Pump it Up: Data Mining the Water Table

Antriksh Agarwal

Sarvesh Pandit

Abstract

The task of this project is to predict which water pumps throughout Tanzania are functional, which of those need repairs, and which do not work at all. The prediction is based on a number of variables about what kind of pump is operating, when was the pump installed, and how are the pumps managed. We are comparing the performance of a couple of boosted classifiers with Support Vector Machines (SVMs) and the performance of SVMs with different kernel functions. The selected boosted classifiers for this project will be XGBoost and LightGBM. These classifiers will be evaluated using accuracy as the evaluation criteria as given on the challenge website. The ultimate goal would be to find the best classifier and suggest improvements.

1 Introduction

Water is a fundamental need of the human beings, mainly for their survival. We also need water for washing, cooking, drinking, bathing and a lot of other things. Water helps to fulfil so many of our domestic needs and is also useful for so many of our industrial needs such as powering plants, producing electricity, irrigation of agricultural lands, etc.

There are a large number of countries in the world not having adequate sources of water and even lesser resources to harvest it. Such countries require serious laws and regulation of water so that the people are not dying due to limited supply of water. One such country seems to be Tanzania where lots of pumps were installed by various firms and have now been non functional or needs repair for some time. Tanzania, officially the United Republic of Tanzania, is a country in eastern Africa within the African Great Lakes region. Tanzania's population in 2016 has been estimated to be 55.6 million, composing of various ethnic, religious and linguistic groups. Tanzania's water supply can be characterized as decreasing access to improved water sources, intermittent water supplies and generally a low quality of service [1]. This project aims to single out the factors which lead to a water pump not being functional.

The project aims to identify problems with various factors that affect the functioning of water pumps, how to use these factors to a favorable outcome and how to succeed in dealing with the problems associated with these factors. This is to be

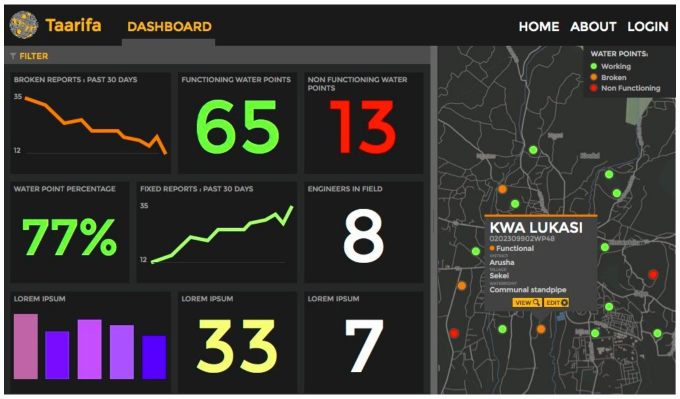


Fig. 1. Pump It Up: Data Mining the Water Table; Taarifa.com visualization of condition of water pumps. Source: Driven Data Competition [2]

done by gaining a smart understanding of features because of which the water pumps installed by various organizations in Tanzania have stopped functioning properly. The training dataset divides each water pump into being functional, non functional or functional but needing repairs.

We are going to be using various supervised machine learning algorithms and evaluate them based on the data obtained from the Driven Data [3] competition Pump it Up: Data Mining the Water Table [2]. The first algorithm we would be working on is Support Vector Machines which have proven to be very useful in the supervised learning world transforming data from one dimension to another dimension where it is linearly separable. There are also various boosting algorithms like Random Forests, XGBoost, Gradient Boosted Trees etc. which perform very well on all sorts of classification problems. We would only be experimenting with XGBoost and LightGBM which are two tree boosting algorithms. The rest of the report explains these algorithms briefly and displays various distributions of data.

A smart understanding of these water pumps using the algorithms mentioned in this report, and other techniques which perform relevantly well will help us and the government of Tanzania improve the condition of water supply in the country.

2 Related Work

There has not been a lot of work trying to predict the functional or non functional water pumps using the data provided in the challenge, but a lot of work has been done using data mining techniques to identify related fields like future healthcare, bioinformatics and even pump and pipe failures.

There has been some work conducted on trying to predict failures of pipes by Tran and Ng [8] where they compare SVM with Back-Propagational Neural Network (BPNN) concluding that SVM outperforms BPNN. This suggested that we use SVM instead of experimenting with Neural Networks. A similar work on attempting to assess deterioration of stormwater pipes in Guelph, Ontario has been done by Harvey and McBean [9]. They use classification trees on a dataset gathering insight on the influence of construction year, diameter, length and slope on pipe condition.

Another research closely resembling the one this project aims to work on has been conducted previously by Arymurthy [10] where they are comparing the performance of Random Forests, XGBoost, Gradient Boost Machine (GBM) and SVM. They also perform a data analysis trying to exclude unnecessary features. They conclude that XGBoost outperforms any other classifier in the domain. They also identify *quantity, waterpoint_type, longitude and latitude (coordinate location), gps_height, wpt_nama, subvillage, ward, construction_year and date_recorded* as the 10 most important features.

3 Dataset

The dataset for the task has been provided in the challenge Pump It Up: Data Mining the Water Table [2] posted on the Driven Data [3] website.

The features provided in the dataset have been described on the challenge webpage and are shown with their description in Fig. 2. There are 40 columns in the training dataset with over 59,000 instances of data. The labels for each of the instances have been provided in a separate file. Each of these instances have a classifying label which classifies the data point to either "functional", "non functional" and "functional needs repair". The labels mean the following:

- functional - The waterpoint is operational and there are no repairs needed
- functional needs repair - The waterpoint is operational, but needs repairs
- non functional - The waterpoint is not operational

4 Preprocessing Techniques

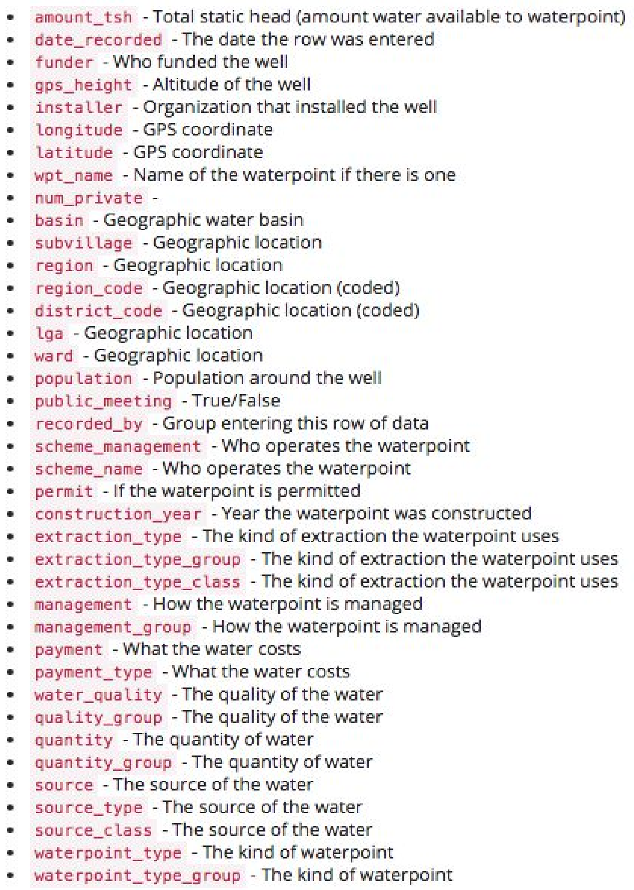
In this project, python will be used as the coding language. To begin with, training dataset contains all the attributes without any labels and a separate file containing just the labels. The decision to combine the two by using the id attribute was taken, which resulted in all the attributes with their labels in a single table.

For generating test data, 20% of the entire dataset was used. The remaining 80% was used for generating training and validation datasets. 10 fold cross-validation technique was used for the creating training and validation dataset.

After visualizing the data on the bar graph, it was found out that the categories with similar names were recognized differently due to different cases (uppercase/lowercase) of some part or whole of the category or some abbreviations (e.g. Gover and Government, Commu and Community). Our assumption is that if the name (in it's exact given form) is different, then the categories are different, else they are already being recognized as same.

The missing values in the dataset were identified as 0 which can be seen in Fig. 11 for the attribute "construction_year".

Some of the features (such as features showing Geographic location, management and management_group, extraction_type_group and extraction_type_class) had identical results and therefore only one of them seemed useful for the experiments and for reducing the complexity.



- `amount_tsh` - Total static head (amount water available to waterpoint)
- `date_recorded` - The date the row was entered
- `funder` - Who funded the well
- `gps_height` - Altitude of the well
- `installer` - Organization that installed the well
- `longitude` - GPS coordinate
- `latitude` - GPS coordinate
- `wpt_name` - Name of the waterpoint if there is one
- `num_private` -
- `basin` - Geographic water basin
- `subvillage` - Geographic location
- `region` - Geographic location
- `region_code` - Geographic location (coded)
- `district_code` - Geographic location (coded)
- `lga` - Geographic location
- `ward` - Geographic location
- `population` - Population around the well
- `public_meeting` - True/False
- `recorded_by` - Group entering this row of data
- `scheme_management` - Who operates the waterpoint
- `scheme_name` - Who operates the waterpoint
- `permit` - If the waterpoint is permitted
- `construction_year` - Year the waterpoint was constructed
- `extraction_type` - The kind of extraction the waterpoint uses
- `extraction_type_group` - The kind of extraction the waterpoint uses
- `extraction_type_class` - The kind of extraction the waterpoint uses
- `management` - How the waterpoint is managed
- `management_group` - How the waterpoint is managed
- `payment` - What the water costs
- `payment_type` - What the water costs
- `water_quality` - The quality of the water
- `quality_group` - The quality of the water
- `quantity` - The quantity of water
- `quantity_group` - The quantity of water
- `source` - The source of the water
- `source_type` - The source of the water
- `source_class` - The source of the water
- `waterpoint_type` - The kind of waterpoint
- `waterpoint_type_group` - The kind of waterpoint

Fig. 2. Feature names and their descriptions as provided on the challenge website

4.1 Feature Engineering

Most of the features in the dataset were categorical features. So the features had to be converted to numerical from categorical for any sort of training to be performed over the data. We tried all types of encoders out of which LabelEncoder from Scikit Learn library was selected. Another option was one-hot encoding of all the categorical features, which resulted in more than 59,000 features. This approach was abandoned because the number of features was almost equal to the number of instances, which would had never gotten done with even principal component analysis. Label Encoder on the other hand converts categories to nominals which can be accessed by the classifier as numbers but may not always be the best approach to machine learning classification tasks. We had to take this approach in order to compensate for time consumption.

After encoding of the features, we found that some of the features were almost exact copies of some other features and some features were not changing. These attributes like, num_private, funder, scheme_name, installer etc have not been used in any experiments from this point on.

4.2 Principal Component Analysis

For this project, after encoding of the categorical values in the attributes was done, principal component analysis was run over the attributes to reduce the dimensionality of the data. The final selection was a dataset with 20 attributes which essentially seemed to give the best performance among all tested. Results and various tests have been shown in Table 2. It also uses 31 principal components which means that no principal component analysis was performed as after reduction of the known columns only 31 attributes were selected in the final dataset.

5 Experiment

For this project, the three primary techniques to be experimented with are Support Vector Machine, XGBoost and LightGBM. A brief explanation of all the three techniques is provided in this section.

5.1 Support Vector Machines (SVM)

SVM is a supervised learning model with associated machine learning techniques for classification and regression. SVM is a model which will split the data in the best possible way by separating the two groups with the widest possible margin. SVM works as a constraint optimization problem where the constraints are the data points and we want to maximize the margin of classification between any two classes.

The equation of the hyperplane for linearly separable classes for an SVM is $w \bullet x + b = 0$ such that

$$w \bullet x_i + b < -1 \quad : if\, y_i = -1 \tag{1}$$

$$w \bullet x_i + b > +1 \quad : if\, y_i = +1 \tag{2}$$

where w is the weight vector, x_i is the ith instance provided in the dataset, b is the bias term, and y_i is the actual class value.

SVM relies on kernel functions for transformation of dimensionality from one dimensional space to a higher dimensional space. A few of the most common functions that have been used with SVM are shown in Eq. 3, Eq. 4 and Eq. 5 representing the polynomial kernel, hyperbolic tangent kernel and the Gaussian radial basis function kernel, respectively. This project will be experimenting with all of these kernel functions to see which one gives the best accuracy.

$$k\,(q, q') = (1 + q.q')^k \tag{3}$$

$$k\,(q, q') = tanh(aq.q' + b) \tag{4}$$

$$k(q, q') = exp(-\| q - q' \|^2 / \sigma^2) \tag{5}$$

The Scikit-Learn [7] implementation of the multi-class SVM, with the already implemented kernel functions for the purpose, would be used for this project.

5.2 Extreme Gradient Boosting (XGBoost)

XGBoost [5] is an optimized open source gradient boosting library, developed at Distributed Machine Learning Common (DMLC), designed to be highly efficient, flexible and portable. It implements machine learning algorithms under Gradient Boosting framework. This library has been designed to run on a single machine as well as distributed clusters over Apache Hadoop. It has been developed by Chen and Guestrin [11] at the University of Washington. This library has been developed for Python, R, Java and Scala programming languages.

The Gradient Boosted Trees implementation of XGBoost package library was used, which is explained in the next subsection.

5.2.1 Gradient Boosted Trees (MART)

Gradient Boosted Machine or Gradient Boosted Trees or Multiple Additive Regression Trees (MART) is an ensemble method widely used for classification problems. As a general explanation, MART computes derivatives of the loss function and adds a regression tree to the ensemble which can fit the inverse of the derivatives. So, essentially, it is a gradient descent algorithm.

Formally, MART takes as input value of type (x, y), where x is the instance, or the points in some input space X and y is a point in the label space. The algorithm also uses a loss generating function which is dependent on the task at hand. The loss generating function and the input define the loss at each point x, $L_x : Y \rightarrow R$ where Y is the predicted space.

At every iteration, MART produces a model $M : X \rightarrow Y$. Let $M(x)$ denote the prediction of M at point x and $L'_x(M(x))$ be the derivative of the loss function at x. MART simulates an intermediate dataset which assigns $-L'_x(M(x))$ as the output label on every data point x. It then fits a regression tree over the new dataset, the inverse derivative, which is then added to the regression trees ensemble to minimize the loss.

In a classification task, which is also what is being intended for solving the problem at hand, the loss function used is logistic loss. The loss function is defined as

$$L_x(\hat{y}) = \frac{1}{(1 + exp(\lambda \hat{y} y))} \quad (6)$$

where λ is a parameter, $\hat{y}$ is the predicted label, y is the true label and x is the input point.

5.3 Light Gradient Boost Machine (LightGBM)

Light Gradient Boost Machine is a gradient boosting framework which is based on decision trees. Since it is based on decision trees, the split is done leaf-wise rather than the conventional depth-wise or level-wise split. The leaf-wise split results in better accuracy. Advantages of using LightGBM are faster training speed, lower memory usage, better accuracy, compatibility with large datasets.

There has been no research paper written on this implementation but this implementation has proven to be fast and more accurate than a lot of other machine learning implementations of gradient boosted trees and its derivatives.

The Dropouts meet Multiple Additive Regression Trees (DART) algorithm implementation of the LightGBM library was used for all testing, which is explained in the next subsection.

5.3.1 Dropouts meet Multiple Additive Regression Trees (DART)

DART algorithm [11] builds over MART algorithm (also known as Gradient TreeBoost or boosted trees). MART algorithm, as explained earlier, generates a tree after every iteration and keeps model $M = \sum_{i=1}^{N} T_i$ where T_i is the model learned in the *ith* iteration, and N is the number of iterations completed till this point. DART on the other hand selects a random subset of the trees $I = \{1, \ldots, N\}$ such that a new model $\widehat{M} = \sum_{i \in I} T_i$ and fits a new tree T over the inverse of the derivative which is added to the ensemble.

DART, as originally explained in [11], with the extension mentioned above, overshoots the target if both dropping of trees and adding a new tree to the ensemble are done at the same time. Thus, an improvement which is made in DART is scaling down the new tree by $1/k$ to make it have the save order of magnitude as the dropped trees. Following this, the dropped trees and the new tree are scaled by a factor of $k/(k+1)$, where k is the number of dropped trees for model M to make I.

DART is helpful in overcoming the problem of over-specialization. The dropping of trees can be viewed as regularization where the number of dropped trees controls regularization. It seems like an improvement over gradient boosted trees or MART algorithm and hence was a good choice to experiment with in the project.

6 Training

All the three models have been run on different parameters for trial and error and checking out which of them gives the best accuracy. Scikit-Learn implementations or APIs of all the above discussed models have been used. So, it was easy to modify parameters in the models to see which parameters accounted for the best accuracy.

Gradient Boosted Trees have been used with a learning rate of 0.03, with maximum allowed depth of a single tree being 18. We generate a maximum of 400 estimators or trees for the ensemble and keep a regularization constant gamma as 3 for the complete training. It was run on 6 cores of an octa core machine.

DART algorithm worked very well on a completely different set of parameters and gave us results similar to Gradient Boosted Trees. In DART, we set the maximum number of leaves generated as 173, and maximum allowed depth of a single tree as 14. The learning rate was set to 0.2 which was about ten times what we used with MART algorithm. The number of estimators was set to a maximum of 350 and just like MART algorithm, it was run over 6 cores of an octa core machine. The objective in both of these algorithms was multiclass classification by maximizing the logistic loss function.

SVM was again running on 6 cores of an octa core machine. The kernel was set to 'rbf' and the regularization constant gamma was set to 0.001. We tested with various values of C to see an improvement in performance, but the best performance was with C around 1 or 2.

All of the classifiers were also tested with a 5-fold cross validation. The results section discusses the outputs of all the classifiers compared to the expectations we had with them.

7 Results

The challenge website uses accuracy as the metric for classification into a good or a bad classifier. We also used another metric as precision to be sure of the classification score we were achieving. Table 1 and Table 2 show the classification accuracy and precision achieved using the three classifiers with their best parameters. A classifier is deemed good only if it outperforms every other classifier on all the metrics used, i.e. precision and accuracy.

MART algorithm or gradient boosted trees was expected to work the best over the given data, but the DART algorithm might have outperformed it with an accuracy of 79.2% as the best performance of all with 20 principal components extracted using the Scikit Learn PCA.

SVM, which was expected to run better than or at least similar to other algorithms gave us some disappointing results. With kernel set to "poly", SVM took a lot of time to converge given 6 cores on an octa core machine. Hence, kernel "poly" was not used in any classification. The 'rbf' kernel was pretty fast in converging as well as gave the best accuracy among all the others. The 'sigmoid' kernel runs considerably fast but fails to give an accountable accuracy (< 50%).

Table 2 shows the best achieved accuracy and precision with each of the classifiers used in the experiments. DART is helpful in overcoming the problem of over-specialization. The dropping of trees can be viewed as regularization where the number of dropped trees controls regularization. It seems like an improvement over gradient boosted trees or MART algorithm and hence was a good choice to experiment with in the project.

What can also be observed about the DART algorithm from the training curve shown in Fig 3 is that it did solve the problem of over-specialization a bit through time in. This is also prominent from the cross validation scores which seem to increase with the number of training examples. Fig 4 and Fig 5 show the training and cross validation curves for MART and SVM classifiers, respectively.

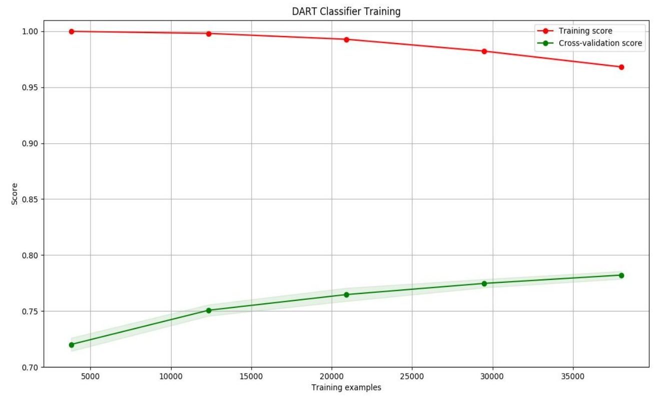


Fig. 3. Training and cross-validation curves for DART classifier.

Table 1. Precision and Accuracy Scores for each classifier with variation in Principal Component Analysis

Principal Component	Classifier	Accuracy	Precision
18	MART	77.415	76.845
18	DART	78.047	77.284
18	SVM	56.540	56.624
20	MART	78.207	77.642
20	**DART**	**79.276**	**78.649**
20	SVM	55.261	56.948
25	MART	78.114	77.568
25	DART	79.057	78.326
25	SVM	55.774	56.704
31	MART	78.788	78.441
31	DART	79.082	78.415
31	SVM	56.085	55.227

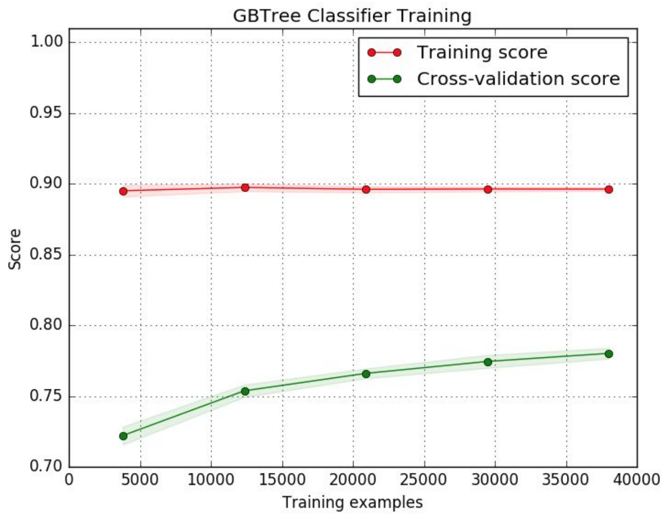


Fig. 4. Training and cross-validation curves for MART classifier.

8 Conclusion and Future Work

DART outperforms all the other algorithms on the given dataset. MART algorithm comes close to the performance of DART algorithm and has better range of precision values but is still outperformed by DART algorithm.

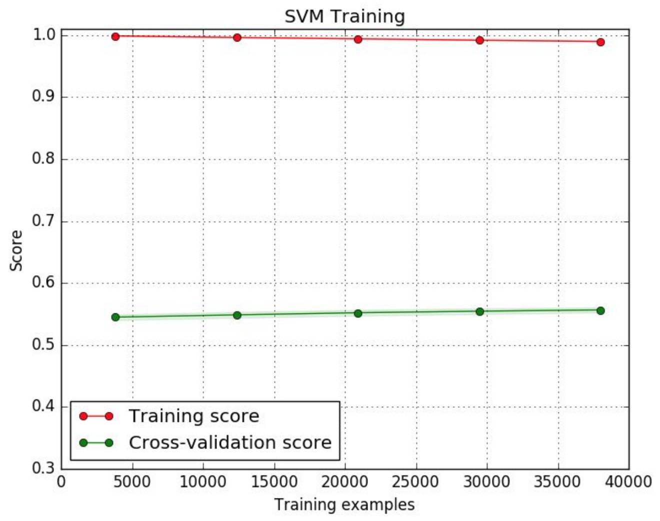


Fig. 5. Training and cross-validation curves for SVM classifier.

A minor setting in pre-processing which could have been improved was the encoding technique used to convert the categorical values to numerical. A label encoding approach was taken in this project since the one-hot encoding was increasing the dimensionality of the data by almost infinity. Another improvement could be the use of better classifiers for performance testing and comparisons. This includes implementing maybe a better kernel with SVM which was more in accordance with the data produced. This also includes other widely used ensembles like Random Forest or AdaBoost.

9 References

[1] https://en.wikipedia.org/wiki/Tanzania

[2] https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/page/23/

[3] https://www.drivendata.org/

[4] https://github.com/dmlc/xgboost

[5] https://lightgbm.readthedocs.io/en/latest/Features.html

[6] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *Journal of Machine Learning Research* 12.Oct (2011): 2825-2830.

[7] Tran, Huu Dung, and A. W. M. Ng. "Classifying structural condition of deteriorating stormwater pipes using support vector machine." *Pipelines 2010: Climbing New Peaks to Infrastructure Reliability: Renew, Rehab, and Reinvest*. 2010. 857-866.

[8] Harvey, Richard, and Edward McBean. "Understanding Stormwater Pipe Deterioration Through Data Mining." *Journal of Water Management Modeling* (2014).

[9] Arymurthy, Aniati Murni. "Predicting the status of water pumps using data mining approach." *Big Data and Information Security (IWBIS), International Workshop on*. IEEE, 2016.

[10] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016

[11] Vinayak, Rashmi Korlakai, and Ran Gilad-Bachrach. "DART: Dropouts meet multiple additive regression trees." *Artificial Intelligence and Statistics*. 2015.

[12] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." *Annals of statistics* (2001): 1189-1232.