data, and possibly combining with Random Forest models, this model could yield much higher results.

2 Problem Statement

Stokhos is a software company that tries to use statistical models to advise on the optimal emergency vehicle coverage possible per region. Their software uses algorithms that predict where future incidents are most likely to happen. This results in reduced response times of an emergency vehicle, when an accident occurs and thus ultimately save lives. Their system has several components, including sending the right ambulance to the accident. However the system only uses the distribution of accidents over a whole year and no timespan shorter than this. The problem that needs to be solved is creating a model that can possibly see patterns on shorter timeframes and create better predictions. Of course it is not possible to predict the future, but using historical data, one can look for patterns, like whether more accidents happen at a certain location and time. The challenge is to predict where accidents will occur as accurately as possible over a shorter timeframe than a year, so that preemptive placement of emergency vehicles can be improved.

3 Data

The available data provided by Stokhos consists of two datasets covering two years from the Zuid-Holland-Zuid (ZHZ) Veiligheidsregio. The first dataset contains the data from 15-May-2018 to 1-May-2019 and the second dataset contains the data from 1-May-2019 to 1-May-2020. The data entries in both datasets are calls made to the Emergency Medical Call Center (EMCC), containing a subjective patient acuity decision made by the ambulance crew. Before starting with the data preprocessing it is important that the raw data is analysed to make the data preprocessing as efficient as possible. If you don't know anything about the raw data one could argue that it would simply be enough to (when present) delete all the records in the dataset that have a NaN value for some of the features. In our case this would result in an 80% drop of available records from 84456 records before removing the NaN values to 14183 after. It does, however, have a lot of NaN values that can not be used in any model so it was important to find a way to get rid of a lot of NaN values without losing too much of the data. That is the reason why we started with a data analysis on the raw data before preprocessing.

3.1 Data Analysis Raw Data

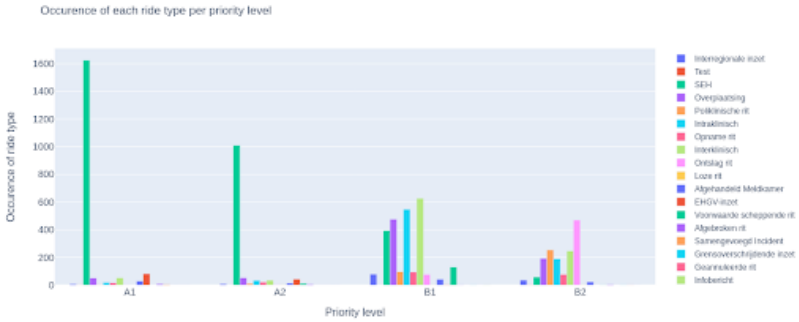


Figure 1: The occurrence of each ambulance ride type for every priority level of incoming calls.

In figure 1, the features "Prioriteit CPA" and "Rit soort afsluiting" where used to take a look at the importance of the different values of these features. From the figure it is clearly visible that there are certain ride types that are more common among certain priority levels. For example, you can see that the "SEH" rides (intensive care) seem to be the only significant rides for the A1 and A2 priority types. Thus you can fill in the NaN values for the priority type with a decent amount of certainty if you see that the ride type is "SEH". It is visible that, to some extent, the same thing holds true for the ride types: "Interklinisch", "Intraklinisch", "Voorwaarde scheppende rit" and "Ontslag rit". We can thus fill in their corresponding codes. Another important distinction that will have to be made to see what data actually will be used for modeling is that of the different priority types. It really is not wise to apply one single measure with this because it might be more important to be on the spot faster for one priority level over another. One way to get a good valuation of the weights that will be applied on these priority levels is to ask it to a professional but, since data never lies, it will probably be helpful to look for patterns in the data. To do this we took a look at the difference in time between the moment that the emergency message came in and the time that the ambulance was on the spot. This was done for all the four different priority levels.

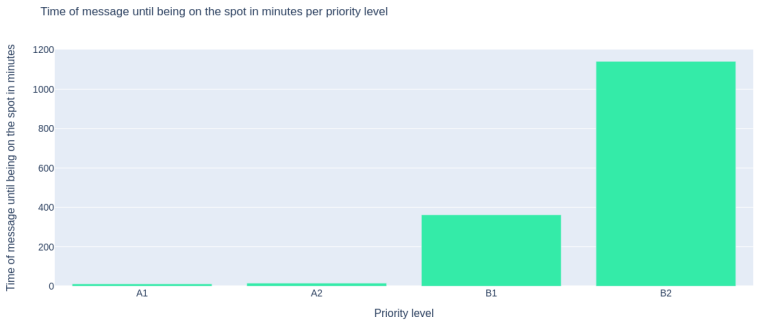


Figure 2: This figure shows the average time that it takes for the ambulance to get to the spot of the accident from the moment the emergency message comes in. The different priority levels are displayed over the x-axis and the average time on the y-axis.

The above figure uses the features "Prioriteit CPA", "DTG Melding" (time of message) and "DTG TP" (time that ambulance is on the spot) to take a look at the importance of each class. The average time is calculated by taking the difference between "DTG TP" and "DTG Melding". The figure clearly shows that there is a huge difference between the "A" priority levels and the "B" priority levels. The average time of the "B2" priority level is more than a hundred times larger than that of the "A1" priority level while the average time of "A2" is only 1.5 times as big as that of "A1". It seems that the average time of priority levels in one class (A and B) don't differ significantly. It does seem that the average time of "B2" is significantly higher than that of "B1" but in comparison with the difference between the A class and the B class this is almost negligible. From figure 1 it is also visible that, in the "A" priority levels, the "SEH" rides are by far the most common so it could be concluded that these rides are the most important to look at. Every record of the data also corresponded to a certain ambulance type ("Voertuigsoort afk"). In figure 3 the different ambulance types are plotted against the different priority levels to see if there was a significant correlation to fill in the NaN-values.

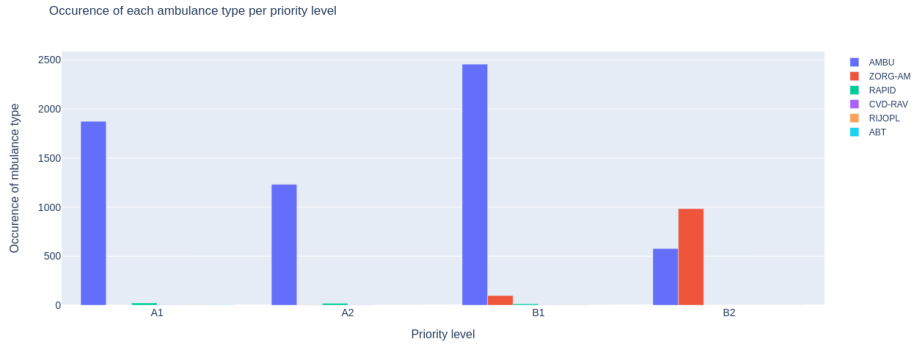


Figure 3: This figure shows the amount of records that correspond to a certain ambulance type on the y-axis and the different priority levels on the x-axis

Figure 3 uses again the feature "Prioriteit CPA" but it also uses "Voertuigsoort afk" which is the ambulance type to look for significant correlations between them. This figure clearly shows that the ambulance type "AMBU" is significantly used more over any other ambulance type for the important "A" priority levels. This could be used for filling in missing values since one could assume that a missing value in one of the "A" priority levels would belong to the ambulance type "AMBU".

3.2 Preprocessing

To work with raw data it has to be preprocessed. Removing, adding or changing features and replacing or removing NaN values. What we did with a feature and a NaN value was mainly determined by conclusions drawn from the data analysis mentioned above. Both datasets started with 26 feature columns and ended up, after preprocessing, with 12 left. A similar decrease was seen in the amount of rows/entries when the NaN values were removed or replaced, going from 41293 to 26552 entries in the 18-19 dataset and from 43163 to 29379 entries in the 19-20 dataset. The datasets were also combined in a 18-20 dataset. These prepared datasets, primarily the prepared 18-19 and 19-20 dataset, were the ones used in our solutions. Below you can find explanations for the removed and added features, and further comment on the removal or replacement of NaN values. A

complete feature table with more insight on specific features (including definitions of abbreviations and usage) can be found in the appendix.

3.2.1 Removed Features

1. Unnecessary timestamps: DTG UI, DTG TP, DTG OV/VT, DTG VP, DTG AB, DTG VR, DTG ER, DTG UI.
   These eight datetime features are not relevant for the goal of predicting the time and location of emergencies, they could be relevant if you want to predict when specific ambulances are available and when they are not.

2. Unnecessary Location features: Naam locatie 1, Naam locatie 2, Huisnummer, Plaats Naam, Postcode
   These five location based features tell something about the location of the emergency, but are not used because our solution uses the to latitude and longitude converted Van_X_Coord and Van_Y_Coord.

3. Unnecessary target locations: NAAR_NAAM_LOCATIE1, NAAR_NAAM_LOCATIE2,
   NAAR_HUIS_PAAL_NR, NAAR_PLAATS_NAAM, NAAR_POSTCODE,
   Naar_X_Coord, Naar_Y_Coord.
   These features are similar to the previous one, but these give the location to which the patient is taken. These features are not useful because we are looking at the emergency and its location, not the accident handling when the ambulance arrives at the place of emergency.

4. Roepnaam eenheid: This feature gives the call sign of the ambulance who handled the emergency. It could be useful when predicting availability of specific ambulances, like the datetime features mentioned above, but is not relevant for the current goal.

3.2.2 New Features

1. DoY Melding, Month Melding
   These two features give the day of the year and the month of the DTG Melding, useful for predicting the emergencies per day or month. They are generated with the dt.dayofyear and dt.month function of the pandas library. In a similar way you can generate the day of the week (dt.dayofweek), day of the month (dt.day), hour (dt.hour), minute (dt.minute) and many others. The first two were the ones primarily used in our solution.

2. Weights
   These weights are based on the Prioriteit CPA feature, giving higher weights to A1 and A2 cases than the B1 and B2 cases. Our solution has not used this feature but there are possibilities how it could be used, for example if you want to define priority of the emergency as a numerical value.

3. Latitude, Longitude
   These two features are generated with the Transformer function of the pyproj library, changing the epsg 28992 coordinate system (Projected coordinate system for Netherlands) of Van_X_Coord and Van_Y_Coord to the epsg 4326 coordinate system (Geodetic coordinate system for World). Our solution required these features because the mapping library (Folium) used for the plots and gridmaps used the epsg 4326 system.

4. Square_no
   The grid the emergency was assigned to, for more information about the grid system see section 4.2.1.

3.2.3 Removal or Replacement of NaN Values

1. Van_X_Coord, Van_Y_Coord All entries were removed when one of these coordinates was missing because these features are crucial for determining the location of the accident. This resulted in a total loss of 82 entries in the 18-19 dataset and 33 entries in the 19-20 dataset.

2. Prioriteit CPA
   The NaN values were replaced with A1 if the entry had SEH in the Rit soort afsluiting feature column. See figure 1. We did the same with A2 if the entry had EHGV-inzet in the Rit soort afsluiting column. The rest of the entries with NaN values in this column were removed because they made no difference to the final dataset.

3. Voertuigsoort AFK
   Every NaN value in this column was replaced with AMBU because this was the emergency vehicle primarily used in any emergency.

3.2.4 Other Preprocessing Steps

1. Rit soort afsluiting
   With our goal of shortening response times of emergencies in mind we only looked at unplannable emergencies, these being SEH and EHGV Inzet. All entries with other values than those two were removed. This led to the majority of B1 and especially B2 entries being removed.

2. Set boundaries grid
   Specific latitude and longitude boundaries were set loosely based on the Zuid-Holland-Zuid veiligheidsregio, the region where the datasets were primarily focused on. All entries outside these boundaries were removed, this led to the removal of almost all outliers. See section 4.2.1 for more information.

3. Sort values on DTG Melding
   The datasets were sorted on DTG Melding to get a chronological dataset. Lastly we did a reset of the index to finalize the datasets.

4 Method

4.1 Possible approaches

With all predictive models training data and a independent held-out validation dataset have been used. Different machine-learning algorithms like regression, random forests, neural network, and LSTM models are all applicable and have all been looked into.
Regression works well if there are clear trends and there is continuity and lots of data. To make a simple regression model one can make time and the number of incidents the only variables, while looking only look at a given area. This however, drastically decreases the number of data points making it unreliable and useless. Nevertheless, regression can be used to see seasonal, weekly, and monthly trends with the total number of incidents across the whole operational area 'Zuid-Holland-Zuid'.
Instead of using raw latitude and longitude data incidents can be clustered into groups making classification tools more useful. Clustering can be done with either ML-tools like GMM or by creating geographical clusters. With this, population information in a single grid could be incorporated. This is further explained in section 4.2.1 Usage of grid map.
A random forest implementation can be done in two ways. Predicting in which cluster an incident is going to occur on a given time, or predicting the amount of accidents per day per cluster. LSTM can also be used with the multivariate dataset, but needs lots of training data. This can thus be used for a yearly prediction of total incidents. Or it can be used for a yearly prediction when looked at one cluster level. This later one can only be done if and only if the training data for that one cluster is big enough.
Since our data consists of only positive samples (datetime 'with' accidents) we can't directly make it a classification problem. However, generating negative samples (datetime 'without' accident) could resolve this. This however needs to be done very carefully since, this could lead to a model rejecting the possibility of an incident in an area.

4.2 Chosen solution

Our solution is twofold. First, a approach which consists of a Random Forest and LSTM model. Besides this findings from a analytical methods were used for seasonal or frequency patterns that may give insides when looking only at time as a factor. The reason for multiple models is that the processing of spatial-temporal data is not fully developed yet. So choosing multiple approaches will assure us that we are making the best use of different ML-tools. Doing so will give us a system that can validate and maybe compare strange predictions with the results of different models.

4.2.1 Usage of Grid Map

To cluster the incidents we looked at our operational area Zuid-Holland-Zuid and made a rectangle in which almost all accidents occurred based on this region. The boundaries chosen are 51.7 to 51.98 latitude and 4.23 to 5.15 longitude. This big rectangle was divided into smaller equally sized ones, based on the average distance that can be travelled in a maximum of 15 minutes. See figure 4. Each rectangle was given a number and each data entry in this rectangle was given this number in the square_no feature column. The size of these smaller rectangles was determined by testing several sizes in google maps, calculating the travel time from one corner to the one diagonally opposite of it. The 15 minute norm was used since the aim for A1 ambulance deployment is to arrive on site within 15 minutes of receiving the call from the operator, according to *RIVM and AZN, 2018*.
This norm can also be used to calculate how many ambulance parking places are needed in the

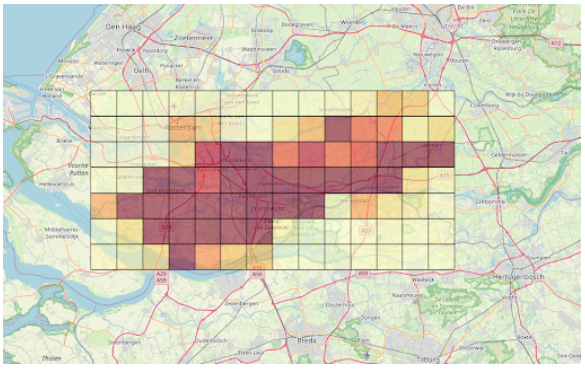


Figure 4: Grid Map - Large rectangles

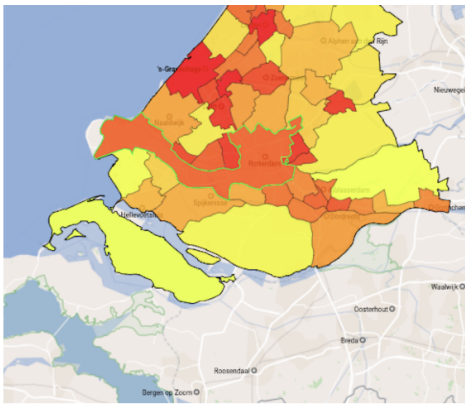


Figure 5: Population density - Zuid Holland from citypopulation.de, 2020

Netherlands to meet the demand for ambulance care. However, the population density seen in figure 5, which roughly speaking can be correlated with the number of accidents, differs greatly per area. This fact can cause a model to grasp towards a higher populated area and ignore other parts or see them as less relevant. See figure 14 in the Appendix, for a better overview of the spread of incidents. Having a closer look at the data revealed that almost 25% of incidents occurred in one of the rectangles. This can make a model really over-fit on a specific cluster, predicting almost no other cluster. This happened in one of our Random Forest models when trying to classify a square_no with a time of emergency. To fix this case of over-fitting the larger rectangles with 2000 emergencies or more were divided into 4 smaller ones. And if these smaller rectangles still had more than 1000 cases they were divided again in 4. See figure 6. This caused the over-fitting to largely disappear. We eventually decided to approach each rectangle individually, see 5.2, choosing the grid with larger rectangles over the one with the smaller divided ones. It was easier to work with and the square numbers of the 18/19 grid matched with the ones in the 19/20 grid.

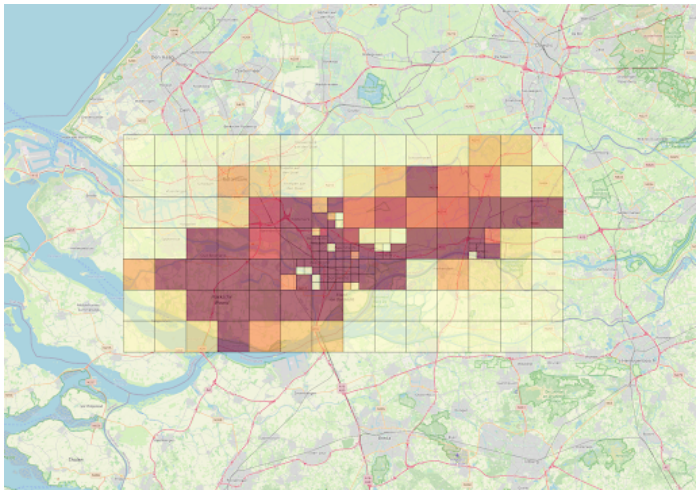


Figure 6: Grid Map - Smaller divided rectangles

4.2.2 LSTM / RNN

Long short-term memory (LSTM) is a type of recurrent neural network (RNN). A RNN is a type of neural network that, unlike regular neural networks, uses prior information to learn. They have loops in them which allows information to persist. Regular RNN's do, however, have a hard time carrying long-term information, according to Olah, 2015. So if a sequence or time series is too long, information gathered in the beginning may have been lost later on. LSTM models on the other hand perform well with longer time sequences. That is because a LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell can 'remember' values over a longer time and over arbitrary time intervals. So missing days in the sequence should not be a problem. The three gates in the cell determine how information flows into and out of the cell. The gates regulate if information is important enough for in the long run and if so, it gets stored in the cell.

Pseudo-code LSTM-Rnn:

```
1: Set input/output units, lstm units, batch size, dropout,
epochs, loss function, and optimizer to define LSTM Network (L)
2: Normalize the dataset into values from 0 to 1, for faster computation
3: Select training window size (tw) and organize Di accordingly
4: for n epochs and batch size do
    5: Train the Network (L)
6: end for
7: Run Predictions using L
8: Calculate the loss
```

Referenced from Kumar et al., 2018

Implementation

To make the LSTM model for our research we used sample code from Brownlee, 2016 on time series prediction with LSTM recurrent neural networks in python with keras which was posted on his website "machinelearningmastery.com". For the LSTM model we used keras sequential models with LSTM, Dense and dropout layers. The dense layer can model non-linearity properties and the dropout sets random activations in the layer to zero, as a counter to overfitting. We have used the Mean Absolute Error (MAE) loss function in combination with the adam optimizer. Empirically we found that an epoch of around 100 works well. The number of epochs determines how many times you go through your training set and can be useful for logging and periodic evaluation. The batch_size was usually set to 1, 64 or 72, but it should not affect the model much.

```
model = Sequential()
model.add(LSTM(250, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
```

4.2.3 Random Forest

Random forest is a ML-algorithm for classification and regression problems which uses multiple different decision tree's. The number of decision trees can be varied according to the complexity of the problem. Every tree is trained on different parts of the training data, which should prevent any negative effects of outliers on predictions. All trees are put together to create an uncorrelated forest of trees who together predict better than all trees separately. A benefit of using random forest is that the tree depth and thus the complexity of the model can be easily varied with a parameter. Random forest can also work with numeric as well as categorical variables.

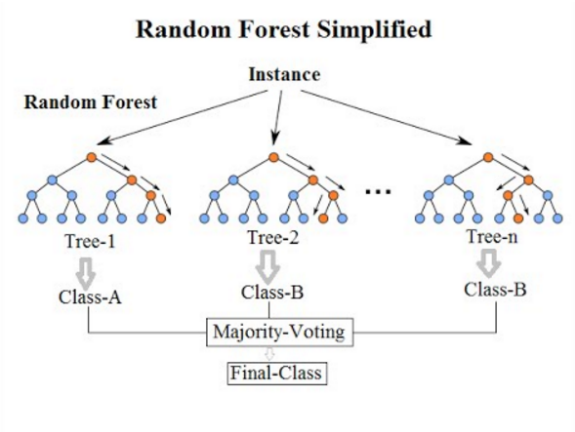


Figure 7: "Random Forest complete, from V. Jagannath, *Wikipedia*. Copyright 2017 by Venkata Jagannath"

Implementation

Although the general random forest algorithm remains the same for classification and regression tasks, both are optimized differently. Like it has been said before, the implementation can be done by either predicting; which cluster on a given time, or the number of incidents per day per cluster. The first one being more a classification problem, and the last one having more regression characteristics. However from empirical findings, it can be said that a predictive classification model for the incidents per day performs better.

Step 1: predicting the number of incidents per day per cluster

1. Instantiate RandomForestClassifier model with N decision trees
2. Train the model on the training data
3. Calculate MAE
4. Use rf.predict(test_data) to make predictions

Step 2: Interpret and visualize

Possible options:

1. Visualize small version of the tree using export_graphviz and pydot

– Instantiate tree with a max_depth so to limit the size of the tree (step 1.1)
– fit the small tree (step 1.2)
– Save and export

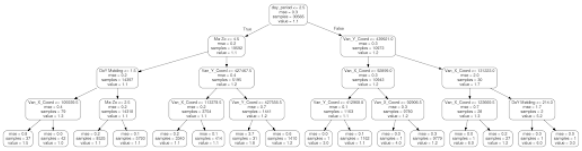


Figure 8: Arbitrary example with features e.g. Van_X and Van_Y

2. Visualize predictions over days
– With the yearly predictions for every cluster, the grid map can be made into a heatmap like below.

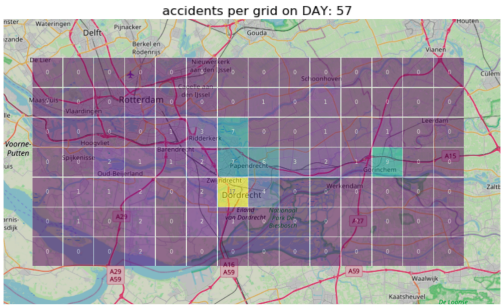


Figure 9: Grid map with number of predicted incidents on a given day

4.2.4 Evaluation Methods

It's useful to know how well the model performs on a given dataset. The random forest model classifier outputs a category, {0,1,...,N} incident per day. Here the Mean Absolute Error (MAE) can be used, to give a helpful indication. Also F1 is used to evaluate the precision and recall of the model. F1 score ranges between zero and one, with one as a high recall and high precision.

For the LSTM model, our predictions are given not per grid, but for the whole area. Getting a wrong prediction here means that is has impact on in multiple grids. That why the LSTM model was evaluated with the root-mean-square error (RMSE) metric. RMSE gives a higher weight to larger errors.

5 Results

5.1 LSTM sequential model

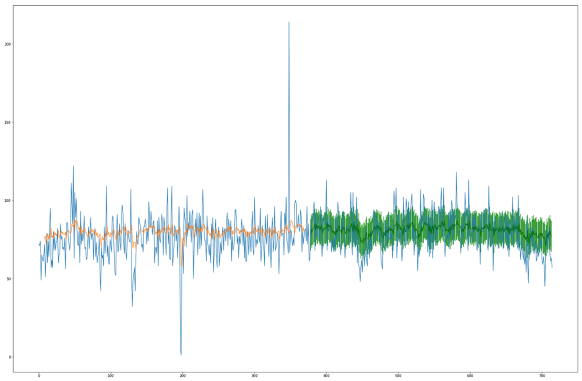


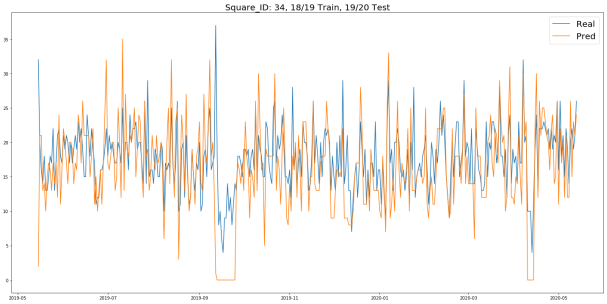
Figure 10: This figure shows the amount of emergency messages of the whole data per day in blue. The predictions of our LSTM model for the training dataset is shown in orange, and the predictions on the unseen test dataset in black with green error bars.

The results of our first LSTM model are shown in figure 10. The orange part of the figure covers approximately the first (2018/2019) dataset and is used for the training samples for the model. The green part on the other hand is used for test samples and covers approximately the second (2019/2020) dataset. Even though it seems like the model is finding some sort of pattern in the data, the RMSE error on both the train and test data is still fairly high. The model has an average error of about 15 accidents (14.85 RMSE) on the training dataset, and about 12 accidents (11.98 RMSE) on the test dataset. The MAE, which gives information about the average number between the models guess and the true number. For this model the MAE is about 10 incidents above or below the predicted daily number of incidents. Therefore we use an error bar of 10 incidents. Compared to the baseline average, which is 80 incidents a day, that's a 8% margin on average.

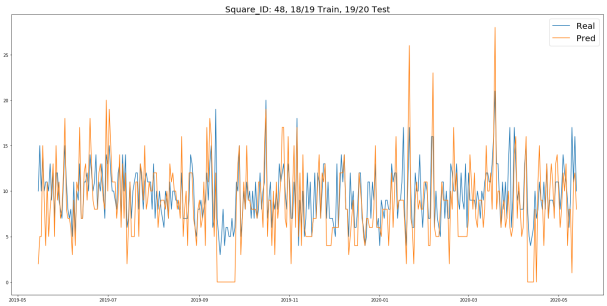
5.2 Random Forest

Predicting the number of incidents per day per cluster.
The following plots below show the prediction (in orange) for clusters over time for the validation set. There are a number of things that stand out. First of all, in plot a and b a period around 2019-09 where the predictions are zero, is noticeable. This has to do with the training data, which is missing that specific period.

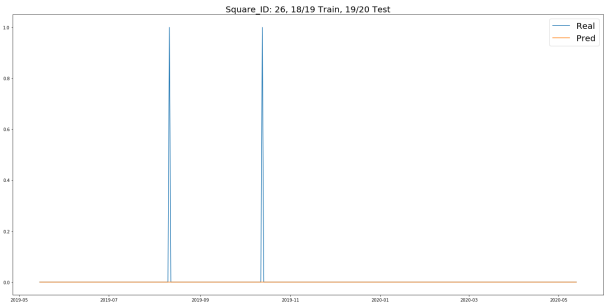
Secondly, training a model on a cluster where with little to no incidents took place, will result in a weird prediction as expected. Only cluster with lots of data like for example square_no 34 and square_no 48 will produce better predictions. One can say that clusters in which incidents rarely take place, a wrong prediction is not as bad, since it will have the slightest effect on the total care in the area. However, ignoring these cluster or expecting no incidents year around, can not be done.



(a) Prediction gird nr. 34



(b) Prediction gird nr. 48



(c) Prediction gird nr. 26

Figure 11: Examples of grid predictions with Random Forest

Taking a closer look at the model predictions, it shows strong similarity with the number of incidents of the training data. To combat this overfitting and make the model generalize more, specific features were used: $n\_estimators = 1000$, $max\_depth = 30$ and $min\_samples\_leaf = 2$.

The used evaluation metric for this model are mainly MAE and F1 score. The Mean absolute error is around 0.6 incidents, while the RMSE sits a bit higher on 2.7. This means overall predictions a stable but once in a while outliers can occur. This is also confirmed by the max error of 44. The models good MAE can be confirmed by looking at the F1 score, which is 0.73. This means a high precision (0.71) and a high recall (0.75). Because of the way we output our predicting as a classifying category and not as a continuous variable, the recall of 0.75 is the metric for the accuracy of the model.

5.3 Analytical Findings

Although there is not enough data to accurately predict how many accidents will occur over the course of a year, there is enough data to deduce conclusions about the frequency and amount of occurrences over shorter time spans. Examples are the weekly cycle and the changes over the course of a day. As is demonstrated below in graph 12 and 13. These analytical findings are found, using a tool by Facebook, called Prophet, which helps organize the data and make predictions with time series. However it also creates clear plots about the timeseries, which helps in the analysis. Note that only the shorter time spans yield significant results, because there is not enough data to predict a good trendline for longer periods.

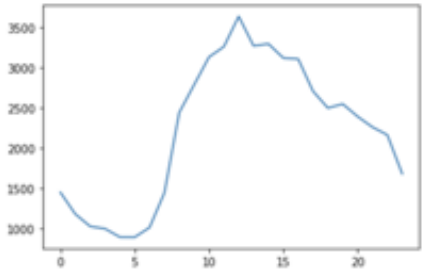


Figure 12: Occurrences per hour

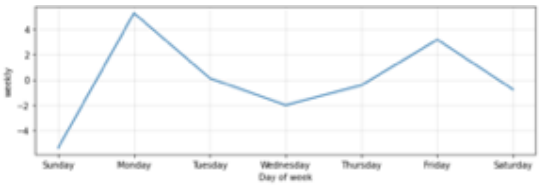


Figure 13: Occurrences over the week

We will now display our significant results in two different groups over the timeframe: 24 hours and the timeframe: 7 days:

Daily deviation One can clearly see a significant difference in occurrences per hour. During the night, especially in the early hours of the morning, significantly less accidents and ambulance rides occur. This can possibly be explained by the fact that during these hours most people are still asleep and thus are not involved in accidents. The number of occurrences shoots up at around 7 to 8 o'clock, when most people wake up and go out to work. During the evening it climbs back down, before reaching the average of 1500 for the night.

Weekly deviation During the week, there is also a significant shift visible. Where during the start and end of the week, the most accidents occur: Monday and Friday. Why this is, is not known to us. What can be seen from the data is that this amount declines during the middle of the week and reaches its minimum on Wednesday. Another minimum is during Sunday, when the least amount of accidents happen.

6 Conclusion

It can be concluded that over shorter periods of time, a trend can be seen and this creates hope that, when the dataset is increased, possible trends can also be found in the seasonal and yearly data. In conclusion, we can see that during the day, with the top at 12 o'clock, the amount of rides is at its highest and that during the weekdays Monday and Friday, there are more rides than other days, with Sunday and Wednesday being the lowest. As of now, we can only speculate as to why that is, besides the face that planned rides are more common during the day and on weekdays. In addition to these analytical findings, there is just not enough data available to create a suitable model that has satisfactory accuracy. This is why we have no definite answer on the question which of the models is the best: An LSTM or a random forest. To answer this question, extra data is needed and then the best of the two models can be used.

7 Discussion

There are solid methods that can be used to predict how many accidents will occur at a certain time. However this is only the case on shorter time frames. Thus, to make an accurate prediction over the whole year, additional data is needed, as there are now only two data points in some cases. This means that the data has a high variance and that the model has trouble with pinpointing exactly how many occurrences are predicted at that point in time. This can be seen in the Mean Squared Error for the LSTM being 10. To improve this, extra data is needed. As previously stated, on shorter time frames, it is possible to make more significant predictions, because there are far more data points available to extract this information. This also shows that the LSTM model over the whole year has the potential for high accuracy on predictions, if more data is supplied.

Using negative data sampling has been done to create more data and make classification boundaries more distinct, but model results did however not increase. This could be due to the fact that ratio positive and negative samples were 1:1 or due to the method used for generating data. It must be remembered that the last two months of the dataset (2020:03:15 - 2020:05:15), captures the start of the pandemic in Holland. However, no distinct increase was found during that short period, so normalization was not used.

In the end, the project has been a success in finding the right models for shorter time frames, but because of the shortage of data, finding patterns over seasonal periods, with statistical significance, is not yet possible.

8 Recommendation

With regards to this research, an LSTM model or Random Forest model would be recommended. If additional data is acquired these models could be improved and one of the two can be chosen. The recommendation to Stokhos would thus be: collect additional data and use either of the models that is supplied with this report. This would give the best shot at answering the question: How many incidents will take place on a given day on shorter timeframes. The Grid system can be used to make the problem a classification issue, so that a random forest model can be used. It is also key to decide on an error margin, as the model, as previously stated cannot be accurate on the amount of incident exactly. This error margin could be decided upon the collected data or substantiated by thinking about it using knowledge of the problem. In summary the following list of things is recommended:

1. Gather more data over more years, to increase accuracy of longer time frames, like seasons.
2. Continuing with more data on the supplied LSTM or Random Forest model.
3. Make use of the analytical findings of this project to substantiate the predictions. (A mean can be extracted for all of the graphs and use the deviation from the mean to predict the amount of cases from this baseline).
4. If the grid method is used, additional data about weather and population data per cluster can be useful.

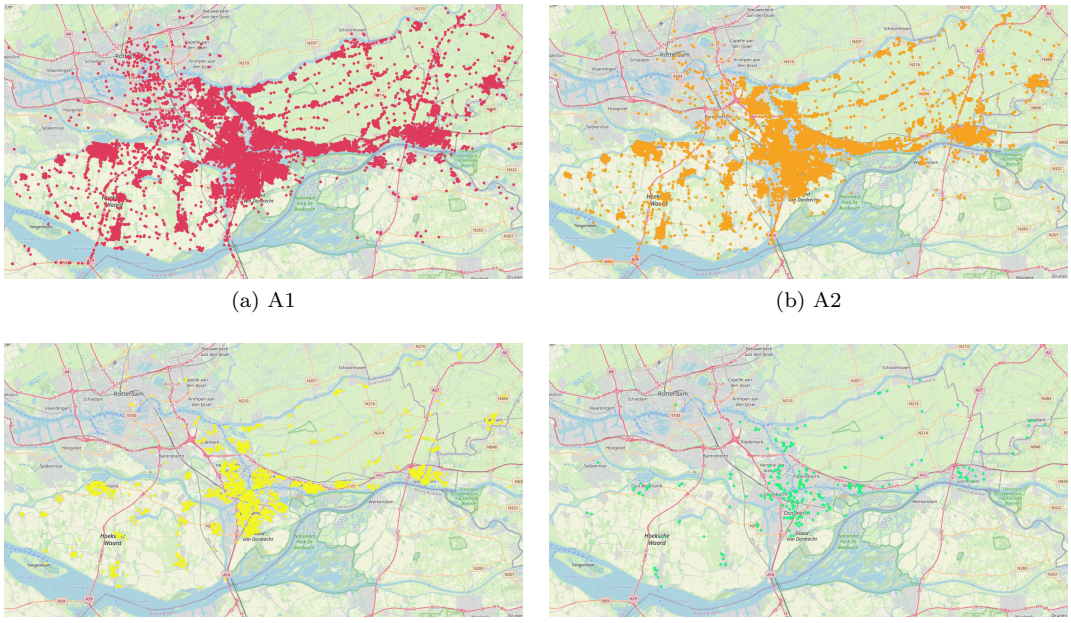
9 Appendix

9.1 Feature Table

Feature	Description	Dtype	Used?
DTG Melding	Time of call to Emergency Medical Call Center (EMCC).	datetime YYYY-MM-DD HH:MM:SS	
DTG OV/VT	Time of assignment emergency to an ambulance crew.	datetime DD-MM-YYYY HH:MM:SS	
DTG UI	Time ambulance crew drive away.	datetime DD-MM-YYYY HH:MM:SS	
DTG TP	Time ambulance crew is at location of the accident.	datetime DD-MM-YYYY HH:MM:SS	
DTG VP	Time ambulance crew leaves the accident.	datetime DD-MM-YYYY HH:MM:SS	
DTG AB	Time ambulance crew arrives at place for treatment patient. (Hospital)	datetime DD-MM-YYYY HH:MM:SS	
DTG VR	Time ambulance crew is available again.	datetime DD-MM-YYYY HH:MM:SS	
DTG ER	Time ambulance crew ends their drive. (Closure of the journey for reporting, can be back at ambulance station).	datetime DD-MM-YYYY HH:MM:SS	
Van_X_Coord	X Coord of accident (epsg 28992).	float	
Van_Y_Coord	Y Coord of accident (epsg 28992).	float	
Naam Locatie 1	Street name of location accident.	string	
Naam Locatie 2	Alternative name of location accident (if available).	string	
Huisnummer	House number of location accident.	int	
Postcode	Postal code of location accident.	string	
Plaats naam	Name of city or village of location accident.	string	
Naar_X_Coord	X Coord of treatment (epsg 28992).	float	
Naar_Y_Coord	Y Coord of treatment (epsg 28992).	float	
NAAR_NAAM_LOCATIE1	Street name of location treatment.	string	
NAAR_NAAM_LOCATIE2	Alternative name of location treatment (if available).	string	
NAAR_HUIS_PAAL_NR	House number of location treatment.	int	
NAAR_POSTCODE	Postal code of location treatment.	string	
NAAR_PLAATS_NAAM	Name of city or village of location treatment.	string	
Roepnaam eenheid	Unique ID corresponding to ambulance crew.	int	
Voertuigsoort afk	Type of vehicle used by ambulance crew.	string	
Prioriteit CPA	Priority of emergency (A1, A2, B1, B2)	string	
Rit soort afsluiting	Type of emergency.	string	
DoY Melding	Day of year of DTG Melding.	int	
Month Melding	Month of year of DTG Melding.	int	
weights	Assigned weight according to Prioriteit CPA.	float	
latitude	Van_X_Coord converted (epsg 4326).	float	
longitude	Van_Y_Coord converted (epsg 4326).	float	
square_no	Assigned square number according to grid system.	int	

Table 1: Feature Table

9.2 Additional Graphs and Maps



(a) A1 (b) A2

(c) B1 (d) B2

Figure 14: Bubbleplot - Location of emergency per priority level

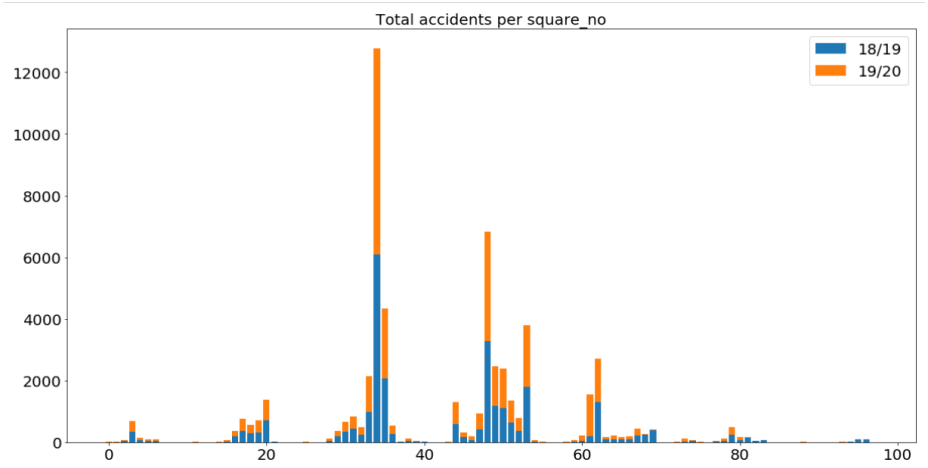


Figure 15: Gridcount - Amount of entries per rectangle

References

Brownlee, J. (2016). *Time series prediction with lstm recurrent neural networks in python with keras.* https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

citypopulation.de. (2020). *Zuid-holland (web).* https://www.citypopulation.de/en/netherlands/admin/NL33__zuid_holland/

Kumar, J., Goomer, R., & Singh, A. K. (2018). Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters [The 6th International Conference on Smart Computing and Communications]. *Procedia Computer Science*, *125*, 676–682. https://doi.org/https://doi.org/10.1016/j.procs.2017.12.087

Olah, C. (2015). *Understanding lstm networks (web).* https://colah.github.io/posts/2015-08-Understanding-LSTMs/

RIVM, & AZN. (2018). *Ambulance-inzetten: Bereik a1-inzetten binnen 15 minuten.* https://www.staatvenz.nl/kerncijfers/ambulance-inzetten-bereik-a1-inzetten-binnen-15-minuten

To see all files, maps, and models see gitlab or email erencantatar@gmail.com. As we do not want to put the data online, in terms of the Non-Disclosure Agreement that was signed.