Predicting Train Occupancies based on Query Logs and External Data Sources*

Gilles Vandewiele
Joachim Van Herwegen
Femke Ongenae
Pieter Colpaert
Ruben Verborgh
Filip De Turck
Olivier Janssens
Erik Mannens

{firstname}.{lastname}@ugent.be
Internet and Data Lab - imec
Ghent University, Belgium

ABSTRACT

On dense railway networks –such as in Belgium– train travelers are frequently confronted with overly occupied trains, especially during peak hours. Crowdedness on trains leads to a deterioration in the quality of service and has a negative impact on the well-being of the passenger. In order to stimulate travelers to consider less crowded trains, the iRail project wants to show an occupancy indicator in their route planning applications by the means of predictive modeling. As there is no official occupancy data available, training data is obtained by crowd-sourcing using the iRail web app[1] and the mobile Railer application for iPhone[2]. Users can indicate their departure & arrival station, at what time they took a train and classify the occupancy of that train into the classes: low, medium or high. While preliminary results on a limited dataset conclude that the models do not yet perform sufficiently well, we are convinced that with further research and a larger amount of data, our predictive model will be able to achieve higher predictive performances. All datasets used in the current research are, for that purpose, made publicly available under an open license on the iRail website[3] and in the form of a Kaggle competition[4]. Moreover, an infrastructure is set up that automatically processes new logs submitted by users in order for our model to continuously learn. Occupancy predictions for future trains are made available through an API[5].

Keywords

linked data; public transport; predictive modeling

1. INTRODUCTION

In Belgium –as well as in other countries with dense railway networks– train travelers are frequently confronted with overly occupied trains. In 2016, the CEO of the national railway company suggested peak-load pricing[6] in the hope that a more uniform load over the course of the day could be achieved. As an alternative, travelers that have the luxury to take a train earlier or later could be informed about the crowdedness of that train. To that extent, the iRail initiative, an independent non-profit project founded by Pieter Colpaert, to stimulate digital creativity concerning mobility in Belgium, wants to introduce a feature that indicates the occupancy of each train in its data feeds.

A system that accurately predicts the occupancy level of a train in the near future can have positive implications as the capacity of that train could be adapted, if possible, according to these predictions. This results on the one hand in a decreased probability of crowdy trains, thus an increase in the quality of service. On the other hand, a decrease in operational costs can be realized by reducing the capacity of trains that are expected to have a low occupancy.

The continuously increasing use of smart cards for automated fare collection offers a unique opportunity to understand passenger behavior at a massive scale. Unfortunately, in Belgium, such an automated system is not yet used. Thus, the Belgian railway company does not have real-time occupancy data at their disposal. IRail can therefore only rely on usage statistics of their API, feedback from their users, as well as other public datasets.

The most popular user agents reusing this API are the Railer App for iPhone, the BeTrains app for Android[7], and the iRail Web app. These are classic route planning applications following a straight-forward user story: a user selects a departure stop, a destination stop and a desired time of

*(Produces the permission block, and copyright information). For use with SIG-ALTERNATE.CLS. Supported by ACM.

[1] https://iRail.be
[2] http://railer.be
[3] http://gtfs.irail.be/nmbs/querylogs
[4] https://inclass.kaggle.com/c/train-occupancy-prediction
[5] https://github.com/GillesVandewiele/SpitsGidsREST

[6] http://www.knack.be/nieuws/belgie/bus-en-treintickets-worden-duurder-in-de-spits/article-normal-525199.html
[7] https://play.google.com/store/apps/details?id=tof.cv.mpp

WWW'17 Companion, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4914-7/17/04.
http://dx.doi.org/10.1145/3041021.3051699

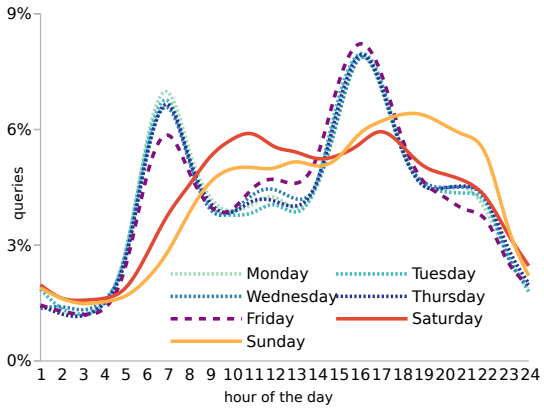

Figure 1: The average distribution of the iRail api query logs per day was the first inspiration to use the query logs as an indication of travel demand.

departure; the app then suggests up to 6 possible itineraries. Other user agents exist, such as Next Train[8] (an app for the Pebble smart-watch), chat bots, data harvesters or search engine bots. As they only represent a minor part of the query logs, they are discarded in this research. Colpaert et al. [4] showed that, when enough query log data is gathered, it looks similar to actual travel demand, as illustrated in Figure 1. In this paper, we go one step further: we try to classify the occupancy level of a train using these query logs and external data sources.

2. RELATED WORK

Tirachini et al. [13] provide many interesting insights concerning the impact of high occupancy levels in public transport systems regarding different dimensions. First, when the occupancy level of a train is low, passenger transfer occurs smoothly and passenger-related disruptions that impose unexpected delays are less likely to happen. As the number of passengers increases, some users need to stand inside vehicles, hindering the movement of other passengers. This in turn results in an increase in riding time or an increase in the probability that an unexpected delay arises. Puong [11] showed that the average boarding time in uncrowded conditions is on average 2.3 seconds per passenger. This increases to 4.4 seconds per passenger when the number of standees per door reaches a threshold of 15 or more. Milkovits [9] showed that this effect is even more significant for the alighting time, explained by the difficulties of alighting passengers walking among too many standees. Second, high occupancy levels can give rise to a phenomenon called train bunching. When a train is too full, not all passengers can board it, leading to an increase in waiting time for these passengers and a higher number of expected passengers for the next train. Finally, crowding has a significant negative impact on the passengers' well-being. Authors have documented increased anxiety, stress, feeling of exhaustion and perceptions of risk to personal safety amongst others [7, 1].

A few private initiatives exist to predict occupancy scores. As an example, the Open Capacity project[9] is a consultancy firm that creates occupancy scores for public transport agencies. It does this by measuring passenger load using existing public transport data sources, such as weight sensors, CCTV cameras, door sensors, and ticketing information.

The Dutch railway system introduced a feature in its app to report the occupancy of a train[10]. Three scores are possible, based on the sentiment of the passenger: positive, neutral and negative. The railway system uses this commuter feedback as a transparent means to research occupancy on trains. When many high occupancies are reported on a certain train, caused by a structural problem, its capacity is increased if possible.

Over the past years, a few research attempts tried to map the occupancy levels of public transport. In Nuzzolo et al., Short Term Occupancy Prediction (STOP) is proposed [10]. STOP is a system that enables predicting the number of passengers on a bus in the nearby future, using available real-time information on smart cards of passengers. The system was evaluated by integrating it in the bus management system of the public transport company in Santander, using data collected in a limited timespan of one day. The system achieved a mean squared error of 3.25, in predicting the number of occupants on a bus, which is equal to a relative root mean squared error of 46% due to the rather small capacity of a bus.

Silva et al. conducted an analysis of data collected by the Transport for London railway system [12]. Hundreds of millions smart-card readings over 70 days from February 2011 till February 2012, containing a time stamp, location code and event code were used to create regression models that predict the number of passengers on a train. In this research, real-time information was used as well, leading to a small increase in the predictive performance when the forecasting horizon decreases. A RMSE of 6.76 and 6.82 are reported, using 5-fold cross-validation, for 1-minute-ahead and 30-minute-ahead forecasts respectively. The authors are also able to quantify the effects of a shock in the system, such as a line segment or station closure.

Zhang et al. [14] collected over 6.5 million records in China by using crowdsensing over a timespan of five months. Crowdsensing is the collection of data through different kinds of sensing devices by a large mass of users. In contrast to crowd-sourcing, which was used to collect our data, the collection of this data happens automatically and requires no human input. They tried to solve two tasks using machine learning techniques: (i) predict whether a certain passenger will take public transport within a given week and (ii) forecast the number of passengers on a bus. For the classification task, F1 scores of around 0.43 are reported. For the regression task, an RMSE of around 25 is reported. Two main contributions were done by this research. First, they showed that weather and semantic trajectory information (such as the number of companies within a certain radius of a station) have a positive impact on the predictive performance of the machine learning model. Second, their results show that the eXtreme Gradient Boosting (XGBoost) algorithm outperforms other prominent algorithms on both tasks.

[8] https://apps.getpebble.com/en_US/application/52cf056321e5796173000081

[9] http://opencapacity.co

[10] http://www.ns.nl/reisinformatie/service-verbeteren/drukte-melden-in-de-trein.html

While the discussed research attempts provide many interesting insights, there are some fundamental differences with our research. First, the exact numbers of passengers was always available and, thus, a regression problem could be solved. Second, the amount of samples used is several orders of magnitude larger than the amount of samples in this research, due to smart card and crowdsensing mechanisms. Third, real-time information about the trains was often used. While this information could indeed be very useful in predicting the occupancy, it no longer enables a railway company to increase or decrease the size of the train, since the train must have already departed for this information to be available. Moreover, this information is not available to predict the occupancy of a train in the future.

3. GATHERING FEEDBACK

First, we ran a questionnaire to retrieve initial trains that would be structurally occupied. The questionnaire was disseminated by the Belgian railway company (SNCB) over Twitter[11] and helped us to gather 334 trains that usually have a high occupancy. With this initial data, the iRail API was extended with two features. The first is an occupancy indicator, which provides the occupancy on the following levels:

- **Low occupancy:** there are plenty of seats left.
- **Medium occupancy:** it is hard to find a seat and it is difficult to sit together.
- **High occupancy:** there are no seats left and people have to stand up.
- **Unknown occupancy:** the occupancy of the train is currently not known.

A second feature introduced in the API is the ability to post feedback. On a specific departure of a train, a user would then be able to specify the occupancy level. This feature, launched in August 2016, was picked up by the Railer App and the iRail.be web app by September, as can be seen in Figure 2. This led to 3818 feedback entries by the 19th of December 2016.

4. PREPARING THE DATASET

Of the 3818 collected records up until the time of writing, 256 contained wrong information (such as wrongly formatted station and vehicle ids) and could not be parsed, resulting in a dataset with a size of 3562 rows. All these records occur in the start of our dataset, and can probably be explained by the fact that the occupancy indicator was still being tested during that time. An occupancy log entry contains the following information:

- **Querytime:** the time at which the record (or log entry) was sent to the system. From this timestamp, many different features such as the *seconds since midnight*, the *day of the week* and the *month* are extracted. Moreover, two binary variables indicating whether or not a *morning or evening jam* is ongoing are used. These variables are equal to one when it is not a day in the weekend and the departure time is from 6 to 10 AM or 3 to 7 PM respectively.

[11] https://twitter.com/NMBS/status/758572996617465856

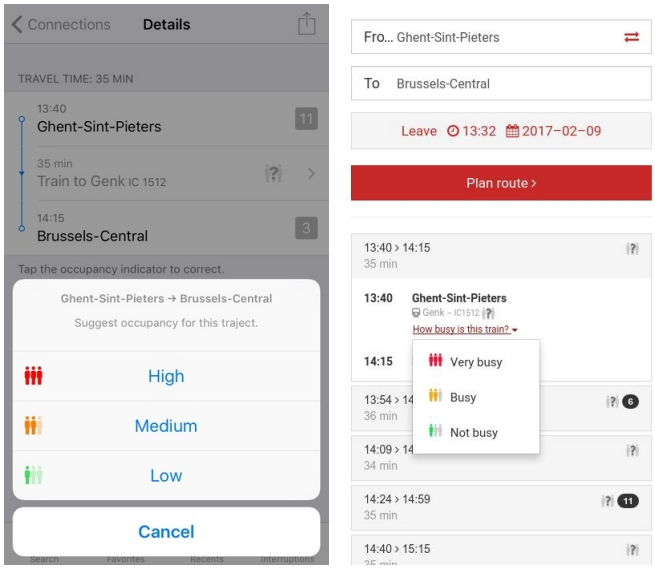

Figure 2: Screenshots of the feedback feature in Railer (left) and iRail (right)

- **Vehicle:** a structured identifier of the train the user is taking. This identifier is composed of three components: the *vehicle type*, the line number (*line category*) and the hour of the departure time from the first station on the line. The departure hour is added to the line number and the vehicle type is prepended. As an example, the IC500 line are the intercity trains going from Oostende to Eupen, while the IC507 train is the train going from Oostende to Eupen at 07:40 AM.
- **From & To:** an identifier of the station from which the user departs or where the user wants to go. It is important to note that it is the departure and arrival location of the user, not the train.
- **Connection:** a URI that links to connection information of that train, such as delay time and the stations where it stops.
- **Occupancy:** the reported occupancy level (low, medium or high). This is the target variable.

All categorical variables are one-hot encoded, which is defined as a mapping of a variable to a binary vector of length equal to the number of categories. All elements in the vector are equal to zero except at the index corresponding to the category of that sample. The from- and to-station identifiers could be one-hot encoded as well but this leads to an explosion of the dimensionality of the features and deterioration of the predictive performance of the model. Therefore, information from two external sources was used. First, a file from iRail with the name, the identifier and the coordinates for each station in Belgium. Second, a static file published by the Société Nationale des Chemins de fer Belges (Belgian railway company, i.e. SNCB) containing the number of passengers visiting a station on a weekday, Saturday or Sunday. This *number of visitors and the coordinates for the from- and to-station* were used as features. Moreover, by using these coordinates, we requested different weather

parameters, such as *the weather type (which required one-hot encoding), the temperature and the humidity*, through an API. A calendar API was used to provide a *holiday type* feature.

To gather additional data, the connection URI –provided in the feedback data– was used. For each sample, we extracted the delay of the train on that departure time and created a vector, with a length equal to the total number of stations in our dataset. For every sample in our dataset and every station in Belgium, we calculated the following function f:

$$f(v,c,s) = \begin{cases} 0, & \text{if v does not stop in s} \\ k, & \text{if v will stop in s in k stations from c} \\ -k, & \text{if v stopped in s, k stations ago from c} \end{cases}$$

With v the vehicle identifier, c the current station identifier and s the station for which we want to calculate whether or not v stops there. As an example, the first three stations of the IC507 from Oostende to Eupen are Oostende, Brugge and Gent-Sint-Pieters. If the log entry is created from Brugge (i.e. the second station on the line), then the indices corresponding to Oostende, Brugge and Gent-Sint-Pieters contain a -2, -1 and 1 respectively. This procedure applied to each sample in the dataset results in a very sparse matrix. Again, to avoid an explosion in dimensionality, and thus a deterioration of the generalization capability of the model, these vectors are not directly used as features. For each station –which are the columns in this sparse matrix– we count its frequency or the number of times it occurs in the matrix, which is equal to the number of elements not equal to zero in the corresponding column in the matrix. Then for each train-ride (or row in the matrix), the *sum of frequencies and a weighted sum of frequencies* are calculated and used as features. For the *weighted sum of frequencies*, we multiply the frequency by the inverse of its element in the matrix. The intuition behind this is that in the morning, a lot of commuters get on the train at smaller stations and alight the train in a larger station. Close to these larger stations, this value becomes large. Since the size of our dataset is still rather small compared to the total number of different lines in Belgium, these features were also calculated with the reported visitors per station from SNCB in 2015 instead of their frequencies in our dataset in order to get a less biased view on the crowdedness of a station.

In total, 1270 features are used in the model, including all binary variables due to one-hot encoding. To measure the quality of a feature, we calculated the feature importances using XGBoost [3], which is also used to create our predictive model. XGBoost calculates the feature importance by counting in how many trees of the constructed forest a certain feature occurs, taking into account the depth of the nodes where the features occur. A bar plot of the 40 most important features and their corresponding value can be found in Figure 3. Here we can clearly see that the number of seconds since midnight of a train departure (i.e. the departure time expressed in seconds) is the most important feature, followed by the calculated frequency features, the number of visitors per day and the humidity in the departure and arrival station. The delay features, which are real-time information and can thus not be used to predict the occupancy of future trains, do not have a significant impact

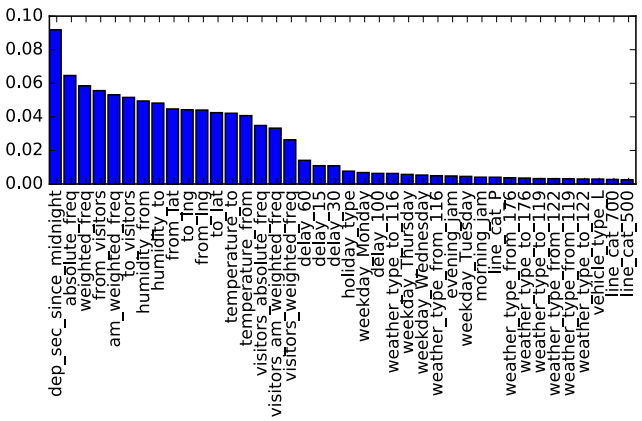

Figure 3: Feature importances of the extracted features.

on the model and can therefore be discarded. We tried to apply two prominent feature selection techniques, Boruta [6] and LASSO [5], but it did not increase the predictive performance of our model.

5. MACHINE LEARNING AND RESULTS

The designed machine learning approach is composed out of multiple steps, depicted in Figure 4. In a first phase, logs that have the same vehicle identifier and from-station identifier on the same day are grouped together. The equality of these three parameters also implies a similar query time, since the departure hour is incorporated in the vehicle identifier. Then, the mode of the labels is calculated. This enables a simplistic form of anomaly detection, as a wrong label can get corrected if more correct labels are given for that train on that day. Moreover, the labels are mapped to an integer where low is equal to 1, medium equal to 3 and high equal to 5 in order to calculate a mean score. Of the 3562 records collected from 1 September 2016 until 19 December 2016, 506 duplicate logs were combined with others, resulting in 3056 samples. For each of these samples, a feature vector was calculated, as explained in Section 4. The feature extraction failed for 25 records, because of external API errors, resulting in a 3032×1270 matrix to train our predictive model on. The predictive model consists of two components. In the first component, a neural network is trained for a regression task using the feature vectors and the calculated mean score. The neural network consists of three inner layers with 750, 250 and 100 neurons and dropouts of 0.33, 0.25 and 0.1 after each layer respectively. Then, the out-of-sample predictions, which are predictions of samples where the model is not trained on, of this neural network are used as an extra feature for our final XGBoost model. Since the XGBoost algorithm contains a lot of different hyper-parameters that can have values in a large range, the search space becomes too large in order to feasibly optimize these hyper-parameters with a brute-force GridSearch technique. Therefore, a Bayesian optimization library, BayesOpt [8], was used, which also supports the optimization of hyper-parameters. Moreover, to deal with the imbalance in our dataset (41% low, 28.6% medium and 30.4% high), more weight was given to medium and high

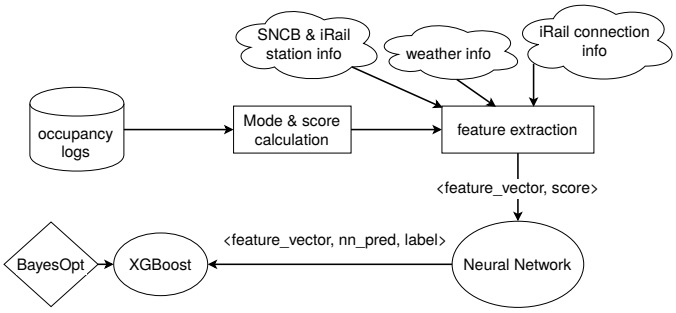

Figure 4: Schematic overview of our machine learning approach.

Folds	Accuracy	Precision	Recall
3	0.52660 ± 0.005	L: 0.621 ± 0.006 M: 0.404 ± 0.011 H: 0.515 ± 0.01	L: 0.633 ± 0.007 M: 0.417 ± 0.013 H: 0.487 ± 0.01
5	0.53279 ± 0.005	L: 0.626 ± 0.006 M: 0.411 ± 0.008 H: 0.525 ± 0.01	L: 0.631 ± 0.008 M: 0.43 ± 0.012 H: 0.496 ± 0.01
10	0.54012 ± 0.005	L: 0.631 ± 0.006 M: 0.423 ± 0.008 H: 0.532 ± 0.009	L: 0.635 ± 0.007 M: 0.444 ± 0.012 H: 0.502 ± 0.008

Table 1: The mean accuracy, precision per class and recall per class with their corresponding standard deviations using 20 measurements of 3-fold, 5-fold and 10-fold cross validation. L, M and H stand for Low, Medium and High respectively.

samples in our dataset. We also experimented with SMOTE [2] to balance our dataset, but it did not increase the precision and recall scores for the lower populated classes that much, while deteriorating the total predictive performance significantly.

In order to evaluate the predictive performance of our model, we measured the mean and standard deviation of the accuracies together with the mean and standard deviation of the precision and recall for each class using the result of twenty trials of 3-fold, 5-fold and 10-cross validation on the 3032×1270 data matrix. The results can be found in Table 1. For completeness, a confusion matrix averaged over these twenty trials is plotted for 10-fold cross-validation in Figure 5. As expected, the predictive performance increases slightly when the number of folds are increased. When we use 10-fold cross validation, we achieve an average accuracy of 54.012%, which is quite low but already better than random guessing or always predicting low occupancy. Moreover, there is a big difference between the precision and recalls of the three classes, even with higher weights for the medium and high occupancy class. The low recall and precision scores for the medium occupancy class could perhaps be explained by the fact that people do not know the definition of the medium occupancy class well and tend to classify the occupancy of a medium-filled train as low or high.

Although the results are not yet what we expected, we are convinced that given more data, these results will improve. This is confirmed by the increasing trend of the learning curve of our model, which can be seen in Figure 6.

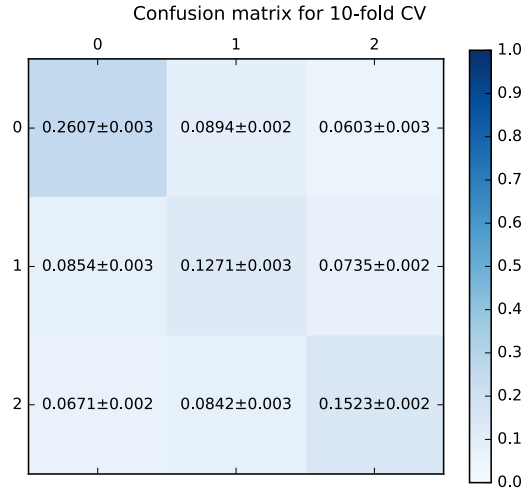

Figure 5: Confusion matrix generated by taking the sum over the 10 folds and then averaging these sums over the 20 measurements.

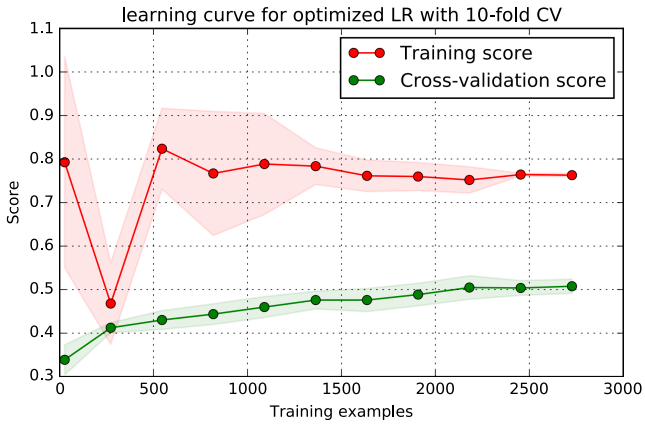

Figure 6: Learning curve using Logistic Regression with optimized hyper-parameters through GridSearch and 10-fold cross-validation.

5.1 Benchmarking

A first effort to create a public benchmark with this data is done by launching a Kaggle competition. A Kaggle competition is a machine learning competition wherein every contestant has to use the same data as any other contestant to create and test their machine learning model. Data from July 2016 till October 2016 serves as the training set and data from the end of October 2016 till the end of December 2016 serves as testing set. The Kaggle competition provides a leaderboard which lists the scores contestants achieved using their approaches. Very often, these contestants gladly share their approaches, such that we can extract all interesting insights and implement them in our system.

5.2 Architecture and Web resources

In order for the predictions generated by our model to be available for everyone and to enable data re-use, an API was set up that allows users to query the occupancy for a certain

train from a station on a certain day. It continuously polls the iRail API to check for new occupancy records. When such a record is found, it is automatically processed and added to our NoSQL MongoDB. Every night, two processes are run. On the one hand our predictive model is re-trained with the newly collected data. On the other hand, the hyper-parameters are tuned using Bayesian optimization. The API is accessible through the following IP: 193.190.127.247

6. CONCLUSION AND FUTURE WORK

In this paper, the first steps towards a system that can predict the occupancy level of a train in the nearby future based on query logs are presented. Such a system can have a significant positive impact on the quality of service while decreasing the operational costs. We discussed the different phases of constructing such a system: (i) adding a functionality to a widely used application in Belgium in order to collect data through crowd-sourcing; (ii) extracting numerical features from these raw JSON logs and (iii) creating a predictive model on this extracted data. Moreover, an API was created in order to expose the predictions of our model and a Kaggle competition was set up to enable collaborative benchmarking.

We conclude that, in this early phase, our predictive model, which is trained on a limited amount of data, is good at predicting trains with a low occupancy. This comes at no surprise, as the low occupancy of trains outside peak hours is easy to predict and as it is the largest populated class (currently, around 41% of all samples have the low occupancy label). When more samples are collected, we are convinced that the system's predictive performance will increase. The strength of the approach in this paper is that the data used can be gathered for any public transport system. At this moment, data has only been collected over a limited timespan. The current dataset thus contains only a limited amount of samples, but is growing steadily with more than 1000 query logs per month.

7. ACKNOWLEDGMENTS

Gilles Vandewiele is funded by a PhD SB fellow scholarship of FWO (1S31417N). Thank you to iRail, TreinTramBus and Metro Time, and crowd-funding supporters for their time and financial effort in the Spitsgids campaign. Thank you to SNCB for the support to gather first data. Thank you Serkan Yildiz, Stan Callewaert and Arne Nys for their enthusiasm implementing the features in the apps during open Summer of code. Thank you Kris Peeters, Nathan Bijnens and other Twitter users who helped discussing the data publicly.

8. REFERENCES

[1] M. Cantwell, B. Caulfield, and M. OâĂŹMahony. Examining the factors that impact public transport commuting satisfaction. *Journal of Public Transportation*, 12(2):1, 2009.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[3] T. Chen and C. Guestrin. 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, 2016.

[4] P. Colpaert, A. Chua, R. Verborgh, E. Mannens, R. Van de Walle, and A. Vande Moere. What public transit api logs tell us about travel flows. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 873–878. International World Wide Web Conferences Steering Committee, 2016.

[5] Y. Kim and J. Kim. Gradient lasso for feature selection. In *Proceedings of the twenty-first international conference on Machine learning*, page 60. ACM, 2004.

[6] M. B. Kursa, A. Jankowski, and W. R. Rudnicki. Boruta–a system for feature selection. *Fundamenta Informaticae*, 101(4):271–285, 2010.

[7] U. Lundberg. Urban commuting: Crowdedness and catecholamine excretion. *Journal of Human Stress*, 2(3):26–32, 1976.

[8] R. Martinez-Cantin. Bayesopt: a bayesian optimization library for nonlinear optimization, experimental design and bandits. *Journal of Machine Learning Research*, 15(1):3735–3739, 2014.

[9] M. Milkovits. Modeling the factors affecting bus stop dwell time: use of automatic passenger counting, automatic fare counting, and automatic vehicle location data. *Transportation Research Record: Journal of the Transportation Research Board*, (2072):125–130, 2008.

[10] A. Nuzzolo, U. Crisalli, L. Rosati, and A. Ibeas. Stop: a short term transit occupancy prediction tool for aptis and real time transit management systems. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pages 1894–1899. IEEE, 2013.

[11] A. Puong. Dwell time model and analysis for the mbta red line. *Massachusetts Institute of Technology Research Memo*, 2000.

[12] R. Silva, S. M. Kang, and E. M. Airoldi. Predicting traffic volumes and estimating the effects of shocks in massive transportation systems. *Proceedings of the National Academy of Sciences*, 112(18):5643–5648, 2015.

[13] A. Tirachini, D. A. Hensher, and J. M. Rose. Crowding in public transport systems: effects on users, operation and implications for the estimation of demand. *Transportation research part A: policy and practice*, 53:36–52, 2013.

[14] N. Zhang, H. Chen, X. Chen, and J. Chen. Forecasting public transit use by crowdsensing and semantic trajectory mining: Case studies. *ISPRS International Journal of Geo-Information*, 5(10):180, 2016.