Broken Bikes Detection Using CitiBike Bikeshare System Open Data

Rémi Delassus*†, Romain Giot†, Raphael Cherrier*, Gabriele Barbieri * and Guy Mélançon†
*QUCIT, parc Newton, 213 cours Victor Hugo, 33130 Bègles, France
†Univ. Bordeaux, CNRS, LaBRI, UMR 5800, F-33400 Talence, France

***Abstract*—It seems necessary to detect a broken bike rooted at a station in near realtime as the number of bikes within bikeshare systems has reached more than a million in 2015. Indeed, a bike that cannot be moved is not cost effective in terms of number of trips. This brings frustration to users who were expecting to find a bike at that station without knowing that it is actually defective. We thus propose a methodology from feature extraction to anomaly detection on a distributed cloud infrastructure in order to detect bicycles requiring a repair. Through a first step of K-means clustering, and a second step consisting of spotting samples that do not clearly belong to any cluster, we separate anomalies from normal behaviors. The proposal is validated on a publicly available dataset provided by Motivate, the operator of the New-York bikeshare system. The number of distinct bikes that have been classified by this algorithm as broken at least once during a month is close to the number of repairs given in monthly reports of Motivate.**

I. Introduction

Bikeshare systems generally refer to a set of stations, maintained by an operator, within a city in which bikes can be picked up and dropped off. As of 2015, more than one million bikes have been shared within more than 700 bikeshare systems worldwide [1]. Each one of those bikes has been heavily used, traveling up to an average of ten times a day in Barcelona as reported by Bikesharing Napoli [2]. The high number of bikes as well as the high usage level make breakdowns very likely to happen. Hence, many bikes are broken, waiting to be repaired, immobilized in a station.

This is so common that bikeshare users have picked up the habit of turning the saddle backwards when encountering a broken bike, thus signaling to others that it is not worth renting [3]. Thanks to this personal investment, other users avoid waisting time on renting and returning unusable bikes.

In the city of New York, although CitiBike's maintenance staff regularly performs roundtrips to inspect each bike's state, an alert system has been setup. Each dock has a button that can be pressed in order to alert that the docked bike is broken [3]. The need for such a system proves that rounds are not efficient enough and operators are aware of it.

Bikes are repaired on site whenever possible (for example when the tire is deflated). Otherwise, the maintenance staff picks up the faulty bike and brings it to a specialized workshop that will fix it.

In a previous work, we have shown that as a bikeshare system sees its number of stations doubled, the number of bike trips is multiplied by three [4]. Therefore the number of trips in an expanding bikeshare system grows fast, implying that the number of failures and the number of users wanting to rent a bike will also quickly grow. This leads to a greater number of frustrated users that are in a situation where the only available bikes are broken.

To overcome this problem, we propose a model to classify, for each day, the status of a bike, deciding if it needs repairing or not. Our classification would let maintenance staff know whether a bike needs to be repaired so they could optimize their rounds. Moreover, it could be used within mobile applications where users can plan their trips, displaying the number of bikes not only available, but also usable.

The originality of our study relies on the fact that it is bike centered. This means for example that if a broken bike stays immobile a long time in a station, it is more relevant to study the fact that the bike is not moving rather than the fact that the station is not emptying. In order to follow each bike we work with the New York trips data [5] shared by Motivate, the company operating the CitiBike bikeshare system. Most of bikeshare systems operators have a website displaying the number of bikes and the number of free docks available at every station, which is used by the users to plan their trips. Few operators share their trips data (station and time of departure and arrival of identified bikes) and none of them share their maintenance data. Motivate exposes nearly three years of trips [5], along with monthly reports stating, among others informations, the number of repaired bikes in their workshops [6]. We propose a machine learning process to detect broken bikes. First we extract features from the trips dataset, and then perform an anomaly detection on the extracted features. At that point we use the data from the monthly reports as labels. Those labels are an aggregation of the repairs over a month. Since there is not a one-to-one correspondence between the features and the labels, but a many-to-one correspondence between the features and aggregated labels, we face a problem of Aggregated Output Learning [7].

This paper is organized as follows. Section II presents the existing work in anomaly detection applied to bikeshare systems. Section III describes the proposed method, including the feature extraction and the anomaly detection modules. Section IV depicts the protocol, such as the parameters selection and the infrastructure used. Finally sections V and VI respectively details the results of our method and discusses its limits.

II. Existing Work

Although Bikeshare systems have been heavily studied [1], there is a lack of work related to anomaly detection. In an example of such a proposition [8] Kaspi *et al.* use the same trips we are using to simulate the behavior of a station. This study is thus station centered. A stochastic model is then described. This model is parametrized by the probability of failure of bikes which must be known *a priori*. It is supposed that users do not rent bikes that are broken, which is only true if there are other possibilities. This model has not been confronted to the reality so we cannot know if the assumptions made to make a model were detrimental.

Bertens *et al.* defined a co-occurrence anomaly [9] as two normal events occurring together where they are usually exclusive. They tried to detect such anomalies in a bikeshare system. They used rental data as well as weather data. Unfortunately, the only co-occurrence anomaly they found was a very rare co-occurrence of low real temperature and high perceived temperature.

As far as we know, no machine learning algorithm has been applied to bikeshare systems with the intent of detecting broken bikes, which is the focus of our work. One explanation for this lack of studies is that no label is available to train machine learning models. There is no operator that shares detailed maintenance data, such as the dates and nature of reparations of each bike.

However some operators, such as Motivate, share the number of repairs that occur each month. Musicant *et al.* [7] described machine learning models that can use such an aggregated output to be trained instead of usual labels.

Also, Fanaee *et al.* [10] explain how without any prior label, they were able to label events with bikeshare data. But the events labeled did not occur within the bikesare system. The bikeshare system is rather used as an event detector and background knowledge retrieved from Google is used to provide labels.

III. Proposed Method

The method can be applied in two independent steps. The first step consists to extract features from the raw data, while the next one uses this enhanced and reshaped data to detect anomalies. A diagram recapitulating the process is presented in Figure 1.

A. Features extraction

The features extraction step is strongly linked to the available source of information. So far of our knowledge, only Motivate proposes data that could be used for fault detection. This data is an aggregation of the desired labels, which leads us an Aggregate Output Classification problem.

1) Raw data: We have at our disposal two kinds of data.

(i) The first one is massive and accurate; it describes the trips done with the CitiBike bikeshare system which can be downloaded on CitiBike's website [5]. Each trip is characterized by heterogeneous data described in Table I. We refer to that table as the trips dataset.

(ii) The second one is rather small and imprecise. It consists of a set of monthly reports [6] describing the activities of both the operator and the users during the month. It provides various informations such as the average number of daily trips per bike, the number of users, the respect of the service level agreement and, the one we use to optimize our model, the number of bicycle repairs. We refer to that set as the ground truth dataset. Note that we do not know what happens when a bike has been repaired twice in a month (does it counts for one or two bicycle repairs ?). We do not know how bad the breakage has to be to lead to the bike being picked up by the maintenance team.

We use the trips dataset to build a set of features and the ground truth dataset to train and evaluate our model.

2) Extracted Features: This study is not trip centered so it does not characterize a trip to know if it is a broken bike's trip or not. Indeed the lack of trip can be a sign of abnormality. Instead, it builds a sample per day and per bike, then determine if at a given day, a given bike was broken or not. For each of those samples, the tailored features are statistics on the life of the bike during a given interval of T days. They characterize the last of those days. By sliding a window of T days for each bike, several millions of samples are computed. If during the duration of a window (of period T) a given bike has not been used at all, then this sample is not created.

The features extracted are as follow: number of trips made, number of distinct departure stations, number of distinct arrival stations, number of loop trips (same arrival and departure station), average distance traveled over the trips, maximal distance traveled over the trips, minimal distance traveled over the trips, average duration over the trips, maximal duration over the trips, minimal duration over the trips.

That means that each sample exists in a 10 dimensions space, each of those features being one of those dimensions.

Those features are chosen based on three criteria. Firstly, they are an easy way to aggregate several days of trip for a bike. Secondly, once the classification has been made, it is possible to define heuristics to check the coherence of the classification. For example, a broken bike would not make a lot of long trips across the city compared to other bikes. Lastly, the shape of the distribution of each feature is either Gaussian, or at least with a long tail of outliers events, see Figure 2. This shape is desirable for anomaly detection since it allows us to easily separate rarely occurring events from normal ones.

B. Anomaly detection

After the extraction of the features for each dichotomy of day and bike, it is necessary to classify them into one of the two categories: normal if the bike is running smoothly, abnormal if the bike has to be picked up by the maintenance.

Musicant *et al.* [7] proposed algorithms to perform machine learning on aggregated outputs such as these. However there is a major difference with their data that makes their solution not directly applicable. In their case the aggregated output is the sum of the output they would normally use. For example, given a set of five balls, classify each ball as a blue ball or

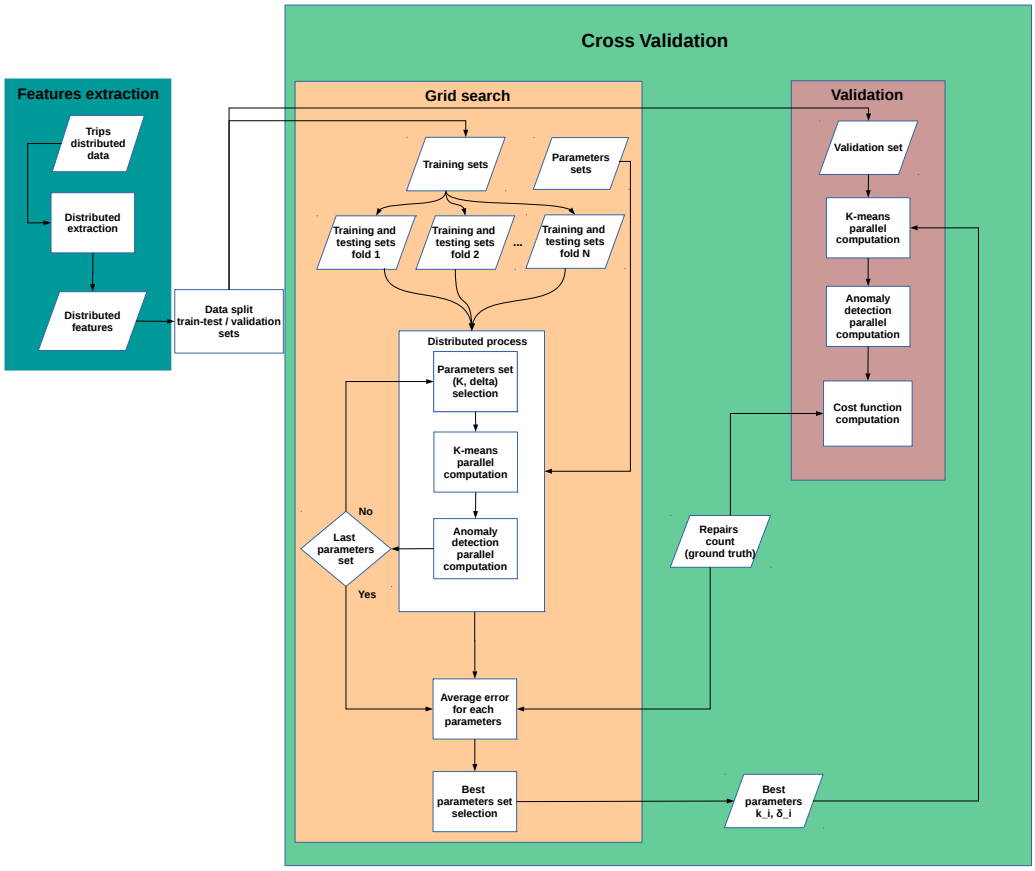

Fig. 1. **Process overview.** We can see here the two main distinct processes; first the features extraction creates a features set from raw data. Then the cross validation process that trains and evaluates the model. During the grid search an error is computed for each parameters set and each fold. An averaged error over the folds is then computed for each set of parameters and the one with the smallest error is selected. It is then used to perform a classification on the validation set which are features that have never been used to train the model. The error obtained on the validation set is an indactor of how well our model can generalize to new data.

TABLE I
RAW DATA DESCRIPTION

Feature Name	Description	Frequency
Departure location	Departure station coordinates	Always
Arrival location	Arrival station coordinates	Always
Departure station	Departure station name	Always
Arrival station	Arrival station name	Always
Departure date	Departure timestamp	Always
Arrival date	Arrival timestamp	Always
Bike	Bike unique identifier	always
Birth	User birth year	When the user is a subscriber
Gender	User gender	When the user is a subscriber
Repairs	Reported repairs number	Once a month

a red ball, given that there are 3 blue balls and 2 red balls. The aggregated output is the sum of balls of a given color. However, in this work, the aggregated ouptut is the sum of the repairs that occurred during each month. Thus their method would be directly relevant if we were trying to classify each day of a bike's life as being one where a repair occurred. In this case, each day classified as a repair day would be counted and the resulting sum should approach the aggregated output.

But this is not the classification that is performed because the aim of this work is to classify each day of a bike's life as a day where it was broken or not. Since there is no information if a bike can be repaired several times in a month, or how that would impact the given number of repairs, it is assumed here that a bike is only repaired once a month. The aggregation function used is thus a count of the distinct bikes that have been classified as broken each month. In consequence, the algorithms proposed by Musicant *et al.* can not be used.

The classification methodology is done in two steps. (i) First step consist in computing a clustering that separates the features in different groups representing different normal behaviors of a

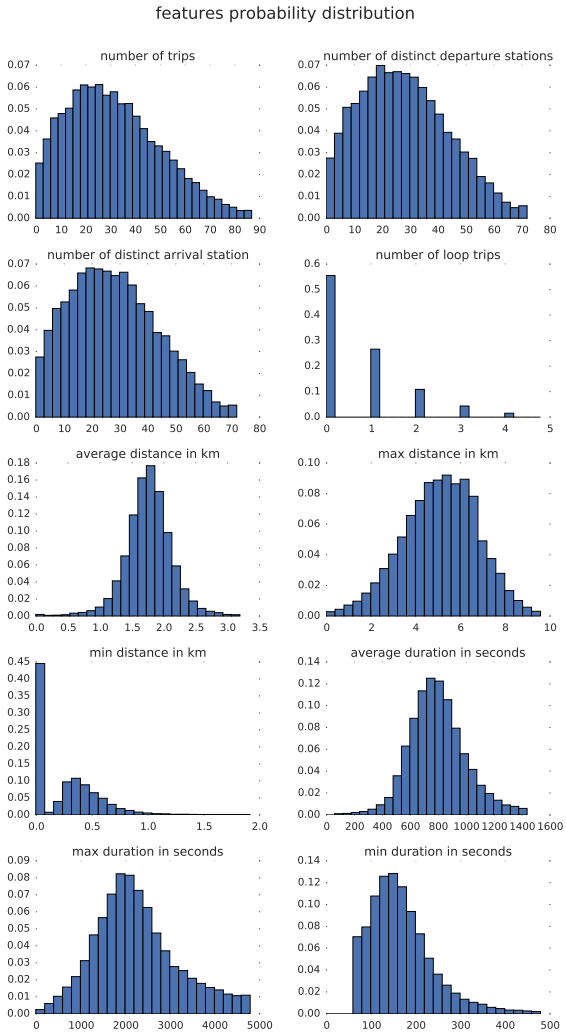

Fig. 2. Probability distribution of all tailored features

bike, (ii) then the second steps consists in identifying samples that do not clearly belong to a cluster. They are the one that are too far from the closest cluster center.

1) Clustering: First, we suppose that a bike has a different behavior depending of its situation. For example, a bike docked near a green area such as a park or a water plan is not used in the same way than a bike parked near a residential area. The former is more likely to be used for a stroll, finishing in the same station it started. The latter might be used early in the morning to go to a working area and in the evening, to go back to a living area.

This leads us to perform a clustering on our samples to separate different normal behaviors. To make this clustering, we use the K-Means algorithm [11]. K, the number of clusters and thus the number of relevant normal behaviors, being a parameter of our model. Each sample belongs to a cluster k, with clusters indexed by $k \in [1; K]$ of size M^k. Each sample of a cluster k is designated by $S_i^k, i \in [1; M^k]$. Each cluster possesses a centroid, C^k, which is the average position of all the points of the cluster k. We can place a centroid in our multidimensional space by defining each of its features $\{C_j^k\}, j \in [1; n]$ as the average of the values of this feature for the samples belonging to this cluster; $C_j^k = \frac{\sum_{i=1}^{M^k} S_{ij}^k}{M^k}$.

2) Distance from cluster center: Once the samples have been assigned to a cluster, we still have to determine which ones are not normal. Since our clusters are supposed to represent different normal behaviors, we assume that the abnormal ones are those that are not clearly in a cluster. We consider that a sample is not clearly in a cluster when its distance to the centroid is significantly larger than the one of the other members of the cluster to the same centroid.

This distance is defined as follow:

$$dist(S_i^k, C^k) = \sqrt{\sum_{j=1}^{n} \left(S_{ij}^k - C_{j^k}\right)^2} \quad (1)$$

Then we can define the function that is used to label sample as normal or abnormal as

$$anom(S_i) = \begin{cases} 1 & \text{if} \quad dist(S_i^k, C^k) > \delta * \sigma^k \\ 0 & \text{else} \end{cases} \quad (2)$$

where $anom$ is a function that returns 1 if the sample S_i is abnormal (the bike is supposed to be broken) and 0 if it is normal (the bike is not broken), σ^k is the standard deviation of the distances to the centroid, δ is another parameter of our model, it is used to modulate the distance from the centroid at which a sample is considered abnormal.

IV. Experimental Protocol

We describe here the protocol followed to train and evaluate our model.

1) Parameters optimization: We are left with two parameters to optimize; K the number of clusters (*i.e.*, the number of relevant behaviors) and δ the factor of the standard deviation of the distance from the centroid at which a sample is classified as an anomaly. T, the number of days used to build the features is not optimized here but fixed to seven.

The parameter selection is controlled by a process of cross validation. The selection has been made over a year and a half, start from June 2013 down to June 2014. Then once the parameters have been fixed, we performed a classification over a period of 15 months, start from January 2015 down to March 2016, which is our validation set. During the selection phase, the features dataset has been separated five times in a training set and a testing set (5-folds cross validation). Each time all months from June 2013 to December 2014 are used, 80% of them for the training and 20% of them to compute an error (the result of the cost function described in Equation 3) on the testing set. As for now we are using the Root Mean Square Error (RMSE). The error of a set of parameters being

the average testing error of the five different folds. Then the parameters set with smallest average error on the testing set is selected and the classification can be run on the validation set.

We find the best set of parameters thanks to a grid search since the space search is little enough to perform an exhaustive search. K is searched between 1 and 5; 1 means that there is no need for a clustering step, 5 is high enough so several behaviors may appear and low enough so the computation time is reasonable. If the error would have presented a tendency to decrease as K increases we would have had to push this limit farther but it has not been the case. δ is searched between 1 and 3, with steps of 0.25. That's because many of our features have a Gaussian-like shape, so we know that with $\sigma < 1$ we have too many abnormal samples and with $\sigma > 3$ we have too few of them. In order to judge if a set of parameters is better than an other one, we define a cost function (equation 3) that the models have to minimize. It measures the distance between the number of distinct bikes that have a sample detected as abnormal during a month and the number of repairs reported for this month.

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(Anomalies_{K,\delta}(i) - Repairs(i)\right)^2} \tag{3}$$

with $Anomalies_{K,\delta}(m)$ the number of distinct bikes that have been classified as broken at least once during the month i and N the number of months used. This particular cost function is called Root Mean Square Error (RMSE)

The domains of the extracted features are different which could lead the computed distance between two vectors of this multi-dimensional space to depend almost exclusively of the difference between one of their dimensions. It is necessary to reduce and center them, so every feature has the same average and a similar value range.

2) Execution platform: This procedure has been implemented with Spark [12] 2.6.0. We ran the feature extraction and the parameters selection on both a desktop machine with an 8 cores i7 CPU, 32GB of RAM, and 100GB SSD hard drive and a cluster of 15 machines, each one with 50 GB of RAM and 48 cores, using the data stored in Hadoop Distributed File System. The configuration allocates 50 workers with 5 cores per worker (there are more than one worker per machine) and the possibility to use 5GB of RAM per worker.

V. Results

This section presents for the results obtained with the optimal set of parameters.

This optimal set of parameters is $K = 1$ and $\delta = 2.25$ as illustrated in Figure 4. These parameters provide us a model that performs better with the chosen features and cost function on unseen data.

Equation 3 presents to minimize. It implies that we are trying to minimize the distance between two values; the given number of repairs for each month and the number of distinct bikes that have been detected as broken at least once during said month. Figure 3 presents the evolution of those values. The training period where we performed cross validation ends in December 2014. The validation period starts in January 2015. During the latter we perform a classification on data that we have never seen before. Using the root mean square error (RMSE), we compute an error depending of the distance between the two curves.

On the training set the RMSE is 852 and on the validation set 1304. Since these two RMSE values are of the same order, there is no overfitting.

We computed a normalized cross-correlation (Equation 4) between the two curves during the training period and the validation period. The closer it is to 1, the more our classification and the ground truth follow the same variations The normalized cross-correlation is defined as follows:

$$\frac{1}{n}\sum_{x}\frac{(f(x)-\overline{f})(t(x)-\overline{t})}{\sigma_f\sigma_t} \tag{4}$$

where $t(x)$ is the value of index x in the vector containing the values of the green curve, $f(x)$ is the value of index x in the vector containing the values of the red curve, n is the number of months in $t(x)$ and $f(x)$, $\overline{f}$ is the average of f and σ_f is the standard deviation of f.

The normalized cross-correlation during the training period is 0.47 and during the validation period 0.35. Having low values for both the RMSE and the correlation metrics indicates that the selected δ is helpful to fit the average number of repairs, but does not enable us to predict the variability.

The implementation on a cloud architecture is justified by the size of the data. On a regular machine the features extraction requires 25 minutes of processing while the optimisation of the parameters is done in 2 hours without cross validation, and can not be performed with cross validation executed with a parallel Spark implementation: after several hours of computation, the disk is full and the process stops. On the cluster the features extraction is performed in 2 minutes and 30 seconds, the optimization without cross validation in 17 minutes and with the cross validation 2 hours.

VI. Discussion

Since $K = 1$ is the best parameter, we failed to separate multiple normal behaviors. Three explanations are possible. Firstly, it is maybe not optimal to try to separate these behaviors and no solution with $k > 1$ can be better. Secondly, the most probable explanation is that the chosen features do not allow the behaviors to be separated. Indeed, we have not been able to visualize a separation between the clusters obtained with the Kmeans algorithm: scatter plots on each pair of dimensions do not separate them. We have to find other features that will lead to a meaningful clustering.

We could use t-sne [13] or other advanced techniques for high dimensional data visualization in order to distinguish them. If it is not possible we will train a linear model on our features with our classification as target in order to analyze the parameters of such a model. We will then gain some insight into what makes a feature more susceptible to separate clusters.

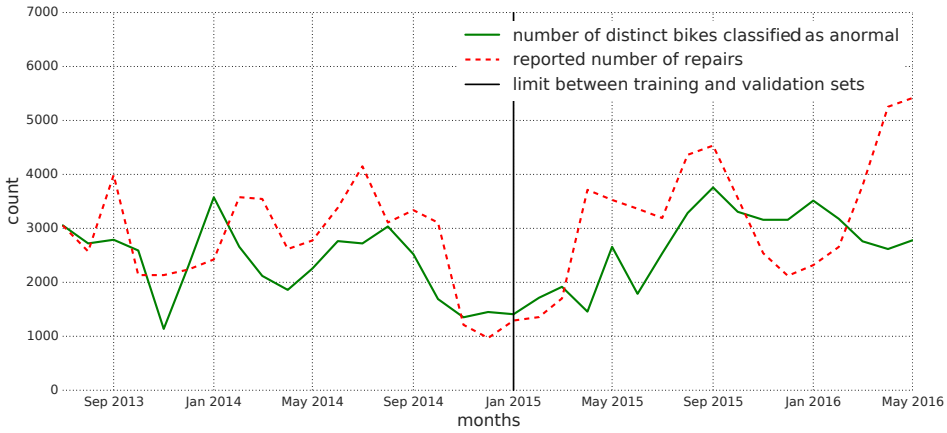

Fig. 3. **Classification versus ground truth.** The red dotted curve shows the number of repairs that have been reported each month. The green curve represents the number of distinct bikes that have been classified as abnormal at least once this month. The training period goes from the beginning to December 2014. The validation set starts on January 2015. They are separated by a black vertical line.

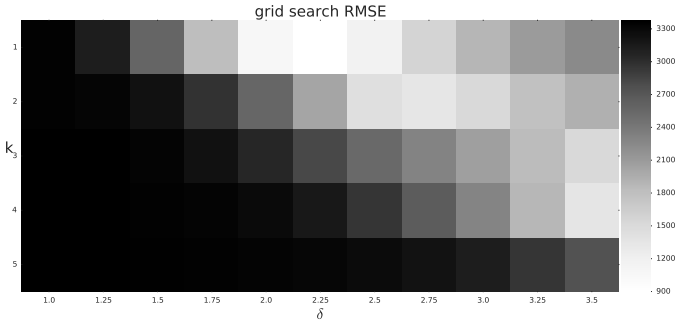

Fig. 4. **Best parameters.** This represents the average error on the 5 testings sets with each set of parameters. The lighter the color, the smaller the error. The minimum error is reached at $K = 1$ and $\delta = 2.25$.

The third possibility is that the cost function does not favor such a separation. Several clues indicate that this function is currently suboptimal. For example, if it is not unusual to see a bike being repaired several times a month, and if we are sure that every broken bike is repaired during the month, then the distinct number of broken bikes our model outputs should always be lower than the registered number of repairs. We will thus heavily penalize the opposite case in our cost function. On the other hand, the correlation is expected to be high and will be rewarded. Finally, a bike being detected as broken only one day or all days of the month makes no difference. A regularization term will help maintaining the overall number of anomalies lower.

Another limitation of our approach is that we cannot yet figure out if the detected anomalies match broken bikes. The curve which represents the performance of our classification (green curve) can fit very closely to the curve representing the real number of repairs (red curve), without representing broken bikes.

One impactful thing to do can be using a classic Aggregate Output Classification algorithm to try to detect for each bike / day pair if this is a day at which the bike has been taken by the maintenance team for a repair. The count of these occurrences is supposed to be exactly the given aggregated labels. Thus we know that those Aggregated Output Classification algorithms, designed for this very case, will have better performances. This also unambiguously handles the case when a bike is picked up more than once. So we will be trying to fit a number of repairs to the given number of repairs, which will be less prone to error of modeling or error of interpretation. Then if that new model is good enough, we can use its output as unaggregated new labels. It will be necessary to have a new way of measuring the correctness of this model so we will be able to use it as a new "ground truth" giving for each bike the pick-up time. Using this new information, we will have a better way to determine how our current model is performing. All the anomalies we detect should occur just before the bike has to be picked up.

Finally, the factor of distance determining if a sample is too far to its centroid to be classified as normal, δ, can be unique for each cluster. The advantage would be that we might have whole clusters of anomalies. Also, the parameter K will be used to control the complexity of the model. As for now we cannot allow this because it would make our model very complex. With so many parameters to optimize both the computation time and the probability of overfitting are very high.

VII. Conclusion

Bike sharing systems suffer of broken bikes and this paper presents a methodology which helps to find these broken bikes.

Using the CitiBike trips data as well as the given number of monthly repairs, we propose a method to classify each day of each bike's life as a day where this bike is not used, broken or normal. We do so by building features through the computation of statistics on the bike usage during the previous days, applying a clustering algorithm on the obtained samples and then detecting outliers.

We find that with the features designed and the cost function used, the clustering is not needed. By classifying samples too far from the average of all samples as broken bikes, we obtain a good fit to the curve representing the number of repairs each month. However, a small normalized cross correlation indicates that the number of distinct bikes classified at least once as broken during the month does not follow the same variations than the number of repairs. That is a clue to the fact that anomalies detected do not correspond to broken bikes.

Future work will aim at improving the model by using a better cost function, tailoring a new set of "raw data" via the use of standard Aggregate Output Classification methods, and finally factoring features that will let us take full advantage of the clustering algorithm.

ACKNOWLEDGMENT

The authors would like to thank Motivate for their open data, since they are one of the few bikeshare operators who are sharing their trips. This work was supported by the CIFRE n°1303/2014 of the French Association Nationale de la Recherche et de la Technologie.

REFERENCES

[1] E. Fishman, "Bikeshare: A review of recent literature," *Transport Reviews*, vol. 36, no. 1, pp. 92–113, 2016.

[2] B. Napoli, "Bike share boom: 7 cities doing it right," http://www.bikesharingnapoli.it/it/best-practices/, 2013, online ; accessed 18/02/2016.

[3] Susi, "How to be a good citi bike citizen," http://velojoy.com/2013/07/02/turned-bike-share-saddle/, 2013, online ; accessed 18/02/2016.

[4] N. Bonnotte, R. Cherrier, R. Delassus, and Y. Alouini, "Real-time data analytics and optimization of shared-vehicles networks," in *2nd ITS World Congres*, 2015.

[5] Motivate, "System data," https://www.citibikenyc.com/system-data, 2016, online; accessed 15/02/2016.

[6] ——, "Nycbs monthly operating reports," https://www.citibikenyc.com/system-data/operating-reports, 2016, online; accessed 15/02/2016.

[7] D. R. Musicant, J. M. Christensen, and J. F. Olson, "Supervised learning by training on aggregate outputs," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, 2007, pp. 252–261.

[8] M. Kaspi, T. Raviv, and M. Tzur, "Detection of unusable bicycles in bike-sharing systems," *Omega*, 2015.

[9] R. Bertens, J. Vreeken, and A. Siebes, "Beauty and brains: Detecting anomalous pattern co-occurrences," 2016, arXiv:1512.07048.

[10] H. Fanaee-T and J. Gama, "Event labeling combining ensemble detectors and background knowledge," *Progress in Artificial Intelligence*, vol. 2, no. 2-3, pp. 113–127, 2014.

[11] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, 1967, pp. 281–297.

[12] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *HotCLoud*, vol. 10, 2010, pp. 10–10.

[13] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.