2018 Sixth International Conference on Enterprise Systems

Data-driven Smart Bike-Sharing System by implementing machine learning algorithms

Jia Qian*
Department of
Applied Mathematics and Computer Science,
Technical University of Denmark,
Kongens Lyngby, Denmark
Email: jiaq@dtu.dk

Livio Pianura
Logos Technologies s.r.l.,
Mestre, Italy

Matteo Comin*
Department of
Information Engineering,
University of Padova,
Padova, Italy
Email: comin@dei.unipd.it

***Abstract*—This paper aims to solve a real-life problem: the bike-sharing management system arises the requirement of offering the customers the accessibility of the bikes in different bike-stations concerning the potential demands in every time-slice. The prediction of needs is critical to the distribution of the limited resources (bikes and empty slots to place the bikes) and the management of the system. We propose addressing this problem by using the *regression* model, which is trained by the raw data collecting from the different sensors. Thanks to the wide distribution of the edge devices, the machine learning algorithms, and the advanced computing ability, we may incorporate the intelligence to the database-related system. We will demonstrate that the boosting gradient method as a predictor to forecast the quantities of rentals and returns of bikes, outperforming the other means, e.g., random forest, support vector machine, etc. It reaches a promising result; the average accuracy reaches 75%.**

I. Introduction

Bike-sharing is a world-scale project; it has been promoted and implemented in many countries across different continents. It receives high popularity and attention due to the richness of meaning behind it. It suggests a healthy, eco-friendly lifestyle: alleviating the green-house emission by reducing the usage of cars [1]; diverting the traffic flow that highly burdened the load of the traditional public transporting network [2]. BikeX is a Bike Sharing project implemented in a metropolitan city with the population around 4 million (due to the confidential business information we cannot point out the project name). It is used for short trips in the place of the public transport system. It offers a service that it is free with the first thirty minutes to encourage the usage. Currently, it has more than 270 available bike-stations, from where the rental and return services can be offered, they are distributed throughout the whole city, including the downtowns and the outskirts with different density accordingly (bike station is shown in figure 1).

The placement of bikes is challenging since the variant geographical locations (e.g., center and suburb) and the various time (e.g., rush hour, idle time) differentiate the number of customers' demands and the type of service (e.g., rent or return). The issue mentioned above increases the difficulty of the system management, which turns out two consequences: 1) sometimes the requirement of customers are far away from fulfilling, 2) the resources cannot be entirely used (resource waste). For instance, the customers cannot find the available bikes in the nearby station or cannot find the empty position to return the bikes; some stations are fully mounted of bikes while no rentals and returns occurred. The resource distribution bears the pain of efficiently and systematically positioning the limited resources in response to the requirements [3], while it is highly required to solve since it is directly related to the expense of management. We suggest incorporating a foresight function in the system that may envision the underlying needs of the clients through the implementation of Machine Learning techniques. The predictor is typically generated by gaining the insight from the historical records sitting in the database. While BikeX management system has a massive, integrated database, which is jammed by different genres of data, e.g., customer profiles, log of bike usages and as well as climate information from the external sensors. It complicated our work as the different categories of data suffer from the data variety and data producing velocity. We have to determine the objective that is feasible under those constraints. After the discussion of the database designer, we finally decided the target as the prediction of the quantities separately for rentals and returns at every hour in different stations. We realize it through observing the records where a set of variables related to the target and the actual quantities of rentals/returns are collected. Alternatively, we may build a different setting, where the activities of clients, present/absent at every hour in the stations, may be classified. Then, the to-predict quantities are obtained by summing the classification result (0/1). As this method will ask for the grants from the clients, we do not adopt it, though, it may be a strengthener to improve the result.

II. Related work

In the following text, we will move from the introduction of the project and determine the objective to present the machine learning algorithms. The prediction is a ***supervised regression*** problem [4] in the machine-learning lingo, namely it means the variable that we try to predict are real value and data samples used to train and build the model are input-output

* Corresponding authors

2572-6609/18/$31.00 ©2018 European Union
DOI 10.1109/ES.2018.00015

Fig. 1. An example: Bike Station where the clients may rent and return the bikes.

paired. Many methods have been proved to be effective in different applications correspondingly, e.g., linear regression (with/without shrinkage) [5][6], support vector machine (SVM [7]), decision tree(DT [8]). The linear regression attempts to estimate the relationship between the set of input variables and the output variables by automatically learning from a set of training data. It gains its popularity due to its simpleness and interpretability, the set of features are combined with the different weights to determine the output finally. The coefficients imply the significance of the corresponding features if the weights are high indicating that the corresponding variable is significant and vice verse. Notably, the preprocess of re-scaling the variables is critical due to the nature of linear regression model [9]. Furthermore, it also spawns some variants, with the purpose to alleviate the over-fitting problem, which means that a model describes the random errors in the data instead of the relationship between variables. It happens when the model is too complicated, some methods subject the cost function to a constraint function of penalizing the features, e.g., least absolute shrinkage and selection operator (LASSO [10]). Usually, it works correctly when the features are continuous numerical values, and the standardization is implemented appropriately. SVM is another pleasurable approach; it may tolerate the noises to some extent by the use of slack variables, which may bypass the over-fitting issue since it often arouses from trying to fit all the data samples including noises. Another favorite character is that SVM builds the hyperplane and margin solely by a subset of the whole data set, which may significantly liberate memory and computing time. Additionally, the size of the subset may be adjustable by the hyper-parameters. Admittedly, it also has its limitation, the model has to come with the support vectors (the subset of data set), and it fails when dealing with the sparse data in some application ([11]), e.g., the recommender system. DT is another category of methods, and it is generally constructed from top to button, where the leaf nodes may predict the values or do the classification. The internal nodes are represented by the variables to generate the branches along with the break-point values. The selections of the features are executed by minimizing the sum of squared error between the predicted value and actual value, with considering the whole dataset. Thus, it suffers from over-fitting. Moreover, the mixture of categorical and numerical values are acceptable, the data scale and monotonic transformations are not necessary to apply to tree-like methods. It is a fundamental method, based on which a series of approaches are further developed and investigated. Among the ensemble category, Random Forest (RF, [12]) is a collection of trees, which are trained by a subset of data samples and some features instead of all as decision tree does. It attempts to solve the over-fit problems by reducing the variance, more precisely, by averaging a set of trees. These trees are fully-grown (low bias, high variance), they are uncorrelated by the random generation fashion to maximize the reduction in variance. However, the bias may not be reduced during the process, the biases of initial trees have to be as small as possible. Extreme random tree (ETR, [13]) is another tree-based ensemble method, unlike RF that optimizes the splitting node, it randomizes the cutting point. Therefore it is computationally cheaper. In general, machine learning has proven useful in a variety of different applications, including many problems in computational biology [14], [15], [16], [17], [18], [19], [20].

In this paper our contribution can be summarized as i) converting a practical issue into a realizable machine-learning model, ii) compare and find a method that fits the model (predictor) best, iii) embed the predictor into the management system. The paper is organized as follows: in the first section, we present the background of this Bike-sharing project by pointing out the problems and pinning down the objectives; in the second section briefly introducing the related algorithm; followed by presenting the gradient boosting method and the dataset, subsequently, the result will be illustrated and compared with the state-of-art techniques, and completing the work with the conclusion.

III. Material and Method

A. Database

Our work is carried out based on the real data set, and some data points are abnormal, incomplete and even erroneous, e.g., the connecting problems between sensors equipped on the stations and the server. Therefore, a purifying procedure is required to implement when we extract the data from the database. Note that all the information about the clients is anonymous. The whole dataset contains the records of 276 stations, spanning from 2016 to 2017. The different positions of the stations lead to the variant demands from the clients. Some stations located in the center or close to the entrances of public transport, where the service is highly required. Whereas,

some others have few usages because they are lately built or in the rural and developing area. As a result, The number of records is also diverse. We remove the stations which contain the number of records less 1000 since the information is not enough to train the model. We split the data into two parts, 70% for training, 30% for the validation. The average number of data samples for training is 2100; for testing is 850. We selected the features that are promisingly related to the objectives that we want to predict, describe as table I, and the targets that we attempt to predict are shown in table II. The first subset of features indicate the date and time, and we split it into five variables: year, month, day, hour and the day of the week. The second block of features reflect the number of subscriptions, "AnnualCustomerCount" is the number of clients who subscript at least one year; whereas the "OccasionalCustomerCount" is the number of clients who temporarily use the service. Subsequently, the four variables represent the climate: temperature, precipitation, wind, and humidity. The last subset indicates the status of the previous hour by four foci: the number of rentals and returns occurred in the precedent hour, the number of available bikes and empty slots at the end of the precedent hour.

TABLE I
FEATURES TABLE

Feature	Type	Annotation
StationId	int	identify the different stations
Year	int	specify the years
Month	int	specify the months
Day	int	specify the days
Hour	int	specify the hours
DayOfWeek	int	specify the weekday
Season	int	specify the season
AnnualCustomerCount	int	number of annual-subscription customers
OccasionalCustomerCount	int	number of temporal-usage customers
Temperature	int	media degree per hour of the corresponding day
Precipitation	float	media precipitation per hour of the corresponding day
Wind	int	media wind velocity per hour of the corresponding day
Humidity	int	media humidity per hour of the corresponding day
EmptySlotCount	int	number of empty slots at the previous hour
AvailableBikeCount	int	number of available bikes at the previous hour
PreviousRelease	int	number of rentals occurred in the previous hour
PreviousReturn	int	number of returns occurred in the previous hour

B. Feature Analysis

Before introducing the methods, we'd like to analyze the features in the first place. We want to evaluate the relations between them, i.e., how likely they are independent of each other. It is studied by computing the Kendall correlation [21] between pairs of variables. The heat-map of Kendall correlations between features is shown in Figure 2. The heat-map is symmetric, and the off-diagonal entries indicate the correlation between two different variables. The Kendall correlation ranges from -1 to 1 if the rank of the two variables are perfectly agreed the correlation is one if they disagree the value is -1 when the two variables are independent the correlation is close to zero. As we can tell, they are weekly correlated (relatively independent), this benefits us when we build and train the model in the future work. The Kendall's correlation defines as the below, know that we have two variables m,n.

$$\tau_{m,n} = \frac{(\#\text{ of concordant pairs}) - (\#\text{ of disconcordant pairs})}{N(N-1)/2}$$

Where N is the number of observations of the two variables that we want to compute the correlation. Concordant pairs (m_1, n_1), (m_2, n_2) are pairs of values in which ranks coincide: $m_1 < m_2$ and $n_1 < n_2$ or $m_1 > n_2$ and $n_1 > n_2$. Otherwise, disconcordant pairs. The denominator is the number of possible combinations of two variables.

C. Method

After we determined the objectives (predict the number of rentals and returns) and the category of machine-learning problem that may be mapped to (supervised regression), we propose a stage-wise method Boosting Gradient Regression (BGR)[22][23], which is a member of ensemble family. The choice is made based on the characteristics of the dataset: the variables stay in different magnitudes, and some are the integer, some are the float. If we use the linear models, the transform is supposed to apply on the variables, usually referring to the data-reduction. In some cases, it is not trivial. BGR may take care of these problems without the need for any manipulation. BGR sets off by creating a weak learner in every stage, and add it to the learner pool that keeps increasing after every iteration. Note that many models can form the weak leaner; it is just a broad concept. So in theory, a well-coded gradient boosting module would allow you to plug in various classes of weak learners at your disposal. Here it refers to the decision tree. During every stage, the weak learner is the winner that may minimally reduce the residual errors, selecting from a set of different trees that are built according to various combinations of features, sometimes even a stumble (tree with only one node). Besides, the construction of weak leaner may control by adjusting the maximal depth of trees or the number of nodes for every tree; the residual error function can be least square[24], absolute square, Huber(combination of least square and absolute square)[24] and so on. The absolute loss function has its disadvantage, and it is not differentiable when

it comes to zero, whereas, the least square loss function is sensitive to the outliers. Huber takes the advantages of them; when the difference between predict value and the real value is small, the function is quadratic, rather it is linear for the large value. We use Huber as the loss function, and it is defined as:

$$L_\delta(y,\gamma) = \begin{cases} \frac{1}{2}(y-\gamma)^2, \text{ for} |y-\gamma| \le \delta \\ \delta|y-\gamma| - \frac{1}{2}\delta^2, \text{ otherwise} \end{cases}$$

Where $\gamma = f(x)$, δ is hyperparameter, we need to pre-set.

The main algorithm describes as the followings:

- Firstly, we start the model by the first weak learner, it formalizes as $F_0(x)$, we initialize it with the mean of the training target values (the case when $m = 0$).

$$F_m(x) = \begin{cases} argmin_\gamma \sum_{i=1}^{n} L(y_i,\gamma), \text{ for } m \neq 0 \\ \frac{1}{n}\sum_{i=1}^{n} y_i, \text{ for } m = 0 \end{cases}$$

  Where $L(y_i,\gamma)$ is Huber error function, defined above.
- For iteration m from 1 to M (the case $m \neq 0$):
  - $F(x_i)$ in iteration m defines as

$$F_m(x_i) = F_{m-1}(x_i) + \gamma_{im} h_m(x_i)$$

  - Residual errors after iteration m is defined as the difference between the real values y and the predicted values $F_m(x)$ generated by the collection of learners in stage m:

$$e_{m+1}(x_i) = y - F_m(x_i)$$

  - γ_{im} and $h_m(x)$ are computed separately by the first and second order of loss function L.

$$\gamma_{im} = -[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{F(x)=F_{m-1}(x)}$$

$$h_m(x) = [\frac{\partial^2 L(y_i, F(x_i))}{\partial F(x_i)^2}]_{F(x)=F_{m-1}(x)}$$

  $F_{m-1}(x)$: the collection of learners up to stage m-1.
  γ_{im}: the learning size in stage m for sample data i.
  $h_m(x)$: a new leaner generated in stage m.
  $e_m(x)$: the residual error.

Despite the performance, the BGR has many advantages that make it outstanding among other approaches.

- It is not necessary to normalize or standardize the variables, and the algorithm enables the feature node splitting automatically during the training process with the purpose of minimizing the residual errors. The normalization is not exclusively benefiting the result, e.g., SVM in some cases it improves, in some cases, not [25].
- It may somewhat alleviate the overfit issues since in every stage the tree is built shallow, in other words, it is a simple tree made of few features.
- It may automatically handle the data missing problem, knowing proper direction to traverse and adequate leaf to sit. For the other methods, we have to deal with the missing data before we implement the methods.

TABLE II
TO-PREDICT VARIABLES TABLE

Prediction	Type	Annotation
ReleaseCount	int	number of rentals occurred in the next hour
ReturnCount	int	number of returns occurred in the next hour

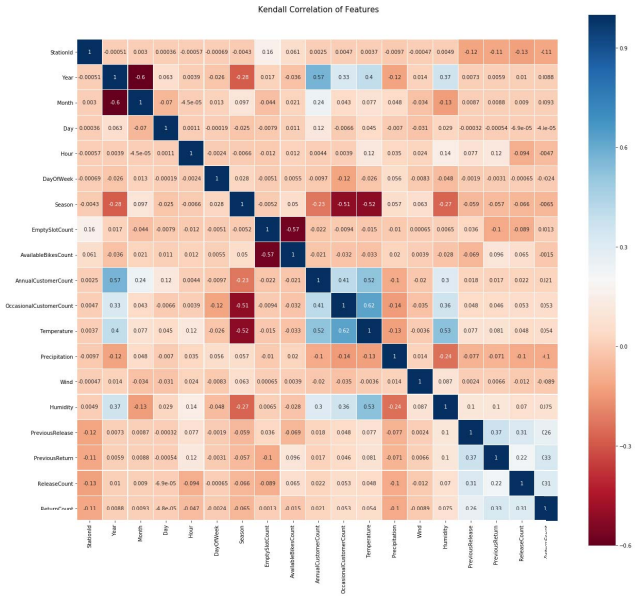

Fig. 2. Feature analysis: study the independence between the feature pairs.

IV. EMBEDDING THE PREDICTOR INTO THE SYSTEM

In the last section, the BGR algorithm is introduced, in this section, we will briefly introduce how to add the predicting function into the existed system and the interaction between them. The core frame is shown in figure 3.

- Initially, the system automatically loads the predictors (from the database) and feature vector (from different sensors) into the RAM at the specific time according to the requirements recorded on the configuration file. Since the predictor is trained based on single station, the loading model signal should clarify the corresponding model to load. The parallel operation is possible, and we may create multi-threads to deal with multiple requirements at the same time. When we form the feature vector, we should pay attention to the timing of collecting the features related to the previous hour (the last four features in table I).
- Consequently, the predictor outputs the to-forecast values and feed them into another module (resources distributing) along with a reliability indicator to place the bikes according to the potential needs. The indicator is computed based on the records, implying how likely that the predicted value may be correct.
- In the same time, the system collects the real values and the feature vectors and saves them into the dataset for

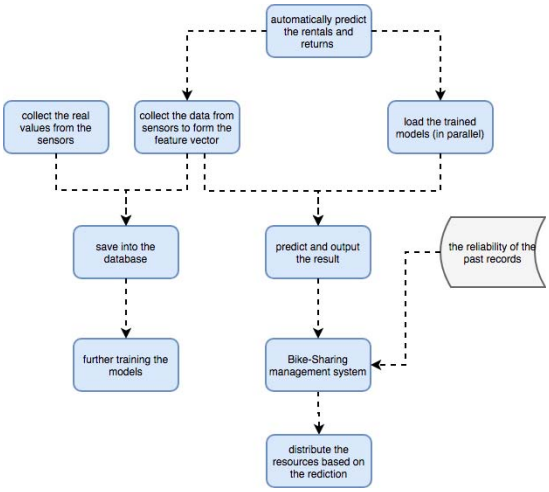

Fig. 3. workflow and the interaction between the predictor and the sensors, database of the bike-sharing system.

the further training, which will occur after an interval to track and update the underlying changes of the customers' demands.

V. Experiment and Analysis

A. Measure Matrix

The performance is evaluated firstly by the average precision of all bike stations, and then we look into the accuracy of every single station. Assume that we have m stations to predict, every station has n data samples to test (for the simplicity, actually the number of data samples of every station is different). The error for station t is:

$$E_t = \frac{\sum_{i=1}^{n}(y_{ti} - y_{ti}^{'})^2}{\sum_{i=1}^{n} y_{ti}^2}$$

Therefore, the precision for station t is:

$$P_t = 1 - E_t$$

The average precision P of m stations is:

$$P = \frac{\sum_{s=1}^{m} P_i}{m}$$

B. Performance

To find the best-fit algorithm predicting the possible number of rentals and returns, we tried the different methods. To validate and demonstrate the performance, we compare the average precision produced by BGR with SVM, LASSO, RF, and ETR. GBR outperforms the other methods for both release and returns predictor, shown in figure 4. For the release predictor, the average precision of GBR is 75%, almost the double of SVM, 10% higher than LASSO, 5% higher than ETR. Here we applied both SVM with linear-kernel and radial-basic-function (RBF) kernel, in our case they don't show the

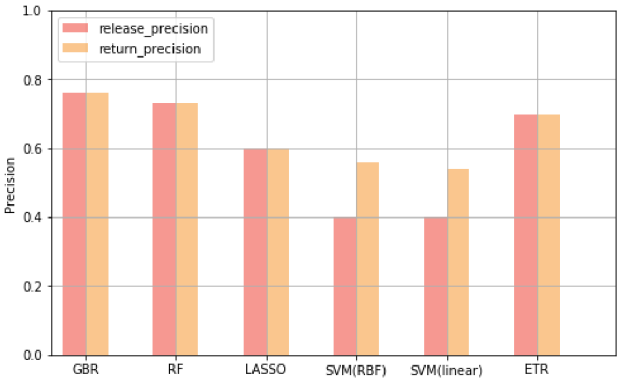

Fig. 4. Average release/return precision for different methods.

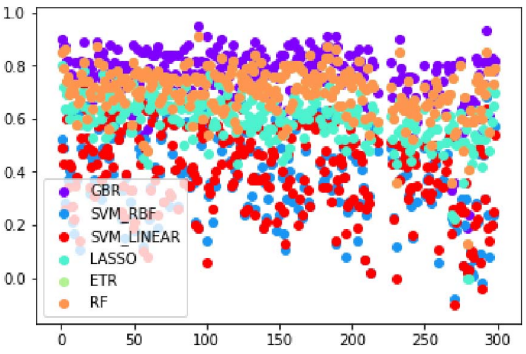

Fig. 5. Release precisions for all stations.

significant difference. Additionally, our result aligns with the conclusion drawn by the other authors [26] that BGR has the better outcome than RF and DT. The result generated by the return predictor is similar to release predictor. The only difference is that the precision of SVM is improved, and we can tell the ensemble algorithms (GBR, RF, and ETR) seem to be more robust, the accuracies of rental and return are almost identical.

In figure 4 we illustrated the average precision of all 276 stations, in this series of the experiment we want to take a closer inspection of the accuracy for every station, shown in figure 5 and figure 6. The accuracies of all the station produced by GBR for both rental and release predictors are marked by the purple circles, hanging above the markers generated by the different methods. It means that virtually the precision of every station outraces the results of the other approaches. Notably, there are few stations that the LASSO turns out to contribute a better result. Therefore, we can use LASSO as a predictor in those stations to further improve the performance.

VI. Conclusion

This paper thoroughly presents how to enable the machine-learning algorithms to solve the real-life application. It starts from introducing the background of the application and the

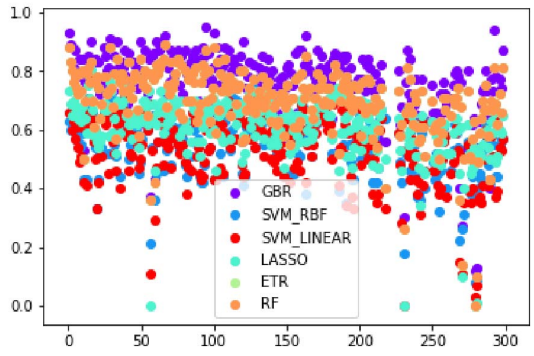

Fig. 6. Return precisions for all stations.

problems, and then determining the objective that is implementable by the machine learning methods, searching the best solver, and finally embedding it to upgrade the system. We want to bridge the gap between the theories and the applications by giving a practical case. The future work can be carried out from three aspects. The first one is to forecast the rentals and returns by building a classifier that can predict the activities of the customers to strengthen the smart module. The second one is to study the batch size for the further training as the underlying needs of the customers may vary over time, further training is necessary. Moreover, we can make the dimensions of the feature vector scalable if we can collect more useful variables from the external sensors in the future, e.g., the event tracker, hence, the predictor may be further empowered.

ACKNOWLEDGMENT

We would like to thank the SQL designer who introduces us the database and contributes the idea during feature selection and data extraction. This work is supported by FSE project "Smart Management of Next Generation Bike Sharing code 2105-1-2216-2016.

REFERENCES

[1] J. D. Figueroa, T. Fout, S. Plasynski, H. McIlvried, and R. D. Srivastava, "Advances in co2 capture technologythe us department of energy's carbon sequestration program," *International journal of greenhouse gas control*, vol. 2, no. 1, pp. 9–20, 2008.

[2] B. Chapman, H. Iseki, B. D. Taylor, and M. Miller, "The effects of out-of-vehicle time on travel behavior: Implications for transit transfers (deliverable# 1)," CA: California Department of Transportation, 2006.

[3] C.-J. Huang, Y.-W. Wang, C.-T. Guan, H.-M. Chen, and J.-J. Jian, "Applications of machine learning to resource management in cloud computing," *International Journal of Modeling and Optimization*, vol. 3, no. 2, p. 148, 2013.

[4] A. M. Legendre, *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot, 1805.

[5] D. Nguyen, N. A. Smith, and C. P. Rosé, "Author age prediction from text using linear regression," in *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*. Association for Computational Linguistics, 2011, pp. 115–123.

[6] G. Y. Kanyongo, J. Certo, and B. I. Launcelot, "Using regression analysis to establish the relationship between home environment and reading achievement: A case of zimbabwe." *International Education Journal*, vol. 7, no. 4, pp. 632–641, 2006.

[7] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[8] C. J. Stone, "Classification and regression trees," *Wadsworth International Group*, vol. 8, pp. 452–456, 1984.

[9] A. Gelman, "Scaling regression inputs by dividing by two standard deviations," *Statistics in medicine*, vol. 27, no. 15, pp. 2865–2873, 2008.

[10] H. Xu, C. Caramanis, and S. Mannor, "Robust regression and lasso," in *Advances in Neural Information Processing Systems*, 2009, pp. 1801–1808.

[11] S. Rendle, "Factorization machines," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 995–1000.

[12] T. K. Ho, "Random decision forests," in *Document analysis and recognition, 1995., proceedings of the third international conference on*, vol. 1. IEEE, 1995, pp. 278–282.

[13] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[14] J. Qian, D. Marchiori, and M. Comin, "Fast and sensitive classification of short metagenomic reads with skraken," in *Biomedical Engineering Systems and Technologies*, 2018, pp. 212–226.

[15] M. Comin and M. Antonello, "On the comparison of regulatory sequences with multiple resolution entropic profiles," *BMC Bioinformatics*, vol. 17, no. 1, p. 130, 2016.

[16] M. Comin, A. Leoni, and M. Schimd, "Clustering of reads with alignment-free measures and quality values," *Algorithms for Molecular Biology*, vol. 10, no. 1, pp. 1–10, 2015.

[17] M. Comin and M. Schimd, "Assembly-free genome comparison based on next-generation sequencing reads and variable length patterns," *BMC Bioinformatics*, vol. 15, no. 9, pp. 1–10, 2014.

[18] M. Comin and D. Verzotto, "Beyond fixed-resolution alignment-free measures for mammalian enhancers sequence comparison," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 4, pp. 628–637, July 2014.

[19] M. Comin and M. Antonello, "Fast computation of entropic profiles for the detection of conservation in genomes," in *Pattern Recognition in Bioinformatics*, 2013, pp. 277–288.

[20] M. Comin and D. Verzotto, "Whole-genome phylogeny by virtue of unic subwords," in *Database and Expert Systems Applications (DEXA), 2012 23rd International Workshop on*, Sept 2012, pp. 190–194.

[21] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[22] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[23] L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean, "Boosting algorithms as gradient descent," in *Advances in neural information processing systems*, 2000, pp. 512–518.

[24] S. M. Stigler, "Gauss and the invention of least squares," *The Annals of Statistics*, pp. 465–474, 1981.

[25] T. Kraska, A. Talwalkar, and J. Duchi, "Mlbase: A distributed machine-learning system," in *CIDR*, 2013.

[26] J. O. Ogutu, H.-P. Piepho, and T. Schulz-Streeck, "A comparison of random forests, boosting and support vector machines for genomic selection," in *BMC proceedings*, vol. 5, no. 3. BioMed Central, 2011, p. S11.