Towards Station-Level Demand Prediction for Effective Rebalancing in Bike-Sharing Systems

Pierre Hulot
Polytechnique Montréal
Montréal, Canada
pierre.hulot@polymtl.ca

Daniel Aloise
Polytechnique Montréal
Montréal, Canada
daniel.aloise@polymtl.ca

Sanjay Dominik Jena
ESG-UQAM
Montréal, Canada
jena.sanjay-dominik@uqam.ca

ABSTRACT

Bike sharing systems continue gaining worldwide popularity as they offer benefits on various levels, from society to environment. Given that those systems tend to be unbalanced along time, bikes are typically redistributed throughout the day to better meet the demand. Reasonably accurate demand prediction is key to effective redistribution; however, it is has received only little attention in the literature. In this paper, we focus on predicting the hourly demand for demand rentals and returns at each station of the system. The proposed model uses temporal and weather features to predict demand mean and variance. It first extracts the main traffic behaviors from the stations. These simplified behaviors are then predicted and used to perform station-level predictions based on machine learning and statistical inference techniques. We then focus on determining decision intervals, which are often used by bike sharing companies for their online rebalancing operations. Our models are validated on a two-year period of real data from BIXI Montréal. A worst-case analysis suggests that the intervals generated by our models may decrease unsatisfied demands by 30% when compared to the current methodology employed in practice.

CCS CONCEPTS

• **Applied computing** → **Decision analysis**; • **Information systems** → *Uncertainty*; • **Computing methodologies** → *Model development and analysis*;

KEYWORDS

bike sharing systems; traffic modeling; decision support; dimensionality reduction; station-level demand prediction

ACM Reference Format:
Pierre Hulot, Daniel Aloise, and Sanjay Dominik Jena. 2018. Towards Station-Level Demand Prediction for Effective Rebalancing in Bike-Sharing Systems . In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3219873

1 INTRODUCTION

Bike-sharing systems are nowadays used world-wide. Given that the system tends to be unbalanced, particularly at peak demand hours, challenging analytical problems arise such as accurately predicting the demand and how to rebalance the system. Rebalancing has been addressed in the literature, and several approaches have been proposed, mostly using mixed integer programming and constraint programming techniques [3, 8, 11, 23, 25, 27, 31]. To propose effective rebalancing strategies, those methods require accurate demand estimation models. In this work, we aim at filling a gap in the traffic prediction literature for bike sharing systems. Some papers [2–4, 9, 18, 21] use only the estimated traffic, others [1, 25, 29, 31] suppose a Poisson distribution of trips, but most of the time the prediction model is very simple. Bike sharing traffic has also been analyzed, and exogenous factors have been proved to have effect on the demand. Most articles [5, 13–16, 20, 24, 26, 30] show that the traffic largely depends on the month of the year, the day of the week, the hour, the precipitation and the temperature. Some works on network design [5, 14, 16, 20, 26] also suggest that the traffic depends on the neighborhood of the stations, the job density, the business and population densities, the presence of universities, the length of the trip, the available connections, the number of bicycles in service and the sociology of people. Most of these features can only be used for design purposes as they do not change during the year and are included in the station behavior. Finally, some works also focus on effective traffic prediction. However, most of them estimate the traffic of the entire system [5, 15, 16, 22, 32, 33], i.e., the total number of trips per hour in the entire system.

Bike sharing traffic prediction has also been explored at a station level perspective. Cagliero et al. [6] proposed a model that predicts critical states in the network, but the work is biased by the actions of the operator, and the model is not accurate enough to build a robust model on it. Rudloff & Lackner [26] proposed a model for traffic prediction using a linear regression on time and weather features, augmented by network status features. The authors use the neighboring station status to achieve a better precision. However, this feature is not always significant. They also tried to fit a distribution on the stations and found that negative binomial and hurdle models fits the data better than the usual Poisson hypothesis. However, they learned only linear regressions and a model per station, resulting in some overfitting. Yoon et al. [34] propose a model that predicts the evolution of the network in a short period of time (up to 1 hour). A similarity based approach is then proposed to group stations. Their approach is biased by the operator actions because recurrent critical statuses are confused with the lack of demand. Therefore, their work cannot be used for rebalancing. They also predict the network evolution in a very short period of time

(<60min). Vogel et al. [29] propose a cluster-based approach that predicts trips on clusters, and then reallocate them to the stations. However, clustering stations naively approximates station behavior as will be shown in Section 3.1.

The multitude of objectives of papers dealing with the bike-sharing traffic makes the proposed models hardly comparable. Furthermore, the associated works consider different datasets, and hence, scores cannot be directly compared. In our work, we implemented several algorithms proposed in these works (e.g. linear regression, clustering of stations). We adapted them to optimize the objective of this work (i.e., station-level traffic prediction), in particular by removing geographical constraints.

BIXI, the Montreal bike-sharing system, uses an online strategy for rebalancing their system based on inventory intervals. These intervals are manually computed every month and define the acceptable number of bikes at each station for each hour. Once a station does not satisfy this condition, a truck is sent to rebalance it to its target amount of bikes. These intervals and target values are based on BIXI's expected demand in the corresponding time period. In this paper, we propose a traffic prediction model that is able to automate the generation of the inventory intervals, adapting them to different weather conditions.

This paper is organized as follows. Section 2 presents the data from BIXI Montreal together with the description of the features used in our study. Section 3 presents our solution architecture divided into a problem reduction step, a learning step, a reconstruction step and a distribution fitting step. Section 4 presents our methodology for defining inventory intervals for rebalancing operations in bike sharing systems. Section 5 reports the selection of hyperparameters for the methods employed in our model. Section 6 concerns our computational experiments and results. Finally, the last section presents our concluding remarks.

2 DATA

This paper presents a model that makes use of trip history data together with weather data. The first type of data consists of the trip history of BIXI for the period between April 2015 and November 2016 (the Montreal bike sharing system is closed in the winter period). The trip data is aggregated per hour and station into a matrix of dimension (number of hours) $\times$ (2 $\cdot$ number of stations). This matrix represents for each hour (rows) and for each station the number of arrivals (1 column) and departures (1 column) during that hour. Arrivals and departures are considered separately to achieve a more accurate prediction. Thus, for each station, two behaviors are learned independently: the arrival behavior and the departure behavior. The data matrix containing all trips information per hour is noted D, the set of stations S and the set of hours T. The weather data is collected from *données Canada* [7].

2.1 Features

Several features are used for this model. They can be classified into two categories: the time related features and the weather features. The time related features are the year, the month, the day, the week day, the hour, the off days and the holidays. The weather features are composed by the temperature, the humidity, the visibility, the pressure, the wind speed and the weather (rain, cloudy, sunny, showers, snow, etc.). Categorical variables are built for the weather (text feature), hours and the week days. This transformation facilitates to learn nonlinear behaviors on hours and week days. A set of nearly 50 features is then determined for each hour. All these features have been shown to have a significant impact on the traffic [5, 14, 15, 24, 26, 29]. For each hour the set of features corresponding to that specific hour is determined. Gaps in the weather data have been completed using a forward fill process. The small number of big gaps (>1) and the small variation of weather between two hours justify this approximation.

3 MODEL ARCHITECTURE

Our architecture builds a specific model per station. It lies between a fine-grained approach that builds a model per station but tends to overfit the training data and a large grained model that builds a model per cluster and tends to underfit the data being too general. Our model extracts main behaviors of stations and combines them to build a specific model for each station. The model is built in two steps: the first one, the reduction step, extracts behaviors, and the second one, the prediction step, models the different behaviors and is able to forecast them. Figure 1 gives an overview of the two steps.

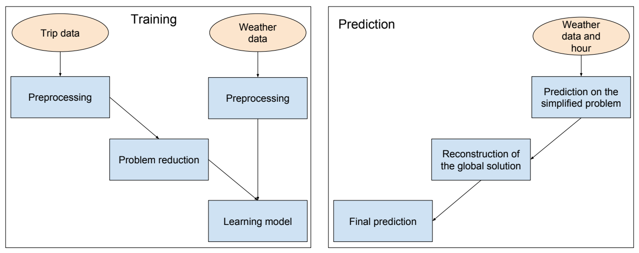

Figure 1: Trip estimation model

The left box of Figure 1 presents the tasks for building a reduced model for predicting the expected traffic, while the right box shows the tasks needed for reconstructing the prediction on the whole non-aggregated network. The model first preprocesses the data. The trip data is then reduced in order to simplify the problem and avoid overfitting. This process keeps the same number of rows while reducing its complexity (columns) from almost a thousand dimensions to just a few. The last part trains a machine learning algorithm to predict on the reduced problem. It uses the reduced matrix as objective and weather and hour data as features.

The right box of Figure 1 indicates how this process is inversed to recover a prediction over the whole bike sharing system. First, it uses weather forecast and hour properties to predict the values of the reduced problem. Then it extrapolates these values inverting the reduction process to recover predictions over the whole network.

Figure 2 presents the remainder of the model. Based on the prediction of the expected traffic and the estimated variance, it builds a statistical model that estimates the probability distribution of the number of trips at each station. This model is then used to design the inventory intervals that satisfy a minimum service level.

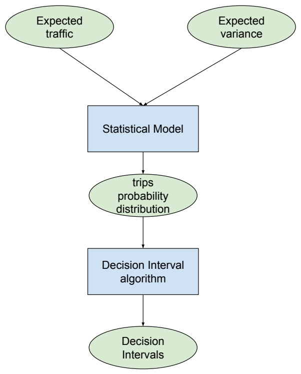

Figure 2: Construction of inventory intervals

3.1 Reduction methods

The reduction method aims to transform the problem from a complex one (about one thousand predictions per hour) to a simpler one. Reducing problem complexity makes the problem more tractable and helps avoiding overfitting. The reduction aims to erase the noise and to keep only important information. Thus, the columns (each representing the number of departures or arrivals in a station) are replaced by some carefully chosen representatives. These new columns represent main behaviors. Explored techniques perform a linear transformation of the data. Then the transformation can be written in the following form:

$$D' = D \cdot R, \tag{1}$$

where D is the data matrix of size (number of hours)×(2 · number of stations), D' is the data matrix after the transformation whose dimension is (number of hours)×(reduction size), and R is the reduction matrix of size (2 · number of stations)×(reduction size). The following techniques are explored:

- **Identity**: No reduction performed; baseline used to compare reduction improvement. The R matrix here is the identity matrix.
- **Sum**: this reduction method groups all stations into one cluster. The resulting data is a vector containing for each hour the average traffic on the network. The R matrix is a vector of ones. It is a very naive method, also a baseline.
- **Kmeans**: Stations are first clustered into k clusters $\{K_i\}_{i\in\{1..k\}}$ using the kmeans method on a aggregated version of the data (data is aggregated per hour of the week). The $R = (r_{i,j})$ matrix represents the assignment of stations to clusters. $r_{i,j} = \frac{1}{|K_j|}$ means that station i has been assigned to cluster j. The number of clusters will be determined in Section 5. Clustering methods are used in most articles modeling the traffic [5, 10, 29].
- **SVD**: The Singular Value Decomposition method maximizes, for a linear method the explained variance with the selected number of dimensions. This method finds the best linear transformation of the data minimizing the loss for a given dimension. This transformation builds two orthogonal matrices O and Q and a diagonal matrix $Diag$ ordered by decreasing values, containing the singular values of D, such that $D = O \cdot Diag \cdot Q^T$. To apply a SVD reduction, the matrix $Diag$ is truncated after the k-th singular value. The R matrix is then composed by the first k columns of Q:

$$R = [Q_1, .., Q_k], \tag{2}$$

with Q_i being the i-th column of Q.

3.2 Prediction methods

Prediction models try to predict the values of the extracted behaviors (output of the reduction). The following prediction methods are explored:

- **Linear regression**: The linear regression is the simplest prediction algorithm for continuous objectives. Some dummy variables (hour and week day) are created to help the learning of non-linear features. Regularized linear regression is also used (Lasso and Ridge regression).
- **Multi-layer perceptron (MLP)**: The non-linearity of the problem incite to use non-linear predictors. The neural network seems fitted for this task. We trained a multilayer perceptron network for this task.
- **Gradient boosted tree (GBT)**: A gradient boosted tree is built for each reduced dimension separately.
- **Random forest (RF)**: A multidimensional random forest is also built.

3.3 Reconstruction into the original data space

Once a prediction has been made on the reduced problem, the station-level predictions need to be reconstructed. For that purpose, the reduction made in Section 3.1 is inverted considering the respective reduction method that has been previously applied. This method is a pseudo inverse because the original data cannot be entirely recovered from the reduced one. As all methods are linear, the reduction inversion can be written as:

$$D'' = D_r \cdot R' \tag{3}$$

where D_r is some reduced data of dimension $l \times k$, D'' is the rebuilt data of dimension $l \times$ (2·number of stations) in the original dimension of D, and R' is a matrix of shape $k \times$ (2·number of stations) representing the inversion method. Let us define $\bar{d}_s$ the average number of trips per hour of station $s \in S$

- **Identity**: This method has no inversion to perform. R' is simply the identity matrix.
- **Sum**: To invert this reduction, trips are distributed to stations proportionally to their average number of trips per hour:

$$R' = \frac{1}{\sum_{i\in 1..n} \bar{d}_i}[\bar{d}_1, ...\bar{d}_n]. \tag{4}$$

- **kmeans**: To invert the Kmeans clustering, the trips of each cluster are distributed to its stations proportionally to their average number of trips. Let us note K_j the set of stations of cluster j. Thus,

$$R'_j = \frac{1}{\sum_{i\in K_j} \bar{d}_i}[\bar{d}_1 \times \mathbf{1}_{1\in K_j}, ...\bar{d}_n \times \mathbf{1}_{n\in K_j}] \tag{5}$$

where R'_j is the j-th row of R'.

- **SVD**: To invert the SVD, the inversion matrix is given by $[Q_1, ..., Q_k]^T$ because of its orthogonality.

$$R' = R^T = [Q_1, ..., Q_k]^T. \tag{6}$$

3.4 Distribution selection

Bike-sharing demand is a stochastic phenomenon. We will therefore model the demand at each station and hour as a probability distribution. The Poisson distribution is the most used probability for such problems. However, Rudloff [26] showed that the Zero inflated distribution and the Negative binomial distributions fit better the data. These three distribution hypotheses have been tested. To determine the second parameter of the zero inflated and Negative Binomial distributions, a variance predictor has been learned. The estimation of the mean and variance of the traffic is used to compute the probability of the number of trips per station and hour.

3.5 Variance estimation

An analysis of the mean estimation residuals has shown a non-constant variance in terms of the hour of the day, temperature, pressure and humidity. Thus, a linear regression is trained on these features to predict the variance for each station each hour. We found out that a linear regression was sufficient for variance estimation as a more complex model is more likely to overfit the data with little perfomance gain.

4 INVENTORY INTERVALS

The model built in previous sections is used to improve BIXI's rebalancing operations. Inventory intervals define the desired inventory level for each of the stations. Thus, whenever a station exits its desired inventory interval, an alert is raised, and the operator sends a truck to rebalance it. The station is then filled to its target value. Currently, these inventory intervals and target values are manually computed by BIXI based on historical demand.

Our approach is inspired by the work of Schuijbroek et al [27], in which the authors define the *service level* as the expected satisfied demand over the expected total demand. The rental and return service levels (Sl_{rent} and Sl_{return}, resp.) of station s with f bikes at time t_0 for a maximum capacity of C_s are given by:

$$Sl_{rent}(f, s, t_0) = \frac{\int_{t_0}^{t_0+T} \mu_s(t)(1 - p_s(f, 0, t))dt}{\int_{t_0}^{t_0+T} \mu_s(t)dt} \tag{7}$$

$$Sl_{return}(f, s, t_0) = \frac{\int_{t_0}^{t_0+T} \lambda_s(t)(1 - p_s(f, C_s, t))dt}{\int_{t_0}^{t_0+T} \lambda_s(t)dt}, \tag{8}$$

where $p_s(f, N, t)$ is the probability that station s has N docked bicycle at time t knowing that there were f bikes at time t_0, and $\mu_s(t)$ (resp. $\lambda_s(t)$) is the expected demand at time t and station s for renting (resp. returning) bicycles.

The proposition of Schuijbroek et al. is modified here to be able to prioritize either returns or rentals, by adding a weight α. The modified service level for each station inventory is computed as:

$$Sl(f, s, t_0) = Min(\alpha Sl_{rent}(f, s, t_0), (1 - \alpha)Sl_{return}(f, s, t_0)) \tag{9}$$

This function is defined between 0 and the capacity of the station. If the operator wants to maximize the number of trips, an α bigger than 0.5 may be chosen, while if he wants to maximize the satisfaction of users, an α smaller than 0.5 should be selected.

The inventory x that maximizes equation (9) for a fixed s and t_0 defines the target value ($T(s, t_0)$).

$$T(s, t_0) = argmax_{x \in \{1..C_s\}}(Sl(x, s, t_0)) \tag{10}$$

Let us define $m(s, t)$ ($M(s, t)$ resp.) the minimum (maximum resp.) service level which is possible at time t at station s.

$$m(s, t) = min_{x \in \{1..C_s\}}(Sl(x, s, t)) \tag{11}$$

$$M(s, t) = max_{x \in \{1..C_s\}}(Sl(x, s, t)) \tag{12}$$

The inventory interval computed for each station should ensure a given minimum service level. Some service levels may not be possible to be achieved at some stations. We therefore define the desired service level to be satisfied at station s and time period t as:

$$c(s, t) = m(s, t) + \beta(M(s, t) - m(s, t)) \tag{13}$$

where $\beta \in [0, 1]$ indicates how exigent the operator is about the network. A $\beta = 1$ means that stations are requested to stay at their best service level value. A $\beta = 0$ means that stations are free to unbalance. Moreover, this definition assures that the value $c(s, t)$ is feasible. The inventory interval for a station s at time t is defined as the set of values having a service level bigger than a minimum threshold $c(s, t)$.

$$I(s, t) = \{x \in \{1..C_s\} | Sl(x, s, t) > c(s, t)\} \tag{14}$$

The monotonicity of $Sl_{rent}(f, s, t_0)$ and $Sl_{return}(f, s, t_0)$ guarantees that the interval values are consecutive. Besides, the definition of $c(s, t)$ ensures that the interval is not empty or full ($I(s, t) = \emptyset$ or $I(s, t) = \{0..C_s\}$) unless desired ($\beta = 0$ or $\beta = 1$).

From the operator point of view, β is used to calibrate the size of the intervals, and hence, the number of alerts, which is a delicate trade-off. The available trucks may not be able to handle too many alerts, while too few alerts result in a suboptimal rebalancing strategy and idle trucks. The parameter can be adjusted during the day to meet the rebalancing capacity.

5 SELECTION OF HYPERPARAMETERS

The methods proposed in the Section 3 make use of several hyperparamters to optimize their results. For the reduction methods, these parameters consist mostly in selecting the best output dimension (number of clusters or number of singular values to keep). For prediction methods, hyperparameters help the predictor to better fit the data.

5.1 Reduction

To select the dimensionality of the reduced data we define the reconstruction loss. We denote *red* the reduction method and *inv_red* its pseudo inverse. The reconstruction loss (RL) is defined as:

$$RL = mean((D - inv\_red(red(D)))^2) \tag{15}$$

In the linear case, using our previous notation, this is equivalent to:

$$RL = mean((D - D \cdot R \cdot R')^2) \tag{16}$$

The mean operation consists in computing the overall mean of the elements of the matrix. To select the reduction dimension, we tried to minimize the reconstruction loss while minimizing the dimension. Figure 3 presents the loss of the SVD and kmeans reduction methods as a function of the size of the reduced dimension.

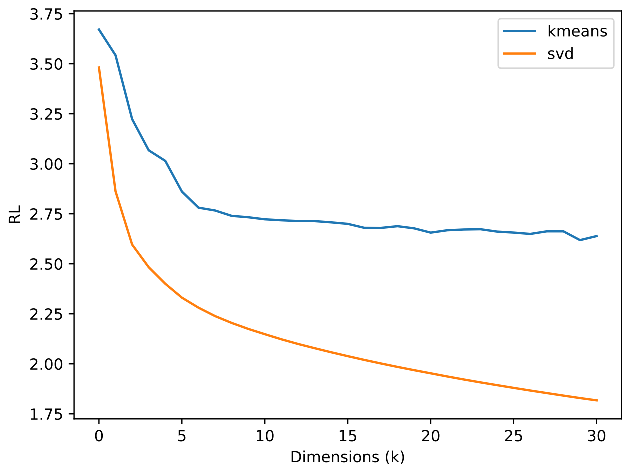

Figure 3: Reconstruction loss per number of dimensions used

5.2 Prediction

Prediction algorithms have specific hyperparameters to optimize the learning algorithms. These hyperparameters were optimized using a grid search.

- **Linear regression**: Regularized linear regression have one hyperparameter that values the weight penalization. Its optimal values were found for alpha = 0.1 (Lasso) and 0.2 (Ridge)
- **MLP**: The multi-layer perceptron has been optimized in terms of the number and the size of layers, dropout value [28], regularization, skip connections [17], batch normalization [19], activation functions, and learning rate. The adopted neural network had: 3 layers of 60, 30 and 15 neurons, a learning rate of 0.01, a tanh activation function, a dropout of 0.6 and a batch normalization after each layer. No regularization was applied. A Nadam optimizer [12] is used.
- **GBT**: The gradient boosted tree was optimized in terms of number of estimators, learning rate and max depth. A minimum number of samples per split did not improve the performance. The final GBT had 130 estimators, a learning rate of 0.1 and a max depth of 5.
- **RF**: the random forest was optimized in terms of the number of trees, and the minimum number of samples per split. The best configuration found had 100 trees and a minimum of 30 samples per split.

6 RESULTS

This section presents an analysis of each model. The BIXI data was split into three parts, the training data, representing the trips from 01/01/2015 to 30/07/2016, the validation data from 01/08/2016 to 31/09/2016 and the testing data from 01/10/2016 to 31/11/2016.

The model was implemented using Python, and machine learning libraries (numpy, pandas, scipy, keras). The experiments were executed in a Windows 10 platform with a Intel core i7-6700HQ, 2.6GHz, 4 cores, and 8 processors.

6.1 Reduction

The simplest way to compare reduction methods is the reconstruction loss (equation 15). This metric is able to compare them without building a complete model. This metric gives a first feedback on the performance of reduction methods. Table 1 presents the reconstruction loss (*RL*) for the selected reduction methods and the output dimension. The identity has a perfect *RL* score, by definition, but keeps all dimensions, imposing a large and complex prediction model. The sum algorithm loses most of stations information as expected since the specificities of the stations are neglected. Kmeans is able to conserve most of the data with only 10 dimensions. The SVD scores slightly better. In fact, only three dimensions are sufficient for the SVD to score as good as the kmeans reduction.

Table 1: Reconstruction loss for each reduction technique

algorithm	dimension	RL
Identity	1062	0
Sum	1	6.36
kmeans	6	3.01
kmeans	10	2.97
SVD	6	2.50
SVD	10	2.34

Figure 4 presents the first components of the SVD aggregated in one week starting on Monday. Each station traffic behavior is expressed as a linear combination of these components. This figure shows the difference between week ends (first and last days) and working days. The first component (blue) displays the average traffic in the network. The second one (orange) balances traffic between morning and evening and the third (green) is able to resize peak hours and weekends. The remaining components are harder to interpret, but main behaviors are already visible on these first three components.

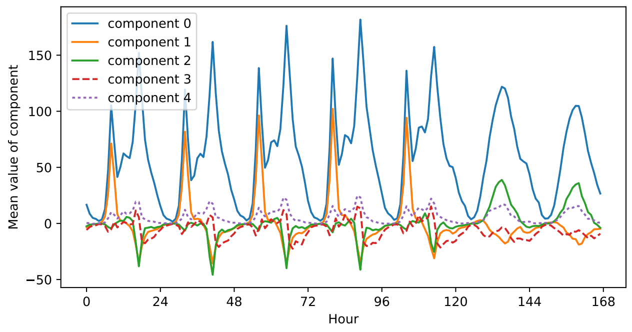

Figure 4: First components of the SVD averaged over one week (168 hours)

6.2 Mean and Variance prediction

We first analyze the precision of each mean estimation model. Several measures were used to compare them, but only the R^2 score and the *RMSE* were kept since the *RMSLE*, *MAE*, *MAPE* had very strong correlation with the *RMSE*, and did not add valuable insights. Table 2 shows the precision results of several mean estimation models. Models are named from the combination of the reduction and prediction methods used. A *RMSE* of 1.394 has been achieved on the test data. The best algorithm used a SVD reduction and a Gradient Boosted Tree predictor. This model outperforms all trained models in terms of prediction performance. We also notice that the SVD reduction speeds up the training time spent by the GBT algorithm by a factor of approximately 100. The random forest (RF) predictor was also able to achieve very good scores. The neural network (MLP) was able to correctly learn the data. However, it was constantly outperformed by the GBT.

Table 2: Prediction scores of different algorithms

Algorithm	*RMSE*	R^2	training time (s)
SVD, GBT	1.394	0.597	22.4
SVD, RF	1.420	0.582	16.2
kmeans, GBT	1.479	0.547	26.8
kmeans, RF	1.446	0.567	16.2
SVD, MLP	1.564	0.493	172.8
id, GBT	1.466	0.555	2241.4
sum, GBT	1.986	0.183	16.7

In order to refine the SVD and kmeans hyperparameters, we built several models with a GBT predictor and a variable number of reduced dimensions. Figure 5 presents the evolution of the precision (*RMSE*) for different numbers of reduced dimensions for the kmeans and SVD methods. A plateau is visible after 10 dimensions for both the SVD and the kmeans reduction methods. This graph shows that the information contained in the remaining dimensions is mostly noise and does not help to build a better predictor. Thus, the reduction method acts as a denoiser of the data. Table 3 shows the best *RMSE* score achieved with each reduction method while fixing the GBT predictor.

Table 3: *RMSE* for each reduction technique

algorithm	dimension	$best_{RMSE}$
Identity	1062	1.466
Sum	1	1.986
kmeans	6	1.494
kmeans	10	1.446
SVD	6	1.452
SVD	10	1.420

For the variance estimation, Table 4 compares the error of the linear regression to the error made by the unbiased variance estimator (constant mean) in terms of *RMSE*, concerning the SVD,GBT algorithm. A clear improvement is observed with the use of the linear regression.

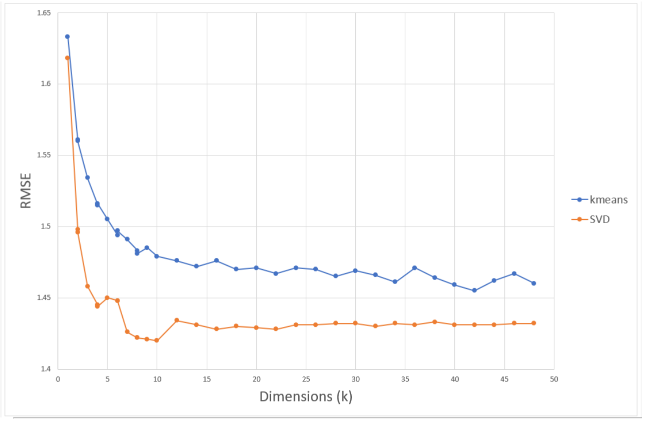

Figure 5: R2 score for several reduction dimensions of kmeans and SVD using a GBT predictor

Table 4: RMSE of variance predictors

algorithm	*RMSE*
unbiased estimator	3.608
Linear Regression	2.901

6.3 Precision per station size

In order to get a better understanding of the model, we analyze the dependency between the station size and the model precision. We define the size of a station as the average number of trips per hour. We computed the *MSE* for each station using a SVD-GBT model. Figure 6 plots this measure per station size. A clear linear dependency is visible in the graph. The orange line corresponds to the line $y = x$. This line corresponds to the theoretical Poisson error (variance of the data). The figure shows that the precision decreases when the station size increases, which is also true for Poisson processes.

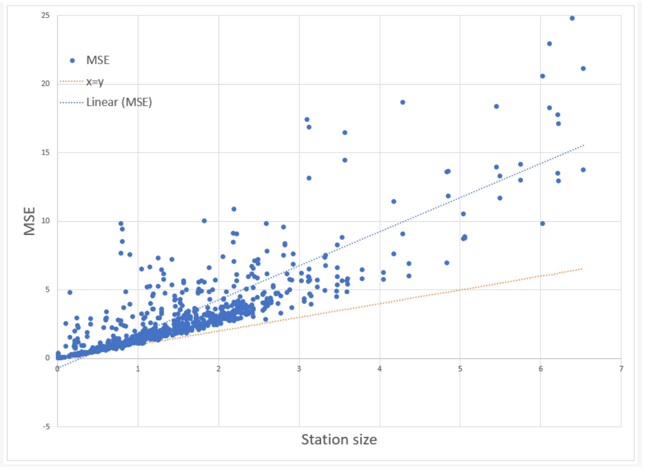

Figure 6: *MSE* score by station size using a SVD-GBT model

Figure 7 plots the R^2 score per station size. This time the score improves as the station size increases (the bigger R^2 the better). The R^2 score represents the proportion of explained variance. Hence, even if bigger stations have bigger MSE errors, the model explains more of the data variance.

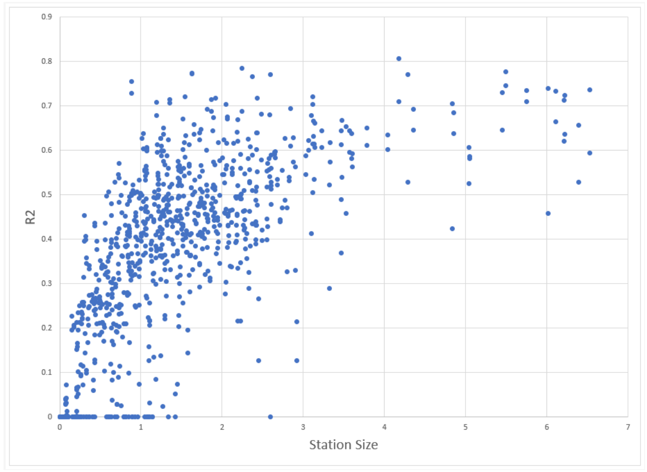

Figure 7: $R2$ score by station size using a SVD-GBT model

6.4 Distribution Choice

Once the expected mean and variance of the traffic is predicted, our model fits a distribution on the station data. To measure the goodness of fit, we compute the average log likelihood for each algorithm and distribution hypotheses. A selection of them is presented in Table 5. The Poisson distribution gives the best results, being the best distribution for 85% of stations. However, we remarked that small stations (approx. 0.27 trips per hour) are better fit by zero inflated distributions. This conclusion is supported by the work of Rudloff [26], which suggests that these distributions work better on the Vienna network, composed by small stations. The Poisson distribution hypothesis is taken for all stations as the difference is not significant.

Table 5: Log likelihood of best models

mean estimation model	distribution	Log likelihood
SVD GBT	Poisson	-1.13
SVD GBT	NB	-1.35
SVD GBT	ZI	-inf
kmeans GBT	Poisson	-1.13
kmeans GBT	NB	-1.26
kmeans GBT	ZI	-1.39

6.5 Performance of generated inventory intervals

Our inventory intervals were evaluated through two worst-case analyses. The first one computes the number of lost bike rentals (departures) if all stations were at their lowest inventory level, inside the tested interval, at the beginning of each period. The second one measures the number of lost bike returns (arrivals), using the same approach, but assuming that stations are at their largest inventory level inside the interval. We define:

$$max_I(s,t) = max(i \in I(s,t)) \quad (17)$$

$$min_I(s,t) = min(i \in I(s,t)) \quad (18)$$

The number of departures (resp. arrivals) during period t and station s is noted $d(s,t)$ (resp. $a(s,t)$). Thus, the average number of lost departures and arrivals are computed as:

$$lost\_dep = mean_{s \in S, t \in T} max(0, (d(s,t) - min_I(s,t))) \quad (19)$$

$$lost\_arr = mean_{s \in S, t \in T} max(0, (a(s,t) - (C_s - max_I(s,t)))) \quad (20)$$

The average of these two numbers defines the *lost trips* score in the worst case analysis. These two scores can also be computed replacing $min_I(s,t)$ and $max_I(s,t)$ by $T(s,t)$ (target value). This new score is called *lost target*.

These two scores are highly dependent on the size of intervals: thinner intervals will get better scores. So, we also compute a third score that counts the average number of rebalancing operations needed to keep the network balanced (i.e. by keeping the inventory level inside the inventory intervals). This last score is computed by running a simulation with real data traffic from two months, counting the number of rebalancing operations performed in the system. This number is averaged per week and called *number of alerts*.

Table 6 presents the scores for BIXI based on the real intervals used by the operator (non-public data provided by BIXI), and for intervals generated by our solution architecture using a SVD reduction, a GBT prediction and a Poisson distribution. This score is computed for $\alpha = 0.5$ (balance between arrivals and departures) and different values of β. The table shows that our inventory intervals outperfoms those used by BIXI. Using $\beta = 0.5$, the number of lost trips is similar to those of BIXI, but our intervals require r less rebalancing operations. Using $\beta = 0.7$, the generated intervals result in the same number of rebalancing operations as BIXI, but our intervals satisfy more trip demand.

Table 6: Inventory interval scores

model	BIXI	SVD GBT P	SVD GBT P
β	-	0.5	0.7
lost departures	2.8	2.11	1.50
lost arrivals	2.55	2.93	2.31
lost trips	2.68	2.51	**1.9**
lost target	**0.55**	0.68	0.68
mean alerts	688	417	687
mean interval size	12.8	16.71	**14.35**

Figure 8 shows the evolution of the *lost trips* score and the number of alerts as a function of β. These graphs provide an understanding of the tradeoff between the exigence of the operator regarding the quality of the service and the provided service level. The horizontal dotted lines correspond to the scores achieved by the intervals used by BIXI. The first graph shows that the number of lost trips decreases with β. The second graph shows that, at $\beta = 0.7$, the number of alerts are about the same. At the same β value in the first graph, one can see that our generated intervals result in smaller demand losses than the intervals generated by BIXI.

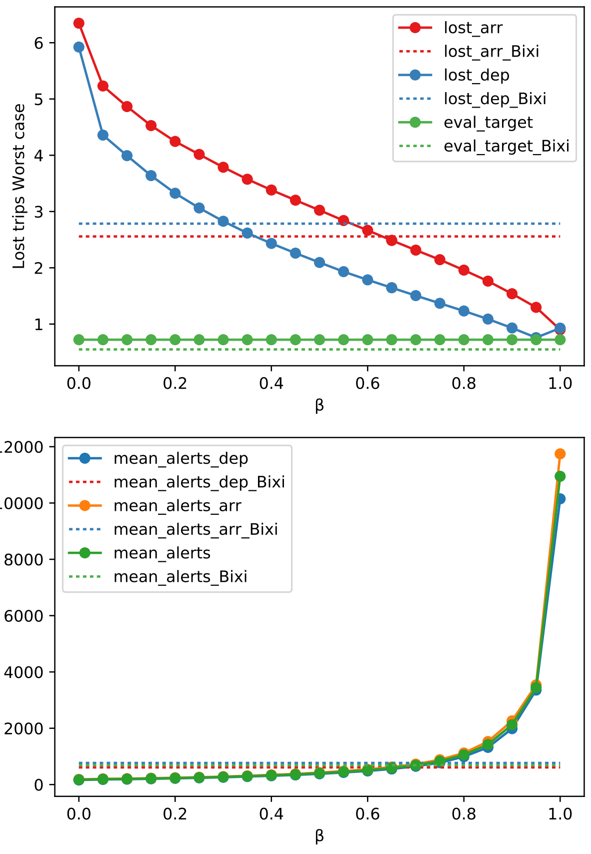

Figure 8: evolution of *lost trips* score (left) and alert numbers (right) with β

The first graph of Figure 9 shows the evolution of *lost departures* (blue) and *lost arrivals* (red) with α. The second graph shows the evolution of the number of alerts. The figure shows that an α of 0.5 yields approximately the same amount of losses in departure and arrivals. It also shows that to improve slightly the performance in terms or departures (resp. arrivals), the performance in terms of arrivals (resp. departures) will be greatly decreased. Operators are then encouraged to value departures as much as arrivals. The intersection point for the two curves illustrated in Figure 9 is at $\alpha = 0.45$. This point should theoretically be at $\alpha = 0.5$. This deviation may be caused by a bias in the test data. Finally, Table 7 shows that for an α of 0.45 and a β of 0.6, our model is able to build inventory intervals with comparable service levels for departures and arrivals, without deteriorating the *lost trip* score too much. Only the *lost targets* score is worsened.

Our worst case analysis on the inventory intervals revealed that the intervals built with our traffic model are significantly more robust than those provided by the bike-sharing company, losing significantly less trips (-30%) than BIXI intervals on the test period of our study. These inventory intervals can also be adapted to the operator needs in terms of rebalancing capacities and tradeoff between departures and arrivals importance. Regarding the restrictiveness of the desired service levels and the quality of the service provided, ideal values of β and α can be easily chosen by analyzing the graphs shown above.

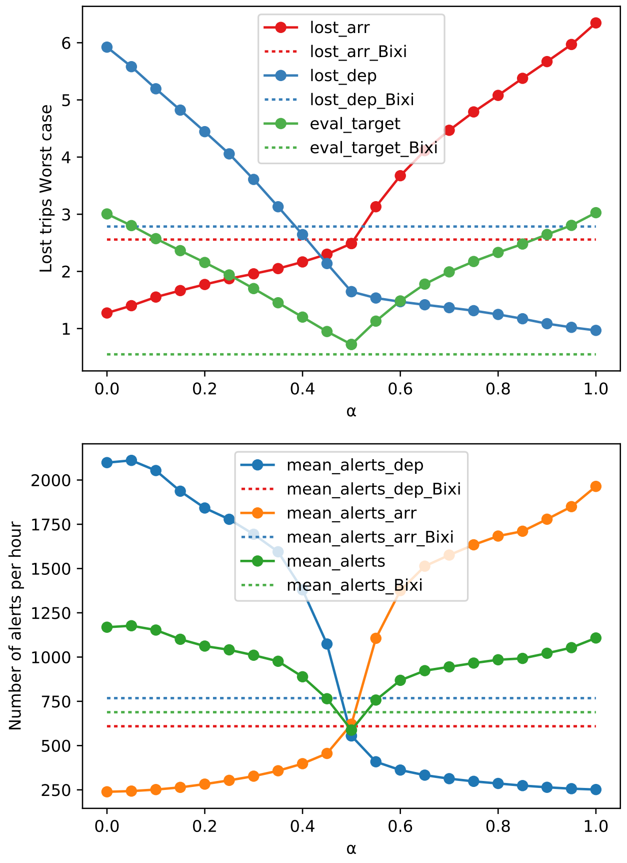

Figure 9: evolution of *lost departures* and *lost arrivals* scores with α

Table 7: Inventory interval scores

model	BIXI	SVD GBT P	SVD GBT P
β	-	0.7	0.6
α	-	0.5	0.45
lost departures	2.8	1.5	2.27
lost arrivals	2.55	2.3	2.36
lost trips	2.68	1.9	2.31
lost target	**0.55**	0.68	0.94
mean alerts dep	609	648	926
mean alerts arr	768	726	394
mean alerts	688	687	660
mean interval size	12.8	14.35	**14.97**

7 CONCLUSIONS

In this paper we proposed a model able to predict station-level traffic in a bike sharing system in order to define inventory intervals for online rebalancing of the system. We compared several approaches for demand prediction and proposed a dimension reduction technique to explore station similarities, while keeping station specificities. Our solution approach was able to effectively reduce the complexity of the prediction problem, reducing computing time and memory requirements, while increasing prediction accuracy.

The model was trained on trip data from the Montreal bike sharing system using time and weather related features. As a direct result, a methodology for the construction of inventory intervals was devised using the learned traffic model. A worst-case analysis was performed and showed that the proposed intervals performed 30% better than the intervals used by the Montreal bike sharing system in the same time period, while conserving the same number of rebalancing operations.

Deviation in terms of arrivals and departures balancing can provide some insight on the network natural bias. The deviation observed in the paper might be explained by relief differences in Montreal. Users may tend to go to work by bike, but return via other public transportation modes. Then, they tend to depart from uncongested stations (living areas) to arrive on congested ones (working areas) introducing more pressure on downtown stations. Our proposed approach also allows the operator to build demand intervals that prioritize lost arrivals over lost departures, and vice-versa, therefore compensating for this kind of natural unbalance.

ACKNOWLEDGMENTS

The authors would like to thank BIXI Montreal, especially Nicolas Blain and Antoine Giraud, for their data and their feedback. This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 2017-05224 and 2017-05617, and by the Canada Excellence Research Chair in Data Science for Real-Time Decision-Making.

REFERENCES

[1] Ramon Alvarez-Valdes, Jose M Belenguer, Enrique Benavent, Jose D Bermudez, Facundo Muñoz, Enriqueta Vercher, and Francisco Verdejo. 2016. Optimizing the level of service quality of a bike-sharing system. *Omega* 62 (2016), 163–175.
[2] Shoshana Anily and Refael Hassin. 1992. The swapping problem. *Networks* 22, 4 (1992), 419–433.
[3] Mike Benchimol, Pascal Benchimol, Benoît Chappert, Arnaud De La Taille, Fabien Laroche, Frédéric Meunier, and Ludovic Robinet. 2011. Balancing the stations of a self service "bike hire" system. *RAIRO-Operations Research* 45, 1 (2011), 37–61.
[4] Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. 2007. Static pickup and delivery problems: A classification scheme and survey. *TOP* 15, 1 (2007), 1–31.
[5] Pierre Borgnat, Patrice Abry, Patrick Flandrin, Céline Robardet, Jean-Baptiste Rouquier, and Eric Fleury. 2011. Shared bicycles in a city: A signal processing and data analysis perspective. *Advances in Complex Systems* 14, 03 (2011), 415–438.
[6] Luca Cagliero, Tania Cerquitelli, Silvia Chiusano, Paolo Garza, and Xin Xiao. 2017. Predicting critical conditions in bicycle sharing systems. *Computing* 99, 1 (2017), 39–57.
[7] Donnée Canada. 2018. Hourly meteorological data. http://climat.meteo.gc.ca/historical_data/search_historic_data_f.html
[8] D Chemla. 2012. *Algorithms for optimized shared transport systems.* Thesis.
[9] Daniel Chemla, Frédéric Meunier, and Roberto Wolfler Calvo. 2013. Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization* 10, 2 (2013), 120–146.
[10] Longbiao Chen, Daqing Zhang, Leye Wang, Dingqi Yang, Xiaojuan Ma, Shijian Li, Zhaohui Wu, Gang Pan, Thi-Mai-Trang Nguyen, and Jérémie Jakubowicz. 2016. Dynamic cluster-based over-demand prediction in bike sharing systems. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing.* ACM, 841–852.
[11] Claudio Contardo, Catherine Morency, and Louis-Martin Rousseau. 2012. *Balancing a dynamic public bike-sharing system.* Vol. 4. Cirrelt Montreal.
[12] Timothy Dozat. 2016. Incorporating nesterov momentum into adam. (2016).
[13] Come Etienne and Oukhellou Latifa. 2014. Model-based count series clustering for bike sharing system usage mining: a case study with the Vélib system of Paris. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 3 (2014), 39.
[14] Ahmadreza Faghih-Imani and Naveen Eluru. 2016. Incorporating the impact of spatio-temporal interactions on bicycle sharing system demand: A case study of New York CitiBike system. *Journal of Transport Geography* 54 (2016), 218–227.
[15] Kyle Gebhart and Robert B Noland. 2014. The impact of weather conditions on bikeshare trips in Washington, DC. *Transportation* 41, 6 (2014), 1205–1225.
[16] Robert C Hampshire and Lavanya Marla. 2012. An analysis of bike sharing usage: Explaining trip generation and attraction from observed demand. In *91st Annual meeting of the transportation research board, Washington, DC.* 12–2099.
[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 770–778.
[18] Hipólito Hernandez-Pérez and Juan-José Salazar-González. 2004. A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics* 145, 1 (2004), 126–139.
[19] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning.* 448–456.
[20] Neal Lathia, Saniul Ahmed, and Licia Capra. 2012. Measuring the impact of opening the London shared bicycle scheme to casual users. *Transportation research part C: emerging technologies* 22 (2012), 88–102.
[21] Andréanne Leduc. 2013. *Modèle d'optimisation de la redistribution des vélos d'un systeme de vélopartage.* Ph.D. Thesis. École Polytechnique de Montréal.
[22] Yexin Li, Yu Zheng, Huichu Zhang, and Lei Chen. 2015. Traffic prediction in a bike-sharing system. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems.* ACM, 33.
[23] Meghna Lowalekar, Pradeep Varakantham, Supriyo Ghosh, Sanjay Dominik Jena, and Patrick Jaillet. 2017. Online Repositioning in Bike Sharing Systems. (2017).
[24] Mohamed Salah Mahmoud, Wafic El-Assi, and K Nurul Habib. 2017. Effects of built environment and weather on bike sharing demand: Station level analysis of commercial bike sharing in Toronto. In *94th Transportation Research Board Annual Meeting, Washington, DC.*
[25] Tal Raviv and Ofer Kolka. 2013. Optimal inventory management of a bike-sharing station. *IIE Transactions* 45, 10 (2013), 1077–1093.
[26] Christian Rudloff and Bettina Lackner. 2014. Modeling demand for bikesharing systems: neighboring stations as source for demand and reason for structural breaks. *Transportation Research Record: Journal of the Transportation Research Board* 2430 (2014), 1–11.
[27] Jasper Schuijbroek, Robert C Hampshire, and W-J Van Hoeve. 2017. Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research* 257, 3 (2017), 992–1004.
[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[29] Patrick Vogel, Jan F Ehmkeb, and Dirk C Mattfelda. 2016. *Service network design of bike sharing systems.* Springer, 113–135.
[30] Patrick Vogel, Torsten Greiser, and Dirk Christian Mattfeld. 2011. Understanding bike-sharing systems using data mining: Exploring activity patterns. *Procedia-Social and Behavioral Sciences* 20 (2011), 514–523.
[31] Patrick Vogel and Dirk Christian Mattfeld. 2010. Modeling of repositioning activities in bike-sharing systems. In *World conference on transport research (WCTR).*
[32] Wen Wang. 2016. Forecasting Bike Rental Demand Using New York Citi Bike Data. (2016).
[33] Yu-Chun Yin, Chi-Shuen Lee, and Yu-Po Wong. 2012. Demand Prediction of Bicycle Sharing Systems.
[34] Ji Won Yoon, Fabio Pinelli, and Francesco Calabrese. 2012. Cityride: a predictive bike sharing journey advisor. In *2012 IEEE 13th International Conference on Mobile Data Management (MDM).* 306–311.